UNIVERSITA' DEGLI STUDI DI TRENTO
Facoltà di Scienze Matematiche, Fisiche e Naturali

Corso di Laurea triennale in Informatica

Elaborato finale

Progettazione e Sviluppo del client ANDIAMO - Car pooling per dispositivi Symbian

Relatore:
Ing. Paolo Giorgini

Laureando:
Paolo Rallo

Anno Accademico 2006 - 2007

Indice

Introduzione

Nell'ultimo decennio l'elemento predominante che ha guidato lo sviluppo della tecnologia è senza dubbio la mobilità. Dall'avvento dei cellulari e dei computer portatili, gli utenti (quindi il mercato) hanno sempre più “voglia di libertà”: abbracciando le tecnologie senza fili e svincolandosi dalle restrizioni dei cavi sulle loro scrivanie.

Nuove tecnologie wireless sono nate per venire incontro alle esigenze del mercato. Da qualche anno le reti wireless non sono più un'esclusiva delle grandi aziende, ma grazie allo standard IEE 802.11, l'utenza può creare le proprie reti senza fili. Un'altra importante innovazione è il bluetooth: una versione del wi-fi semplificata che permette la comunicazione a corto raggio tra dispositivi mobili di limitata energia.

Grazie a queste nuove tecnologie, all'incremento esponenziale delle vendite di cellulari e all'esigenza dell'utenza di maggiori prestazioni; nell'ultima decade si è assistito al porting del wi-fi e del bluetooth sui cellulari. Tutto ciò ha favorito la nascita di cellulari “rivoluzionari”. Questa generazione di cellulari o piccoli (per grandezza) computer, non solo permette la comunicazione tra persone via voce, ma anche lo scambio di informazioni con qualsiasi dispositivo.

Nuovi sistemi operativi per cellulari sono nati per offrire un miglior servizio all'utenza media con un'interfaccia intuitiva, ma soddisfano anche le esigenze dei programmatori grazie ad accordi tra le aziende leader. System calls standard permettono la portabilità di codice tra cellulari e tra cellulari e pc; le macchine

virtuali non sono più una chimera e la Java MicroEdition è supportata oramai da quasi tutti i cellulari o dispositivi mobili di ultima generazione.

I software per cellulari attualmente sono di numero limitato, ma la domanda di programmi è in continuo aumento. Per sopperire a questa esigenza, i creatori di software hanno spaziato in altri ambiti che non fossero il semplice utilizzo del cellulare come mezzo comunicazione vocale.

Uno degli scenari che maggiormente interessa l'utenza è il tema della viabilità, essendo ancora un problema irrisolto. Il traffico è in aumento e una coscienza sociale per avere un mobilità sostenibile è agli albori. In questa realtà è nato il progetto “Andiamo – Car pooling” (il quale sfrutta la tecnologia del cellulare per offrire una possibile soluzione a questo problema): un servizio pensato per l'utenza e costruito grazie all'ambiente di sviluppo Java, sfruttando appieno le potenzialità del cellulare.

Questo elaborato si colloca nell'ambito del progetto “Andiamo – Car pooling”, nato per permettere a chiunque di offrire o ricercare con facilità un passaggio in automobile. Il nodo critico del client Andiamo consiste essenzialmente nella portabilità e nella pesantezza del programma; essendo costruito in Java, il programma dovrebbe rispettare appieno l'obiettivo per cui è stato creato, funzionando su qualsiasi piattaforma che abbia una macchina virtuale integrata.

L'attuale client è legato alle restrizioni del sistema operativo sottostante e non si adatta ai cellulari che hanno caratteristiche grafiche e strutturali differenti. Una soluzione possibile, ma non efficiente, sarebbe la creazione di differenti versioni di Andiamo per ogni marca e modello di cellulare, pda, smartphone o simili.

L'obiettivo della tesi è offrire un'alternativa allo scenario descritto precedentemente. Si vuole proporre un miglior approccio per risolvere il problema senza aggirarlo, attraverso una programmazione ad un livello più basso con l'aiuto delle Canvas (classe base per scrivere applicazioni in cui bisogna disegnare sul display o comunque averne un controllo a livello grafico) ed estenderne le carenti librerie grafiche. Grazie a questa metodologia d'intervento si elimina ogni problema di portabilità di Andiamo e come effetto secondario si dispone di una nuova libreria grafica da riutilizzare per nuovi progetti. Ciò facilita la risoluzione dei problemi di adattabilità grafica della libreria standard, partendo da un modello grafico di riferimento che si adatta al contesto.

La tesi si articola in tre capitoli. Il primo capitolo riguarderà la descrizione del progetto Andiamo e il suo funzionamento, approfondendo i problemi riscontrati.

Il secondo capitolo verterà la riprogettazione di Andiamo: partendo dalla scelta del sistema operativo Symbian, passando alla classe Canvas e arrivando infine alle possibili soluzioni dei principali problemi.

Nel terzo capitolo, infine, sarà presentata la parte implementativa del client e si descriverà come si è realizzato il software.

Capitolo 1

Il sistema Andiamo – Car pooling

In questo capitolo si spiegherà il funzionamento di Andiamo e il concetto di portabilità applicato alla grafica. Verrà inoltre fatta una panoramica sui principali problemi che rendono Andiamo poco adattabile ai vari modelli di cellulare, smartphone o pda.

1.1 Funzionamento di Andiamo

Andiamo è un client che permette ad un utente qualsiasi di offrire o richiedere un passaggio in macchina in un particolare giorno, ad un data ora e potenzialmente per qualsiasi destinazione. In particolare si possono indicare dei punti d'incontro intermedi per massimizzare l'efficenza del car pooling[1].

Un'altra possibilità fornita all'utente è la verifica di disponibilità attive: questa opzione permette di conoscere se altri utenti hanno già offerto o richiesto il servizio relativo alle sue esigenze (il giorno e il tragitto desiderati, in un determinato lasso di tempo).

La comunicazione tra gli utenti che si avvalgono di Andiamo avviene tramite sms; un servizio server telefonico di messaggistica è addetto allo smaltimento e al filtraggio dei dati dei vari utenti. Dopo un'analisi incrociata delle richieste e delle offerte attive, verrano inviate le risposte agli interessati. Per garantire la serietà del

1Il car pooling o pool-car o carpool, termine inglese traducibile come "auto di gruppo", è una modalità di trasporto che consiste nella condivisione di automobili private tra un gruppo di persone, con il fine principale di ridurre i costi del trasporto

sistema e di coloro che usano il servizio, esiste un sorta di punteggio che viene assegnato al guidatore e/o i passeggeri dopo il tragitto.

1.1.1 Nel dettaglio

Il software è diviso essenzialmente in quattro parti. La componente principale che si andrà ad analizzare in questo elaborato è la sezione grafica; essa guida l'utente in varie finestre grafiche che gli permetteranno di offrire o richiedere un particolare servizio.

La navigazione dell'utente tra le varie finestre di Andiamo avviene tramite menù, composti da bottoni, i quali rappresentano il vero problema del programma insieme a liste poco efficienti. Queste componenti grafiche non sono performanti, essendo poco duttili graficamente.

Le altre sezioni che compongono Andiamo creano una sorta di sottoprogramma che supporta la grafica, cioè un programma nel programma che si occupa di fornire le scelte all'utente nei vari menù, controllarne la validità ed eseguirle in caso positivo.

Nel seguito del capitolo si approfondiranno i problemi derivanti dalla scarsa adattabilità delle finestre grafiche.

1.2 Usabilità e portabilità

Tralasciando la parte server, ogni opzione del client Andiamo su cellulare è gestita tramite componenti grafiche come menù, bottoni e icone che rendono l'interfaccia

utente di facile da usare e intuitiva per chiunque.

Andiamo garantisce l'usabilità[2] del prodotto; infatti dispone di un'interfaccia user-friendly[3] che rispetta le seguenti caratteristiche [4]:

- *adeguatezza*: devono essere richiesti solo gli input necessari per svolgere un determinato compito;
- *facilità di apprendimento*: l'utilizzo deve essere chiaro ed intuitivo, rendendo minima la lettura di manuali o istruzioni d'uso (anch'essi devono essere chiari e comprensibili);
- *robustezza*: l'impatto dell'errore deve essere inversamente proporzionale alla probabilità dello stesso.

L'usabilità non è il solo elemento determinate per valutare il prodotto finale, è necessario considerare anche l'efficacia della sua interfaccia. Se non è chiara o difficilmente comprensibile, il programma può non soddisfare l'utente e nel peggiore dei casi quest'ultimo abbandona il software e rivolgendosi ad altri servizi [5].

Senza soffermarsi sulle regole relative alla domanda e all'offerta del mercato, l'utente predilige un'interfaccia chiara, interattiva e il più personalizzabile possibile.

Per misurare il grado di soddisfazione dell'utenza verso Andiamo, l'usabilità non è l'unico elemento da prendere in considerazione: è indispensabile introdurre il

2 L'usabilità è definita dall'ISO (International Standard Organization), come l'efficacia, l'efficienza e la soddisfazione con le quali determinati utenti raggiungono determinati obiettivi in determinati contesti. In pratica definisce il grado di facilità e soddisfazione con cui l'interazione uomo - strumento si compie

3 Sinonimo di usabile; di facile usabilità

concetto di ergonomia[4] e di livello ergonomico, il quale determina la qualità del rapporto tra l'utente e il mezzo utilizzato. Il requisito più importante per determinare questo livello è la sicurezza[5], seguito dall'adattabilità, l'usabilità, il comfort, la gradevolezza e la comprensibilità [6].

