Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione
Corso di Laurea in Ingegneria Informatica

Il framework Phonegap vs Java Nativo per applicazioni mobile: il caso di un'applicazione multipiattaforma per l'acquisto di biglietti tramite telefono cellulare

Laureando:
Nicolò Calzavara

Relatore:
Ch.mo Prof. Federico Filira

Anno accademico 2010/2011

. . .

Sommario

Di seguito viene presentata una rassegna dei capitoli della tesi e il loro contenuto.

Capitolo 1: Si spiegano i due approcci fondamentali per lo sviluppo di applicazioni per piattaforme mobili. Si illustra la differenza tra applicazioni native e web application. Si fornisce un riassunto sulle motivazioni che spostano l'attenzione del mercato verso le piattaforme mobili;

Capitolo 2: Viene introdotto un approccio allo sviluppo delle applicazioni per dispositivi mobili rivolto a molteplici piattaforme, Phonegap;

Capitolo 3: Si illustrano le caratteristiche peculiari di questo framework con attenzione alle problematiche e metodi di sviluppo differenti rispetto alle varie piattaforme. In particolare si illustra come orientare lo sviluppo delle applicazioni per le piattaforme prese in esame: Symbian, Blackberry;

Capitolo 4: Si illustra lo studio seguito per una prima realizzazione dell'applicazione per acquisto di biglietti tramite SMS su piattaforma Symbian. Particolare attenzione viene rivolta ai limiti delle varie piattaforme e alle soluzioni consigliate dal progetto Phonegap;

Capitolo 5: Si illustrano gli approcci allo sviluppo delle applicazioni in linguaggio Java per le piattaforme Symbian, BlackBerry, Android. Viene fornito un'introduzione per ciascuna piattaforma sul modo in cui un'applicazione funziona e come interagisce con il sistema operativo del telefono stesso;

Capitolo 6: Si illustra lo sviluppo nativo per il sistema operativo Android di Google. Concetti fondamentali e linee guida per la creazione di applicazioni.

Indice

Elenco delle figure

Capitolo 1

Introduzione

Lo sviluppo di tecnologie wireless e l'enorme diffusione di dispositivi sempre più potenti e facilmente trasportabili ha spinto gran parte del mercato nella direzione del mobile business. Cresce costantemente il numero delle aziende che ricorrono ad applicazioni e piattaforme mobile per rimanere vicine ai loro clienti. Nel corso di questa tesi si illustrerà, mediante un caso pratico, come è possibile realizzare applicazioni per telefoni cellulari.

Esistono due approcci fondamentali per far interagire l'utente di un telefono cellulare con un'applicazione. Uno è rappresentato dall'interazione con una pagina web, tramite la navigazione in un network[1] o la navigazione in pagine offline. L'altro approccio è la realizzazione di un applicativo mirato e specifico per quel tipo di cellulare o sistema operativo. Nei due casi rispettivamente si parla di Applicazione Web e di Applicazione Nativa, vediamo di seguito le differenze principali e come i due approcci sono relazionati all'argomento della tesi.

Applicazione Web

Un'applicazione Web è un applicazione accessibile via web mediante un network. Nel caso di uno smartphone è possibile accedere tramite una connessione GSM o Wireless. In generale, si parla di applicazione Web quando la funzione svolta dalla pagina è più che la semplice consultazione. Il suo contenuto è in genere dinamico e interattivo. Ciò che può essere definito una web-app sono software come webmail, e-commerce, web forum, blog, giochi online e altro.

La "finestra"che consente all'utente l'interazione con queste applicazioni è il browser. In genere questo tipo di software è realizzato impiegando dei linguaggi di programmazione che lasciano la computazione e la gestione del comportamento della pagina al *Server*. Il codice della pagina in questo caso è compilato *lato-server*, al browser viene fornita la pagina web senza il codice di programmazione utilizzato al suo interno. Questo rende la gestione della pagina sicura in quanto l'utente non verrà a sapere come vengono gestiti i suoi dati. L'alternativa per realizzare una web application è quello di appoggiarsi a codice Javascript.

[1] In questo caso si intende l'insieme globale delle pagine disponibili sul World Wide Web

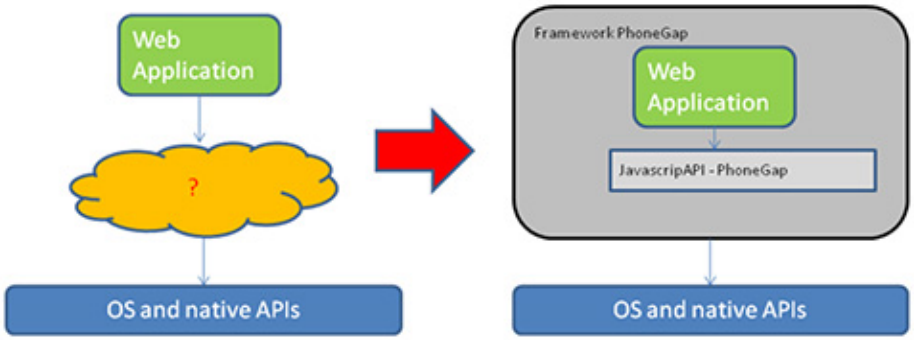


Fig. 1.1: Phonegap racchiude la web application e accede alle API dell'OS

Questo codice sarà all'interno della pagina e sarà compito del web Browser interpretarlo e gestirlo.

In questo lavoro il passo fondamentale che ci consente di portare le web-app dal web al mobile è il framework Phonegap [1]. Esso rappresenta una *sandbox*[2] che "contiene "la/e pagina/e web e consente di effettuare il salto verso la quasi totalità delle piattaforme mobili presenti sul mercato. Vedremo nei prossimi capitoli come funziona lo sviluppo mediante il framework Phonegap.

Applicazione Nativa

Un'applicazione mobile nativa è un software realizzato *ad hoc* per una piattaforma. Ciò comporta l'utilizzo del linguaggio di programmazione adatto, l'installazione di una SDK[3], la configurazione di eventuali piattaforme di sviluppo legate al sistema target. Per alcune piattaforme proprietarie è necessario utilizzare hardware adeguati per compilare le applicazioni. I guadagni derivanti da un approccio nativo sono legati a:

- un incremento delle prestazioni;
- una precisione superiore nella creazione dell'interfaccia utente;
- un maggiore controllo nella gestione degli eventi;
- possibilità di interfacciamento con tutte le possibilità hardware del dispositivo.

A fronte di un maggiore impegno progettazione e codifica esiste la possibilità disegnare l'applicazione in ogni dettaglio in modo da rendere il suo aspetto unico. Si pensi alla realizzazione di un gioco per un cellulare dove ogni aspetto deve essere disegnato manualmente per garantire che si presenti correttamente in aspetto e comportamento.

[2]Letteralmente: scatola di sabbia. Termine che indica, per analogia con il posto dove i bambini giocano con la sabbia, un ambiente isolato e sicuro rispetto al resto del sistema.

[3]Software Development Kit o Pacchetto di Sviluppo Applicazioni: indica un insieme di strumenti per lo sviluppo e la documentazione di software. Gli elementi fondamentali che contiene sono: un compilatore, le librerie standard dotate di interfacce pubbliche dette API, documentazione.

Nel corso di questa tesi verrà illustrato, mediante un caso pratico, come sia possibile realizzare applicazioni in un modo e nell'altro. Sebbene si sfrutti un solo caso di esempio, le piattaforme toccate sono molteplici. Come vedremo emergeranno pregi e difetti dei due approcci, con riferimento alle rispettive piattaforme.

Cosa spinge verso il mobile

Come approfondimento alle motivazioni per cui lo sviluppo di software si muova verso le piattaforme mobili, si espongono alcune delle ragioni di questo fenomeno.

Oltre alla grande disponibilità di dispositivi mobili con capacità di calcolo notevoli è di considerevole rilevanza la spinta degli utenti privati a voler partecipare in maniera più attiva alla comunicazione attraverso il web.

L'esigenza di essere sempre connessi agli affetti o alla vita lavorativa rappresenta un bisogno che gli utenti sentono sempre più forte, inoltre spinge le piattaforme mobili e le case produttrici di software a fornire gli strumenti per permetterlo. Leggere le ultime notizie, consultare la propria casella di posta elettronica, verificare il proprio conto in banca o l'andamento del mercato, condividere dei contenuti o solo il proprio stato d'animo; sono tutte comodità a cui gli utenti moderni di smartphone sono poco disposti a rinunciare. Il pubblico, in Italia più che in altri paesi [2][4], è disposto a investire su uno smartphone piuttosto che su un cellulare più economico. Ciò che ci si può aspettare è che nel prossimo futuro una base di utenti sempre più larga disponga di apparecchi sempre molto potenti e si aspetti di sfruttare la le comodità che offrono [3].

La sempre crescente attività dei social network di ogni tipo ne è un chiaro esempio.

Assieme agli sviluppi dell'hardware è determinante la sempre maggiore spinta di internet a portare l'utente ad avere un'esperienza sempre piú coinvolgente nel navigare. Questa interazione con il web avviene non solo grazie al dispositivo hardware stesso; ma grazie anche ai web browser. Le case produttrici di browser svolgono un grande ruolo nel mercato di internet. Cercano di rendere l'esperienza di navigazione la migliore possibile per i loro utenti, in termini di velocità, facilità di utilizzo, compatibilità con il maggior numero di pagine; il tutto volto a guadagnare una maggiore fetta di mercato.

Queste due motivazioni hanno spinto le associazioni che progettano standard a concentrare gli sforzi nel creare una base che metta insieme l'esigenza dell'utente e quella delle grandi compagnie. Le compagnie produttrici di browser chiedono uno standard, gli utenti chiedono una maggiore velocità e interattività con il web.

Per rispondere a queste molteplici esigenze W3C[5] e WHATWG[6] hanno iniziato a lavorare nel 2004 a un nuovo standard. Le successive pubblicazioni da parte del W3C delle

[4]Nel terzo trimestre del 2010 sono infatti 18,5 milioni gli italiani che dichiarano di possedere uno smartphone, erano 12,3 milioni solo un anno prima.

[5] Questo organismo internazionale, fondato nell'ottobre 1994, è composto da università e aziende private (tra cui IBM, Microsoft, Netscape Communications Corporation, Novell Softquad, Spyglass e Sun Microsystems) e coordinato da LCS (Laboratory for Computer Science). Esso ha lo scopo di guidare lo sviluppo del Web e di definirne gli standard.

[6]Web Hypertext Application Technology Working Group è una comunità crescente di persone interessate a sviluppare il Web. Si concentra principalmente nello sviluppo di HTML e le API che servono alle

prime bozze per lo standard cambiano e cambieranno in maniera sostanziale il modo di vedere Internet. L'HTML5 è il nome che è stato dato al progetto che diverrà il nuovo standard per le pagine HTML.

L'HTML5, per le nuove funzionalità che introduce, ha già la fama di "standard delle web app ", vedremo come ha influenzato lo sviluppo della tesi.

applicazioni Web. Il WHATWG è stato fondato da individui di Apple, la Mozzilla Foundation, e Opera software nel 2004, dopo un workshop del W3C. Apple Mozzilla e Opera stavano diventando molto interessate circa la direzione del lavori del W3C su XHTML, mancanza di attenzione verso HTML e apparente disprezzo per la necessità di sviluppatori capaci. Quindi, in risposta, queste organizzazioni hanno posto il loro obiettivo nel risolvere questi problemi e hanno dato vita al Web Hypertext Application Technology Working Group

Capitolo 2

Phonegap

Phonegap é un progetto Open Source della *Nitobi Software*, un'azienda che crea applicazioni mobile e web applications da piú di dieci anni. Citando il sito [1]:

"PhoneGap is an open source development framework for building cross-platform mobile apps. Build apps in HTML and JavaScript and still take advantage of core features in iPhone/iPod touch, iPad, Google Android, Palm, Symbian and Blackberry SDKs. Since winning the Web 2.0 Expo LaunchPad competition in 2009, the open source code has been downloaded more than 300,000 times and thousands of apps built using PhoneGap are available in mobile app stores and directories. "

Consiste in un insieme di librerie statiche che permettono di sviluppare velocemente ed efficacemente applicazioni per dispositivi mobili di diverse famiglie. L'idea fondamentale di questo progetto è nel cercare di realizzare lo slogan "Write once, port everywhere ".

Phonegap si propone di focalizzare gli sforzi degli sviluppatori sull'applicazione piuttosto che perdere tempo ad adattarla ad ogni piattaforma. Per fare ciò un'applicazione realizzata con Phonegap richiede solo la conoscenza di HTML, CSS e Javascript. Il porting verso le varie piattaforme viene fatto installando gli ambienti di sviluppo relativi e compilando la webapp realizzata[1]. I requisiti sono quindi di installare le SDK, dove richiesto dalla piattaforma, e gli strumenti per consentire la compilazione delle applicazioni.

2.1 Perché Phonegap

Sviluppare un'applicazione per tutte le maggiori piattaforme richiede la conoscenza dei linguaggi delle stesse, l'installazione di IDE[2] per lo sviluppo, di personale in grado di seguire lo sviluppo. Una pianificazione per l'ingegnerizzazione del software è d'obbligo per garantire la realizzazione di un prodotto uniforme malgrado le eterogeneità dei dispositivi. La realizzazione di un'applicazione su varie piattaforme richiede quindi tempo e denaro,

[1] A seconda della piattaforma e della versione di Phonegap utilizzata, può essere necessario adattare o ridurre l'impiego di codice javascript.

[2] *Integrated Development Enviroment* o *Ambiente di Sviluppo Integrato*; normalmente consiste in un editor di codice sorgente, un compilatore e/o un interprete, un tool di building automatico, e un debugger

per reclutare il personale, formarlo se necessario, e preparare le piattaforme di sviluppo per ogni piattaforma.

Rivolgersi al mercato in questo modo implica l'impiego di importanti risorse che potrebbero essere volte allo sviluppo dell'applicazione stessa o ad una fase di test più accurata. A causa delle consistenti risorse da dover mettere in gioco ciò che alcune aziende fanno è sviluppare la propria applicazione solo per le piattaforme più utilizzate, o concentrare gli sforzi sulle piattaforme con meno restrizioni. Ciò ovviamente porterebbe a ridurre il potenziale mercato dell'applicazione oltre che impedire automaticamente agli utenti delle altre piattaforme di utilizzarla.

Con Phonegap lo sviluppatore può dedicarsi maggiormente allo sviluppo della sua web application senza pensare agli intralci o impedimenti delle varie piattaforme. Chiaramente le possibilità dell'applicazione saranno limitate a quelle di una applicazione web[3].

2.2 Come funziona

Phonegap fà da ponte tra il sistema operativo e la web application realizzata dallo sviluppatore. Si intuisce che occorre che dalla web application esista una modalità standard per invocare le API native in maniera indipendente dal tipo di piattaforma sottostante. PhoneGap è infatti un framework che permette ad una web application di invocare le API native mediante funzioni JavaScript di cui è possibile vedere gli esempi forniti sul sito di riferimento.

