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

Corso di Laurea triennale in Informatica

Elaborato Finale

PROGETTAZIONE E SVILUPPO DI UN SISTEMA DI RICARICA DI CARTE PREPAGATE PER DISPOSITIVI MOBILI

Relatore: Paolo GIORGINI Laureando: Dario DEL SANTE

Anno Accademico 2006 – 2007

INDICE

INTRODUZIONE

La tecnologia GSM è attualmente lo standard di telefonia mobile più diffuso al mondo. Le sue caratteristiche le hanno permesso in circa 10 anni di soppiantare totalmente quella precedente, la TACS; in modo particolare il canale di comunicazione che adesso è digitale consente lo scambio di dati (commutazione a pacchetto) oltre alle semplici comunicazioni. Vi è dunque da parte degli utenti la possibilità di accedere ad una nuova serie di servizi a costi contenuti e la diffusione di dispositivi mobile sempre più ricchi di caratteristiche (dalla multimedialità alla connettività) ha accentuato le esigenze degli stessi utenti di volerne usufruire.

In particolare il servizio SMS l'ha fatta da padrone, forte della sua semplicità e del costo relativamente basso. Da diversi anni è utilizzato non solo per comunicare ma anche per scopi commerciali di aziende o degli stessi operatori telefonici e recentemente è diventato un mezzo per offerte di contributi in maratone benefiche e iniziative sociali.

Soltanto da poco tempo però l'utente "mobile" è stato preso in considerazione anche da pubbliche amministrazioni o enti per offrirgli direttamente servizi personalizzati: informazioni su traffico, richieste di estratti conto dalla propria banca, ecc… .

Considerando ad esempio l'insieme di servizi messi a disposizione dal Comune di Bolzano ai cittadini, si è scelto di estendere quello relativo alla ricarica di una School Card per la ristorazione. In questo caso applicando proprio la tecnologia SMS si permetterebbe agli utenti di ricaricare il proprio badge semplicemente inviando un messaggio di richiesta dal proprio cellulare alla banca. Questo consentirebbe di non doversi recare allo sportello bancomat personalmente come avviene attualmente. Partendo quindi dall'analisi di questo semplice sistema di ricarica per studenti, si è cercato con questo lavoro di provvedere ad estendere tale servizio a altri tipi di carte prepagate, quali ad esempio PayPal e quelle telefoniche.

L'obiettivo è dunque quello di studiare un sistema di ricarica di carte prepagate per dispositivi mobili che consenta all'utente di inoltrare una richiesta alla propria banca, la quale eseguirà l'ordine ricevuto e notificherà all'utente l'esito dell'operazione sul cellulare.

Quello che ci si aspettava di ottenere è da un lato un applicativo semplice da utilizzare e intuitivo che rispondesse alle esigenze di qualsiasi categoria d'utenza (giovani, anziani e tutti coloro che usano abitualmente un telefono cellulare, senza che debbano avere conoscenze particolari aggiuntive); dal lato back-end invece un sistema leggero e funzionale, che ascoltasse le richieste che arrivano, eseguisse l'ordine e rispondesse con un messaggio di notifica. Tutto questo senza dimenticare ovviamente il discorso sicurezza, prestando attenzione quindi all'affidabilità del servizio, alla sua robustezza e alla tutela dei dati sensibili.
Questa tesi si propone dunque di analizzare con il Capitolo 1 le motivazioni per le quali un tale sistema di ricarica possa essere sviluppato argomentando le scelte fatte per l'elaborazione della soluzione ideata.
Nel Capitolo 2 si studierà il problema di partenza per elaborarne una soluzione adatta e allo stesso tempo flessibile, in modo che possa eventualmente essere applicata ad altre tipologie di problema, senza radicali modifiche della struttura.
Sarà poi proposto nel Capitolo 3 un prototipo di soluzione che ne rispetti i requisiti, che ne mostri le scelte strutturali e logiche e che sia modulabile per possibili aggiornamenti futuri (Capitolo 4).

1. CAPITOLO 1: Tecnologia mobile e pubbliche amministrazioni

Questa sezione si pone principalmente di introdurre la situazione della tecnologia mobile nei suoi vari aspetti e applicazioni.
Si pone inoltre l'obiettivo di delineare il rapporto del cittadino e delle pubbliche amministrazioni con queste tecnologie per poter affrontare la questione da risolvere in maniera più esaustiva e corretta.

1.1 La telefonia cellulare

L'accesso privato alla Rete Telefonica Generale per mezzo di ponti radio posti sulla superficie terrestre e dispositivi mobili utilizzabili in ampie aree geografiche è chiamata telefonia cellulare. Essa, come la telefonia satellitare (l'altra tipologia di telefonia mobile), si contrappone alla telefonia fissa che serve zone geografiche limitate.
Il nome cellulare deriva dalla porzione di territorio servita da una Stazione Radio Base, chiamata appunto *cella*. Le Stazioni Radio Base ricevono i segnali dai dispositivi cellulari e li inoltrano attraverso antenne direttive, agendo all'occorrenza attraverso più stazioni, all'utente destinatario della comunicazione.
La prima generazione di rete per telefoni cellulari fu il Total Access Communications System (TACS), introdotto in Italia verso la fine degli anni '80; si basava su tecnologia analogica ed ebbe il merito di diffondere la telefonia mobile nel nostro paese, ma lo svantaggio di essere vulnerabile: i terminali potevano essere semplicemente clonati.
Le succedette la tecnologia GSM (Global System for Mobile Communication), l'attuale standard di telefonia mobile, con le sue 2 miliardi e più di utenze in 200 paesi. Essa è una tecnologia digitale, la quale permette l'accesso a tipologie di servizi differenti come SMS (Short Message Service) e GPRS (General Packet Radio Service) per la comunicazione di dati.

La terza generazione, ovvero UMTS, Universal Mobile Telecommunications System, è la tecnologia che sta prendendo piede negli ultimi anni ed estende le potenzialità (e quindi i servizi) della precedente GSM: voce, dati a pacchetto , videoconferenze e internet ad alta velocità.
Nel 1999 la telefonia mobile superò quella fissa e oggi solo in Italia ci sono circa 43 milioni di apparecchi, quasi uno per individuo.