Andiamo non prevede sistemi di sicurezza, quindi in questo caso non è un elemento determinate del livello ergonomico; infatti è il sistema operativo che gestisce la sicurezza con opzioni di settaggio dei software e tramite certificati di licenze. Il metodo delle licenze verrà sviluppato nel proseguo dell'elaborato, fornendo valide argomentazioni sul suo svantaggio.

L'adattabilità, o meglio dire la portabilità, è il vero problema del software.

Andiamo purtroppo non è adattabile graficamente e insieme ad un livello di confort e gradevolezza insoddisfacenti, il livello ergonomico si abbassa complessivamente

1.3 Problemi di portabilità per Andiamo

Di seguito si mostreranno degli esempi, corredati da foto[6] e da dettagliate descrizioni, delle cause principali che rendono Andiamo scarsamente efficace per catturare l'attenzione del pubblico.

Il programma è perfettamente funzionante, ma l'utenza non prende in considerazione il software in esecuzione sotto la grafica (come nel caso delle

4 L'ergonomia (dal greco "*ergon*" (lavoro) e "*nomos*" (legge), secondo la I.E.A., è quella scienza che si occupa dell'interazione tra gli elementi di un sistema (umani e d'altro tipo) e la funzione per cui vengono progettati (nonché la teoria, i principi, i dati e i metodi che vengono applicati nella progettazione), allo scopo di migliorare la soddisfazione dell'utente e l'insieme delle prestazioni del sistema (def. I.E.A.). In pratica è quella scienza che si occupa dello studio dell'interazione tra individui e tecnologie

5 salvaguardia dei sistemi informatici da potenziali rischi e/o violazioni dei dati

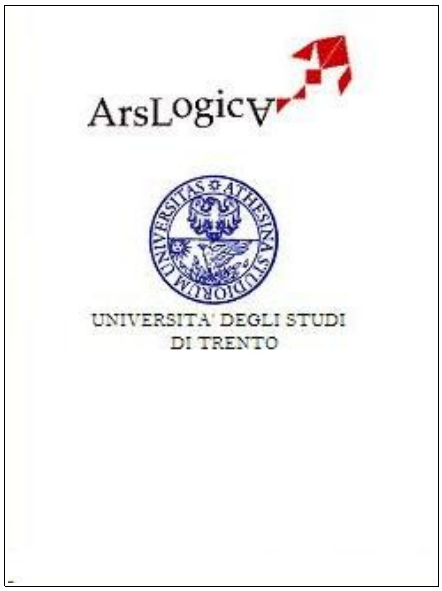
6 In realtà si parla di screenshots, ovvero immagini catturate dallo sfondo del pc

pagine web). Gli utenti si basano inizialmente sulla GUI[7] per giudicare il prodotto [7], se si passa l'ostacolo dell'interfaccia, in seguito presteranno attenzione all'usabilità.

La presentazione renderà comprensibile al lettore la scelta di ricostruire da zero un'interfaccia grafica scadente.

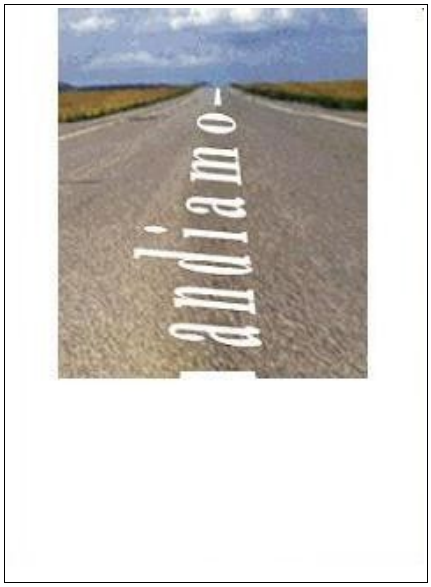
Le immagini mostrano un'istantanea del programma Andiamo in esecuzione sull'emulatore del pc[8]; benché l'elaborato riporti illustrazioni di una simulazione, l'emulatore rispetta fedelmente le caratteristiche grafiche di un cellulare, di uno smartphone e di una pda.

Le figure 1 e 2 mostrano la presentazione in sequenza del programma Andiamo all'avvio a tutto schermo.



Prima schermata introduttiva

Figura 1



Seconda schermata introduttiva

Figura 2

7 Acronimo di graphical user interface, è un tipo di interfaccia utente che permette all'utente di interagire con programma

8 Sun Java Wireless Toolkit 2.5.1, un'emulatore per pc che simula un cellulare, uno smartphone o un pda a discrezione del programmatore

Si può immediatamente notare che l'altezza e la larghezza delle immagini centrali non coincidono con le dimensioni dello schermo; questo significa che qualsiasi sia la risoluzione dello schermo, l'immagine rimarrà invariata.

Oltre a questo scenario, su alcuni modelli di cellulari l'immagine può sforare l'altezza o la larghezza del display. Già da questo esempio si può comprendere l'inesistenza di una adattabilità o portabilità grafica.

La seguente figura mostra un menù caricato dopo la presentazione iniziale (figura 3). Nell'illustrazione è presente una zona in cui il refresh[9] non ha avuto buon fine e preclude la visione completa della stringa animata sottostane; il ricaricamento del display con una nuova finestra grafica è avvenuto, ma con un errore significativo. Questo tipo di malfunzionamento mina la serietà del software e abbassa il livello di soddisfazione dell'utente compromettendo il livello ergonomico sopracitato.

In questo caso l'errore si è limitato a una piccola porzione dello schermo. Non è possibile prevedere le conseguenze di un errato refresh in altri cellulari, ma si possono ipotizzare almeno quattro situazioni:

- l'errore non avviene e il refresh della nuova finestra carica tutti gli oggetti senza disfunzioni;
- l'errore è simile: la stringa animata o qualsiasi altro componente grafico non è visibile perché coperto dall'immagine della precedente finestra;
- il refresh non ha avuto effetto, il programma si blocca e non procede regolarmente;
- l'errore causa il crash[10] del programma.

9 Dall'inglese rinfresco, in informatica il termine indica lo spegnimento e l'istantanea riaccensione dello schermo che comporta il ricaricamento di tutta la grafica.

10 Un programma o più in generale un sistema collassa causando l'interruzione immediata del suo

Prima finestra del programma: con errata visualizzazione

Figura 3

Queste ipotesi rendono il sistema non adattabile al contesto in cui viene eseguito; infatti nei casi limite il programma funzionerà solo su alcuni modelli di cellulare, pda o smartphone.

Nella figura 4 si possono notare delle incongruenze del menù: le icone e le opzioni di scelta non sono allineate sulla stessa riga. Come risultato si ha un bottone del menù diviso a metà. In fondo alla figura si può notare un'icona isolata senza il resto dell'etichetta. Questo metodo di costruzione del menù può erroneamente suggerire all'utente che si possono selezionare le icone oltre alle opzioni sottostanti.

Come precedentemente suggerito, una possibile soluzione è allineare le icone con le proprie labels e opzioni. Nella figura seguente (figura 5) viene mostrato che questa scelta non è una soluzione.

funzionamento; nei casi peggiori un collasso si può propagare anche ad altri programmi o sistemi

Selezione errata nel menù

Figura 4

Errata visualizzazione degli oggetti del menù

Figura 5

Sia nel metodo con le icone e le opzioni disgiunte, sia nell'ipotetica soluzione proposta dell'allineamento, non si risolve il problema che si nota nella figura 5.

L'icona in alto è tagliata e si presenta lo stesso problema della figura 1 e della figura 2. Questo comportamento non è riservato solo alle icone, ma anche gli altri oggetti grafici subiscono il medesimo trattamento. Il programma si appoggia su un'architettura a Midlet[11], in cui l'oggetto menù non considera il numero di componenti grafiche caricate e dell'altezza del display.

La questione non si pone se la somma delle altezze di tutti gli oggetti caricati nel menù è inferiore all'altezza dello schermo, ma si ripresenta il problema dell'adattabilità o portabilità grafica. La dimensione dello schermo è un elemento dinamico nella progettazione di qualsiasi componente grafica.

Il menù usato in Andiamo carica tutti gli oggetti che gli si appendo durante la fase

11 Una classe di java, solo per la versione micro edition, che fornisce al programmatore dei metodi per sviluppare un'applicazione per qualsiasi dispositivo mobile, come i cellulari, che abbia una macchina virtuale installata.

di programmazione, e scorrendolo con le frecce direzionali mostra l'oggetto seguente non considerando i precedenti. La situazione rimane invariata anche se si aggiunge un oggetto alla lista del menù ad ogni scorrimento. In conclusione il menù (tecnicamente chiamato Form, del pacchetto grafico per creare una Midlet) non è adattabile.

Come nella figura 5, anche nella figura 6 le dimensioni dello schermo non sono

Oggetti grafici sovrapposti

Figura 6

rispettate. In questa finestra si è adottata la soluzione di allineare labels e icone sulla stessa linea; (come accennato in precedenza) questa soluzione non risolve alcun problema riguardante la dimensione dello schermo e degli oggetti grafici.

In seguito alle suddette riflessioni, si può constare che Andiamo, e in particolare il menù delle Midlet, non tiene conto della larghezza dello schermo.

Ancora una volta si può notare che non esiste una gestione razionale degli oggetti grafici presenti nel menù. In questa foto, in particolare, si vede come gli oggetti sono organizzati a livelli, ma le icone con le rispettive labels coprono quelle precedenti.

I problemi sin qui affrontati per i menù (Form) sono analoghi anche per le liste (List); infatti le liste non sono altro che menù i cui elementi sono inermi, cioè modificabili.