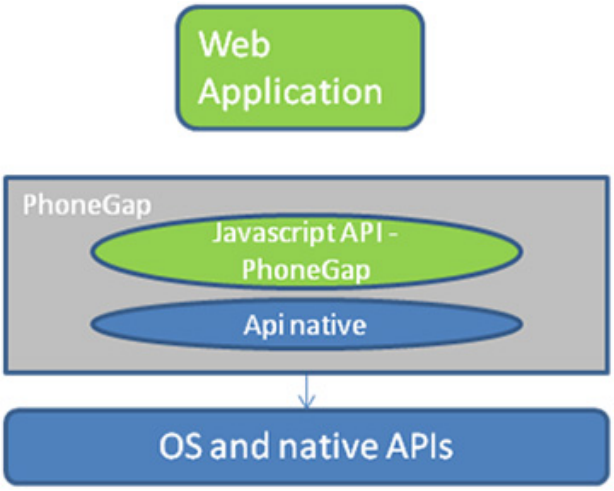


Fig. 2.1: Interfacciamento di Phonegap

Il vantaggio nell'utilizzo di questo framework è che esso fornisce l'aggancio tra la piattaforma nativa e la web application, di modo che uno sviluppatore che non ha conoscen-

[3]Vedremo che sono utilizzabili anche i comandi javascript che mette a disposizione il browser nativo; capitolo 3.

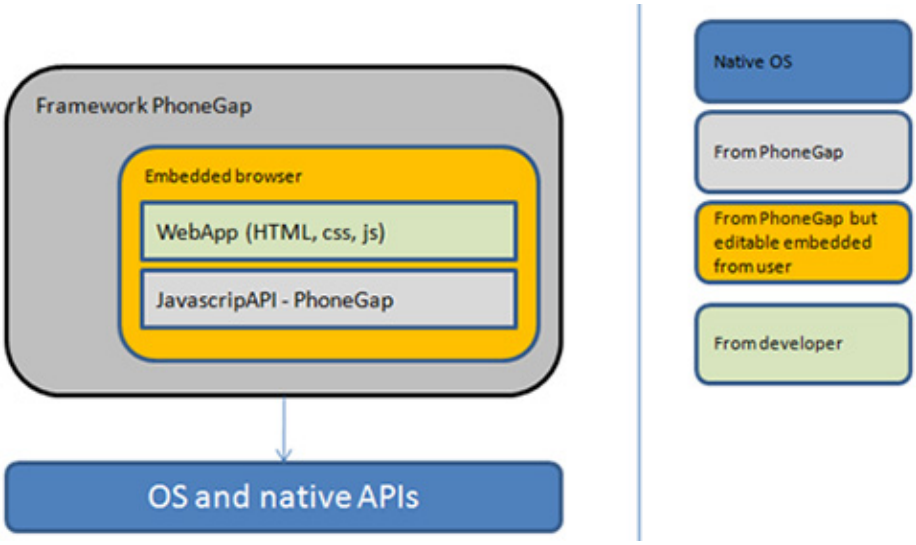


Figura 2.2: Architettura di una applicazione realizzata con Phonegap

ze specifiche sui linguaggi nativi, possa dedicarsi allo sviluppo della web application con tecnologie standard.

La caratteristica fondamentale di PhoneGap sta nel fatto che, una volta realizzata l'applicazione per il mobile con questo framework, l'applicazione si adatta a qualunque piattaforma supportata.

PhoneGap userà il linguaggio nativo della piattaforma per accedere alle risorse hardware e software in modo da aggiungere le funzionalità di base al motore JavaScript e renderle così facilmente utilizzabili dall'applicazione come fossero tradizionali metodi di libreria.

In base alla tipologia di piattaforma con la quale dovrà interfacciarsi, l'implementazione di aggancio sarà di conseguenza sviluppata in Objective C per iPhone, in Java per Android, e così via; e tale implementazione è fornita dallo stesso framework. In pratica esiste un runtime basato su WebKit[4] in cui vengono iniettate le componenti statiche.

Il risultato sarà un pacchetto composto di due elementi principali con differenti responsabilità che però cooperano tra loro per fornire delle funzioni a valore aggiunto. Nel caso specifico il runtime si occupa di dialogare direttamente con il dispositivo e le parti statiche offrono l'interfaccia verso l'utente.

L'uso di JavaScript e di Ajax consente poi di dare vita alle applicazioni che possono avere comportamenti complessi e comunicazioni verso endpoint remoti realizzando di fatto un'applicazione completa in cui il client è un elemento fondamentale in linea con le moderne applicazioni Web 2.0 (e in contrapposizione a quanto accadeva solo pochi anni fa per le applicazioni web la cui interfaccia era completamente elaborata server side).

Ora illustriamo mediante la Figura 2.2 come è strutturata l'architettura di Phonegap. Partendo dal basso verso l'alto, si può notare che la parte in blu è quella del sistema operativo della piattaforma nativa e come tale viene fornita. Al livello immediatamente

[4]É il motore di browser come Safari e Google Chrome

sopra troviamo il framework PhoneGap (in grigio) e anch'esso ci viene fornito insieme alle API JavaScript. È curioso notare il rettangolo in arancione che incapsula la web application e le API JavaScript. Questo è stato fatto per mettere in evidenza che l'oggetto browser può essere aperto all'interno dell'applicazione che si sta sviluppando e dunque dentro PhoneGap. Per comportamento di default il browser viene aperto esternamente all'applicazione. Di conseguenza ad ogni richiesta di una pagina web si ha l'apertura del web browser. L'apertura di una pagina nel browser embedded nasconde la barra degli indirizzi, i bottoni forniti dal browser di default. In questo modo si oscura relativamente il fatto che si tratti di una pagina web, vantaggio indiscutibile nel rendere l'esperienza più simile a quella di un'applicazione nativa.

La scelta di aprirlo o meno all'interno dell'applicazione sta allo sviluppatore e nel momento in cui decidesse di aprirlo all'interno dell'applicazione deve apportare delle semplici modifiche al codice di aggancio nativo. Infine, in verde è evidenziata la parte a carico dello sviluppatore che è la web application, realizzata all'interno di PhoneGap.

Nel nostro caso pratico il problema non si pone, in quanto la creazione di una applicazione per la piattaforma Symbian vincolava le pagine HTML ad una sola per i motivi che vedremo al capitolo 3.

Per cui, il pacchetto applicativo destinato alla distribuzione conterrà file statici come HTML , JavaScript , CSS e le classi native di aggancio. Il pacchetto dovrà essere opportunamente firmato prima di essere distribuito sullo store ufficiale delle piattaforme di riferimento.

2.3 Conoscenze di base

La creazione di un'applicazione richiede la sola conoscenza dei linguaggi: HTML, CSS, Javascript. Questi tre linguaggi contribuiscono a creare la struttura della pagina web (HTML), di personalizzarne il suo aspetto (CSS), e di renderla dinamica[5] e interattiva mediante il codice Javascript. In seguito si illustra in breve in cosa consiste ciascuno dei tre.

2.3.1 HTML

HTML stà per HyperText Markup Language, è il linguaggio predominante delle pagine web [4]. Per descrivere le pagine web viene usato un sistema di tag. I tag sono costituiti da parentesi angolate (come <html>), il browser è programmato per riconoscere i tag e la loro funzione, un volta riconosciuti non li mostra nella pagina finale ma li utilizza per costruire la pagina. E' possibile inserire immagini e oggetti e creare moduli (d'ora in poi *form*) interattivi. Esistono tag per strutturare il documento con link, intestazioni, liste, paragrafi e altri elementi. E' possibile inserire all'interno del codice della pagina altri linguaggi attivi, ad esempio si può incorporare del codice Javascript o utilizzare altri linguaggi più *server-*

[5]Per pagina dinamica si intende una pagina il cui contenuto è del tutto o in parte generato al momento.

side per svolgere compiti più complessi come PHP[6] o Java[7]. Le specifiche del linguaggio HTML sono state pubblicate dal World Wide Web Consortium (W3C; vedi nota a piè pagina n°5 a pagina 3).

Le versioni

Il linguaggio HTML è stato messo a punto da Tim Berners-Lee, allora ricercatore al CERN, a partire dal 1989. Quest'ultimo annunciò ufficialmente la creazione del web su Usenet nell'agosto del 1991. E' comunque dal 1993 che lo stato dell'HTML è considerato sufficientemente avanzato da poter parlare di linguaggio (HTML viene allora battezzato simbolicamenteHTML 1.0). Il navigatore internet usato all'epoca era chiamato NCSA Mosaïc.

Fig. 2.3: Il logo originale di HTML.

L'RFC1866, datato novembre 1995 rappresenta la prima versione ufficiale dell'HTML, cioè l'HTML 2.0

Dopo la breve apparizione dell'HTML 3.0, che non vide mai ufficialmente la luce, l'HTML 3.2 divenne standard ufficiale il 14 gennaio 1997. I contributi più importanti dell'HTML 3.2 erano la standardizzazione delle tabelle nonché di un gran numero di elementi di presentazione.

Il 18 dicembre 1997, l'HTML 4.0 viene pubblicato. La versione 4.0 del linguaggio HTML standardizza soprattutto i fogli di stile e i frame. La versione HTML 4.01, apparsa il 24 dicembre 1999, apporta qualche modifica minore all'HTML 4.0.

HTML5

Chi è abituato a lavorare con lo sviluppo di pagine HTML, sa che per creare la pagina il layout si è abituati a utilizzare una miriade di tag .. , e utilizzare gli attributi *id*= " oppure *class*=" per indicare la loro funzione e caratterizzarli poi nel file CSS. Una pagina in HTML 4 è strutturalmente realizzata come Figura 2.4(a). Ora come si vede in Figura 2.4(b) la pagina in HTML5 presenta le stesse separazioni concettuali, solo che ora vengono rappresentate dai tag stessi.

Quindi il markup per la pagina HTML5 finale è strutturato in questo modo:

<body>
<header>...</header>

[6]PHP : definizione

[7]Mediante pagine JSP è possibile creare pagine web con contenuto dinamico con l'utilizzo di codice java incluso nella pagina html.

(a) La struttura di una pagina in HTML 4 (b) Il sistema di tag di una pagina in HTML 5

Fig. 2.4: Le struttura di una pagina HTML 4 (a), e di una pagina HTML 5 (b)

```
  <nav>...</nav>
  <article>
    <section>
      ...
    </section>
  </article>
  <aside>...</aside>
  <footer>...</footer>
</body>
```

Una tale forma consente allo standard di essere retro compatibile, evitando scelte difficili agli sviluppatori web.

Oltre a questi cambiamenti nella struttura della pagina, vengono messi a disposizione alcuni tag aggiuntivi per gestire multimedialità e personalizzazione delle pagine:

<canvas> Il seguente tag mette a disposizione uno spazio rettangolare di dimensione personalizzabile. All'interno è possibile disegnare a piacere qualunque tipo di forma, immagine, animazione si desideri. La creazione delle illustrazioni avverrà tramite primitive Javascript.

<audio><video> Mediante questi tag è possibile includere e controllare contenuti multimediali. Fino ad ora tutto ciò era reso possibile tramite Flash[8]. Ora mediante javascript è possibile gestire i video senza ricorrere ai metodi usati finora, appesantendo meno la pagina. Sono già pronte alcune librerie per la gestione di video, anche se i semplici comandi javascript messi a disposizione dalle specifiche sono sufficienti.

2.3.2 CSS

Il *Cascading Style Sheets (CSS)* è un linguaggio che definisce lo stile, la formattazione e l'aspetto di un documento scritto in un linguaggio di markup [5]. E' più comunemente

[8]Adobe Flash: software per uso prevalentemente grafico per la realizzazione di animazioni vettoriali per il web. Nelle ultime versioni è divenuto un potente strumento per la fruizione di streaming video/audio.

usato assieme alle pagine web scritte in HTML o XHTML, ma può anche essere applicato ad ogni tipo di documento XML.

Il linguaggio definisce i comportamenti visivi di una pagina web. Ad ogni elemento della pagina viene associata una serie di comandi che modificano l'aspetto dell'elemento stesso, e il suo comportamento all'interno della pagina. L'intero codice va a creare uno script che viene associato alla pagina stessa e definisce i l'aspetto degli elementi in base ad alcune azioni dell'utente (spostamenti del mouse sopra elementi della pagina, click di alcuni elementi o altro).

Definisce il posizionamento, la colorazione, il comportamento del testo, degli elementi, di immagini.

Mentre l'autore di un documento tipicamente associa il proprio documento ad uno specifico CSS, i lettori possono utilizzare un foglio di stile proprio e sovrascrivere quello che il proprietario ha specificato.

Le specifiche del CSS sono aggiornate dal W3C.

2.3.3 Javascript

Javascript è un linguaggio di programmazione interpretato con un rudimentale orientamento agli oggetti [6]. L'obiettivo del liguaggio è quello di consentire l'inserimento di contenuto eseguibile in una pagina web. Significa che una pagina web non resta solo HTML statico ma può includere programmi dinamici che interagiscono con l'utente, controllano il browser e creano dinamicamente contenuti HTML. Può inoltre ottenere informazioni dall'utente e dal browser stesso. Interagisce con il CSS per modificare l'esperienza dell'utente durante la navigazione. Durante la navigazione può, dinamicamente, nascondere o mostrare, alterare le proprietà o gli attributi associate a particolari elementi della pagina.

Nel nostro caso Javascript è utilizzato principalmente per gestire la navigazione dell'utente. Come vedremo nell'applicazione pratica, vengono nascoste tutte le sezioni con cui non si vuole che l'utente interagisca. Questo per molteplici motivi, innanzitutto guidare l'utente nella navigazione, impedire che interagisca con parti della pagina su cui non ha diritto di agire; inoltre ci possono essere motivazioni tecniche[9].

Un esempio della potenza di Javascript è stato riscontrato nella simulazione del browser, in particolare la funzionalità del bottone *Indietro* per tornare alla pagina esattamente precedente a quella visitata (Figura 2.5).

Fig. 2.5: La barra di navigazione superiore per il widget di test.

Per ulteriori funzioni utili che sono state utilizzate si può far riferimento al capitolo 3.

[9]Il riferimento è relativo alle limitazioni su sistema operativo Symbian. Vedi 3.2.

Javascript e la velocità

Sicuramente è capitato a chiunque di visitare una pagina web con un bellissimo aspetto grafico e chiedersi come mai ci metta tanto a caricarsi completamente. E' un problema che può avere molte cause diverse, a seconda di come è realizzata la pagina. Spesso è a causa dell'uso pesante di codice Javascript per la gestione di animazioni, navigazione, dinamicità della pagina. Lo stesso problema si è riscontrato nello sviluppo con Phonegap; pagine contenenti molto codice Javascript tendono a essere lente nel caricamento. C'è da sorprendersi relativamente soprattutto se pensiamo che a fare il rendering della pagina è il browser di un dispositivo mobile. Il problema è stato riscontrato sui vari dispositivi , in forma più o meno accentuata.

Symbian Il rendering funziona bene, non sono disponibili animazioni ma non ci sono limitazioni all'uso di framework Javascript.

BlackBerry La piattaforma che rende più difficoltoso lo sviluppo in assoluto, permette solamente l'uso di framework Javascript realizzati ad-Hoc. OS5 e OS6 promettono di rimediare a questo divario con le altre piattaforme.

L'eccezione per questa sezione è data dalla piattaforma iPhone, che dispone accelerazione hardware per il rendering delle pagine web. Ciò la rende la piattaforma principe per la navigazione web via telefono cellulare e allo stesso tempo per tutti gli approcci di sviluppo con lo stesso principio di Phonegap.