1.2 Il cittadino "mobile"

La corsa alla miniaturizzazione ha premesso una quantità e una qualità di tecnologia all'interno dei device solo qualche tempo fa impensabile e ci ritroviamo oggi con dispositivi di ridotte dimensioni che svolgono quasi totalmente le varie funzioni dei comuni personal computer: internet, video, musica, editing di testi e molto altro.
Alla base di queste apparecchiature oggi come oggi ci sono veri e propri sistemi operativi dedicati, pensati per gestire e ottimizzare le risorse sempre meno limitate, ma comunque non paragonabili a quelle dei calcolatori.
Le aziende che governano tale mercato sono sempre più numerose e sempre più impegnate a concorrere per l'innovazione da applicare ai propri dispositivi.
Altri tipi d'aziende hanno invece capito da tempo il vantaggio di sfruttare le potenzialità di queste tecnologie dal punto di vista di "offerta di servizi": dare la possibilità all'utente di avere degli "extra" a pagamento in qualsiasi momento ma soprattutto in qualsiasi luogo esso si trovi era un'occasione da non perdere. Così si spiega il proliferare di abbonamenti per informazioni delle news da parte degli operatori telefonici, piuttosto che suonerie settimanali o ancora sfondi per il display del proprio terminale. L'utente, che lo voglia o no, è "bombardato" da offerte di tutti i tipi ed è chiaro, visto il successo, che questo "attacco" funzioni in maniera soddisfacente.
Di tutta questa innovazione la pubblica amministrazione sembra però non essersene accorta, almeno fino a poco fa; non è da tanto tempo infatti che vengono offerti servizi basati su questa filosofia che affiancano i più

tradizionali metodi di comunicazione ai cittadini: notifiche per la scadenza di pagamenti, per il saldo di conti correnti, ecc…

Ovviamente per questo genere di operazione l'SMS è appunto la scelta più pratica: comunicazione immediata, semplicità di utilizzo e costo esiguo.

Le motivazioni per utilizzare le tecnologie mobile nell'ambito della pubblica amministrazione non solo non si limitano a questi aspetti: infatti l'utente non è l'unico a trarne vantaggio, bensì anche l'ente che offre il servizio. Innanzitutto tali servizi possono essere personalizzati in base alle esigenze dei singoli richiedenti: l'informazione è dunque mirata a soddisfare necessità precise e l'utente avrà così informazioni su misura, mentre chi la fornisce può così dedicarsi a informare coloro che sono effettivamente interessati ad esserlo. Un esempio: quando c'è la necessità di dare informazioni circa un particolare evento (supponiamo culturale), il comune è solito servirsi di mezzi di informazione di massa quali giornali, volantini, spazi pubblicitari ecc…, spendendo un enormità per dover aggiornare magari solo una piccola parte coinvolgendo dunque anche i non interessati; un servizio simile alle conosciute Newsletter potrebbe senza dubbio far risparmiare notevoli somme e informerebbe solamente chi vuole essere informato.

Un altro vantaggio non indifferente sta nel fatto che le informazioni, se fornite per esempio telefonicamente, possono essere dimenticate, ma se invece vengono recapitate su un dispositivo mobile, esse rimangono scritte e quindi sempre consultabili (sempre se ovviamente non vengono accidentalmente cancellate).

Soprattutto negli uffici pubblici una considerevole quantità del tempo degli impiegati è occupata appunto da varie telefonate per rispondere a richieste d'informazioni, prenotazioni di appuntamenti, ecc: anche in questo caso un servizio automatizzato che comunichi direttamente con i device dei cittadini non occuperebbe tempo agli impiegati stessi rendendoli liberi di proseguire con il loro lavoro, e soprattutto potrebbe soddisfare le richieste in maniera praticamente concorrente: un centralino composto da persone fisiche infatti può potenzialmente essere sempre occupato e un utente non troverebbe mai ascolto alle sue esigenze.

Ovviamente non tutti i servizi possono essere adeguati a queste nuove tecniche che presentano a loro volta purtroppo anche dei limiti: da un lato pratico ci sono per esempio sostanziali differenze tra i vari modelli di dispositivi che implicano diverse versioni di interfacce che devono quindi venire adattate ad hoc, oppure la dimensione del display che discrimina le applicazioni più esigenti, o ancora la tastiera che nella maggior parte dei casi è semplicemente alfanumerica e non QWERTY, il layout standard di tastiere, e complica così la fase di input; dal lato logico, un servizio studiato per tali sistemi deve essere pensato nel suo insieme partendo da un concetto di sicurezza e affidabilità molto delicato in quanto c'è la possibilità che vengano trasmessi dati sensibili che non devono essere "ascoltati", oppure che una richiesta si "perda" per qualche motivo nell'etere quando l'utente ne ha già pagato l'invio.
Applicativi mobile per soddisfare le esigenze della pubblica amministrazione o di un qualsiasi ente che voglia comunicare con il cittadino, o che voglia dare ad esso la possibilità di comunicare con loro in tale maniera deve essere dunque ben studiato e progettato.

1.3 Applicativi mobile: caratteristiche e problematiche

Un software che esegue su un dispositivo mobile necessita di particolari attenzioni. Come già accennato deve svolgere i suoi compiti sfruttando le risorse limitate del sistema su cui è installato, quali display, processore, memoria e sistema di input.
C'è la necessità quindi di semplificare ogni azione che dovrà eseguire l'utente, cercare di visualizzare informazioni relativamente complesse in maniera semplice e chiara in modo da non creare confusione. Per non dimenticare che deve essere assolutamente intuitivo perché deve essere utilizzato da tutte le tipologie di utenza, anche dai meno "aggiornati" tecnologicamente. Il telefono cellulare che ha fatto dell'usabilità la sua carta vincente non può quindi avere al suo interno un applicativo complesso e verboso, che magari richieda conoscenze specifiche per essere utilizzato: le fasi "critiche" devono essere trasparenti a chi lo usa che non si deve perciò

accorgere di quello che sta succedendo; chi utilizza tali dispositivi vuole tutto e subito ottenendo quello che vuole nella maniera più semplice e veloce, senza che sia costretto ad "imparare" troppe cose: non si può pretendere per esempio che, con la tastiera dei telefoni standard, egli debba inserire diverse volte dei parametri anche lunghi e complessi, anche perché un errore in questo caso farebbe perdere tempo e pazienza. Oppure che una risposta ricevuta in seguito ad una richiesta effettuata sia troppo lunga, o ancora, che sia necessario cercare la voce desiderata attraverso un menu difficile e intricato.