Finestra dei punti d'incontro

Figura 7

Scelta dei punti d'incontro

Figura 8

Stessi punti d'incontro con città diversa

Figura 9

Le figure 7, 8 e 9 mostrano una possibile azione che si può compiere usando Andiamo.

In questa situazione all'utente è permesso scegliere i punti d'incontro della città di partenza di un'area a piacere. L'analisi seguente non vuole soffermarsi solamente sulla grafica, ma sugli eventi causati dalla pressione dei tasti in questo momento del programma.

Come si può vedere nella successione, l'utente ha scelto una città, l'area e i relativi punti d'incontro. Dopo aver cambiato la città di partenza, la lista dei punti d'incontro non si è modificata (questo problema è analogo anche per il cambio dell'area).

Il programma non ha correttamente catturato l'evento: si è limitato a cambiare la label della città di partenza, ma i punti d'incontro sono rimasti tali. Questo bug di

Java[12] ME nelle finestre di popup [8], in questo caso, limita la scelta dell'utente.

Come detto precedentemente, su alcuni dispositivi il problema potrebbe essere simile, non esserci o aggravarsi. Il bug è stato eliminato recentemente [8] con l'aggiornamento del sistema operativo e di Java. Ricompilare il programma per i nuovi modelli può essere controproducente. Lo scenario che si prospetta nel breve periodo vede gli utenti provvisti di cellulari o simili impossibilitati nell'usare la nuova versione di Andiamo, a causa delle specifiche tecniche che non supportano i nuovi aggiornamenti.

Gli utenti che prima usavano Andiamo, ora sono costretti ad aggiornarsi per restare al passo con il software. Una minima parte dell'utenza potrà permettersi di usare il nuovo Andiamo.

Pensando in termini di tempo medio-lunghi si avrà inizialmente un numero ristretto di utenti che potrà usare Andiamo, ma si può auspicare che nel futuro il numero aumenti.

Le due visioni prospettate prevedono che la maggior parte dell'utenza si aggiorni tecnologicamente, ma si corre il rischio che gli utenti perdano interesse verso il software che li costringe a spese onerose.

Questa soluzione non è praticabile, in quanto il codice riadattato di Andiamo non è retroattivo e quindi non portabile sui modelli obsoleti.

In questo capitolo si è voluto evidenziare i principali problemi che rendono Andiamo molto poco portabile; altri problemi riguardanti l'estetica dell'interfaccia grafica del programma sono stati deliberatamente saltati per appesantire

12 Linguaggio di programmazione orientato agli oggetti, derivato dal C++ (e quindi indirettamente dal C) e creato dagli ingegneri di Sun Microsystems.

l'elaborato.

Nel capitolo seguente verrano proposte e argomentate soluzioni che miglioreranno la portabilità di Andiamo rendendolo usabile su qualsiasi dispositivo mobile che abbia una macchina virtuale integrata. Nei casi principali, nella parte implementativa della tesi, verrà mostrato una parte di programma che rispecchia la sezione di sviluppo e progettazione del secondo capitolo.

Capitolo 2

Progettazione del client Andiamo

Al fine di sopperire alla mancanza di adattabilità o portabilità (grafica) del client Andiamo, che gioca un ruolo essenziale per rendere il software usabile all'utente su qualsiasi piattaforma (cellulare, pda o smartphone), è nata l'esigenza di sviluppare un nuovo client Andiamo.

Come spiegato nel capitolo precedente, le librerie grafiche di Java Micro Edition esistenti, e usate da Andiamo, non sono sufficienti; partendo da questo presupposto, verrà di seguito presentata la nuova architettura grafica del software. Inoltre si spiegherà l'interazione tra la nuova GUI e la parte del programma non grafica che resterà immutata nella nuova versione.

Si illustreranno quindi le varie sotto-componenti entrando nel dettaglio delle loro interazioni e interconnessioni. Verranno suggeriti inoltre alcuni possibili scenari d'utilizzo della nuova liberia grafica.

2.1 Tra vecchio e nuovo

Come anticipato in precedenza il client Andiamo progettato in Java per un ambiente Symbian.

Come tutti i software, anche Java e Symbian presentano vantaggi e svantaggi; in particolare Java Micro Edition risente di notevoli mancanze rispetto alle versioni SE (Standard Edition) ed EE (Enterprise Edition) a causa delle restrizioni di energia, memoria e velocità di calcolo dei dispositivi mobili [9].

2.1.1 La scelta di Symbian

Symbian è un sistema operativo per dispositivi mobili creato da un consorzio delle più grandi aziende di telecomunicazione del mondo. Erede di Epoc (nato negli anni '80 e sviluppato in C++), ha ricevuto notevoli investimenti portandolo all'attuale versione 9.3. Sul mercato dei sistemi operativi ci sono altri concorrenti come Windows Mobile, Palm OS e Linux, ma la maggior parte dei cellulari è basata su interfacce derivate da Symbian, come Nokia Serie 60 (la più diffusa), Nokia Serie 80, Nokia Serie 90 e UIQ. Come altri sistemi operativi dispone di funzionalità di multithreading, multitasking e protezione della memoria.

Grande importanza è data all'utilizzo della memoria mediante tecniche specifiche di Symbian, che determinano la rarità degli errori dovuti a una cattiva gestione della memoria (memory leaks). Tecniche analoghe permettono un'altrettanta efficiente gestione dello spazio su disco.

Il funzionamento di Symbian è basato su eventi e la CPU è automaticamente disabilitata quando non vi siano eventi attivi: il corretto uso di questa tecnica aiuta ad assicurare alle batterie una durata maggiore.

Operatori e produttori di telefoni cellulari preferiscono la maggiore configurabilità di Symbian rispetto a Windows Mobile, sebbene questa renda poi l'integrazione dei dispositivi basati su Symbian più difficoltosa.

Gli altri due sistemi operatori non sono così diffusi come Symbian e Linux Embeded, in particolare, è ancora agli albori.

Quasi tutto il codice di Symbian è fornito ai produttori di dispositivi mobili e ad

altri operatori. Inoltre la documentazione relativa alle API[13] è disponibile pubblicamente, dunque chiunque ha la possibilità di sviluppare software per Symbian.

La scelta di Symbian, quindi, è dettata dal largo consenso che ha riscosso negli ultimi anni e dal suo "monopolio/standard" sui mercati di cellulari, smartphone e pda [10].

Sebbene Symbian sia diffuso a livello planetario, per raggiungere anche gli utenti che hanno scelto Linux si è preferito sviluppare il programma in Java e non in Symbian C++; così da permettere ad Andiamo di funzionare anche su quei terminali mobili che generalmente accettano soltanto applicazioni Java.

2.1.2 La scelta di programmare in Java

Java per dispositivi mobili è disponibile nella versione Micro Edition. Noto anche come Java ME o J2ME, è un runtime e una collezione di API per lo sviluppo di software dedicato a dispositivi con risorse limitate come pda, telefoni cellulari e simili.

J2ME è la tecnologia più diffusa per lo sviluppo di giochi e utilities per i cellulari.

Il suo funzionamento può essere emulato con un personal computer, cosa che semplifica l'attività di sviluppo e di collaudo.

J2ME può essere utilizzato per sviluppare applicazioni per una ampia gamma di apparati. Diverse tipologie di dispositivi sono identificati da diversi profili e a loro volta riferiti a diverse configurazioni. La configurazione Connected Limited

13 L'acronimo di application programming interface, è un'interfaccia di codice che provvede a supportare le librerie di un programma

Device Configuration (CLDC), per esempio, include un sottoinsieme minimo di classi Java, ed è utilizzata su dispositivi con scarsissime capacità di calcolo. Fra i profili che operano in configurazione CLDC compare il Mobile Information Device Profile (MIDP), pensato per i cellulari. Il MIDP prevede un sistema di GUI orientato a display a cristalli liquidi e una API di base per giochi in 2D.

Molti cellulari moderni vengono forniti con un'implementazione residente del MIDP. Un altro profilo che utilizza la configurazione CLDC è l'Information Module Profile (IMP), usato per esempio nei distributori automatici e in altri apparati dotati di funzioni minime di display e di connettività di rete.

La scelta di programmare in Java Micro Edition è legata alla sua portabilità grazie a profili e configurazioni, ma come analizzato nel primo capitolo la grafica non è adattabile. I vari CLDC, MIDP, e IMP non massimizzano l'adattabilità di Andiamo sui vari dispositivi mobili.

L'uso massiccio nel progetto di classi che utilizzano funzioni ad alto livello (come Midlet, presente nel pacchetto MIDP insieme alla sopracitata Form, List ecc...) nell'implementazione di Andiamo, non ha giovato alla sua grafica (vedi l'approfondimento della componente grafica: menù, nel capitolo precedente).

La vecchia architettura a Midlet sarà rimpiazzata con l'uso di librerie grafica che agiscono a un più basso livello, così da ricostruire gli oggetti grafici forniti da Java e renderli realmente adattabili ad ogni dispositivo con un proprio design dello schermo.

L'uso della classe Canvas[14], fornita anch'essa nello standard di J2ME, è una

14 Una classe del pacchetto MIDP che, a differenza delle altre classi che la compongono, fornisce

soluzione per rimediare ai bug delle delle classi che usano le Midlet.

Grazie a questo pacchetto si potrà aumentare il grado di personalizzazione delle applicazioni disegnandone i vari componenti e le schermate.

2.2 Architettura di Andiamo

Prima di iniziare le modifiche del codice, è necessario comprendere l'obiettivo che si vuole raggiungere e i passi intermedi da conseguire che porteranno al risultato finale atteso.