2.3.4 Client Side

Essendo un'applicazione Client Side, è possibile gestire la visualizzazione tramite il rendering del browser nativo. Non è utilizzabile nessuna forma di linguaggio Server-side.

Nel nostro caso sarebbe stato utile gestire la navigazione e le informazioni inserite dall'utente tramite PHP, uno dei linguaggi più usati per gestire la comunicazione Server-Client. Qualora l'utente abbia bisogno di inserire dei dati all'interno della pagina (nome del giocatore per un minigioco, lingua desiderata o altro) una delle possibilità poteva essere quella di gestire il tutto tramite PHP. La natura del linguaggio è tale che si rende necessario avere un interprete/server avviato sulla piattaforma. Chiaramente non esiste la possibilità di disporre di linguaggi Server Side su normali smartphone.

2.3.5 Osservazioni

Come è stato spiegato la gestione di tutti i dati che verranno inseriti dall'utente dovrà essere fatta tramite javascript. La gestione di particolari eventi sulla pagina html (click su bottoni, liste, checkbox, form), e interazione in genere con la stessa devono essere supportati con funzioni javascript se l'html di base non lo prevede. Questo fatto lascia poche alternative all'implementazione. L'avvento di HTML5 in parte semplifica questo, implementando funzionalità che non richiederanno l'utilizzo di plugin Javascript aggiuntivi. Per fare un esempio sulle piattaforme che non supportavano il salvataggio di dati (al

momento dello sviluppo, nella forma di chiave-valore) era consigliato utilizzare un plugin Javascript, Lawnchair[10]. Un iPhone aggiornato all'ultima versione di iOS ha disponibili queste funzionalità nativamente perché Safari stesso le rende disponibili.

2.4 API e lo *status quo* dello sviluppo

Il progetto *Phonegap* é sempre in via di sviluppo. La documentazione sulle api è aggiornata costantemente. Le funzionalità disponibili al momento sono:

Accelerometro Utilizza il sensore di moto.

Bussola Ottiene la direzione in cui il dispositivo punta.

Camera Cattura una immagine con la fotocamera.

Contatti Lavora con il database dei contatti.

Device info Raccogli informazioni sul telefono.

Events Aggancia gli eventi nativi tramite JavaScript.

File Connetti al file system nativo tramite JavaScript.

Geolocazione Rende l'applicazione coscente della locazione.

Media Registra e ascolta file audio.

Network Controlla lo stato della rete.

Notifica Notifiche visuali, acustiche, tattili.

Storage Agganciati alle opzioni dello storage nativo del sistema.

Per avere un'idea di cosa è supportato e su che piattaforma, la tabella presente sul sito principale è rivelatoria, Figura 2.6

Il sito ufficiale ricorda comunque che aggiornamenti più frequenti sono presenti al link:`http://wiki.phonegap.com/w/page/28291160/roadmap-planning`.

2.4.1 Come è cambiato il progetto

Durante gli ultimi mesi del 2010, più o meno quando l'esperienza dello stage entrava nel vivo, il progetto ha subito un importante cambiamento. Gli ultimi draft su HTML5 hanno spinto le compagnie produttrici di smartphone ad aggiornare i browser delle piattaforme alle nuove specifiche. Sebbene non definitivi e soggetti a cambiamento, i draft rappresentano delle direttive a cui serve solo del tempo per diventare uno standard effettivo. Così

[10]Vedere: Progetti Associati al paragrafo 2.6

Fig. 2.6: Le API e il loro supporto per le varie piattaforme.

facendo si sono aperte le porte per i sistemi che ancora non supportavano l'HTML5 alle webApplication.

All'inizio dello studio per lo sviluppo di applicazioni su Phonegap il progetto non si era ancora orientato verso il pieno supporto di HTML5. Le api disponibili in quel momento si possono vedere in Figura 2.7.

Le caselle rosse indicano che la funzione non è accessibile perché la piattafoma stessa non lo permetteva. Come si intuisce le limitazioni piú forti dell'approccio Phonegap erano proprio queste. Le caselle gialle invece indicano che l'implementazione per quella funzione é in via di sviluppo. Le caselle verdi indicano che l'API é disponibile per quella piattaforma. Un altra possibilitá é che non esistano implementazioni al momento. In questo caso la cassella resterebbe bianca.

Dopo il rilascio dei browser aggiornati sulle varie piattaforme, il progetto stesso non è stato a guardare. Per adattarsi al mercato che cambia sono state adattate per ogni piattaforma le primitive per la gestione dei nuovi tag resi disponibili dall'HTML5. Sono andate perse nell'aggiornamento le API che supportavano comandi particolari per ciascuna piattaforma (es. le API per mandare SMS su Symbian). Come vedremo nell'applicazione demo realizzata in Phonegap, verrà messo in risalto il fatto che ciò che rende disponibile Phonegap sotto forma di API è comunque un sottoinsieme delle possibilità della piattaforma. Conoscendo le piattaforme in particolare si possono utilizzare chiamate a funzioni Javascript già disponibili nativamente nel browser del telefono.

	iPhone	Android	Blackberry (OS 4.5)	Symbian[1]	Windows Mobile	Palm	Maemo*
Geolocation							
Accelerometer			OS 4.7				
Camera							
Vibration							
Contacts API							
SQLite Functionality							
XMPP API							
File system IO						Read only	
Gesture / Multitouch		Android 2.0					
SMS API							
Telephone API							
Copy / Paste							
Sounds (Play)							
Sounds (Record)							
Bluetooth							
Wifi Adhoc connection							
Maps							
Orientation change							
Network availability							
Magnetometer	3GS only						
Storage							

Fig. 2.7: La Roadmap del progetto Phonegap sul finire del 2010.

2.5 Phonegap fà al caso mio?

Se siete degli sviluppatori di applicazioni mobile o di applicazioni web e avete bisogno di sapere se Phonegap fà a caso vostro allora conviene fare alcune analisi.

Alcuni aspetti chiave da valutare per decidere se sfruttare il framework Phonegap per il proprio progetto sono:

- Sebbene tutte le piattaforme si stiano allineando agli standard di html5, alcune permettono altre funzionalità che vanno oltre quelle offerte da Phonegap. Occorre valutare quali sono le esigenze e tenere conto della possibilità di utilizzare queste possibilità.

- La fase di sviluppo, malgrado non richieda particolari attenzioni per la stesura della/e pagina/e html, necessita di qualche attenzione nello sviluppo della parte Javascript. Ogni piattaforma ha il suo browser nativo e gestisce il codice in maniera diversa.

- A prescindere del sistema operativo della piattaforma obiettivo, è d'obbligo una fase rigorosa di test per verificare il funzionamento.

- Come prospettive future é auspicabile che ogni piattaforma migliori il proprio browser nativo. Migliorando il rendering delle pagine web, e quindi delle applicazioni realizzate in Phonegap.

- Qualora si disponesse di un'applicazione web funzionante, magari già sviluppata sfruttando html 5, il porting verso le piattaforme mobili diventa molto agevole.
- Come nota finale si può aggiungere che le piattaforme coperte da Phonegap sono al momento la quasi totalitá del mercato.

Fig. 2.8: Sistemi supportati dalla piattaforma Phonegap.

2.6 Progetti Associati

Per aiutare a migliorare l'applicazione finale occorre sapere che esistono dei progetti che nello sviluppo possono tornare utili. Come descritto alla pagina `http://www.phonegap.com/tools` ci sono vari strumenti utili che possono servire a facilitare il lavoro dello sviluppatore. Sono presenti framework Javascript molto stringati, API per lo sviluppo di giochi Javascript, tool per disegnare widget. Alcuni di questi strumenti si sono rilevati utili, altri sono comunque degni di attenzione. Tra tutti:

XUI Framework Javascript particolarmente stringato. E' utile per realizzare applicazioni per i browser più prestanti. Vedremo nel capitolo 4 i risultati del suo impiego.

Sencha/Sencha animation Permette lo sviluppo di applicazioni web che sembrano native per iPhone e Android.

JQuerymobile Plugin per il framework Javascript JQuery in via di sviluppo. Promette di fornire un'interfaccia utente unificata a applicazioni web per ogni tipo di piattaforma mobile. [11]

Ripple Progetto recentemente aggiuntosi come collaboratore. Fornisce mediante un'aggiunta a Google Chrome il supporto per il testing della pagina realizzata su dispositivi mobili. E' possibile visualizzare l'apparenza della pagina su diversi tipi di schermi simulati da questa applicazione.

Lawnchair Plugin client side che aggiunge la possibilità di memorizzazione persistente mediante JSON.

[11] Al momento siamo alla alpha #3.

JQTouch Plugin per JQuery per lo sviluppo web su iPhone, iPodTouch.

Capitolo 3

Sviluppo con Phonegap

In questa sezione vedremo quale procedimento viene seguito e quali sono i requisiti per realizzare un'applicazione con Phonegap. In particolare si esamina lo sviluppo per la piattaforma Symban e Blackberry. Verranno esposti i requisiti e le possibilità che offrono questi sistemi operativi.

3.1 Direttive principali per lo sviluppo

L'utilizzo di Phonegap per lo sviluppo di applicazioni mobile comporta, a seconda del target, l'installazione di un IDE per lo sviluppo e di strumenti per la scrittura, compilazione, pacchettizzazione del software [7]. Ogni piattaforma dispone di una SDK che consente agli sviluppatori di creare software, non sempre è un requisito necessario per lo sviluppo Phonegap. Rappresenta tuttavia una risorsa di cui è utile disporre in quanto con essa sono forniti i simulatori per le varie versioni di sistema operativo.

Ciò che è bene fare prima di iniziale la codifica è effettuare uno studio sistematico di tutte le piattaforme, determinare i limiti di ciascuna e pianificare lo sviluppo in modo da avere applicazioni il più possibile uniformi per tutte le piattaforme.

Come è già stato introdotto al capitolo precedente Phonegap entra in gioco solamente quando si deve realizzare il pacchetto applicazione per la piattaforma desiderata.

3.1.1 Requisiti

L'approccio Phonegap richiede alcuni requisiti comuni a cui le pagine web devono sottostare:

- La pagina html principale deve essere *index.html*. L'applicazione partirá dalla stessa appena avviata.
- Un link al file javascript *phonegap.js* nell'intestazione della pagina principale.

- Per Windows, varie piattaforme hanno come requisito l'installazione di Cygwin[1] con l'utility *make* per poter pacchettizzare l'applicazione.

- Trovare un'icona per l'applicazione. Il nome deve essere "Icon.png".

Come si intuisce l'unico pegno che si paga per avere la completa portabilitá é quello di dover installare una tantum l'ambiente di sviluppo relativo alla piattaforma. Nello specifico per iPhone ad esempio è richiesta una versione recente del sistema operativo OS X, l'ambiente di sviluppo Xcode e la SDK Apple installata. Per sviluppare applicazioni per cellulari Nokia che supportano Web Run Time, non è necessario avere altro che il blocco note e un compressore zip per creare applicazioni con Phonegap. Sebbene si riesca a creare l'applicazione funzionante esistono dei plugin per Eclipse[2] che automatizzano la creazione e fornisce anche una preview della pagina creata all'interno dell'IDE stesso.

A seconda del sistema target altri requisiti possono entrare in gioco:

- La compilazione può essere vincolata all'installazione dell'SDK (Software Development Kit) relativo alla piattaforma target e ad eventuali plugin per gli ambienti.

- Se la piattaforma lo supporta (es. iPhone) è bene rimpiazzare la schermata di caricamento di default con qualcosa di più gradevole.

- Per la piattaforma iPhone, può essere consigliato utilizzare immagini ad alta risoluzione qualora lo schermo della piattaforma obiettivo lo consenta.

3.1.2 Distribuzione

Le politiche di distribuzione per le applicazioni realizzate con Phonegap sono le stesse delle applicazioni native. I requisiti variano molto a seconda di ciascuna piattaforma ma i passi fondamentali da seguire sono sostanzialmente questi:

1. creazione di account da sviluppatori presso i portali;

2. acquisto della licenza di distribuzione (tempi di attesa piú o meno lunghi per l'abilitazione dell'account);

3. rispetto di vincoli sulla realizzazione delle applicazioni;

4. sottomissione dell'applicazione allo store;

5. attesa dell'approvazione da parte dello store;

[1] Cygwin è un software libero che consente di svolgere su Windows alcuni compiti in maniera esteticamente e funzionalmente simile ad un sistema UNIX

[2] Uno degli IDE di sviluppo di maggiore successo.

3.2 Piattaforma Symbian

Phonegap per la piattaforma Symbian si appoggia a "Nokia's Web Runtime "che rappresenta il motore del Web Browser per S60 e S40. Il rendering del contenuto delle pagine è dato dalla tecnologia WebKit. Le piattaforme che possono essere il target di applicazioni Phonegap sono pertanto Symbian S60 5th Edition e S60 3rd Ed. FP2.

Questo significa che tutti i cellulari con sistema operativo Symbian con circa meno di cinque anni di età potranno utilizzare un'applicazione realizzata in questo modo.

Lo sviluppo per cellulari Nokia con sistema operativo Symbian è pittosto semplice, è sufficiente utilizzare quanto fornito dai sorgenti phonegap.

Prima di iniziare lo sviluppo è consigliabile dotarsi di un ambiente con make. Lo script fornito assieme a Phonegap gestisce la pacchettizzazione. Qualora si lavorasse su Windows è necessario dotarsi di Cygwin, durante l'installazione dello stesso occorre selezionare il pacchetto con make e il pacchetto di compressione zip in modo da poter utilizzare lo script dalla shell Cygwin.

Una volta verificato è necessario scaricare Phonegap ed estrarre i contenuti in una cartella apposita e seguire

- Una volta individuata la cartella relativa al sistema Symbian (*/phonegap-symbian.wrt*) l'applicazione andrà a rimpiazzare il file *index.html* situato nella sottocartella */www*.
- Assicurarsi che il file *phonegap.js* sia linkato nell'intestazione della pagina.
- Rimpiazzare l'icona di default *Icon.png* con un'icona a piacere
- Eventualmente modificare il file *info.plist*
- Sviluppare la propria applicazione attorno al fine html, le api Phonegap saranno accessibili solo da questa pagina.
- Raggiungere mediante shell Cygwin o Unix la cartella *phonegap-symbian.wrt* e lanciare make, lo script produrrà *phonegap-symbian.wrt/app.wgz*.

Il file con l'applicazione è *app.wgz*, può essere trasmessa a un cellulare via bluetooth o via cavo usb. Per installarla è sufficente aprirla tramite un file manager dal telefono stesso. Per una fase di testing prima di arrivare all'installazione è possibile ricorrere a due alternative. Una possibilità è quella di installare il simulatore della versione di Symbian che ci interessa testare. Purtroppo per fare ciò è necessario installare tutta la SDK in toto. Ciascuna versione del sistema operativo comporta l'installazione di una diversa SDK. Quella relativa a Symbian S60 5th edition richiede attorno a 1GB di spazio su disco. Qualora non si disponga di un cellulare con la stessa versione del sistema operativo è vivamente consigliato di fare una prova dell'applicazione sul simulatore.