In sostanza l'utente non deve ragionare per districarsi all'interno dell'applicativo, deve raggiungere l'opzione desiderata immediatamente. E soprattutto deve essere sicuro che tutto è andato a buon fine. Quindi, prendendo per esempio in considerazione una qualsiasi richiesta inviata, bisogna ottenerne immediatamente l'esito, o una notifica che lo avverta che la sua richiesta sarà servita.

1.4 Ambiti di utilizzo

I settori in cui può essere utile avere un applicativo mobile che permetta al cittadino di accedere a un insieme di servizi pubblici o privati sono molteplici. L'importante è considerare soprattutto questo aspetto: può l'utente sentire effettivamente la necessità di usufruire di un determinato servizio quando si trova in giro?

Bisogna perciò scoprire qual è la necessità che lo può spingere a utilizzare il cellulare per quello scopo piuttosto che aspettare e magari effettuare le proprie richieste più comodamente da casa via web.

Il concetto di necessità di accedere ai servizi deve essere quindi tenuto in considerazione almeno quanto quello relativo alla possibilità di usufruirne.

Ad esempio ci sono già opzioni precaricate nelle carte SIM (Subscriber's Identity Module) di ogni operatore che danno la possibilità di poter ricevere informazioni sugli avvenimenti politici piuttosto che sportivi o economici, ma non implica che ci sia l'effettiva necessità di utilizzarle. Ovviamente questi sono degli "extra" che gli operatori propongono all'utenza, forti del

fatto che un servizio loro già lo offrono, ovvero la disposizione della loro rete. Mettere in piedi invece un sistema ad hoc esclusivamente per una o più tipologie di servizio è un altro discorso: non c'è la garanzia di un feedback positivo da parte dell'utenza: può essere un successo o un fiasco totale, e in questo caso significa aver perso tempo e risorse per poterlo avviare.
Siccome, tecnologicamente parlando, oggi come oggi è quasi tutto possibile si deve perciò individuare quel servizio che risponda alla domanda posta in precedenza in maniera positiva.
Per trovare, nell'ambito della pubblica amministrazione, una valida risposta bisogna innanzitutto selezionare nel ventaglio dei servizi da essa offerti quelli che potrebbero inizialmente essere "informatizzati" (o che magari già lo sono).
L'elenco in questo caso risulta lungo in quando qualsiasi ufficio fornisce oggigiorno molte informazioni in maniera digitale, dà la disponibilità di scaricare moduli di qualsiasi genere, offre servizi di prenotazioni agli appuntamenti e molto altro ancora. Tra tutte queste possibilità bisognerà dunque individuare quella (o quelle) che offra significativi vantaggi e comodità se implementata anche per dispositivi mobile.
Un sistema di prenotazione on-line per appuntamenti presso un ufficio ad esempio non sarebbe ideale perchè l'utente dovrebbe "compilare" troppi campi, sarebbe necessario un display molto più ampio per rendere il servizio usufruibile in maniera chiara e magari dovrebbe sfruttare un qualche tipo di connessione come può essere il GPRS, servizio ancora troppo costoso per l'utente medio perché lo spinga a utilizzarlo per un motivo del genere. Egli in questa eventualità telefonerebbe per avere un appuntamento o effettuerebbe la prenotazione da casa usando comodamente internet.
Bisogna quindi puntare su qualcosa di molto più veloce, immediato: poche operazioni per ottenere subito ciò che si vuole, semplice magari quanto sia l'invio di un messaggio all'operatore telefonico tramite l'apposita opzione nel telefono per essere aggiornati sul proprio credito residuo.

1.5 School Card – Sistema di pagamento elettronico

Il Comune di Bolzano mette a disposizione al cittadino molti servizi, siano essi accessibili da Internet, siano essi più "tradizionali".

Tra questi, la possibilità per gli studenti di iscriversi a un servizio di ristorazione per il quale il Comune stesso fornisce un badge elettronico, nominativo e non cedibile, spedito direttamente a casa.

Attualmente questa card, strettamente personale, può essere ricaricata presso le banche affiliate oppure presso gli sportelli bancomat abilitati e questo può essere un limite se egli si trova in procinto di pagare e scopre di essere al verde.

Da questa problematica nasce quindi l'idea di modernizzare tale processo: studiare cioè un sistema che dia la possibilità allo studente di poter ricaricare la prepagata usando un applicativo installato sul proprio telefono, in modo che egli non debba recarsi nei siti idonei alla ricarica e che lo possa fare in qualsiasi momento, magari proprio all'ultimo momento; un software che gli permetta velocemente di inoltrare una richiesta alla banca che immediatamente esegua l'ordine e notifichi l'avvenuta transazione all'utente stesso.

In questo modo l'utente è totalmente svincolato dagli orari delle banche o dalla ricerca del bancomat abilitato per compiere l'operazione.

Una soluzione del genere però può tornare utile anche per altri tipi di utente e non solo per poter ricaricare il badge dedicato alla ristorazione. Si è pensato quindi di generalizzare il discorso, estendendo la proposta a tutte le tipologie di utenza e alle altre carte prepagate, PayPal (lo strumento di pagamento utilizzato nell'e-commerce) e telefoniche.

Ogni cittadino quindi potrà ricaricare la carta desiderata solamente usufruendo dell'applicativo installato sul proprio telefono cellulare, senza dover ricorrere ai metodi più tradizionali.

Essendo i sistemi bancari funzionanti 24 ore al giorno, una richiesta di ricarica della tessera telefonica può essere inoltrata in questo modo anche a tarda notte, quando ovviamente un'edicola che vende le carte valore degli operatori sarà sicuramente chiusa; un anziano che non possiede internet

può quindi evitare di uscire appositamente è sarà per lui sicuramente una comodità non indifferente.
Sono tanti i vantaggi di poter effettuare tali operazioni senza doversi muovere e in qualsiasi momento, soprattutto in caso di necessità. La possibilità di essere svincolati da orari e distanze è oggigiorno sempre più importante e utilità del genere non possono che agevolare la vita quotidiana.
Un obiettivo di un tale servizio è che possa essere utilizzato da più persone possibili, giovani e meno giovani, esperti di nuove tecnologie e neofiti.
Bisogna quindi cercare di far convergere tutto il sistema in un concetto molto semplice e conosciuto, ambendo a non far sì che il cittadino desista dall'utilizzare una novità tale come spesso accade con le nuove tecnologie, dove la pigrizia nell'aprire la mente e lo scetticismo vincono.