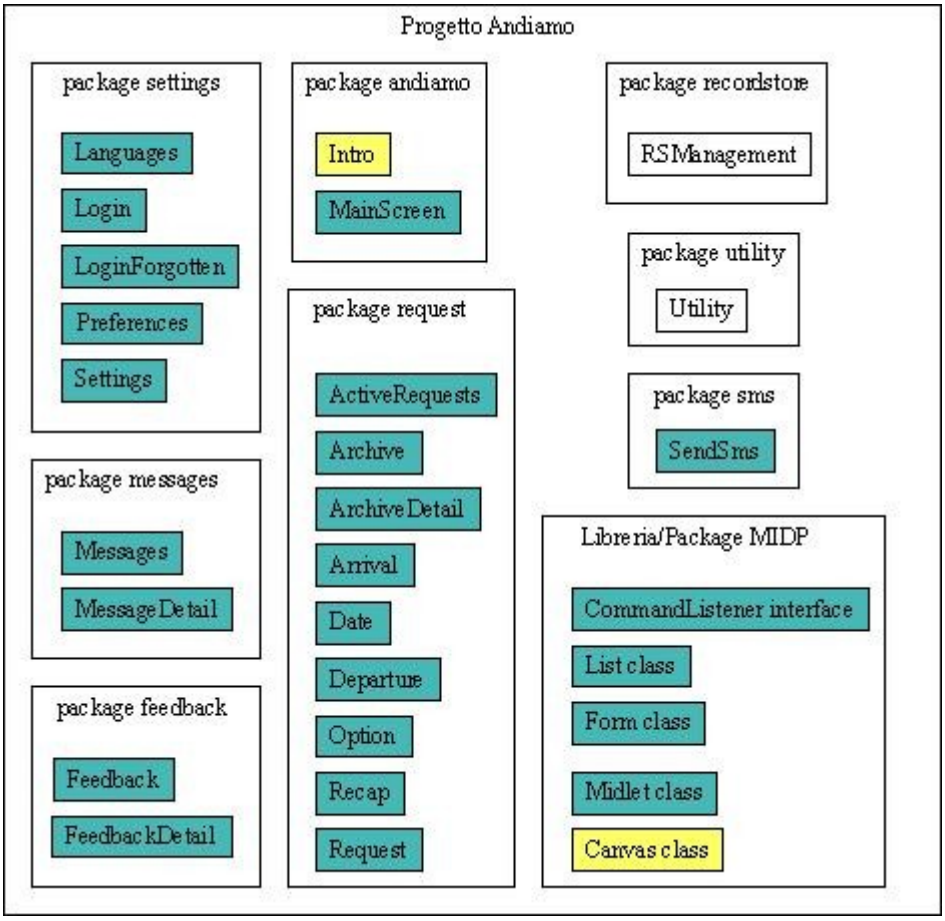
L'obiettivo della tesi è sostituire le classi grafiche di alto livello usate in Andiamo, presenti nel pacchetto MIDP, con altrettante nuove classi che ereditano i metodi e le interfacce dalla libreria Canvas di basso livello, anch'essa presente in MIDP. Non tutte le classi del progetto Andiamo contribuiscono alla costruzione della grafica: alcune servono come supporto all'intero sistema, mentre altre gestiscono gli eventi in input e in output.

Di seguito verrà mostrato uno schema esplicativo di tutto il sistema Andiamo composto dalle varie classi che lo compongono.

Nella figura 10, con il colore blu sono evidenziate le classi da modificare che usano i componenti grafici che creano le Midlet, in quanto finestre grafiche. Il bianco indica le classi del progetto non dedicate alla grafica: indispensabili al progetto, ma non utili al fine di migliorare la portabilità del progetto.

---

metodi e interfacce per costruire oggetti grafici partendo da un livello più basso di programmazione.



Architettura di Andiamo

Figura 10

Nella nuova versione di Andiamo queste classi non saranno eliminate ma riviste e migliorate, aggiungendo nuovi metodi e preservando quelli preesistenti.

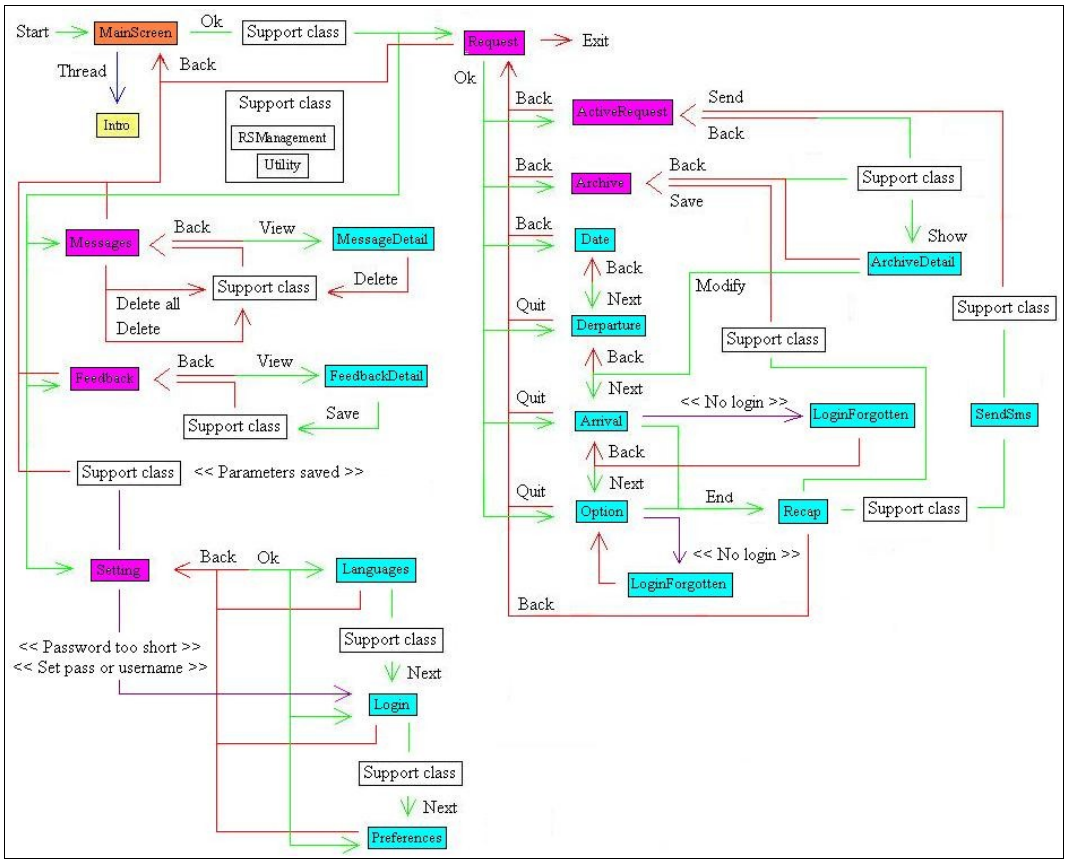
Un'unica classe (di colore giallo) implementa i metodi delle Canvas, ma i risultati in figura 1 e in figura 2 testimoniano la cattiva programmazione della classe Intro.

Si può notare che nel package MIDP, ci sono solamente quattro classi e un'interfaccia; naturalmente ce ne sono altre che fanno parte della libreria [11], ma per motivi spazio non sono state inserite perché poco importanti.

Al fine di eliminare i problemi elencati nel primo capitolo, verranno prese in

considerazione solo le classi List, Form, Midlet, Canvas e l'interfaccia CommandListener.

La classe principale MainScreen che estende la classe Midlet, istanzia le altre classe del progetto di colore blu (liste e menù) e gestisce la loro visibilità sullo schermo. In altri termini, tutte le classi del progetto fanno riferimento a MainScreen, in quanto è l'unica che possiede i metodi per renderle visibili sul display. Da questo vincolo non trascende la classe Intro che estende Canvas; infatti alla base di qualsiasi programma per dispositivo mobile che usi il package MIDP deve esserci almeno una classe Midlet.



Funzionamento delle classi di Andiamo

Figura11

La nuova versione di Andiamo non potrà aggirare questa regola, anche se tutte le classi (meno una) estenderanno Canvas.

In figura 11 si può osservare il funzionamento di Andiamo partendo dalla classe Midlet MainScreen (segnata in arancione), sino ad arrivare con le giuste operazioni ad inviare un'offerta o una richiesta via sms.

Nell'immagine sono differenziate le classi in liste e menù con i rispettivi colori viola e azzurro. Si può notare che le liste permettono solo operazioni di navigazione (Ok, Show, View); infatti il loro compito è di permettere all'utente di selezionare oggetti inermi o non modificabili a loro interno che lo porteranno in una differente finestra grafica. I menù (azzurri) fornisco all'utente strumenti per cambiare le opzioni di settaggio e quindi per costruire una richiesta o un offerta personalizzata. Alla pressione del tasto “Ok” l'utente effettuerà una scelta cambiando il valore di un opzione particolare di un campo di testo, calendario o di una lista di valori (radiobuttons).

Le classi Utility e RSManagement dispongono di metodi che supportano le altre classi, come funzioni per aprire o salvare dati e forniscono le varie scelte per ogni opzione del menù.

2.3 La classe Canvas

La nuova architettura di Andiamo prevede la sostituzione delle classi List, Form, Midlet e dell'interfaccia CommandListener con la sola classe Canvas, ma in cosa consiste e come funziona la classe Canvas?

Fino ad ora si è parlato delle Canvas come un metodo per programmare a basso livello, ma cosa significa?