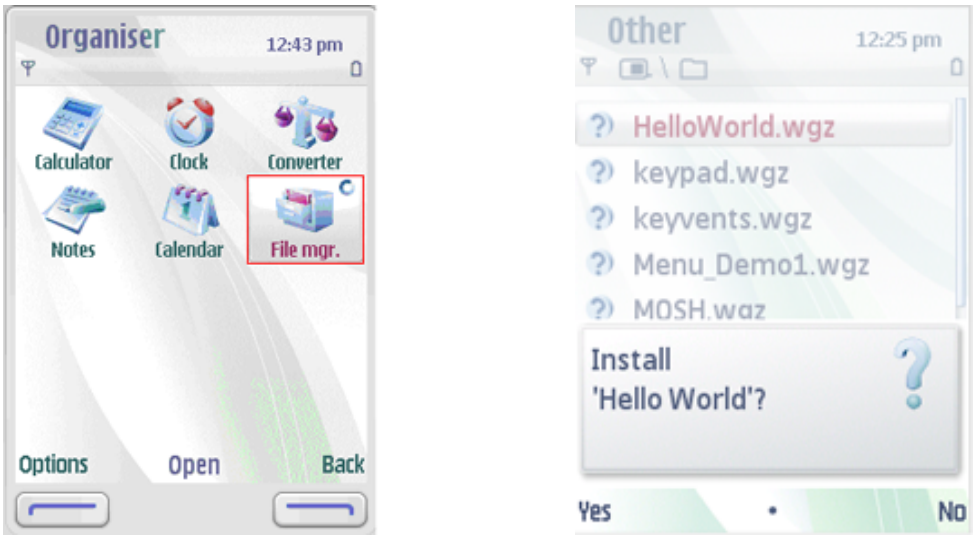
Caricare l'applicazione sul simulatore

Per caricare il file con l'applicativo sul simulatore è sufficiente copiare il file nella cartella: *<cartella installazione SDK>\epoc32\winscw\c\Data\Others*[3].

L'emulatore è avviabile seguendo il percorso: *S60 Developer Tools >[S60 versione piattaforma] >[versione SDK] >Emulatore* dal menù start di Windows. A seconda della potenza del calcolatore, può richiedere anche qualche minuto.

Una volta avviato entrare nel menù e aprire:

- Se si stà usando l'emulatore per S60 3rd Edition FP2 SDK, aprire Organiser.
- Se si stà usando l'emulatore per S60 5th Edition SDK, aprire Applications.



(a) Il file manager (b) Installazione di un'applicazione

Fig. 3.1: Sistema Operativo Symbian S60: File manager (a), e installazione di un'applicazione (b)

Avviare *File Manager* e entrare nella cartella *Others/Altro* di cui sopra. Aprire il file *<nome applicazione>.wgz* e avviare l'installazione.

Conclusa l'installazione, per avviare l'applicazione tornare al menù principale e:

- Se si stà usando l'emulatore per S60 3rd Edition FP2 SDK, aprire Installazioni.
- Se si stà usando l'emulatore per S60 5th Edition SDK, aprire Applicazioni.

Localizzare l'icona dell'applicazione e avviarla.

[3]La cartella *Others* è visibile nel filesystem dell'emulatore italiano con il nome Altro solo quando si ha già lanciato l'emulatore per la prima volta e chiuso.

3.2.1 Linee guida per lo sviluppo

Il sistema Symbian presenta un requisito esclusivo; la presenza di un file *Info.plist* nel quale si dichiara la pagina principale dalla quale partirá l'applicazione. Il file è già presente dello scheletro dell'applicazione una volta estratto il contenuto fornito da Phonegap.

Le API phonegap (Geolocazione, notifiche, informazioni sul dispositivo e altro) sono invocabili solo dalla pagina principale dichiarata nel file *Info.plist*. Uno script invocato da una pagina diversa da questa non riuscirá a, ad esempio, far vibrare il telefono.

Durante lo sviluppo per Symbian si è capito che è opportuno organizzare l'applicazione con le seguenti linee guida:

- Utilizzare una sola pagina HTML, prepararla in modo che funga da struttura sulla quale innestare molte "viste". É consigliato realizzare degli elementi $<div></div>$ per racchiudere le varie sottopagine.

- Gestire tramite Javascript la navigazione, mostrando/nascondendo opportunamente le sottopagine o elementi in esse contenuti, form, bottoni o quant'altro.

- Modificare dinamicamente la pagina mediante Javascript. Qualora ad esempio si debba riportare una scelta dell'utente, o presentare informazioni acquisite dal dispositivo (coordinate gps, informazioni sul telefono o browser), è necessario inserire del contenuto nella pagina html dinamicamente.

Nel corso dello sviluppo, al fine di semplificare il lavoro, si è verificato se alcune librerie o framework Javascript più usati su internet fossero compatibili e utilizzabili sul telefono vero e proprio. Questa prima fase ci ha portato a individuare in jQuery[4] il framework che faceva al caso nostro. Con le stesse regole con cui si assegnano delle proprietà in un CSS è possibile selezionare uno o più elementi all'interno della pagina e modificarne le proprietà. Oltre a ciò fornisce il supporto per semplici animazioni per mostrare o nascondere elementi.

jQuery è stato impiegato nello sviluppo dell'applicazione per gestire anche la navigazione, infatti tramite il plugin *jQTouch* è stato possibile tenere traccia dell'ordine in cui le sottopagine sono state visitate, per mantenere sempre consistente la funzionalità del bottone *Indietro*.

jQTouch non consiste solo in codice Javascript ma è un vero e proprio progetto che comprende un CSS pensato per la creazione di applicazioni per iPhone. Per avere un'applicazione iPhone-like è sufficiente adottare gli attributi corretti per i vari elementi della pagina e linkare il CSS nell'intestazione della pagina.

3.2.2 Testing

Il testing per questa piattaforma è stato svolto per metà sul simulatore inizialmente, poi si è passato a verificare il rendering su un cellulare vero e proprio; il Nokia .... Un altro

[4]jQuery è una veloce e concisa libreria Javascript che semplifica la navigazione nei documenti HTML, la gestione eventi, l'animazione, e l'interazione AJAX per lo sviluppo web.

strumento che il plugin per Eclipse ci rende disponibile è un emulatore del browser nokia con due possibili settaggi, WRT 1.0 e 1.1[5] . Si è rivelato utile inizialmente per verificare che il codice Javascript funzionasse correttamente all'interno della pagina, inoltre funziona bene per avere un'idea di come la pagina appaia sullo schermo di un cellulare.

Nel corso dello sviluppo ci si è resi conto che è più opportuno verificare la correttezza della pagina in un browser desktop e la visualizzazione sulla periferica stessa. Il simulatore fornisce comunque indicazioni quasi pari a quelle della periferica stessa. Come browser desktop è consigliabile utilizzare un browser basato su Webkit, lo stesso motore che effettua il rendering delle pagine su Safari. Google Chrome rappresenta il browser ideale in quanto permette di visualizzare il sorgente della pagina, verificare le proprietà CSS ereditate da ogni elemento, consultare la console Javascript. Inoltre è quasi completamente compatibile con lo standard HTML5, quindi rappresenta una buona base di test anche per le future applicazioni.

3.3 BlackBerry platform

A differenza della piattaforma Symbian il blackberry non ha problemi nel gestire le API Phonegap su più pagine html. Tuttavia proprio nell'idea di scrivere l'applicazione una volta sola si è cercato di modificare il codice il meno possibile.

L'approccio adottato è stato di cercare di semplificare l'applicazione già realizzata per Symbian rimuovendo eventuali comandi non supportati dalla nuova piattaforma, allineandosi ad un minimo comune denominatore. Mantenere la funzionalità su entrambe le piattaforme sviluppate finora era di fondamentale importanza per mantenere fede allo slogan "Write once, port everywhere ".

Al momento dell'analisi dell'utilizzabilità di Phonegap, il progetto era ancora in fase di sviluppo, pertanto molte delle funzioni già disponibili altrove erano in via implementazione. Gli sforzi del progetto erano rivolti alle piattaforme dove si concentravano la maggior parte dei programmatori che volevano testare le potenzialità di Phonegap, vale a dire iPhone e Android. Inoltre la piattaforma stessa, escludendo le versioni più recenti, non è mai stata orientata a fornire la miglior esperienza di navigazione su telefono cellulare; obiettivo che invece iPhone si è prefissata fin da subito.

Recentemente le aspettative del mercato nei confronti degli smartphone sono di fornire un'esperienza di navigazione all'altezza dei soldi spesi per l'acquisto, al di là delle caratteristiche peculiari del telefono stesso. Questo è perfettamente comprensibile se si considera che la connessione ad Internet, tramite Wireless o rete GSM diventa sempe più diffusa e accessibile a tutti.

Per colmare questo divario, RIM, *Research In Motion*, la compagnia che produce gli smartphone Blackberry[6], ha lanciato assieme a dei nuovi modelli, due nuove versioni del sistema operativo Blackberry: OS5 e OS6. Blackberry OS 5 è un aggiornamento disponibile per gran parte dei vecchi BB; rende disponibili la maggior parte delle funzionalità

[5]Web Run Time; denota la versione del motore del browser

[6]D'ora in poi anche BB.

dell'HTML5. Blackberry OS 6 invece è disponibile di default negli ultimi smartphone ed è pienamente compatibile con il nuovo standard. Per i sistemi operativi citati esiste la possibilità di sviluppare widget nella stessa maniera in cui si sviluppano le applicazioni Phonegap, solamente che le funzioni accessibili tramite Javascript rispondono a comandi relativi solo ad API BB.

Ora verranno illustrate le linee guida e i requisiti fondamentali per lo sviluppo sulla piattaforma Blackberry. Esistono due approcci diversi a seconda si realizzino applicazioni per le versioni del SO più recenti: OS5 e OS6; o per versioni 4.X . Dato che il passaggio al nuovo OS è stato fatto solo in tempi recenti, si è potuto solamente sperimentare lo sviluppo nella vecchia maniera. Tuttavia non c'è dubbio che i recenti miglioramenti rendano la creazione di widget molto più agevole [8].

3.3.1 Versioni 4.X

Come è chiaramente dichiarato dalla wiki di Phonegap il seguente tutorial, dedicato a utenti Windows, serve a realizzare applicazioni per versioni 4.6+ dell'OS Blackberry.

1. É necessario avere installato sul PC una versione uguale o più aggiornata della JDK 6 Update 20.

2. Scaricare e installare l'ultima versione dell'IDE di sviluppo Eclipse. Al momento la più recente è la 3.6.1

3. Scaricare e installare l'ultimo *Blackberry Plugin for Eclipse*. É stato recentemente aggiornato alla versione 1.3

    - Per questo passo e il successivo occorre un account per la Blackberry Developer Zone. É gratuito ma richiede una registrazione.

4. Installare i *JDE Component Packs*[7] desiderati:

    - Aprire Eclipse
    - Selezionare Help ¿ Install Software
    - Inserire in Location: `http://www.blackberry.com/go/eclipseUpdate/3.6/java`
    - Selezionare i JDE 4.7, 4.6.1 e altri qualora si desideri. Di default è selezionata la 5.0.
    - Completare l'installazione inserendo la login del Blackberry Developer Zone.
    - Riavviare Eclipse.

5. Installazione e configurazione di Apache ANT:

[7]JDE: Java Development Enviroment. Esiste una versione per ogni aggiornamento del sistema operativo, servono per creare i progetti correttamente a seconda del diverso SO.

- Estrarre nella cartella di installazione desiderata. Es. *C:\apache-ant*
- Aprire Pannello di Controllo >Sistema >Avanzate >Variabili d'ambiente. Creare la nuova *Variabile d'ambiente* ANT_ HOME e impostare il valore con la cartella dell'installazione.
- Alla variabile di sistema PATH, aggiungere alla fine il valore *;% ANT_ HOME% \bin*
- Verificare l'installazione di ANT. Dal prompt dei comandi digitare: *ant -v* . Si dovrebbe ottenere il seguente messaggio: *Apache Ant version 1.8.1 compiled on April 30 2010 Trying the default build file: build.xml Buildfile: build.xml does not exist! Build failed*

Creazione di un Progetto Phonegap

Una volta installato il software necessario per lo sviluppo, ciò che manca la creazione di un progetto su cui lavorare.

1. Creare un Blackberry Project:
   - In Eclipse selezionare File >New >Project >Blackberry Project.
   - Inserire un nome a piacere
   - Selezionare: *Use Specific JRE*, indicare una JRE 4.7 o più bassa.
2. Importare il progetto Phonegap-blackberry:
   - Scaricare Phonegap da `https://github.com/phonegap/phonegap-blackberry` ed estrarlo in *C:\Phonegap-Blackberry*.
   - Copiare e incollare le seguenti directory:
     - *C:\PhoneGap-BlackBerry\app\www =>C:\<Percorso Progetto Eclipse>\src\www*
     - *C:\PhoneGap-BlackBerry\framework\src\com =>C:\<Percorso Progetto Eclipse>\src\com*
     - *C:\PhoneGap-BlackBerry\framework\src\org =>C:\<Percorso Progetto Eclipse>\src\org*
   - Compilare il file phonegap.js
     - In *C:\PhoneGap-BlackBerry\framework\src\com* editare il file *common.properties*. Cambiare la *jde.home* in modo che punti a la Eclipse JDE Component Pack.
     - Copiare i Blackberry ant tools e Ant-Contrib JAR file che si trovano nella directory *\util* dello zip Phonegap estratto *=>ANT_ HOME\lib*
     - Dal prompt comandi usare *ant build-javascript*
     - Copiare il file *phonegap.js* generato in *C:\<Percorso Progetto Eclipse>\src\www*.
   - Selezionare il progetto in Eclipse e premere F5 o refresh. Dovrebbero apparire le cartelle \com, \org, \www; sotto la cartella \src.

3.3.2 Widget per OS5/OS6

Come abbiamo anticipato, l'uscita di queste due nuove versioni del sistema operativo per Blackberry semplifica molto lo sviluppo di Widget. L'aggiornamento dei browser embedded dei due sistemi operativi mette a disposizione anche le prime specifiche di HTML5, aprendo la porta della piattaforma BB anche a nuovi tipi di applicazioni web. Data la relativa novità dell'aggiornamento, non è stato possibile testare le nuove potenzialità offerte. Pertanto viene illustrato solamente la fase di installazione degli strumenti necessari allo sviluppo. Alcuni dei passi seguiti per l'installazione del software di sviluppo Blackberry sono gli stessi della sezione precedente.

1. É necessario avere installato sul PC una versione uguale o più aggiornata della Sun JDK.