1.6 Software mobile per interfacciare l'utente: la MIDlet

Il problema di come far comunicare il cittadino con la banca tramite il cellulare senza effettuare una telefonata è uno dei punti cardini di questa tesi.
La scelta praticamente obbligata è ricaduta per l'appunto su un già citato applicativo mobile, in particolare una MIDlet basata su specifiche J2ME.
J2ME (Java 2 Micro Edition) deriva da Java 2 Standard Edition J2SE, ottimizzato e alleggerito per eseguire su cellulari e PDA (palmari) di ultima generazione. Si affianca quindi a J2EE - Java 2 Enterprise Edition, dedicato alle applicazioni server.
Un dispositivo abilitato J2ME consente di scaricare un'applicazione di pochi kbyte (alcuni terminali permettono ad esempio applicazioni di max 10KB ciascuna, e 50 applicazioni totali sul terminale), la MIDlet appunto: tali applicazioni vengono installate velocemente sul device e possono essere utilizzate anche off-line.
Oltre che per le limitazioni di memoria e di potenza di elaborazione, J2ME deve adattarsi anche a display particolarmente ridotti di un piccolo telefono cellulare che può disporre di 12.288 pixel (96 ×128), oppure di un PDA di 20.000, anche se ultimamente l'evoluzione tecnologica in questo senso tende

ad abbattere queste limitazioni con schermi sempre più ampi e con sempre più colori.

Il tutto si basa su una Java Virtual Machine, presente nei terminali, che interpreta ed esegue l'applicazione. J2ME comprende due livelli di configurazione (CDC e CLDC) ed inoltre il livello "Profile" MIDP. L'idea che sta alla base è quella di partire da una piattaforma comune a tutti i dispositivi, con un set di API, Application Program(ming) Interface, limitato a fornire le funzionalità base per un qualunque device a limitata configurazione. Saranno poi i produttori di particolari categorie ad aggiungere funzionalità al CLDC, sotto forma di librerie, generando così quello che è un profilo.

Le due configurazioni invece sono legate a differenti scenari, e comunque sono in forte evoluzione:

La prima, Connected Device Configuration (CDC), pensata per dispositivi "*things that you plug into a wall*", cioè collegati in rete, possibilmente always on e ad alta banda, ad es. i palmari :

- 512 kB minimo di memoria per l'applicazione Java (codice)
- 256 kB minimo di memoria "runtime", allocata dall'applicazione

La seconda, Connected Limited Device Configuration (CLDC), ideata invece per dispositivi mobili, "*things that you hold in your hand*", caratterizzati da connettività wireless, ridotta banda e accesso discontinuo:

- 128 kB di memoria per Java
- 32 kB di memoria "runtime"
- interfaccia utente ridotta
- bassi consumi, alimentazione a batteria

Il profilo MIDP (Mobile Information Device Profile) riguarda principalmente terminali wireless, in grado di instaurare connessione basata sul protocollo http; è una estensione, una serie di librerie che aiutano a sviluppare applicazioni tramite API predefinite per interfacciarsi con il terminale (modalità di input, gestione degli eventi, memoria persistente, timer, interfaccia video...).

KVM è la Java Virtual Machine che viene installata sui terminali, caratterizzata da un ridotto fabbisogno di memoria (40/80 kB + 20/40 kB dinamica) e ridotte potenze di elaborazione (processori 16bit a 25MHz).
I vantaggi rispetto a modalità preesistenti come il Wap e Sim Toolkit, risiedono in tre punti principali:

- la possibilità di sfruttare le applicazioni in locale, residenti e funzionanti anche off-line
- la capacità elaborativa
- la flessibilità nell'installazione di nuove applicazioni

L'interfacciamento con le piattaforme web che lavorano con JAVA (Servlet o JSP) è in aggiunta pressoché immediato.
Sebbene si tratti di uno standard aperto, non mancano alcune peculiarità che possono, come già accennato, causare problemi di compatibilità tra terminali differenti, per il problema della creazione dei profili. Siemens, ad esempio, ha definito una serie di librerie proprietarie che risiedono solo sui propri terminali, e che permettono di controllare alcune funzionalità del telefono (ad esempio la vibrazione).
Oltre alle librerie proprietarie legate al terminale, vi è la disponibilità di differenti Virtual Machine (gli interpreti delle applicazioni Java), ad esempio:

- Aplix JBlend, diffuso in Giappone, non solo sui telefonini ma anche su PDA e videocamere (Sony, Sharp, Toshiba, Casio...)
- PersonalJava, legato alla piattaforma Symbian e appoggiato su CDC.

Le capacità multimediali sono attualmente limitate, infatti non è prevista ad esempio la possibilità di riprodurre file audio e, soprattutto per ragioni di sicurezza, di permettere ad applicazioni MIDlet J2ME l'accesso alle informazioni della SIM o dei parametri di configurazione del terminale.
Ci sono inoltre due possibilità di far fronte ad applicazioni che richiedono l'appoggio di uno o più database: utilizzare il terminale come semplice interfaccia client (quindi come un browser) verso un sistema server remoto dove risiedono i dati; uno dei vantaggi, in questo caso, è il ridotto impegno nell'aggiornamento del software sul device e non risulta inoltre critica

un’eventuale rottura o sostituzione del terminale, per questo può essere adatta a personale ad elevata mobilità sul territorio, come trasportatori o addetti alle riparazioni. Oppure disporre di applicazioni (ad esempio su PDA) che lavorano su un database locale, sincronizzato con quello remoto. In questo caso sono disponibili versioni ridotte di DB, come l'UltraLight di Sybase o DB tool di IBM, che in pochi KB raccolgono le funzionalità principali. In questo caso è importante che la suite di sviluppo SDK preveda una serie di librerie pronte per l'interfacciamento con tali DB.
Alla luce di questa situazione si è immaginato dunque come poteva essere strutturato l’applicativo da sviluppare prendendo in considerazione inizialmente l’informazione che doveva essere trasmessa per inoltrare la nostra richiesta di ricarica. Si è pensato perciò a quale mezzo di comunicazione messo a disposizione da tali tecnologie fosse quello più adatto, e come ci si aspettava dall’inizio, l’SMS è stato quello con le caratteristiche più idonee per risolvere il problema.