La classe Canvas si può ritenere una libreria per la programmazione a basso livello perché fornisce istruzioni estremamente basilari che saranno elaborate direttamente dal processore e permettono un totale accesso alle risorse del dispositivo.

L'accesso ad una particolare risorsa motiva la scelta delle Canvas; grazie ad una programmazione a basso livello, adesso si può lavorare con una nuova risoluzione e slegarsi dai vincoli del sistema operativo sottostante. Come nelle figure 1 e 2 la nuova grafica di Andiamo coprirà tutto lo schermo e gli unici vincoli del programma saranno la larghezza e l'altezza reale del display.

I programmi scritti in J2ME con l'ausilio di questa classe non lasciano niente di sottinteso, ma esplicitano ogni istruzioni fino all'essenziale rendendoli estremamente efficienti sia in termini di risorse utilizzate che in velocità di elaborazione.

Per sopperire alla mancanza dell'interfaccia CommandListener (scartata per il bug descritto nel capitolo precedente), vengono fornite sei differenti nuove interfacce per catturare gli eventi che caratterizzano l'attività dell'utente che utilizza la tastiera:

- protected void `keyPressed(int kCode)` – invocato se un tasto (identificato da kCode) è premuto e rilasciato;
- protected void `keyReleased(int kCode)` – invocato se un tasto è

rilasciato;

- protected void `keyRepeated(int kCode)` – invocato se un tasto è premuto ripetutamente;
- protected void `pointerPressed(int x, int y)` – se il puntatore viene premuto nella posizione di coordinate (x,y);
- protected void `pointerDragged(int x, int y)` – se il puntatore viene trascinato dalla posizione di coordinate (x,y);
- protected void `pointerReleased(int x, int y)` – se il puntatore è rilasciato nel punto di coordinate (x,y).

In realtà, di queste sei, sarà utile solo la prima e, in quanto interfaccia, come corpo verrà creato un metodo apposito per identificare il tasto premuto dall'utente.

Gli eventi sono gestiti da keyPressed() e il bug di CommandListener e delle finestre popup non è più un problema.

Il disegno della schermata avviene tramite l'invocazione dell'interfaccia paint; di default, la classe appena istanziata che estende Canvas la chiama automaticamente. Se si desidera invocare nuovamente paint lo si può fare attraverso il metodo repaint (viene richiamato quando la nostra Canvas ha subito delle modifiche e deve essere ridisegnata tutto o in parte).

Oltre a questi metodi, ci sono solamente funzioni di settaggio del colore, di disegno di stringhe, rettangoli, cerchi o triangoli pieni o vuoti e controlli per conoscere le caratteristiche dell'ambiente grafico su cui si lavora. Lo svantaggio

nell'uso della Canvas può essere proprio questo: lavorare a basso livello usando una serie di metodi basilari e poco sofisticati, che rendono ostica la creazione di oggetti grafici.

Non esistono pacchetti ufficiali con metodi per disegnare componenti grafiche basate su Canvas; le librerie amatoriali non dispongono di tutte le funzioni richieste per la progettazione di una nuova interfaccia utente portabile per Andiamo e la maggior parte di esse sono inefficienti e il loro sviluppo è stagnante.

Grazie all'opportunità di riscrivere la GUI di Andiamo, sarà possibile rilasciare alla comunità una liberia grafica innovativa: composta da nuovi menù, liste, radiobuttons ecc.. e una nuova gestione degli eventi basata su Canvas.

2.4 Una nuova architettura per Andiamo

La vecchia architettura di Andiamo prevedeva una serie di classi che estendevano il menù Form e altre che si basavano sulla lista List.

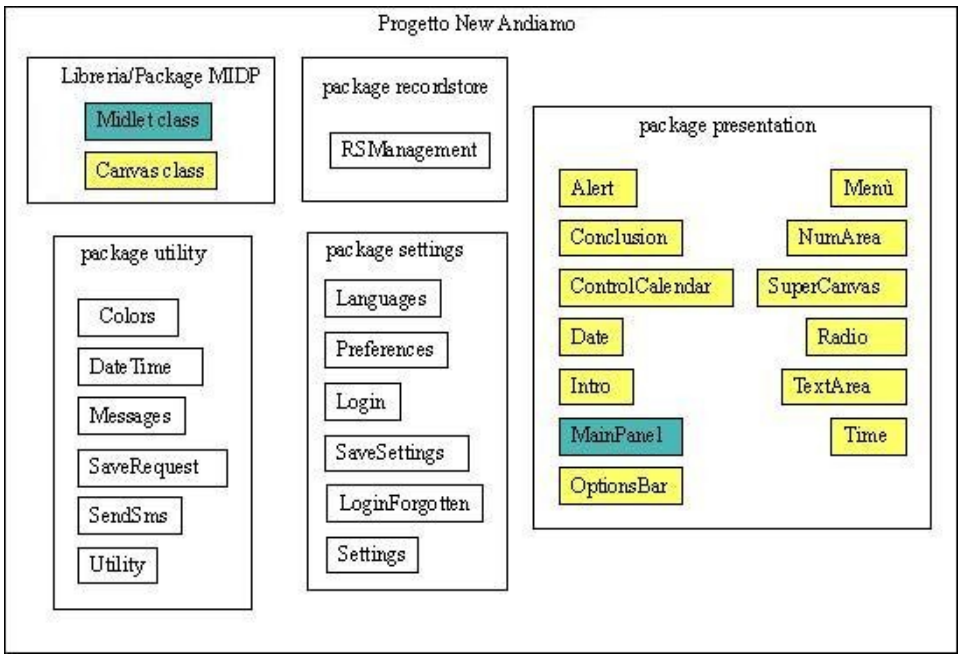
Per migliorare e rendere più efficiente il programma si possono unire le classi comuni e, in base alla posizione e alle scelte dell'utente durante l'esecuzione del programma, caricare le opportune icone, labels e opzioni.

Quindi, con questa nuova visione del software, si passa da 12 classi del vecchio programma che gestivano il menù (Languages, Login, LoginForgotten, Preferences, Arrival, Date, Departure, Option, Recap e ArchiveDetail, FeedbackDetail, MessageDetail) ad una sola. Le liste Messages, Feedback, ActiveRequests, Archive e Settings sono unite in un'unica classe.

Anche SendSms è un menù, ma verrà trattata diversamente trasformandola in una classe che gestisce solo dati.

MainScreen è usata nel vecchio codice come una sorta di classe gestionale, il cui scopo è solo quello di creare e distruggere i vari menù e liste, e renderle visibili a seconda del volere dell'utente. Questa classe non è più indispensabile, la classe Menù che si prospetta sarà "cosciente" della propria posizione nei vari livelli del programma. Essendo il menù l'oggetto grafico predominante di Andiamo (il menù in Andiamo è usato molteplici volte per permettere all'utente di navigare tra le miriadi opzioni del programma), sarà sua responsabilità richiamare opportunamente le liste.

Con queste premesse avremmo una classe menù centrale, una classe che gestisce liste secondaria e una serie di classi minori che conterranno i dati da caricare



Architettura del nuovo client Andiamo

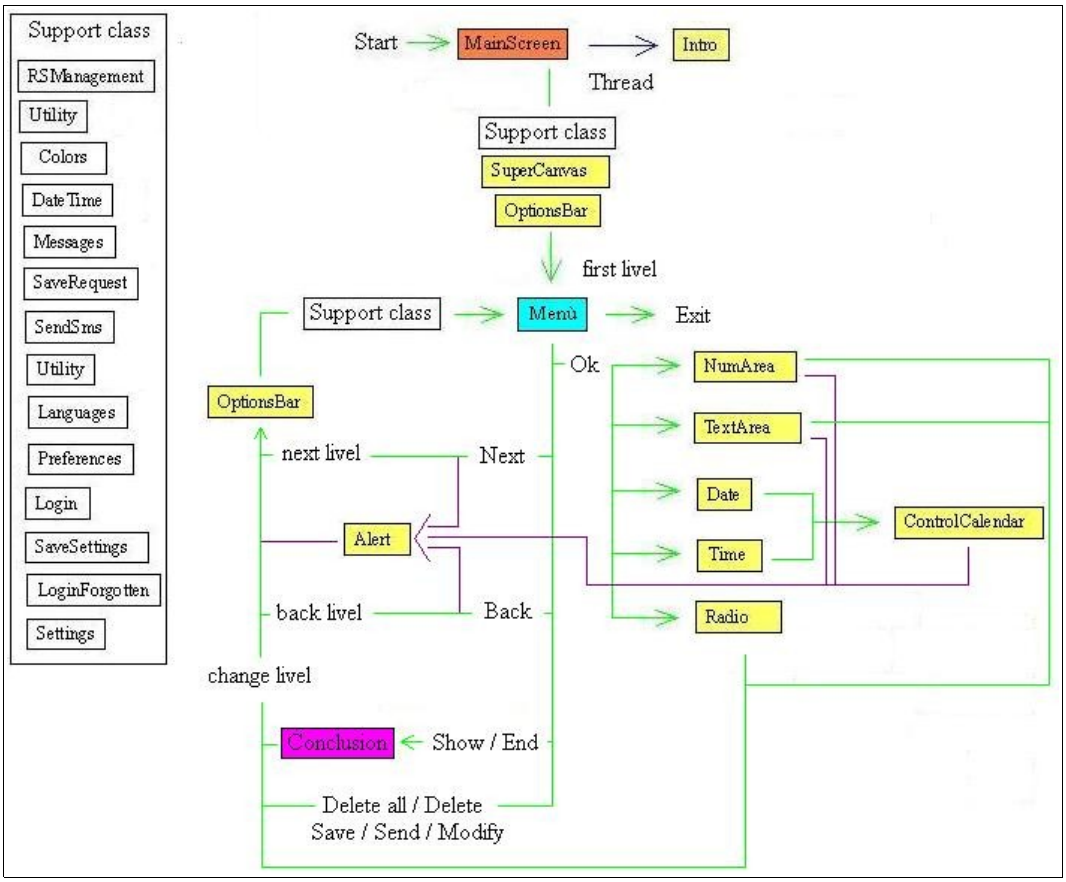
Figura 12

dinamicamente durante la navigazione dell'utente nel programma.