2. Installare ANT e aggiungerlo alla PATH. Come il punto 5 della sezione 3.3.1.

3. Scaricare e installare la Blackberry Widget SDK[8] e installarla in *C:\BBWP*

4. Scaricare l'ultima copia di Phonegap e estrarla in *C:\Dev\Phonegap*

5. Creare un progetto nuovo:

   - Da prompt dei comandi, accedere alla cartella *BlackBerry\Widget*
   - Scrivere *ant create -Dproject.path=C:\Dev\phonegap\BlackBerry\Widget\sample* e premere invio.
   - Entrare nella directory *C:\Dev\phonegap\BlackBerry\Widget\sample*
   - Aprire il file project.properties con un editor e rimpiazzare la riga *bbwp.dir=* con *bbwp.dir=C:\\BBWP*

6. I principali comandi per interagire con il progetto creato sono:

   - Dal prompt dei comandi, mentre si è nella cartella del progetto, per compilare scrivere :*ant build*
   - Per caricare l'applicazione sul simulatore scrivere: *ant load-simulator* . L'applicazione è in downloads all'interno del menù del Blackberry.
   - Per compilare l'applicazione per un telefono scrivere: *ant load-device* . É necessario ottenere delle chiavi da RIM per firmare l'applicazione, previa registrazione[9]

[8] Scaricabile dal sito `http://us.blackberry.com/developers/browserdev/widgetsdk.jsp`

[9] Occorre compilare il form all'indirizzo `https://www.blackberry.com/SignedKeys/` e seguire le istruzioni.

3.4 Applicazione di prova

Per entrambe le piattaforme che abbiamo esaminato è stata realizzata un'applicazione per verificare se l'impiego di Phonegap potesse essere efficace in ambito aziendale. Lavorando nell'ambito dei trasporti l'azienda in cui si è svolto lo stage vorrebbe fornire la possibilità agli utenti di acquistare titoli di viaggio tramite telefono cellulare. Ciò che si è sviluppato è un'applicazione per la scelta di biglietti, l'inserimento di dati e l'invio di un SMS. L'operazione è finalizzata all'acquisto del biglietto stesso. Questo servizio è pensato per essere associato a una carta identificativa del viaggiatore. In questo modo è possibile, mediante una precedente registrazione via web, associare il numero di cellulare dell'utente ad un conto corrente. Tramite il cellulare associato si possono fornire dei servizi. Acquistare uno o più biglietti o rinnovare l'abbonamento sono le opzioni presenti nell'applicazione di prova realizzata con il framework Phonegap. Di seguito sono illustrate le caratteristiche principali delle applicazioni realizzate con questo approccio.

3.4.1 Symbian

Per quanto riguarda la piattaforma Symbian è stata realizzata un'applicazione prendendo vantaggio dall'uso di jQuery e del plugin jQTouch. Si illustrano i problemi fondamentali riscontrati e come sono stati affrontati:

Verifica della scadenza del widget L'elenco dei biglietti e il widget in generale devono avere una data di scadenza. Il comportamento dell'applicazione deve dipendere da questo. Non deve essere possibile utilizzare l'applicazione se tale data è passata, verrebbe mandato una richiesta di acquisto che sarebbe rifiutato. La data di scadenza dell'applicazione è memorizzata all'interno di un file *.js* contenente le funzioni principali. All'avvio viene effettuato un confronto con la data fornita dal browser tramite Javascript.

Realizzata versione multilingua (estendibile) Per poter garantire una maggiore accessibilità al pubblico è stata realizzata una versione multilingua dell'applicazione. Le lingue disponibili sono inglese e italiano, ma è rapidamente estensibile. Ogni stringa di testo all'interno della pagina è stata inserita all'interno di un elemento con uno specifico ID. Esiste un file *.js* per ogni lingua, dove sono associati gli ID alle stringhe della lingua corretta, secondo il paradigma chiave-valore. Al caricamento della pagina viene controllata la lingua del browser e caricato dinamicamente il file javascript corrispondente.

Controllo della scelta dell'utente Ad ogni sottopagina vengono mostrati all'utente bottoni diversi per proseguire la navigazione. La barra di navigazione in alto permette di capire in quale pagina ci si trovi e presenta sempre in alto a sinistra il bottone Indietro. La selezione del biglietto è garantita tramite una lista a selezione unica, una volta selezionata un'opzione è possibile proseguire. Si fa affidamento su jQuery per nascondere o mostrare gli elementi della pagina.

Verifica dei dati inseriti dall'utente All'utente viene richiesto di selezionare il biglietto e successivamente di inserire la quantità desiderata. Di default la quantità viene impostata a 1. Nel primo caso viene controllata la selezione tramite jQuery; nel secondo caso l'inserimento di stringhe di testo viene confrontato con un'espressione regolare. Viene permesso di proseguire alla pagina successiva solamente se il dato inserito è un intero con meno di due cifre.

Composizione e invio dell'SMS Per effettuare questa operazione Phonegap effettua un ponte tra le API javascript del browser nativo. In questo modo si possono usare le API Phonegap per la composizione e l'invio del messaggio. In questo modo, qualora l'invio dell'sms fosse disponibile su un altra piattaforma, non è necessario modificare l'applicazione. É altresì vero che per inviare un sms non è necessario utilizzare per forza le API Phonegap ma è possibile gestire il tutto tramite i comandi Javascript nativi.

La versione finale è stata testata su Nokia Express Music .... Il suo funzionamento è corretto in ogni suo aspetto, la navigazione funziona correttamente fino all'invio dell'sms, sia sulla periferica sia sul simulatore. É osservabile un certo ritardo nell'uso dell'applicazione dovuto al molto codice Javascript impiegato, soprattutto nella versione multilingua. Questo rallentamento è dovuto alla limitata capacità di calcolo della periferica e al browser stesso.

Fig. 3.2: Una schermata dell'applicazione realizzata per Symbian

3.4.2 Blackberry

L'applicazione per Blackberry non è stata ultimata e compilata definitivamente a causa delle molte difficoltà trovate nello sviluppo per il seguente dispositivo. Tuttavia sono state sperimentate varie soluzioni per la risoluzione dei problemi e come vedremo nel capitolo **??** lo sviluppo proseguirà in maniera nativa.

Ricodifica totale Javascript Grandissima parte di ciò che era stato usato per l'applicazione Symbian non è più utilizzabile. Parte del codice Javascript che funziona nell'applicazione browser del Blackberry non funziona nella Web View utilizzata. Le particolarità sono:

- Solo DOM livello 2[10]. Si tratta di specifiche riguardanti la struttura della pagina, la gestione degli eventi e altro.
- Solo CSS livello 2. Questo è stato il problema più grande in quanto prima per impostare l'oggetto di una funzione Javascript si poteva selezionare un elemento anche se non aveva ID. Con questa limitazione è possibile raggiungere un qualsiasi elemento della pagina solo se gli è stato impostato un ID o non esistono possibilità di ambiguità (molto difficile in una pagina html abbastanza complessa).

Pertanto, anche per questo motivo è stato impossibile utilizzare jQuery per la gestione della navigazione e degli elementi dinamici. Ciò che si è tentato di fare ma con scarso successo è di utilizzare un altro framework Javascript più semplice, XUI. Ricompilando il codice sorgente di questo framework è stata ottenuta una versione che prometteva di essere compatibile con la piattaforma Blackberry. É stata ricodificata la parte Javascript con comandi molto meno semplici ma compatibili ai requisiti.

Visualizzazione delle pagine Sono stati riscontrati dei problemi una volta che si è tentato di effettuare un porting dell'applicazione realizzata per Symbian. Esistono due modalità di rendering del browser:

Column View Imbroglia il browser facendo credere una risoluzione di 1024x768 e poi scala il tutto alla risoluzione nativa, 480x320 o 480x360. Viene racchiuso in questo modo ogni tipo di contenuto.

Page View A detta degli sviluppatori Phonegap è quella da impostare per avere la visuale corretta. É possibile inserire nell'intestazione della pagina HTML del codice per impostare automaticamente questa visuale.

Malgrado si siano seguiti i consigli e si siano effettuati i settaggi del caso non si è riusciti a visualizzare la pagina correttamente.

Sviluppo prettamente simulativo Non disponendo di una periferica fisica ci si è affidati al simulatore sviluppato da RIM. Non è decisamente lo strumento ideale con cui testare un'applicazione. A confronto con l'emulatore messo a disposizione da Symbian, è carente in quasi ogni aspetto. Malgrado la lentezza sia comprensibile e comune a entrambi questi strumenti, il simulatore Blackberry non è per nulla user-friendly. Pur essendo rivolto solamente a Windows il simulatore non riconosce i click del mouse e non esiste modo di personalizzare i comandi da tastiera in modo da

[10]Document Object Model: rappresenta il modo in cui sono rappresentati gli elementi di una pagina HTML. Esistono varie versioni per questo standard, la più recente è la versione 3.

gestire più agevolmente la simulazione. Lo stesso tasto destro del mouse svolge più funzioni contemporaneamente, apre il menù, torna indietro e effettua lo zoom della pagina. Ignorando tutto ciò; non esiste modo di sapere quali errori fossero riscontrati durante la simulazione, al contrario del simulatore per S60, dove gli errori, seppur sotto forma di codice, vengono quantomeno mostrati all'utente.

Capitolo 4

Caso Pratico: realizzazione di un'applicazione per ne-t

Le applicazioni, realizzate durante lo stage presso ne-t by Telerete Nordest, permettono l'acquisto di biglietti per il tram di Padova da telefono cellulare mediante l'invio di sms. In questo capitolo si effettua un'analisi dei requisiti e si valutano le alternative all'implementazione con Phonegap. Verranno esposti i motivi che hanno portato all'implementazione nativa dell'applicazione e si illustra la metodologia adottata per l'implementazione su sistema Symbian.

4.1 Pianificazione dell'applicazione

Il progetto dell'applicazione si basa su un progetto sommario già sviluppato all'interno dell'azienda. Il ruolo dell'applicazione è:

1. Mostrare all'utente le possibili opzioni per l'acquisto di titoli di viaggio.

2. Permettere la selezione di un titolo di viaggio.

3. Compilare un sms con la codifica dell'ordine effettuato.

4. Finalizzare l'acquisto con l'invio dell'sms al numero del servizio.

La scelta dell'utente è guidata secondo l'idea di figura Figura 4.1.

Nel progetto originale, oltre al semplice acquisto di titoli di viaggio singoli, è possibile rinnovare o acquistare abbonamenti per il viaggio.

Per pianificare un'applicazione è necessario avere in mente quale sarà la piattaforma che la ospiterà. Nel nostro caso non c'è un unico tipo di sistema mobile che ospiterà l'applicazione. Pertanto l'obiettivo è quello di rendere disponibile l'applicazione al maggior numero di utenti possibili, visto che i cellulari degli utenti del trasporto pubblico sono di massima eterogeneità.

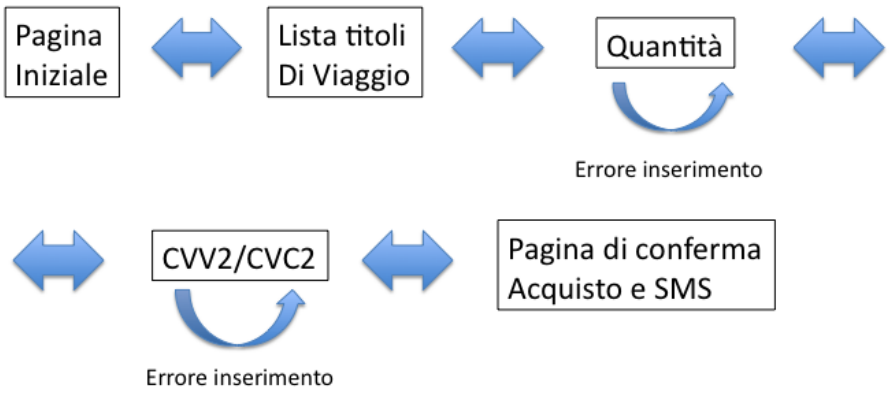


Fig. 4.1: Disegno informale per la navigazione all'interno della applicazione.

Phonegap consente di utilizzare HTML, Javascript e CSS. L'applicazione doveva limitarsi a utilizzare le funzionalità date da questi linguaggi.

Altra osservazione di cui tenere conto è il fatto che non tutte le piattaforme supportate da Phonegap avevano le funzioni necessarie per l'applicazione implementate. Ad esempio la funzione SMS non è disponibile per la piattaforma iPhone[1]

Ulteriore fattore importante da considerare è che tutte le piattaforme target sono limitate dalle capacità del loro browser. Ad esempio le versioni meno recenti di Blackberry (OS <5.0) sono limitate dal browser per quanto detto nella sezione 3.4.2.

4.2 Acquisto di titoli di viaggio con Movincom

Movincom è il consorzio di esercenti attivi sul canale mobile, nato con l'obiettivo di contribuire allo sviluppo sistemico ed organico del mobile business. L'obiettivo di Movincom è quello di permettere agli esercenti consorziati di ricevere dai propri clienti, abilitati al servizio, disposizioni di pagamento di beni e servizi semplicemente attraverso il cellulare. Il numero di telefono rappresenta infatti una chiave univoca ed il cliente può effettuare un acquisto in mobilità senza mai inserire i dati sensibili del proprio strumento di pagamento. Ciò è garantito dalla registrazione a priori del telefono cellulare del cliente, e all'associazione del numero di cellulare alla modalità di pagamento supportata da Movincom.

La Figura 4.2, illustra come il widget funziona da terminale dal quale inizia e finisce la transazione, sia con esito positivo, sia con esito negativo. Tramite il widget l'utente abilitato al servizio effettua la sua scelta e viene compilato l'sms che verrà inviato al numero di servizio. La richiesta di acquisto arriva al servizio sms, viene processata e inoltrata alla Movinbox, il servizio Movincom che si occupa di gestire la transazione con la banca di riferimento associata al numero di cellulare. Una volta approvata o rifiutata la transazione bancaria, il percorso viene fatto al contrario, all'utente viene mandato un messaggio con una ricevuta d'acquisto oppure un esito negativo.

[1] Vedi la figura 2.7.

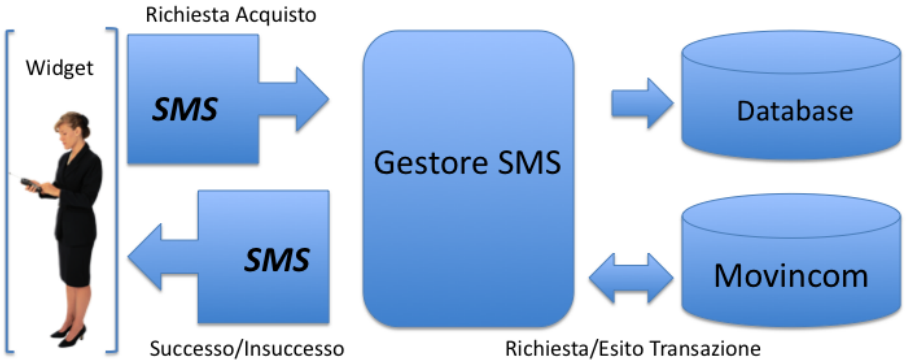


Figura 4.2: Il widget come punto di inizio e fine della transazione.

4.2.1 Presidi di sicurezza e implementazione nativa

Per ogni acquisto effettuato sul web il venditore deve garantire un certo livello di sicurezza della transazione, specialmente se vengono trasmessi dati sensibili relativi a coordinate finanziarie dell'utente [9]. Oltre a presidi di sicurezza imposti dalla legislazione sul trattamento dei dati sensibili le opzioni per garantire la sicurezza sono molte e alcune sono a discrezione del rivenditore. Il commerciante in questo senso è invogliato a garantire la maggior sicurezza possibile per l'utente in modo da renderlo più fiducioso, protetto nell'acquisto, e allo stesso tempo proteggere sé stesso da un eventuale frode. Chiaramente questo sistema è ancora più accentuato se si parla di piattaforme mobili. Difficilmente un utente sarà invogliato a installare un'applicazione se il suo comportamento non è affidabile e garantito da terze parti.