1.7 La tecnologia SMS

Il termine SMS (acronimo dell'inglese: Short Message Service) è comunemente usato per indicare un breve messaggio di testo inviato da un telefono cellulare ad un altro, con un costo esiguo.
Come già introdotto tale servizio è stato originariamente sviluppato sulla rete GSM ed è ora disponibile anche su altre reti, come la UMTS. È possibile inviare SMS ad un telefono cellulare anche da un computer, tramite Internet, e dal telefono di rete fissa.
Tra i principali vantaggi percepiti dell'SMS, alla base della straordinaria diffusione di questo servizio come sistema di comunicazione, c'è il basso costo rispetto ad una lunga telefonata e la possibilità di rendere la comunicazione asincrona, cioè leggere il messaggio in un momento successivo alla ricezione.
Lo standard prevede due diverse tipologie di messaggi: quelli Point-to-Point (SMS/PP), impiegati nella comunicazione da un terminale ad un altro, e

quelli tipo Cell Broadcast (SMS/CB), originati in una cella e distribuiti a tutti i terminali sotto la sua copertura.
Il messaggio ha una dimensione fissa di 140 byte. Questo si traduce in pratica nella possibilità di usare 160 caratteri di testo (a 7 bit). In lingue che usano altri caratteri rispetto all'alfabeto latino, per esempio in russo, cinese, giapponese, il messaggio è limitato a soli 70 caratteri (ognuno di 2 byte, usando il sistema Unicode, il sistema di codifica che assegna una combinazione di bit a ogni carattere in maniera indipendente dal programma, piattaforma).
Dal punto di vista trasmissivo, le unità di dati del messaggio SMS (vengono impiegate 6 diverse PDU, Protocol Data Unit) vengono inserite nei canali di controllo del GSM, in modo che sia possibile ricevere o inviare un messaggio durante una conversazione.
I messaggi Point-to-Point sono i quelli che un utente può inviare ad un altro utente della rete mobile. Ogni messaggio viene inviato ad un Centro Servizi (SMSC, Short Messages Services Centre) che a sua volta si occupa di inviarlo al terminale opportuno, se nella stessa rete GSM, oppure al Centro Servizi dell'operatore della rete del destinatario. Pertanto il singolo messaggio viene, in realtà, diviso in due: il messaggio dal terminale al Centro Servizi (SMS-MO, Mobile Originated), e dal Centro Servizi al destinatario (SMS-MT, Mobile Terminated). Scopo del SMSC è, ovviamente, quello da fare lo store-and-forward dei messaggi, anche in previsione di un'eventuale irraggiungibilità momentanea del destinatario.
Dato che la connessione tra terminale e Centro Servizi è di tipo connectionless, non si ha garanzia né sull'invio né sulla ricezione dei messaggi SMS. Tuttavia è possibile fare richiesta di una "ricevuta di ritorno", lo Status Report del messaggio. In questo modo, una volta che il messaggio è stato correttamente inoltrato al destinatario, viene inviata una segnalazione di "messaggio consegnato" al mittente.
Il servizio di SMS, normalmente operante da utente ad utente, esiste anche in una variante broadcast. Ogni cella ha la possibilità di inviare brevi messaggi a tutti i terminali sotto la sua copertura, su uno specifico canale

numerato, ciascuno dedicato ad un tipo particolare di informazione, quali emergenze, informazioni sul traffico, eccetera.
Attualmente, in Italia, il solo servizio abilitato è il nr. 50: se il cellulare ricevente ha abilitata la ricezione dei messaggi locali su tale canale, ogni 10 secondi circa viene mostrata sullo schermo dell'utente la provincia in cui è posta la cella a cui l'utente è registrato.
Il prezzo applicato a chi invia un SMS è attualmente di 9-15 centesimi di euro, 100 volte maggiore del suo costo pieno industriale che ha un ordine di grandezza dei millesimi di euro (precisamente 0,1 centesimi di euro). Ciò sembra garantire ampi margini di guadagno agli operatori. A livello di costi per la compagnia telefonica, il criterio per calcolarlo è infatti il costo per bit inviato che è indifferente al tipo di informazione inviata (per l'operatore il costo di un bit è lo stesso che si tratti di un SMS, MMS o chiamata vocale); le tariffazioni all'utenza, differenziate per il tipo di servizio offerto, seguono criteri di prezzo diversi da un orientamento ai costi che in questo caso consiste di un'unica tariffa al byte inviato.
Tuttavia, il costo industriale dell'SMS è una grandezza che non tiene conto dei costi di un addebito su scheda SIM. I sistemi di accredito delle compagnie di telecomunicazioni, come quelli delle banche, sono particolarmente costosi. In pratica, il costo per l'operatore per scalare la tariffa di un SMS dal credito di una scheda prepagata (o per fatturarlo sui contratti in abbonamento) è molto maggiore degli 0,1 centesimi spesi per l'invio del messaggio. Senza una tariffa largamente superiore al costo industriale di 0,1 centesimi l'SMS non sarebbe remunerativo per gli operatori, poiché l'operazione di pagamento costerebbe all'operatore più della cifra da incassare.
Vi sono diverse aziende on-line che vendono SMS in pacchetti da 1000 a 500000 SMS utilizzabili solo da PC a cellulare a costi decisamente inferiori.
A partire dal 2005 ha iniziato ad essere impiegato l'invio di SMS come mezzo per effettuare donazioni ad enti senza fini di lucro o organizzazioni umanitarie. A tale scopo viene attivato, in genere per un periodo limitato nel tempo, un numero telefonico speciale verso il quale l'invio di un SMS, che

prende il nome di SMS solidale, viene addebitato con un importo più alto del normale (in genere uno o due euro), che viene interamente devoluto come donazione all'associazione o all'ente organizzatore. Con questo sistema, ad esempio, in seguito al maremoto che alla fine del 2004 ha colpito l'Asia, solo in Italia sono stati raccolti oltre 26 milioni di euro.

L'uso dei messaggi SMS si è diffuso molto velocemente in tutto il mondo. A metà del 2004, il volume di traffico annuo in tutto il mondo era di circa 500 miliardi di SMS; la crescita del fenomeno è impressionante se si pensa che nel 2000 i messaggi erano stati circa 17 miliardi. Gli SMS sono più popolari in Asia, Europa e Australia rispetto agli Stati Uniti. Il business degli SMS cresce ogni anno di più ed è un vero affare per le aziende che, da anni, non abbassano i prezzi.

I moderni telefoni cellulari permettono di inviare, oltre agli SMS, anche MMS, messaggi multimediali dal funzionamento simile agli SMS, che permettono però la trasmissione di musica, immagini e video.