La figura 12 evidenzia la nuova architettura del sistema; come descritto prima, nel progetto è prevista la classe Menù, per i menù, e la classe Conclusion per le liste.
Come il vecchio Andiamo che prevedeva liste e menù strutturalmente identiche, anche nel nuovo software, con Menù e Conclusion, si rispetta questa scelta implementativa. Conclusion è un a particolare tipo di menù che accetta solo stringhe come oggetti da appendere nella sua struttura.
Insieme alle altre classi del package presentation, tranne MainPanel, tutte ereditano da Canvas. Queste classi ulteriori forniscono i dati necessari per creare i menù e le liste, ma integrano il programma con radiobuttons, alerts, campi di testo e opzioni di scelta di tempo e data ridisegnati per Andiamo con le Canvas.
La classe MainPanel è un'estensione di Midlet, indispensabile per avviare il programma. Intro è stata spostata nel package presentation, migliorandone la funzione a livello di codice. Le classi presenti in settings e la classe SendSms, non contribuiscono più alla grafica, il loro compito è stato rivisto e riassegnato: ora gestiscono solo i dati di login immessi dall'utente.
Nel package utility si troveranno tutti i tools per la comunicazione in e out, inoltre saranno disponibili ulteriori metodi per facilitare la gestione dei componenti grafici creati in presentation.
Nella figura 13 seguente si può vedere il nuovo funzionamento del client con le considerazioni appena fatte: una classe centrale Menù che gestisce l'intero sistema e i vari comandi forniti da OptionsBar che determinano l'aggiornamento del livello. Con le informazioni che si dispongono fino ad ora, si può accennare che

per ogni finestra esiste uno codice di livello univoco che le rappresenta; esso determina: la profondità dell'utente nella navigazione nel Menù (la posizione all'interno dei livelli) e quali oggetti grafici caricare in finestra(stringhe per le liste, alias Conclusion, o bottoni per i menù).

A differenza del client precedentemente descritto, le liste non sono costituite da stringhe o bottoni, ma da un elenco dettagliato formato da stringhe che riassumono le scelte effettuate dall'utente lungo tutto il percorso effettuato. La classe Conclusion, infatti, viene richiamata con "End" o "Show": due opzioni forniti all'utente dalla barra del menù (OptionsBar) che permettono di vedere, nel



Schema di funzionamento a classi del nuovo client

Figura 13

primo caso, tutte le scelte effettuate per l'offerta o la richiesta creata; nel secondo caso, l'utente può richiamare il servizio salvato in nella sezione Archivio o Attive e attingere alle informazioni memorizzate.

Support class raggruppa tutte quelle classi che forniscono metodi e dati per costruire la grafica e gestire la grafica, cioè tutte quelle variabili che compongono uno schema preciso di una determinata offerta o richiesta.

Nel capitolo seguente si affronterà la parte implementativa della progettazione del nuovo client Andiamo, seguendo lo schema in figura 12.

Si vedranno le soluzioni ideate ai problemi di portabilità o adattabilità grafica espresse nel capitolo precedente.

Capitolo 3

Sviluppo del client Adiamo

In questo capitolo si mostreranno le soluzioni implementative ai problemi emersi nel primo capitolo riguardanti la portabilità del client Andiamo.
I problemi di adattabilità grafica del software saranno eliminati seguendo lo schema progettato nel secondo capitolo.

3.1 Nuove componenti grafiche

Nel nuovo client basato su Canvas non esistono metodi che richiamano menù, liste o qualsiasi altro oggetto grafico. La programmazione a basso livello delle Canvas consente solamente l'uso di metodi fondamentali che permettono la creazione di oggetti grafici personalizzati.
I metodi forniti da Canvas, che sono il fulcro su cui si basa l'intero progetto sono:

- `setFullScreenMode(boolean visibility)`: funzione ereditata direttamente da Canvas che permette di usare la grafica di tutto lo schermo, i cui vincoli sono le vere dimensioni del display;
- `getClipX()`: funzione che restituisce l'offset, rispetto alle coordinate di origine del sistema dell'ambiente grafico, sull'asse delle ordinate;
- `getClipY()`: ha la stessa mansione di getClipX(), ma fornisce l'offset rispetto all'asse delle ascisse;
- `getClipHeight()`: questa funzione riporta l'altezza del contesto grafico in cui si opera;

- `getClipWidth()`: a differenza della precedente, ritorna un numero che restituisce la larghezza.

Gli ultimi tre metodi si basano su un contesto grafico fornito, come parametro, dall'interfaccia paint di Canvas vista precedentemente.

Queste funzioni che restituiscono le dimensioni display e le coordinate all'interno dell'ambiente grafico, sono il punto focale di tutto progetto; senza di esse non sarebbe possibile la creazione di oggetti dinamici che si adattino alle varie dimensioni del display.

In particolare l'altezza e larghezza del display sono le sole variabili su cui si appoggia l'intera implementazione del codice del progetto; il disegno di tutte le componenti grafiche gira intorno a queste due grandezze.

Il parametro fornito da paint, oltre alle funzioni sopra elencate, fornisce i metodi basilari per la creazione di qualsiasi oggetto grafico:

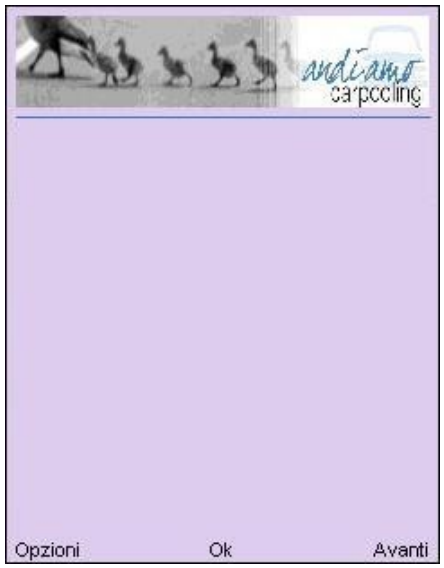
- `setColor(int R, int G, int B)`: con i tre colori fondamentali si può impostare qualsiasi tonalità.
- `draw....(...)`: il metodo draw permette di disegnare qualsiasi forma geometrica, stringa o immagine fornendogli come parametri le coordinate e i punteggi d'ancoraggio.
- `fill...(...)`: a differenza di draw, fill non può disegnare immagini, stringhe, o cornici ma solo forme geometriche piene.

Grazie a questi “3” metodi e a quelli descritti precedentemente, sarà possibile creare nuovi menù, liste, radiobuttons e qualsiasi altro oggetto grafico partendo da semplici funzioni.

3.2 Un rinnovato contesto grafico

Usando il metodo setFullScreenMode, è disponibile un ambiente di lavoro leggermente più grande. Nella vecchia versione di Andiamo, con le Midlet, esisteva una sorta di cornice del sistema operativo che restringeva l'area di lavoro su cui si poggiava la GUI.

Ora si può usare tutto lo schermo e creare una nuova cornice che supporti il programma; la possibilità di sceglierla è facoltativa, ma in Andiamo è stata usata e divisa in due parti.



Nuovo contesto grafico con cornice e barra dei menù

Figura 14

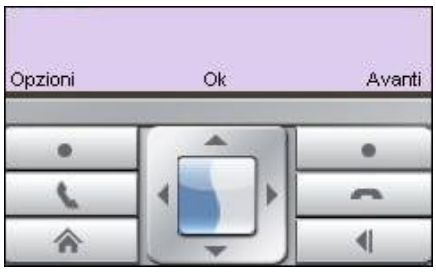
Come si può notare dalla figura, l'ambiente di lavoro è stato ridimensionato dalla presenza della cornice: ospita sopra un'immagine e sotto la barra del menù.

La classe che restituisce il nuovo ambiente grafico è SuperCanvas; essa viene chiamata una sola volta dalla classe Menù all'avvio dal suo costruttore, cioè nel

momento in cui MainPanel istanzia la classe Menù.

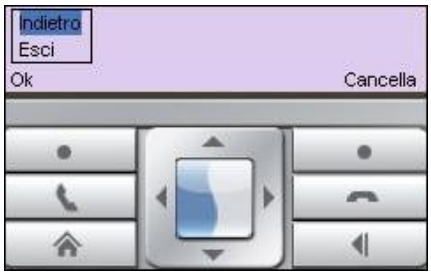
L'utilità di chiamare SuperCanvas una singola volta risiede nell'efficienza. La cornice appena creata, tralasciando la barra dei menù, e divenuta una oggetto statico della grafica; tutte le finestre del nuovo Andiamo avranno la stessa cornice. Grazie alla staticità della cornice, il refresh della finestra dovrà interessare solo gli elementi contenuti all'interno del nuovo contesto grafico; questo migliorerà notevolmente le prestazioni del programma rendendolo più veloce.