Durante lo sviluppo dell'applicazione ci si è resi conto che l'applicazione realizzata con l'approccio Phonegap non rispondeva alle esigenze di sicurezza richieste dal circuito di pagamento Movincom.

I CVC2/CVV2 e la sicurezza nelle transazioni

Esistono due tipi di codice per le carte di credito o prepagate:

CVC1 o CVV1 è memorizzato all'interno della striscia magnetica della carta. Il suo scopo è quello di assicurare che i dati memorizzati siano validi e siano stati generati dalla banca. L'intero valore viene usato come parte della transazione con la banca.

CVC2 o CVV2 Viene chiesto dai commercianti per evitare di avere transazioni a carta non presente e quindi a rischio di frode. Questo codice non è memorizzato con il primo ma è stampato sulla parte che ospita la firma del proprietario.

Il secondo codice ha molti nomi a seconda del tipo di carta su cui è presente:

CVC2 Card Validation Code - Mastercard

CVV2 Card Verification Value - Visa

CID Card Identification Number - Discover, American Express

Per ragioni di sicurezza, è vietato memorizzare i dati sensibili di una transazione, nel nostro caso il CVV, in nessun passaggio della stessa. In questo modo, se il database di un qualsiasi servizio intermedio venisse compromesso, non si potrebbe utilizzare la carta. Questo vincolo di sicurezza tuttavia non si applica solo alla parte interna che si occupa della transazione, ma vale anche ai capi estremi. Ciò implica che nemmeno sul telefono cellulare deve venire memorizzato alcun dato sensibile, nel nostro caso, il CVV. La memorizzazione può aver luogo se il messaggio mandato viene memorizzato nella cartella *OUTBOX* propria dei messaggi in inviati.

Con l'approccio Phonegap questo era inevitabile, la composizione del messaggio era vincolata all'interfacciamento messo a disposizione del codice Javascript della pagina. Chiaramente questo approccio passa attraverso il sistema operativo stesso del cellulare. Ciò implica che l'sms viene gestito come ogni altro, cioè memorizzato una volta inviato. Per questo motivo l'approccio Phonegap è stato abbandonato per la realizzazione di questa applicazione e si è passato allo studio di una implementazione nativa come vedremo al capitolo 5.

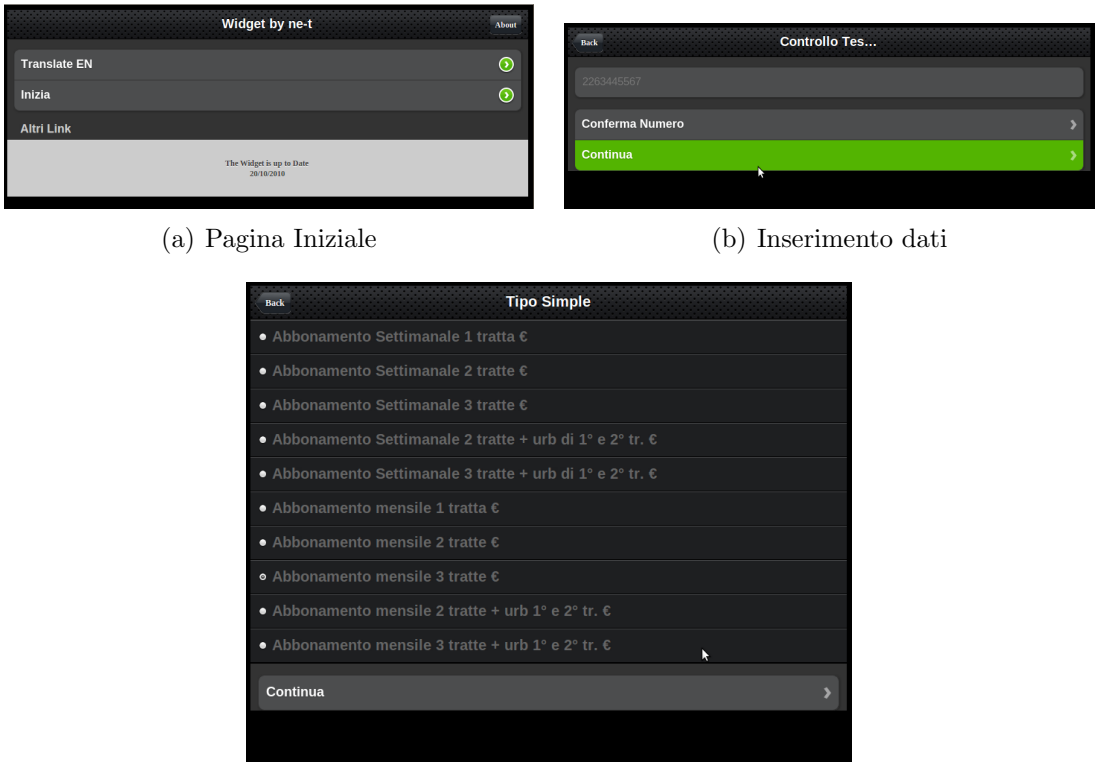
4.3 Versione beta

Le figure seguenti mostrano come appare l'applicazione demo realizzata con Phonegap per Symbian. Sono state seguite tutte le indicazioni della sezione 3.2. Le sottoimmagini di Figura 4.3 mostrano l'applicazione in un browser con motore WebKit, Google Chrome. É stato usato Chrome in quanto è l'unico browser desktop assieme a Safari basato sul motore WebKit. In questo modo si possono meglio visualizzare le particolarità che il CSS di jQtouch impiegato utilizza. Un esempio è il bordo arrotondato di alcuni dei bottoni di inizio e fine delle liste o di conferma. Altra particolarità del CSS è che contorna in automatico il bottone *Indietro/Back* di grigio senza l'uso di immagini in background.

L'applicazione, completa di per sé, controlla l'aggiornamento del widget, consente la selezione del titolo di viaggio, permette l'inserimento del codice della carta, manda l'sms dopo aver ricapitolato la selezione.

Come è stato spiegato poco sopra, questo approccio effettua l'invio dell'sms tramite il sistema operativo. In questo modo l'sms era trattato come se fosse mandato dall'utente e quindi salvato nella cartella inviati. Facendo ciò viene memorizzato il codice di controllo per la validità della carta di credito (o prepagata), inaccettabile per garantire la sicurezza degli utenti del servizio.

La soluzione a queste problematiche è stata trovata nello sviluppo dell'applicazione tramite JavaME. Vedremo nel prossimo capitolo le strategie adottate.



(a) Pagina Iniziale

(b) Inserimento dati

(c) Scelta del titolo di viaggio

Fig. 4.3: Applicazione demo per Symbian realizzata con Phonegap; Pagina iniziale (a), scelta titolo di viaggio (c), inserimento dati (numero tessera) (b)

4.4 Il futuro di Phonegap

Il futuro per phonegap sembra promettente, il progetto prosegue nell'inseguire ciò che accomuna tutte le piattaforme target. Le mosse più interessanti vengono dalle piattaforme stesse. Con l'avvento di HTML5 i browser degli smartphone si adattano alle nuove specifiche, garantendo delle ulteriori funzionalità al browser in modo nativo. Come si è visto nella figura 2.6 le nuove API Phonegap rispecchiano questo cambiamento.

Oltre a implementare gli aggiornamenti ciò che le piattaforme cercano di fare è di garantire agli sviluppatori di applicazioni un modo per sviluppare Wiget. In questo caso per Widget si intende una web application che permette allo sviluppatore, attraverso la pagina web, di interagire con la stessa e con le funzionalità del telefono. A seconda della piattaforma e delle API Javascript fornite al browser, un widget può interagire con più o meno funzionalità del telefono. In questo modo risulta possibile ottenere informazioni dal telefono, ad esempio *Contatti, Messaggistica, Geolocalizzazione* mediante alcuni semplici comandi javascript.

Anche il resto delle aziende protagoniste del web e del mobile non stanno a guardare.

L'uscita di *Jquery mobile* rappresenta un'ulteriore spinta a uniformare lo sviluppo per tutte le piattaforme. Questo framework, tramite la collaborazione con le aziende di telefoni

cellulari stesse, promette infatti di rendersi compatibile a con tutte le piattaforme mobile in circolazione. Racchiude temi per sviluppare applicazioni web di aspetto nativo, supporto Javascript per animazioni e elementi dinamici. Il progetto è ancora in alpha, ha alcuni problemi di velocità su alcune piattaforme, ma c'è grande attesa nel vedere come i futuri aggionamenti possano migliorare il prodotto.

Una riflessione a questo punto nasce spontanea: perché continuare a seguire il progetto se le piattaforme stesse cercano di arrivare a fornire i medesimi strumenti per lo sviluppo di Widget? La risposta è nel motto stesso del progetto Phonegap: "Write once, port everywhere ". Vale a dire che sebbene scrivere il codice di un Widget per ciascuna delle varie piattaforme non sia particolarmente difficile, è ancora più semplice scrivere solamente una volta il codice, utilizzando un solo tipo di API, quelle Phonegap. La stesura di una pagina web e del relativo codice Javascript per la gestione di animazioni, form, calcoli, navigazione ecc, viene fatto una volta sola, spostando la complessità verso l'installazione e la messa in funzione dell'ambiente di compilazione per ciascuna piattaforma.

L'avvento di HTML5 non può che rafforzare un approccio di questo tipo. Quando tutti i browser mobile si allineeranno al nuovo standard sarà ancora più facile per gli sviluppatori web avvicinarsi alle piattaforme mobili. Gli sviluppatori che decideranno di utilizzare Phonegap ritroveranno ulteriori funzionalità a loro disposizione man mano che il progetto cresce.

Nitobi stessa sfrutta la situazione e mantiene i riflettori accesi sul progetto annunciando di proseguire con l'implementazione delle future specifiche di HTML5 nelle versioni successive. Nelle future release si pianifica di fornire supporto a plugin che forniscono altre funzionalità al browser nel gestire particolari esigenze (Childbrowser per GoogleMaps ad esempio), supporto per l'internazionalizzazione, gestire la raccolta di informazioni sul sistema e altro.

Capitolo 5

Sviluppo delle applicazioni native: Symbian

Dopo aver riscontrato i problemi sopra descritti nello sviluppo con phonegap si è deciso di superare i problemi sviluppando applicazioni ad hoc in maniera nativa. Per adattarsi meglio al mercato italiano si è dedicati inizialmente allo sviluppo dell'applicazione per piattaforme Nokia, brand che detiene gran parte dell'utenza del bel paese [10]. Il sistema operativo preso in considerazione è Symbian, nello specifico Serie 40 (d'ora in poi S40) e Serie 60 (S60).

5.1 Requisiti fondamentali

La creazione dell'applicazione deve tenere conto di alcuni requisiti fondamentali, cercando di soddisfarli nella maniera migliore possibile.

- è necessario creare un'applicazione in tempi rapidi, quindi ridurre al minimo la curva di apprendimento del linguaggio per dedicare gli sforzi allo sviluppo dell'applicazione.
- è necessario creare un'applicazione il più possibile uniforme, rivolta alla maggior parte delle piattaforme (Nokia) possibili, in modo da ridurre la complessità nella distribuzione della stessa. La base di utenza infatti deve essere la più larga possibile in quanto il sevizio a cui è associata è pubblico.
- è necessario che l'applicazione sia di facile utilizzo e non rappresenti un ostacolo al servizio stesso. Il successo del servizio dipende anche dalla sua immediatezza e accessibilità.

5.1.1 Alternative

Le alternative che si presentano allo sviluppo di applicazioni sono innanzitutto di linguaggio. Di seguito si riportano le varie possibilità e una descrizione sommaria dei requisiti implicati dalla scelta stessa.

C++ Permette la gestione di ogni aspetto dell'applicazione nei minimi dettagli. Aspetto grafico definibile arbitrariamente; richiede lo studio e la realizzazione di un design specifico per ogni elemento. Sviluppo poco semplice anche pensando alla scarsa portabilità su più versioni dello stesso sistema operativo.

Java Soluzione che permette un controllo accurato su ogni elemento dell'applicazione. Portabilità molto elevata, lo stesso codice è utilizzabile su piattaforme molto diverse grazie alle caratteristiche del linguaggio stesso. Aspetto grafico legato all'implementazione sulla particolare piattaforma. Il linguaggio Java consente di rappresentare oggetti grafici che vengono rappresentati in maniera diversa da telefono a telefono. Oggetti che comunque vengono rappresentati in maniera nativa e che quindi sono in tutto e per tutto elementi che non rappresentano un ostacolo per chi sa usare già il terminale.

WebProject Approccio in tutto e per tutto identico all'approccio di Phonegap, fornisce lo stesso controllo sull'applicazione tramite primitive Javascript. Non risolve i problemi rilevati al capitolo precedente.

5.1.2 Scelta adottata - Linguaggio e Librerie:

Si è scelto di proseguire con lo sviluppo mediante le librerie Java Micro Edition. La scelta inoltre è stata di utilizzare solamente le API CLDC 1.0. L'acronimo che sta per Connected Limited Device Configuration; è la libreria di base dato che non sono necessarie altre classi oltre a quelle per mandare l'sms. Utilizzando solamente queste librerie infatti, si riducono al minimo le richieste a livello hardware, alzando notevolmente il livello di portabilità dell'applicazione. La strategia adottata considera anche che le successive piattaforme di sviluppo, nello specifico Android e Blackberry. Entrambi questi sistemi operativi si basano esclusivamente su Java.

5.2 Design dell'applicazione

Il processo che ha portato al design finale dell'applicazione è stato influenzato da come è possibile realizzare l'interfaccia utente. A seconda del tipo di utenza finale che si vuole servire è possibile scegliere diversi approcci per la realizzazione dell'interfaccia utente (d'ora in poi anche UI). Si illustra un elenco dettagliato dei possibili approcci utilizzabili per realizzare l'interazione con l'utente:

LCDUI

eSWT

Altre API grafiche, che includono:

- Scalable 2D vector graphics API (M2G)

- Mobile 3D graphics API (M3G)
- Nokia UI API

5.2.1 LCDUI

Limited Connected Device User Interface (LCDUI) è la base di ogni interfaccia utente grafica creata in Java ME. Offre un ristretto e efficiente approccio allo sviluppo della UI. Come alternativa a LCDUI, eSWT è anche disponibile da S60 3za Edizione FP2 in avanti.

LCDUI ha una semplice approccio basato su schermate, dove un singolo *Displayable* è sempre attivo in ogni momento sullo schermo del telefono. Questo *Displayable* può contenere elementi predefiniti o essere manipolato in modo più particolareggiato.

L'implementazione LCDUI usa i componenti messi a disposizione dalla piattaforma Symbian. I MIDlet (applicazione per dispositivi mobili) che usano l'API LCDUI su Symbian, prendono le sembianze e il comportamento di applicazioni native. Le implementazioni LCDUI supportano anche i temi, il che significa che un tema selezionato dall'utente influenza anche l'aspetto dell'applicazione.

Anche nell'uso delle API LCDUI, esiste la possibilità di scegliere le facili, veloci e portabili API a più alto livello, e le più personalizzabili API a basso livello. Scegliere le API a basso livello significa, come per la scelta del linguaggio C++, svolgere un minuzioso design per l'aspetto dell'applicazione.