Il servizio di SMS da Internet ha incominciato a diffondersi sul web dai primi mesi del 1997 (il primo sito in Europa a offrire questo servizio è stato quello di SMS Italia) e inizialmente era offerto gratuitamente su molti siti, grazie al fatto che il traffico di SMS scambiati fra i diversi operatori di telefonia mobile, fra loro, era gratuito. Dal 2000 al 2002, progressivamente tutti gli operatori Mobili hanno concordato una tariffa di interconnessione (Interconnection Fee), e questo ha innalzato il costo degli SMS provocando quindi la riduzione di servizi gratuiti di spedizione SMS da Internet, innalzando però il livello di affidabilità globale del servizio.

Dal 2005 si sono diffuse alcune MIDlet, che se installate su cellulari sfruttano la connessione GPRS o UMTS per utilizzare questi servizi che danno la possibilità di inviare SMS gratuitamente ma in numero limitato. Il costo per l'invio degli SMS da cellulare tramite questi programmi è quindi limitato alla connessione Internet e può risultare conveniente solo se il gestore di telefonia offre un piano tariffario adeguato per la connessione a Internet, meglio se a volume o flat.

2 CAPITOLO 2: Analisi e progettazione del sistema

In questo capitolo si affronta l'analisi del sistema che gestisce le comunicazioni utente-banca tramite scambio di messaggi.
Viene quindi proposta la soluzione del sistema esponendo le decisioni prese per quanto riguarda l'aspetto tecnologico, logico e l'architettura per soddisfare i requisiti delle varie parti chiamate in causa.

2.1 Generalità

Il sistema da analizzare è stato suddiviso in due parti logiche: quella relativa all'utente (Front-End) e quella relativa alla gestione delle richieste da parte della banca (Back-End). Questa è stata necessaria per semplificare la comprensione del sistema nel suo insieme e per far sì che le due entità potessero essere sviluppate in maniera parallela durante la fase di implementazione.

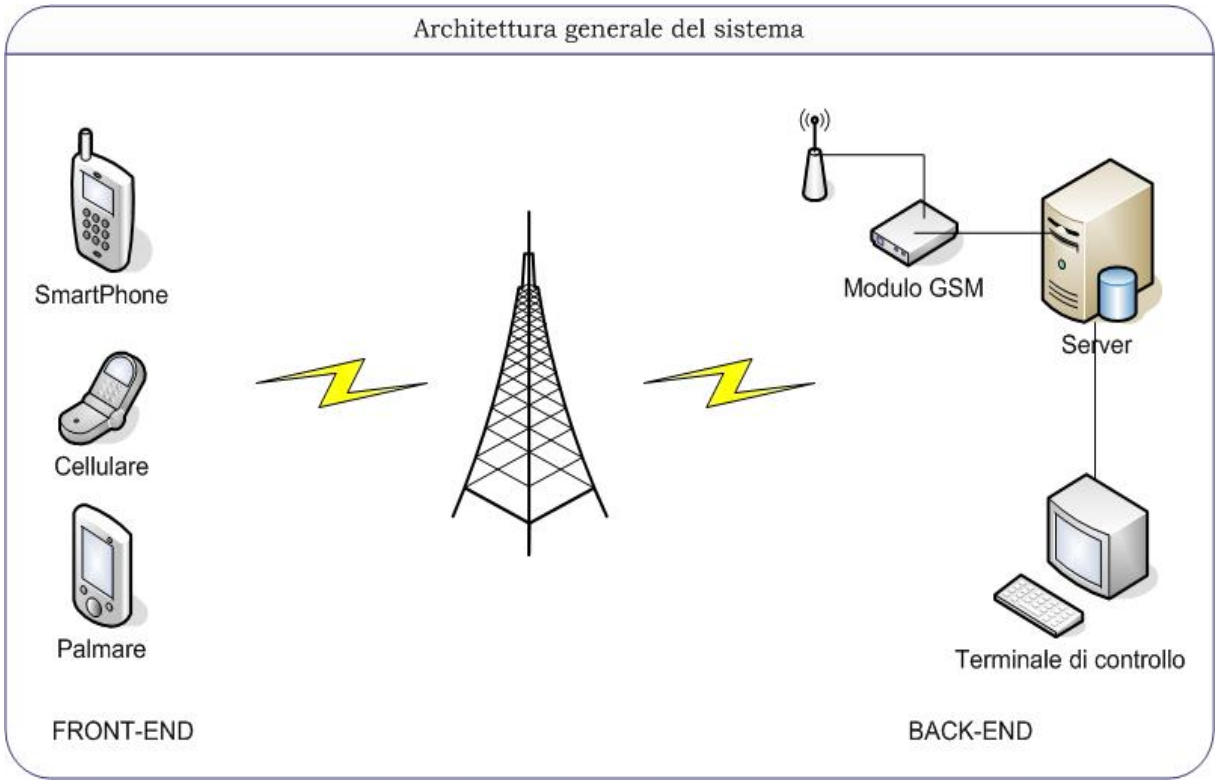

Figura 1

L'immagine in Figura 1 rappresenta il sistema concepito nel suo insieme con le varie componenti chiamate in causa e le due sezioni logiche in evidenza. Per Front-End si intende tutto l'insieme di necessità, requisiti e logica di funzionamento dell'applicazione MIDlet che verrà utilizzata dall'utente, mentre con Back-End intendiamo tutto il funzionamento del sistema server della banca che riceverà le richieste e eseguirà l'ordine.