La barra dei menù è gestita dalla classe OptionsBar, il cui compito è ridisegnare le opzioni di scelta a seconda della posizione dell'utente nel menù. La questione della posizione sarà analizzata successivamente nel paragrafo riguardante il Menù. Nell'esempio sopra riportato si possono notare solo tre opzioni; questo rispecchia la strategia di usabilità dei cellulari, i quali forniscono tre tasti: due laterali e uno centrale per scorrere i menù. La classe OptionsBar fornirà una particolare opzione: Options od Opzioni, in base alla lingua impostata nel programma (al momento il software Andiamo prevede solo due lingue: inglese e italiano), per accedere alle scelte ulteriori previste dalla finestra.



Barra dei menù con tre opzioni

Figura 15



Opzioni avanzate del menù

Figura 16

Dalla figura 15 si può notare che l'uso di tre tasti non comporta una restrizione nelle opzioni. Come in qualsiasi sistema operativo per cellulare o simili, la

gestione di più di tre opzioni per finestra è affidata a un menù a tendina come si può notare nella figura 16. La scelta viene effettuata dall'utente scorrendo con le frecce direzionali up e down e selezionando con il tasto di scelta sinistro.

L'immagine che si nota in figura 14 è stata ridimensionata grazie ad una funzione, progettata e realizzata *ad hoc* per Andiamo, che la adatta alle dimensioni dello schermo. Non solo la risoluzione viene modificata diminuendo la dimensione dell'immagine, ma in contesti dove lo schermo è più grande delle dimensioni dell'immagine originale, questa viene ingrandita. Questo permette la portabilità su qualsiasi display.

Con questa tecnica si è risolto anche il problema grafico riscontrato nel primo capitolo negli esempi 1, 2 e 3. Le immagini all'avvio non si adattavano allo schermo e potevano causare problemi ben più gravi. L'errore dovuto a un errato refresh è stato superato effettuando una pulitura completa a tutto schermo.

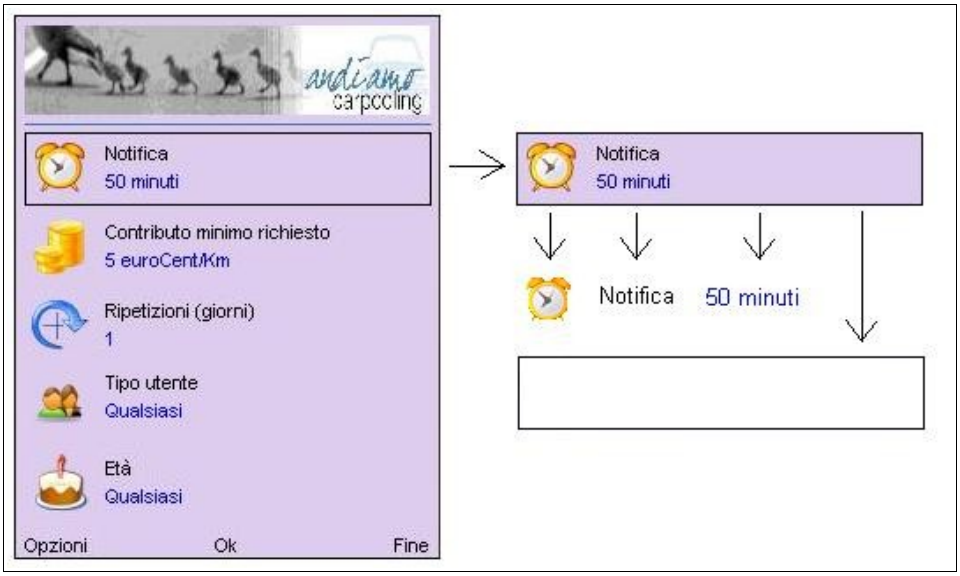
3.3 Il menù

Nei paragrafi precedenti si è accennato ad un nuovo menù che non risente dei cambiamenti dimensionali dei vari display e ad un modello a livelli che risolverebbe il problema della posizione dell'utente durante la sua navigazione.

Prima di affrontare il tema dei livelli è necessario approfondire lo schema del menù riprogettato. Esso non è una struttura unica, ma è composto da una serie di bottoni che insieme formano l'intero menù; ognuno di loro è a sua volta composto da più piccoli elementi implementati con i metodi base forniti dalla classe Canvas.

Sotto è riportato lo schema costruttivo dei vari elementi grafici che, assemblati

opportunamente fra loro, hanno portato alla realizzazione del nuovo menù.



Schema che riassume la creazione dell'oggetto menù partendo da oggetti basi come rettangoli, stringhe e immagini, passando dall'oggetto bottone

Figura 17

3.3.1 L'oggetto grafico bottone

Gli elementi che compongono il bottone sono semplici immagini, stringhe e forme geometriche.

Le due stringhe presenti in figura rappresentano un'etichetta o label associata al bottone. Esse hanno due ruoli diversi: la prima descrive il nome del bottone e suggerisce l'opzione di scelta che l'utente può compiere (*stringa descrittiva*), la seconda stringa, colorata di blu, identifica la scelta effettuata (*stringa di scelta*).

I due tipi di stringhe sono conservate in molteplici copie in un unico file di testo. Ogni copia identifica un diverso vocabolario per ogni lingua prevista dalla classe Languages e una funzione di lettura presente in RSManagement pescherà le

stringhe indicizzate nel file realizzando i due gruppi di stringhe: descrittive e di scelta.

In alcuni display la lunghezza delle stringhe potrebbe non essere supportata; per questa ragione l'oggetto bottone, prima di stamparle, calcola lo spazio rimanente tra l'icona e la fine dell'ambiente grafico.

La stringa o le stringhe saranno troncate lasciando in fondo i classici puntini di sospensione.

Soluzione adottata dall'oggetto bottone

Figura 18

La soluzione qui adottata ovvia il problema sollevato nel primo capitolo nell'esempio numero 6 e rende l'oggetto bottone e in generale il menù adattabile a qualsiasi contesto.

I bottoni che prevedono una scelta da parte dell'utente sono sempre strutturati con una doppia stringa; alla base di ogni scelta ci può essere una struttura a Radiobuttons (classe Radio) o un campo di testo (classe TextArea, NumArea, Date o Time).

Le distanze fra i bottoni nel menù e la posizione degli elementi che compongono gli stessi, sono valori statici impostati nella classe Menù dal programmatore; se, a causa della ristrettezza dello schermo, questi dati compromettessero l'adattabilità grafica del menù, saranno automaticamente ridotti.

Lo stesso ragionamento vale anche per le icone: nell'eventualità che il contesto grafico sia troppo ridotto per contenerle, verranno ridimensionate.

I quattro elementi che compongono il bottone sono creati grazie ai seguenti metodi:

- `drawImage(Image img, int x, int y, int ancor)`: che permette di disegnare un immagine date le coordinate e il punto d'ancoraggio;
- `drawString(String str, int x, int y, int ancor)`: con questo metodo è possibile scrivere stringhe sullo schermo;
- `drawRect(int x, int y, int width, int height)`: la funzione disegna il perimetro di un rettangolo nelle coordinate x e y previste, definendo la lunghezza e l'altezza desiderate.

Naturalmente è possibile disegnare anche rettangoli pieni con l'analoga funzione fillRect() e con il metodo setColor().

Il punto d'ancoraggio o di riferimento di un bottone così creato è in alto a sinistra, la sua lunghezza sarà data dalla dimensione del contesto grafico e l'altezza sarà la somma di varie componenti: l'altezza dell'icona, la somma dei due margini che dividono l'icona dal rettangolo e due volte lo spessore del perimetro del rettangolo.

Ora si conosce la grandezza di un bottone, che non deve essere uguale agli altri; infatti il menù prevede che le icone siano di dimensione diversa.

Un semplice ciclo che controlla lo spazio rimanente tra i bottoni esistenti e la fine del contesto grafico è sufficiente per creare il menù. Ovviamente è possibile che non tutti i bottoni siano disegnati per mancanza di posto. Per questa ragione nella classe Menù sono previste delle variabili che salvano la posizione del menù

selezionato (position), il numero di bottoni presenti (number_buttons) e un offset che tiene conto di quanto ci si è spostati dal primo bottone della finestra (offset_button).

Per creare l'oggetto bottone è prevista una struttura dati, presente nella classe Utility, la quale memorizza le immagini create e le stringhe ad essa associate per ogni finestra del menù[15]:

- `Hashtable images`, contenete per valore le icone e come chiavi di ricerca le *stringhe descrittive*;
- `Vector names`, anch'esso contenente le *stringhe descrittive*.

Le *stringhe di scelta*, al contrario, sono salvate in variabili contenute nella stessa classe Utility (una variabile per ogni *stringa di scelta*) e inizializzate all'avvio della classe Menù. Una funzione appositamente studiata riconoscerà i bottoni (con l'ausilio del vettore names) che necessitano della *stringa di scelta*.

In realtà non esistono più strutture dati, ma un'unica Hashtable images e un solo Vector names che, ad ogni cambio di finestra del menù, vengono svuotati e riempiti nuovamente con nuove icone e stringhe.

Una struttura a livelli dirigerà le operazioni di svuotamento di images e names e il loro corretto riempimento.