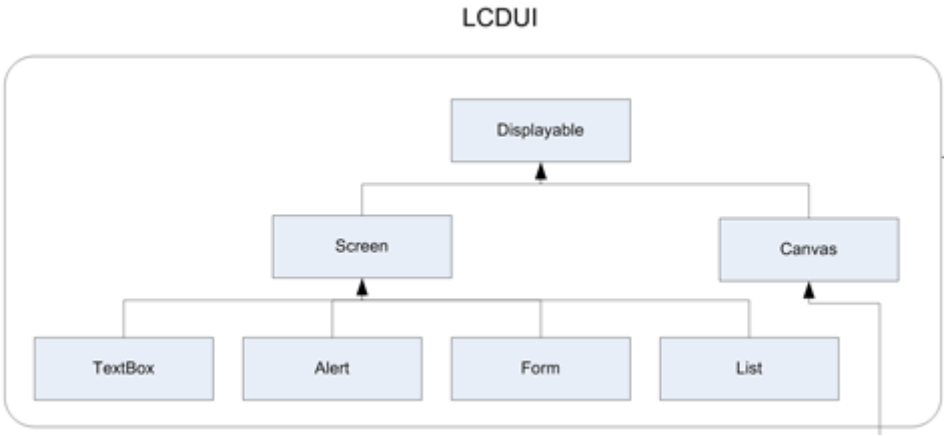


Fig. 5.1: Schema dell'architettura di LCDUI.

5.2.2 eSWT

Su periferiche Nokia l'Embedded Standard Widget Toolkit è stata implementata dalla 3rd Edition FP2 in avanti come strumento alternativo alla interfaccia Utente LCDUI.

eSWT è parte del progetto eRCP nella comunità open source ed è uno stretto sottoinsieme delle API Standard Widget Toolkit (SWT). Lo standard SWT è stato sviluppato

in connessione con il progetto Eclipse ed è stato, per esempio, usato per costruire l'IDE Eclipse. La stessa filosofia di design e usabilità è stata usata per SWT.

In un ambiente desktop, SWT è un'alternativa ad AWT e gli Swing Java GUI toolkits. Consiste in elementi di interfaccia, chiamati widgets, e metodi per crearli e manipolarli. Per visualizzare il widget, l'implementazione SWT accede alle librerie native del sistema operativo. Le API eSWT seguono una simile linea guida, applicata all'ambiente mobile. SWT ed eSWT condividono le stesse API, rendendo le applicazioni realizzate per il mobile, portabili anche su ambiente desktop.

Le API eSWT sono implementate sopra CLDC, che combinate con MIDP, sono la più comune configurazione JAVA ME. Possono essere usati gli stessi strumenti di sviluppo usati per Java.

Il cuore di eSWT consiste in widget di base che si possono combinare per creare interfacce utente come editor di testo, visualizzatori di immagini e altro. Questi widget di base includono elementi come bottoni, etichette e liste. Si possono organizzare widget sullo schermo utilizzando posizioni relative o assolute.

5.2.3 Altre API grafiche

Nokia UI API La Nokia UI API è un'estensione al MIDP 1.0. Fornisce funzionalità addizionali nella forma di capacità grafiche e audio, specialmente per sviluppatori di giochi. Parti della Nokia UI API sono state rimpiazzate da MIDP 2.0 e le librerie che lo estendono, pertanto si consiglia l'uso di queste librerie qualora possibile.

Scalable 2D vector graphics API (M2G) o JSR-226 É un'API disegnata per essere compatta ma efficace, e ha come target piattaforme di basso livello con vincoli di memoria, dimensione dello schermo e potenza computazionale. É usata per interfacciare il motore SVG[1] nativo per renderizzare immagini vettoriali scalabili 2D.

Mobile 3D graphics API (M3G) o JSR 184 É un pacchetto opzionale di JavaME capace di effettuare rendering 3D; tiene conto delle limitazione di alcune piattaforme che implementano J2ME. M3G offre un rendering che è veloce abbastanza per applicazioni interattive come giochi e animazioni. M3G prende anche vantaggio dell'hardware come unità floating point o acceleratori 3D trovati in piattaforme di alto livello.

5.2.4 Scelta adottata - Interfaccia Utente

Dato che la priorità del progetto è stata quella di realizzare un'applicazione funzionante in tempi brevi, è stato scelto l'approccio più semplice e che richiede il minor tempo di apprendimento. LCDUI garantisce il funzionamento delle funzionalità base: creazione di

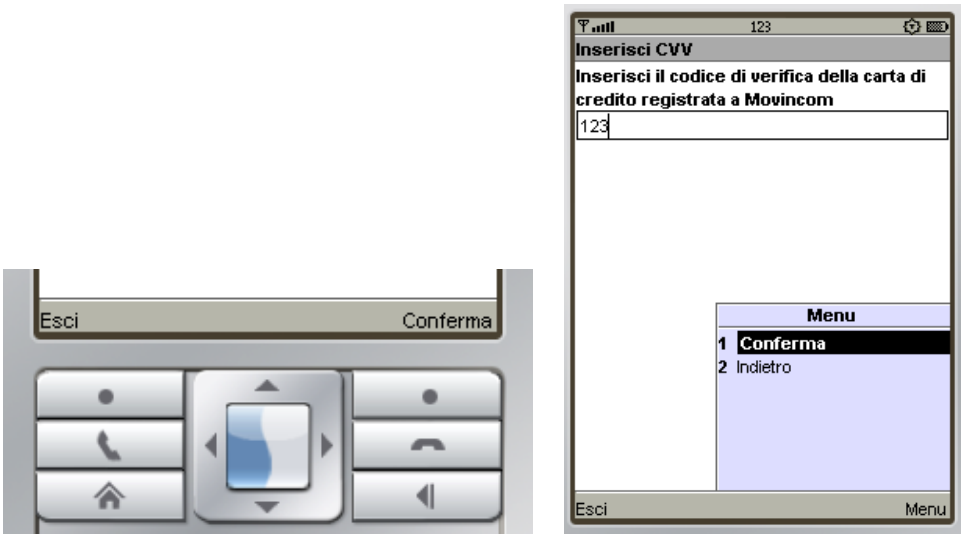
[1]Scalable Vector Graphics (SVG) è un linguaggio definito dal W3C per descrivere grafica 2D in formato XML. Supporta contenuto grafico ricco mediante tre tipi di oggetti: forme vettoriali, immagini raster e testo.

liste, inserimento di testo, stampa a video delle istruzioni per l'utente, gestione della navigazione tramite schermate. Inoltre dato il linguaggio ad alto livello utilizzato da LCDUI viene garantita la massima portabilità dato che eSWT è disponibile solo da S60 3rd Edition FP 2 in avanti. La scelta che è stata presa rispetta i requisiti citati alla sezione 5.1, l'applicazione realizzata è la stessa per tutti i cellulari (tutte le versioni di S40 e S60). La facilità di utilizzo è garantita dal fatto che l'applicazione prende le sembianze native del telefono che la ospita; eredita i temi già presenti sul telefono, utilizza lo stesso stile per bottoni e liste delle funzionalità del sistema operativo stesso. In questo modo l'uso dell'applicazione non diventa più difficile dell'usare il telefono stesso.

5.3 Interazione con l'utente

La gestione della navigazione avviene mediante schermate che vengono presentate all'utente in ogni momento. Gli eventi vengono monitorati mediante la classe listener che resta in attesa delle azioni dell'utente. E' possibile assegnare dei comandi ai due pulsanti standard che tutti i Nokia possiedono.

Di default se il comando Exit viene creato per la pagina esso viene assegnato ad un singolo bottone; sinistro o destro a seconda del telefono, diventando l'unico comando disponibile per quel tasto. Se vengono creati altri comandi, personalizzati o sul modello di quelli di default, è possibile dare un ordine secondo il quale essi compaiono nel menù a comparsa.



(a) Pulsanti di default (b) Apertura del menù

Fig. 5.2: Comandi di default Nokia destra e sinistra. (a). Il menù che mostra i comandi disponibili per la pagina. (b)

Questo metodo è il modo più semplice per realizzare l'interazione, per gestire la navigazione viene richiesta la conferma dei dati pagina dopo pagina. Viene fornito un comando

indietro nel caso si voglia reinserire i dati. Nel caso si siano confermati i dati e si sia arrivati all'ultima pagina non è più possibile tornare indietro.

5.3.1 Selezione dei biglietti

Nella prima pagina viene mostrata la lista dei biglietti acquistabili, qualora non fosse chiaro l'utilizzo dell'applicazione è possibile accedere al menù e ottenere più informazioni aprendo la pagina più info.

La pagina più info contiene informazioni sui biglietti e su come avanzare nella immissione dei dati. Una volta scelto il biglietto la visualizzazione passa alla pagina di inserimento quantità. Qualora il biglietto selezionato fosse il Biglietto Veloce la quantità viene impostata a 1 e la pagina quantità viene saltata. Una volta immessa la quantità si passa all'inserimento Cvv, il codice di sicurezza presente sul retro delle carte di credito. Esso viene visualizzato a schermo sotto forma di asterischi come una qualsiasi password, in modo da evitare che qualcuno lo legga durante la digitazione. Una volta inserito il Cvv e confermato viene inizializzato il costruttore che permette la connessione al sistema GSM del telefono, e vengono compilati i campi destinatario e testo del messaggio. Nel frattempo l'utente visualizza la pagina di riassunto dove può rileggere i dati e il tipo di biglietto che ha inserito, in modo da confermare ciò che ha scelto. Raggiunta la pagina di riassunto non è possibile tornare indietro, per riselezionare il biglietto è necessario chiudere e riaprire l'app. Una volta confermato il viene spedito il messaggio. Eventuali errori vengono visualizzati nella pagina di riassunto, con un chiaro messaggio che identifica le cause nel fallimento dell'invio del messaggio.

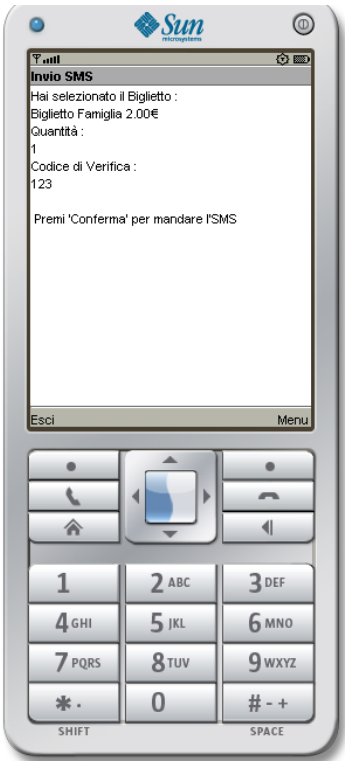


Fig. 5.3: La pagina di riassunto e invio dell'SMS

Nel caso di eventuali errori è possibile ritentare l'invio del messaggio dalla stessa pagina. Es. Il telefono ha riscontrato una bassa copertura GSM per poco tempo, impedendo la connessione.

Capitolo 6

Sviluppo delle applicazioni native: Android

Android è un sistema operativo open source per dispositivi mobili, basato sul kernel Linux. Fu inizialmente sviluppato da Android Inc., startup acquisita nel 2005 da Google. Il 5 novembre 2007 l'Open Handset Alliance presentò Android, costruito sulla versione 2.6 del Kernel Linux.

La piattaforma è basata sul kernel Linux, usa il database SQLite, la libreria dedicata SGL per la grafica bidimensionale e supporta lo standard OpenGL ES 2.0 per la grafica tridimensionale. Le applicazioni vengono eseguite tramite la Dalvik virtual machine, una Java virtual machine adattata per l'uso su dispositivi mobili. Android è fornito di una serie di applicazioni preinstallate: un browser, basato su WebKit, una rubrica e un calendario.

Il 31 gennaio 2011 viene ufficialmente dichiarato che Android è il sistema operativo di smartphone e tablet più diffuso nel mondo: infatti nell'ultimo trimestre del 2010 Android è riuscito a superare Symbian, l'incontrastato sistema operativo di Nokia per oltre 10 anni, vendendo nel mondo ben 32,9 milioni di smartphone contro i 30,6 milioni di Symbian. Dal 2008 Android è cresciuto, anno su anno, del 615.1% [1].

6.1 Piattaforma Android

La piattaforma Android assieme a quella Apple rappresenta quella dalle più ampie possibilità e potenzialità per uno sviluppatore di applicazioni. A favore di Android c'è la relativa novità della piattaforma stessa, associata alle molte possibilità offerte dagli smartphone che ospitano questo sistema operativo. I dispositivi su cui Android è utilizzato sono specialmente smartphone. Il sistema operativo è nato per essere la combinazione perfetta con i cellulari HTC [2] anche se sono stati creati cellulari da diverse aziende, tra le quali Samsung

[1] Fonte: Canalys Estimates ©Canalys 2011

[2] HTC: High Tech Computer Corporation. Fondata a Taiwan nel 1997, inizialmente basava i propri prodotti su Windows CE ma nel 2009 ha spostato la propria attenzione a dispositivi basati su sistema operativo Android.

Worldwide smart phone market
Market shares Q4 2010, Q4 2009

OS vendor	Q4 2010 shipments (millions)	% share	Q4 2009 shipments (millions)	% share	Growth Q4'10/Q4'09
Total	101.2	100.0%	53.7	100.0%	88.6%
Google*	33.3	32.9%	4.7	8.7%	615.1%
Nokia	31.0	30.6%	23.9	44.4%	30.0%
Apple	16.2	16.0%	8.7	16.3%	85.9%
RIM	14.6	14.4%	10.7	20.0%	36.0%
Microsoft	3.1	3.1%	3.9	7.2%	-20.3%
Others	3.0	2.9%	1.8	3.4%	64.8%

*Note: The Google numbers in this table relate to Android, as well as the OMS and Tapas platform variants
Source: Canalys estimates, © Canalys 2011

Fig. 6.1: Dati di vendita e crescita di Android.

e Motorola. Nel corso del 2010 sono stati presentati al mercato dei tablet vari modelli, il più degno di nota è il Samsung Galaxy Tab. La versione più recente per smartphone è la 2.3, anche detta Gingerbread. La versione 3.0 invece viene utilizzata sui tablet. In questo capitolo analizzeremo come vengono sviluppate applicazioni native.

6.2 Installazioni preliminari

Per iniziare lo sviluppo occorre installare sul sistema la SDK, ora giunta alla versione 9. La cosa molto utile che si riscontra installando la SDK è che non si tratta di un unico componente monolitico ma l'installer lascia decidere all'utente cosa vuole installare a seconda delle versioni di Android su cui voglia fare il testing. In questo modo è molto più semplice gestire l'installazione aprendo l'eseguibile e lasciare che gestisca lui il processo piuttosto che cercare il link esatto sul sito per gli sviluppatori.

Ciò che è stato fatto per l'applicazione realizzata è installare la versione più recente e quella più vecchia del sistema operativo. In questo modo, se l'applicazione non utilizza API introdotte solo di recente, lo sviluppo può essere condotto per la versione più vecchia di Android, in modo che il risultato finale sia portabile su tutte le versioni di Android.