Per quanto riguarda il processo completo, ovvero dall'inizio dell'utilizzo del dispositivo fino alla conferma dell'avvenuta operazione, il sistema è concepito per comportarsi in questo modo:

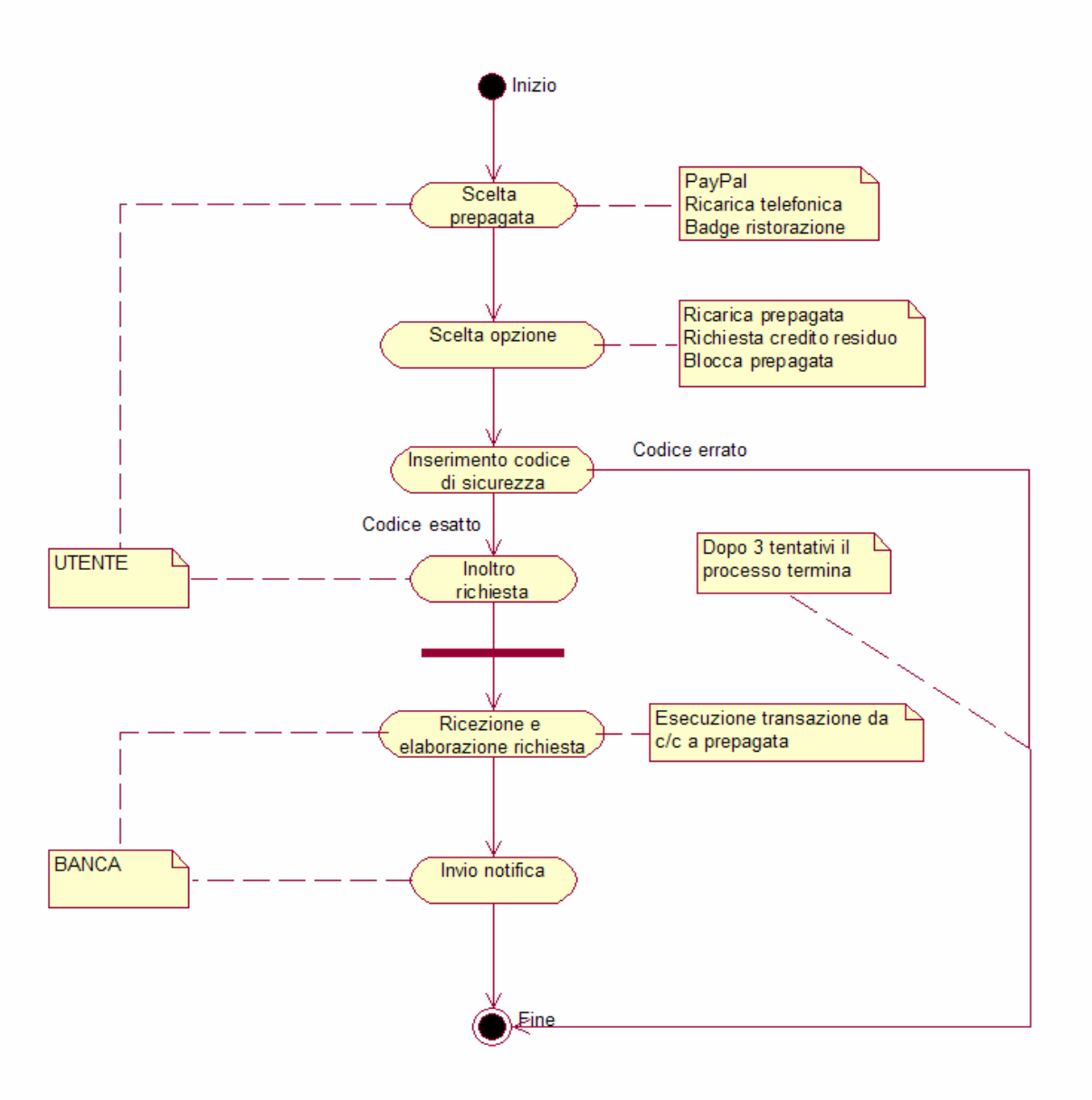

Figura 2

L' activity-diagram in Figura 2 mostra l'intero funzionamento logico che ci si aspetta di ottenere dal sistema una volta implementato; anche in questo caso si può notare la suddivisione nei due sottosistemi logici che saranno trattati in maniera parallela durante la fase di sviluppo.

2.2 Sottosistema Front-End

Inizialmente si è studiato il sistema dal punto di vista dell'utente, in quanto riguardava la parte più innovativa dell'applicativo.
Inoltre, sempre nel rispetto della sicurezza, le esigenze di chi utilizzerà questo software hanno avuto la precedenza rispetto al resto, quindi usabilità e semplicità saranno le parole chiave di questa sezione.
Lo scopo della applicazione MIDlet da sviluppare è generare un semplice SMS con le informazioni necessarie alla banca per soddisfare la richiesta.
Tali informazioni nello specifico saranno il tipo di carta prepagata da ricaricare, quali Badge ristorazione, PayPal e carta telefonica e l'importo che può essere scelto da un menu con voci proposte oppure digitato direttamente in un campo apposito.
Oltre alle opzioni di ricarica si è inserita la possibilità di richiedere un informazione riguardante il credito residuo e quella relativa al blocco della tessera nel caso di smarrimento.
Tutte queste informazioni verranno inserite nel messaggio durante la navigazione all'interno del programma: l'utente crea il messaggio quindi senza sapere come effettivamente è composta la stringa risultante e qual è la sua sintassi.
Prima della conferma dell'invio del messaggio verrà chiesto all'utente di inserire un codice PIN (Personal Identification Number) di sicurezza in modo tale che il servizio non possa venire utilizzato da qualcuno non autorizzato. Sarà consentito all'utente di sbagliare per tre volte il codice di sicurezza, dopodichè il processo terminerà.
Inizialmente si è pensato di non introdurre dati sensibili all'interno del messaggio, in quanto si è ipotizzato che il numero di telefono del mittente fosse sufficiente per poter identificare l'utente registrato nel DataBase della

banca per poter effettuare le operazioni. Questo aspetto verrà poi trattato in maniera più dettagliata tenendo in considerazione gli aspetti relativi alla sicurezza forniti dalla tecnologia GSM.
Come accennato più volte, per far sì che chi usa un'applicazione mobile non si senta ostacolato dai limiti tecnici del terminale è necessario che le opzioni che possono essere scelte siano chiare, ben organizzate, non ambigue e facili da rintracciare, dunque vengono riportati i requisiti che l'applicativo dovrà rispettare per evitare queste problematiche.

Requisiti Non Funzionali:

- L'utente necessita che l'applicativo sia semplice, immediato e non ambiguo.
- Qualsiasi categoria di utenza deve poter usufruire del servizio: giovani, anziani, disabili, ecc…
- L'utente deve ottenere tutto quello che desidera in maniera veloce.

Inoltre per sfruttare in pieno tale tecnologia si deve dare la possibilità di poter compiere più azioni distinte tra loro, carpendo dalla realtà quali possano essere le più idonee per il nostro scopo.
Si è pensato dunque quali fossero le caratteristiche adatte che dovevano essere prese in considerazione, quelle cioè che sarebbe state utilizzate dagli utenti

Requisiti Funzionali:

- L'utente deve poter scegliere l'importo da ricaricare da un elenco fornito o decidere quanto ricaricare immettendo la cifra desiderata.
- L'utente deve sapere se l'operazione è andata a buon fine.
- L'utente deve poter richiedere un'informazione riguardante il credito residuo della carta prepagata.
- In caso di necessità l'utente deve poter inoltrare una richiesta per bloccare la carta prepagata.