3.3.2 La struttura a livelli

La variabile livel nella classe Menù ha il compito di conservare la posizione

15 Ogni menù può essere composto da differenti icone e stringhe; un particolare disposizione di icone e stringhe nei vari bottoni è chiamata finestra. Ogni finestra ha una disposizione univoca di icone e stringhe, che però fa parte di un insieme chiamato finestre del menù

dell'utente all'interno delle varie finestre del menù. Ogni finestra sarà descritta da un particolare codice identificativo riconosciuto dall'oggetto grafico bottone, il quale ha il compito effettivo di caricare le informazioni da images e names per stampare a video la corretta finestra. Ad ogni cambiamento di finestra, causato dall'utente, la variabile sarà modificata.

```
public static void setMenu(String livel){
if(images == null)
    images = new Hashtable();

if(names == null)
    names = new Vector();

    switch(Integer.valueOf(livel).intValue()){
        case(1):{
            names.removeAllElements();
            images.clear();

            images.put(settings.Language.REQUESTS, createImage("/resource/request.png"));
            images.put(settings.Language.SETTINGS, createImage("/resource/settings.png"));

            images.put(settings.Language.MESSAGES, createImage("/resource/messages_no.png"));

            images.put(settings.Language.FEEDBACK, createImage("/resource/feedback.png"));

            names.addElement(settings.Language.REQUESTS);
            names.addElement(settings.Language.SETTINGS);
            names.addElement(settings.Language.MESSAGES);
            names.addElement(settings.Language.FEEDBACK);

            break;
        }
        .
        .
        .
        .
        case(11):{
            names.removeAllElements();
            images.clear();

            images.put(settings.Language.ACTIVE, createImage("/resource/active.png"));
            images.put(settings.Language.DB, createImage("/resource/archive.png"));
            images.put(settings.Language.NEWREQUEST, createImage("/resource/newRequest.png"));
            images.put(settings.Language.VERIFY, createImage("/resource/verifyOffers.png"));

            names.addElement(settings.Language.ACTIVE);
            names.addElement(settings.Language.DB);
            names.addElement(settings.Language.NEWREQUEST);
            names.addElement(settings.Language.VERIFY);
         break;
        }
        .
        .
        .
```

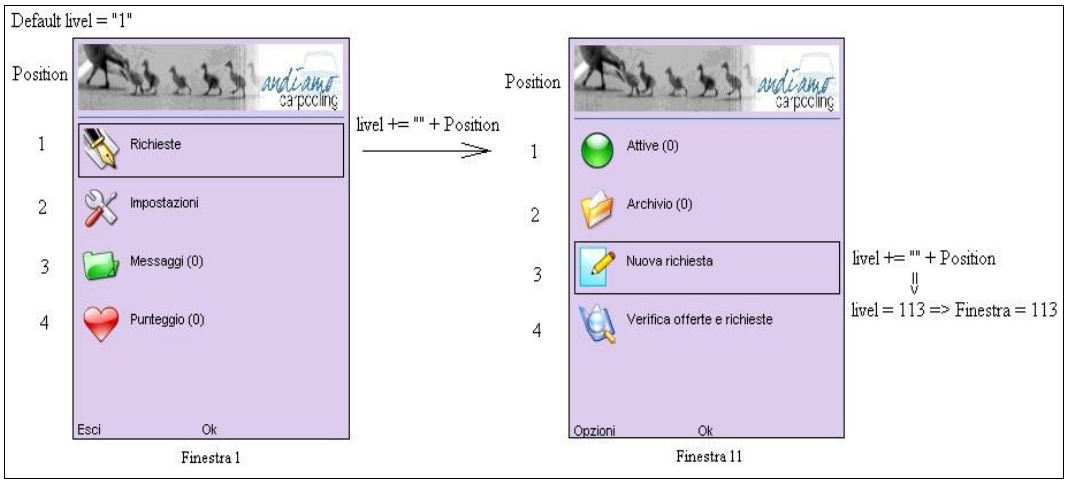
Codice della struttura a livelli

Figura 19

Livel è una stringa di tipo String utilizzata nelle fasi di modifica, ma per settare correttamente images e names viene trasformata in un intero per essere gestita al meglio nella funzione `setMenu()` di Utility.

Come si può vedere nella figura 19, grazie a livel e alla funzione setMenu, images e names contengono le immagini e le *stringhe delle descrizioni* opportune. Questo sistema (a struttura di livelli) vale per qualsiasi finestra del menù.

Nella figura 20 si può osservare un esempio pratico della navigazione a livelli attraverso le varie finestre del menù. Nell'immagine in questione l'utente passa dalla finestra di default "1" alla successiva, in questo caso "11".



Cambiamento di finestra grazie alla struttura a livelli

Figura 20

Prima si è ipotizzato che, a causa dell'elevato numero di bottoni in una finestra, non tutti venissero caricati a video. Il codice visto in precedenza carica tutti i dati nella struttura di memoria, quindi la soluzione non va ricerca in quel codice.

Lo stesso ciclo che crea i bottoni in una data finestra, prevede la possibilità che non tutti stiano nel display.

Grazie a position, offset_button e number_buttons si possono caricare

all'occorrenza i bottoni mancanti.

Se l'utente (figura 21) decide di visionare il bottone successivo all'ultimo visibile, attualmente selezionato, la classe Menù eseguirà le istruzioni di seguito riportate.

Controllerà se esistono altri bottoni oltre quello selezionato confrontando position e offset_buttons con images o names. In caso di esito positivo, offset_buttons verrà incrementato e il ciclo che crea l'elenco di bottoni, basandosi su images e names, inizierà da offset_buttons le sue operazioni.

Prima di disegnare sul display il nuovo elenco, una refresh dell'ambiente grafico eviterà sovrapposizioni tra i vecchi e i nuovi elementi grafici.

Corretta gestione dei bottoni del menù

Figura 21

Questa tecnica risolve i problemi del primo capitolo lasciati in sospeso; infatti il refresh dei bottoni, unito ad un corretto uso dell'ambiente grafico, evita le disfunzioni degli esempi 3 e 4.

Tutti gli interventi approntati ad Andiamo per rimediare alla scarsa portabilità

grafica del software, sono stati descritti e argomentati in tutti e tre capitoli.

Conclusioni e sviluppi futuri

L'obiettivo di una nuova interfaccia grafica completamente portabile e adattabile che non compromettesse usabilità del client Andiamo – Car pooling è stato raggiunto. A riguardo sono stati forniti esempi e spiegazioni seguendo una percorso che ha portato il lettore a comprendere i benefici del software e in particolare della nuova libreria grafica creata.

Il vantaggio di usare la nuova versione di Andiamo sta appunto nella rinnovata portabilità del client. L'adattabilità è stata notevolmente migliorata creando un'interfaccia grafica innovativa e adattabile a qualsiasi contesto. Le dimensioni del display non sono più una limitazione del software, grazie alle componenti grafiche progettate e sviluppate nell'elaborato.

Come si evince dalla nuova architettura del programma, Andiamo ha in sé un package isolato che implementa tutta la grafica; esso può essere ripreso e riusato in altri progetti che richiedono un'elevata portabilità grafica.

Il nuovo client Andiamo è stato testato su parecchi modelli di cellulari, smartphone e pda di varie marche; i risultati ottenuti coincidono con la tesi sostenuta nell'elaborato, ma per alcuni dispositivi i test hanno riportato diversi comportamenti del nuovo software.

I modelli di cellulari che montano una versione di Java Micro Edition con una versione del MIDP inferiore alla 2.0 non permettono l'uso della funzione setFullScreenMode. Questa funzione, che permette l'uso di tutto l'ambiente grafico, non è implementata; il nuovo software sarà condizionato, come con le Midlet, dai limiti imposti dal sistema operativo: oltre alla cornice creata

appositamente per Andiamo apparirà quella di default.

Nei dispositivi con Windows Mobile installato, il sistema delle licenze adottato dal sistema operativo blocca il programma.

Le licenze sono dei certificati a pagamento che permettono l'uso delle risorse del sistema, in questo caso l'uso delle Canvas è precluso.

I software privi di certificati non possono girare su tali dispositivi, ma altri sitemi come Symbian e Linux forniscono opzioni di settaggio per modificare la sicurezza di risorse minori.

In ogni caso i test non si concludono con la stesura di questo lavoro; le prestazioni del prototipo saranno migliorate e nuovi versioni di Andiamo con componenti grafiche aggiunte potranno trasformare questo software in un vero programma distribuibile.

Fra qualche anno Andiamo sarà una realtà nel mercato dei software per dispositivi mobili e potrà contribuire anch'esso nel migliorare la viabilità stradale.

Bibliografia

[1]: M. S. Gast, 802.11 Wireless Networks, First Edition, O'Reilly, 2002

[2]: Sito web ufficiale della tecnologia bluetooth. www.bluetooth.com

[3]: Articolo della Sun riguardante Java e la sua portabilità. www.sun.com/smi/Press/sunflash/2004-02/

[4]: L. Cantoni, N. Di Blas, D. Bolchini, Comunicazione, Qualità, Usabilità, Apogeo, Milano, 2003

[5]: Modelli di business e scelte strategiche riguardanti l'usabilità e l'ergonomia. www.usabile.it/222003.htm

[6]: Luca Spinelli, Ergonomia, usabilità, accessibilità, in «Login», Gruppo Editoriale Infomedia, Pisa, 2006

[7]: Usabilità e il suo funzionamento. http://www.usabile.it/012000.htm

[8]: Discussione nel forum ufficiale della Nokia. http://discussion.forum.nokia.com/forum/showthread.php?t=83759

[9]: Sito ufficiale delle specifiche tecniche di J2ME. http://java2me.org/articoli/arhitetturaJ2ME.html

[10]: Articolo che argomenta le strette relazioni tra standard e monopoli http://www.portel.it/articoli/editoriale/11-2002/se-nokia-ha-ragione.html

[11]: Pagine web della Sun per gli sviluppatori di interfacce grafiche in j2ME. http://developers.sun.com/mobility/midp/articles/ui/