Le versioni del sistema operativo di Google sono numerate in ordine crescente: la versione 2 corrisponde a Android 1.6; la 4 ad Android 2.0; e così via. In questa schermata è anche utile selezionare il driver per effettuare il debugging direttamente su cellulari collegati tramite usb. Una volta scelte le versioni dell' SDK da installare, per effettuare le simulazioni delle applicazioni realizzate è possibile creare una macchina virtuale sempre tramite l'installer dell'SDK. É possibile creare un simulatore per ogni versione di Android

installata. Tuttavia è necessario definire un simulatore di default che verrà avviato da Eclipse alla compilazione.

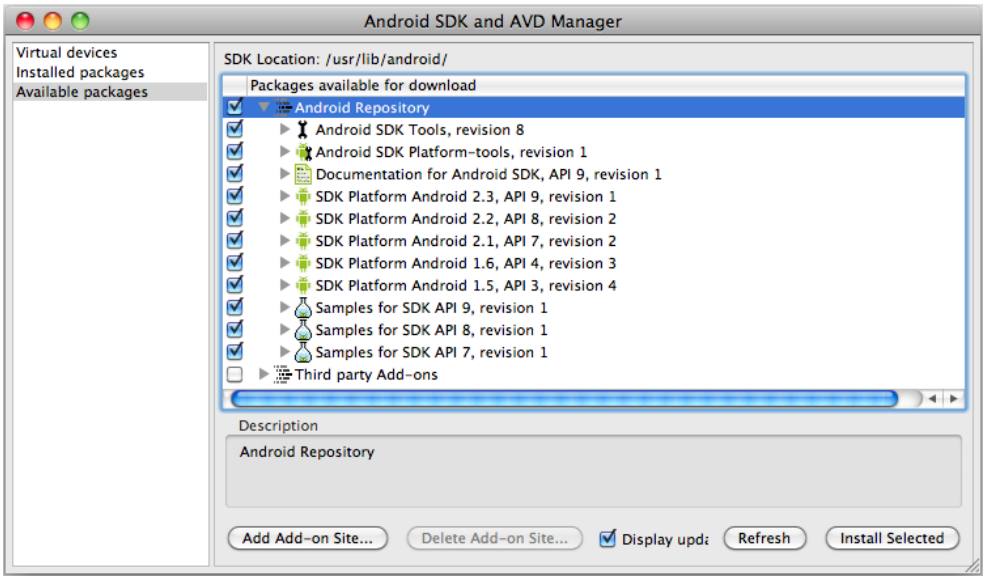


Fig. 6.2: Il menù dell'SDK Manager.

A questo punto occorre munirsi di un IDE di sviluppo, è fortemente consigliato usare Eclipse. Al momento della scrittura della tesi è stato usato Eclipse 3.6.1. La pagina relativa sul sito di riferimento[3] fornisce istruzioni per l'installazione dell'ADT Plugin (Android Development Toolkit) per Eclipse 3.5 Galileo; 3.6 Helios; 3.4 Ganymede. Con questi strumenti è possibile creare un progetto base sul quale poi lavorare e realizzare l'applicazione finale.

Riassumendo, i passi fondamentali da seguire sono veramente pochi:

1. Scaricare l'SDK e AVD manager
2. Scegliere i pacchetti da installare; accettare le licenze e installare
3. Configurare l'ADT plugin per Eclipse: Eclipse ->Preferenze ->su *Browse* inserire la locazione dell'SDK. Riconoscerà automaticamente le versioni installate.
4. Creare una Virtual Device tramite l'SDK manager. A meno di esigenze particolari è sufficiente personalizzare solo il nome.

6.3 Concetti fondamentali

Esistono alcuni concetti fondamentali che è bene sapere durante la realizzazione di applicazioni in Java per Android [11]. Innanzitutto alcuni concetti sul sistema operativo. Di

[3] `http://developer.android.com/sdk/eclipse-adt.html`

fatto la conoscenza degli stessi non è vitale per la realizzazione di semplici applicazioni ma qualora si aumenti la complessità essi possono risultare molto utili.

- Di default, ogni applicazione fa partire il proprio processo Linux. Android fà partire il processo quando un qualsiasi codice dell'applicazione deve essere eseguito, e chiude il processo quando non è più necessario e le risorse di sistema sono necessarie per un'altra applicazione.

- Ogni processo ha la propria virtual machine (VM), quindi il codice applicazione viaggia isolato dal codice di altre applicazioni.

- Di default, ad ogni applicazione è assegnato un unico Linux user ID (l'ID è utilizzato solo dal sistema ed è sconosciuto all'applicazione). Il sistema imposta i permessi per tutti i file dell'applicazione in modo che solo l'ID dell'utente di quell'applicazione possa accedervi.

In questo modo, Android implementa il principio del privilegio minimo. Ciò vuol dire che, ogni applicazione, di default, ha accesso solo alle componenti che richiede per svolgere il proprio lavoro e non di più. Questo crea un sistema in cui ogni applicazione non può accedere parti del sistema per le quali non ha permesso.

In ogni caso, esistono modi per far condividere dati tra applicazioni, e tra applicazione e sistema:

- É possibile organizzare due applicazioni per condividere lo stesso Linux user ID, in questo caso sono capaci di accedere i file dell'altra applicazione. Per conservare risorse di sistema, applicazioni così create possono lavorare nello stesso processo Linux e condividere la stessa VM (le applicazioni devono essere firmate con lo stesso certificato).

- Un applicazione può chiedere permessi per accedere a dati del dispositivo, come i contatti, gli SMS, memorie esterne (schede SD), fotocamera, Bluetooth e altro. Questi permessi verranno garantiti dall'utente durante l'installazione.

6.4 Le Activity

Un'Activity è un componente di un'applicazione che fornisce uno schermo con il quale l'utente può interagire per fare qualcosa, come fare una telefonata, scattare una foto o mandare una email. Ogni attività è fornita di uno schermo sul quale viene disegnata l'interfaccia utente. La finestra tipicamente riempie lo schermo, ma può essere più piccola e galleggiare sopra altre finestre.

Un'applicazione di solito consiste in attività multiple che sono vincolate l'una all'altra. Tipicamente, un'attività svolge la funzione di attività principale, viene presentata all'utente al lancio dell'applicazione per la prima volta. Ogni attività può iniziare altre attività per svolgere diversi compiti. Ogni volta che un'attività inizia, la precedente viene fermata, ma il

sistema mantiene le attività in uno stack (il *back stack*). Quando una nuova attività inizia, viene fatto un push sul *back stack* e guadagna lo schermo (anche detto *focus*) dell'utente.

Il back stack funziona ovviamente con il meccanismo del last in, first out, quindi, quando l'utente ha finito con un'attività e preme il tasto *Back* (presente in tutti i telefoni Android), l'attività viene tolta dallo stack con un pop e distrutta. L'attività immediatamente precendente viene ripresa.

Quando un'attività è fermata a favore di un'altra attività che parte, viene notificato il cambiamento tramite i metodi di callback della vita di un'attività. Vedi Figura 6.3. Ci sono vari metodi di callback che un attività può ricevere, dato che il sistema può crearla, fermarla, farla ripartire, distruggerla. Ogni callback fornisce la possibilità di svolgere specifiche azioni che sono appropriate per quel cambiamento di stato. Per esempio quando un'attività viene fermata, dovrebbero essere rilasciati gli oggetti che occupano più risorse, come la rete o connessioni a un database. Quando un'attività riparte, bisogna far ripartire si possono riacquistare le risorse e riprendere le azioni che sono state interrotte. Queste transizioni di stato sono parte del ciclo di vita di un'attività.

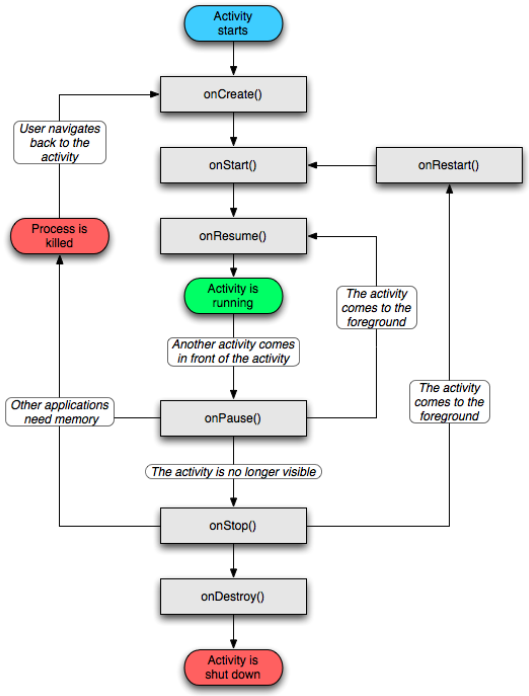


Fig. 6.3: Grafico che illustra il ciclo di vita di un'Activity

6.4.1 Interfaccia Utente

In un applicazione Android, l'interfaccia utente è costruita utilizzando oggetti *View* e *ViewGroups*. Ci sono vari tipi di questi oggetti, ognuno dei quali è discendente della classe *View*.

Gli oggetti View sono l'unità di base dell'interfaccia utente. La classe View serve come base per le sottoclassi chiamate widgets, che offrono oggetti pienamente implementati, come campi testo o bottoni. La classe ViewGroup serve come base per le sottoclassi chiamate layout, che offrono diversi tipi di stile: lineare, tabulare o relativo.

Un oggetto View è una struttura dati le cui proprietà contengono i parametri per lo stile e il contenuto di uno specifico rettangolo dell'area dello schermo. Un oggetto View gestisce, relativamente al proprio rettangolo di schermo, la propria misura, layout, interazione con i tasti o gesti, il cambio di focus. Come oggetto dell'interfaccia utente, un oggetto View è anche un punto di interazione con l'utente e un ricevitore di eventi interattivi.

Il modo più comune per definire il layout e esprimere la gerarchia degli oggetti è con un file XML di layout. XML offre una struttura leggibile ad occhio umano, come l'HTML. Ogni elemento in XML è o una View, o un oggetto ViewGroup (o un discendente). Gli oggetti View rappresentano foglie dell'albero della struttura, gli oggetti ViewGroup sono invece ramificazioni di questo albero.

Il nome di un elemento XML è corrispondente alla classe Java che rappresenta. Un elemento <TextView >crea un TextView nell'interfaccia utente, un elemento <LinerLayout >crea un ViewGroup LinearLayout nell'interfaccia utente. Quando viene caricato un file di layout dalle risorse, Android inizializza questi oggetti run-time, e li fa corrispondere agli elementi del layout.

6.5 Applicazione Android APS

In questa sezione si illustrano le caratteristiche principali dell'applicazione realizzata per la piattaforma Android. Innanzitutto l'applicazione è molto semplice ed è stata compilata in modo da essere portabile su cellulari con Android 1.6, una delle primissime versioni di questo sistema operativo.

In relazione a quanto spiegato nella sezione 6.4 sulle attività e sul layout, ciò che è stato creato per l'applicazione per APS è comprende:

- Un'attività separata per ogni pagina dell'applicazione.
- XML per il design delle attività e degli elementi della lista.

Vengono presentate in successione 3 pagine: Principale, Lista Biglietti, Quantità, CVV2. La navigazione avviene solamente in un senso, se l'utente conferma i dati che ha inserisce dovrà chiudere l'applicazione se vuole cambiare la sua scelta. Questo è dovuto al fatto che viene utilizzata la pagina principale come base su cui visualizzare la scelta dell'utente.

All'avvio la pagina principale appare pressoché vuota, con attivo e visibile solamente un bottone *Avanti*. Questo bottone verrà sempre visualizzato nella pagina principale, per permettere all'utente di avanzare nell'inserimento dei dati alla pagina successiva. Le pagine secondarie sono usate per la selezione del titolo di viaggio o l'inserimento dei dati. Una volta visitata una pagina secondaria, alla conferma dei dati inseriti l'utente viene riportato alla pagina principale. I dati confermati vengono visualizzati nella parte bassa della pagina, in ogni momento l'utente può controllare che biglietto comprerà o quanti ne comprerà. Nella prima implementazione veniva visualizzato in chiaro anche il CVC2/CVV2 nella pagina principale. Questo è stato modificato nella versione successiva e nella pagina principale viene visualizzato solamente il titolo di viaggio e la quantità da acquistare.

Una volta che l'utente ha inserito e confermato l'ultimo dato, viene riportato alla pagina principale per l'ultima volta. Premendo il bottone *Avanti* ancora una volta, viene verificato che non ci siano stati errori fino ad ora e viene attivato il bottone di invio SMS. La conferma che è stato attivato avviene tramite un pop-up con un unico bottone di uscita *Ok*. In questo modo si è sicuri che l'utente ha letto il messaggio.

Il bottone di invio SMS compare a questo punto sotto quello di *Avanti*. Qualora lo schermo fosse troppo piccolo per visualizzare la pagina intera, o eventuali modifiche rendano la pagina più lunga dello schermo, il contenuto della stessa è fruibile scorrendo verso il basso tramite il touchscreen. Alla pressione del bottone per l'invio di sms viene visualizzato un *Toast* che conferma se l'invio è andato a buon fine o ci sono stati eventuali errori. Un Toast è un messaggio che rimane visualizzato sullo schermo per poco tempo e che poi scompare da solo. Consente di fornire all'utente comunicazioni rapide senza che debba interagire ulteriormente.

Una volta inviato con successo un SMS, è possibile utilizzare ancora il bottone per inviare un altro messaggio identico. Ad ogni pressione del tasto verrà ora visualizzato un messaggio che chiede la conferma di invio. In questo modo si evita che l'utente prema più volte il bottone e invii messaggi multipli involontariamente.

(a) Pagina principale all'avvio

(b) Lista dei biglietti

(c) Pagina quantità

(d) Pagina CVV

(e) Dati controllati con successo

(f) Avviso di invio con successo

(g) Avviso per evitare sms multipli

Fig. 6.4: Le figure (a), (e), (f), (g) mostrano la pagina principale modificarsi man mano che si inseriscono dati.

Bibliografia

[1] (2011, 03). [Online]. Available: www.phonegap.com

[2] T. N. Company, "Media monthly report q3 2010," 11 2010.

[3] (2010, 03). [Online]. Available: http://www.comscore.com/Press_Events/Press_Releases/2010/3/UK_Leads_European_Countries_in_Smartphone_Adoption_with_70_Growth_in_Past_12_Months

[4] B. Kennedy and C. Musciano, *HTML: The Definitive Guide*, second edition ed. O'Reilly Media, 1997.

[5] E. Meyer, *Erik Meyer on CSS*, first edition ed. New Riders Press, 2002.

[6] D. Flanagan, *Javascript: The Definitive Guide*, second edition ed. O'Reilly Media, 1997.

[7] (2011, 03). [Online]. Available: http://wiki.phonegap.com/w/page/16494772/FrontPage

[8] (2011, 03). [Online]. Available: http://us.blackberry.com/developers/

[9] (2011, 03). [Online]. Available: http://www.visaeurope.com/en/about_us/what_we_do/payment_security.aspx

[10] (2010, 04). [Online]. Available: http://www.allaboutsymbian.com/news/item/11342_comScore_data_shows_smartphone.php

[11] (2011, 03). [Online]. Available: http://developer.android.com/guide/index.html