Riassumendo, la situazione delle possibilità concesse all'utente è sostanzialmente quella descritta in Figura 3:

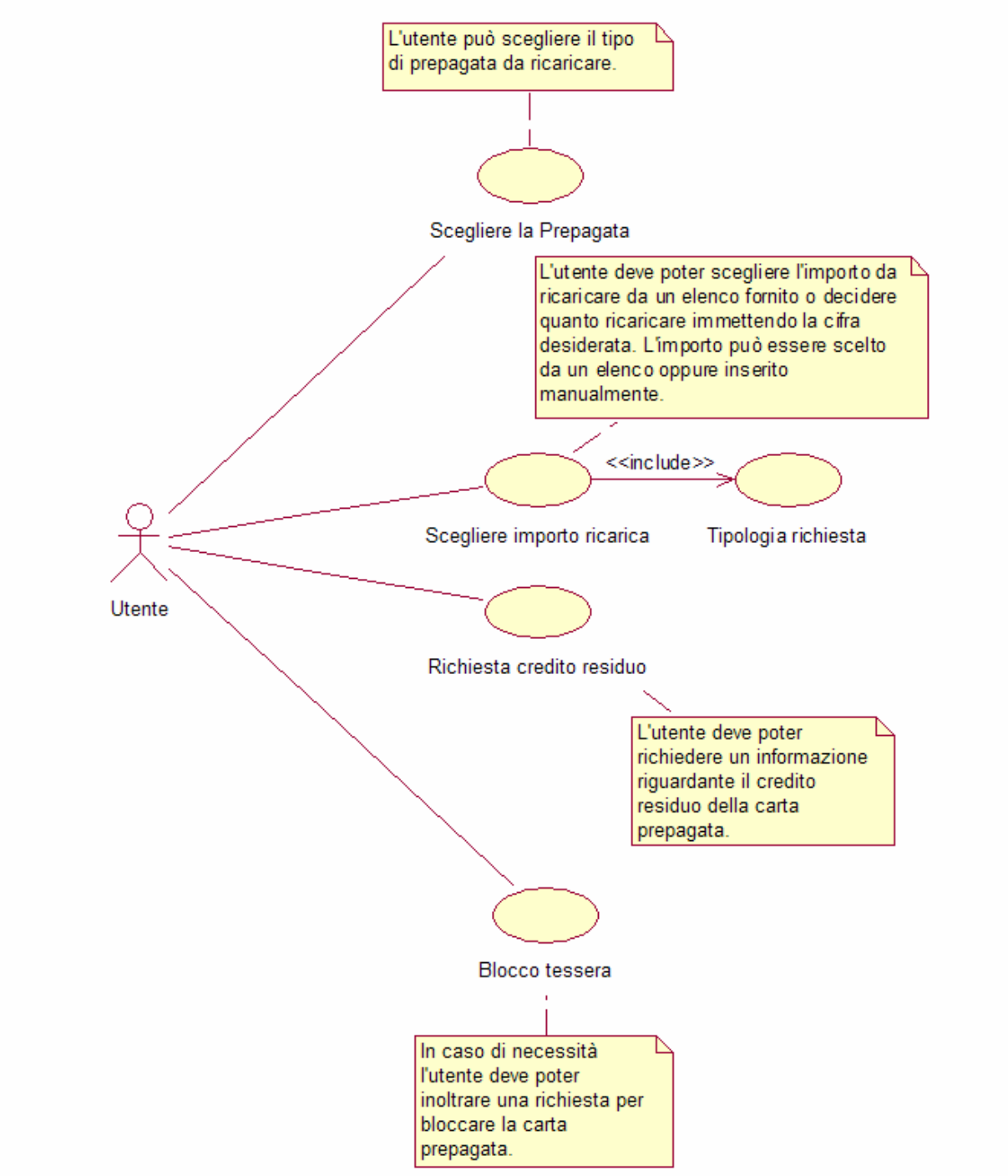

Figura 3

L'applicazione MIDlet da sviluppare deve dunque permettere queste opzioni ma non solo, deve essere infatti pensata in maniera tale che possa essere facilmente aggiornata e, in un futuro, possa anche ospitare servizi differenti.
Se un eventuale utilizzo di tale servizio dovesse decretarsi un successo, la situazione in questo ambito potrebbe velocemente evolversi e molte altre tipologie di servizi potrebbero essere viste sotto questa ottica.
L'idea è far sì che un applicativo mobile possa contenere più servizi piuttosto che prevedere per ogni servizio un diverso applicativo.
Modulabilità e scalabilità devono quindi essere prese seriamente in considerazione se si vuole creare un applicativo aperto, flessibile e senza vincoli logici da impedire aggiornamenti.

2.3 Sottosistema Back-End

L'altra anima del sistema è quella denominata Back-End. Fisicamente essa comprende il sistema server della banca e il sistema di moduli GSM.
L'applicazione residente in sostanza deve rimanere in "ascolto" dei messaggi provenienti dagli utenti, codificarne il contenuto e eseguirne le istruzioni, ovvero provvedere a trasferire l'importo scelto dal conto corrente dell'utente alla carta prepagata specificata nel messaggio.
È in sostanza lo stesso procedimento che funziona ora con l'internet banking, cambia solamente il metodo della richiesta.
Ovviamente ogni operazione deve essere tracciata e quindi un apposito registro consultabile da un operatore o un tecnico deve essere sempre aggiornato.
A esecuzione completata, verrà spedito un messaggio di conferma all'utente che lo informerà dell'avvenuta transazione.

Requisiti Non Funzionali:

- La notifica dell'esito dell'operazione deve essere inviata all'utente sia via SMS che via e-mail.

- Anche senza richiesta da parte dell'utente, il sistema deve notificare se il credito sta per esaurirsi o se è già esaurito.

Requisiti Funzionali:

- Il servizio deve eseguire i compiti richiesti in maniera completamente automatizzata.
- Il servizio deve notificare l'esito dell'operazione all'utente (anche in caso d'errore).
- Il servizio deve registrare tutte le operazioni effettuate.

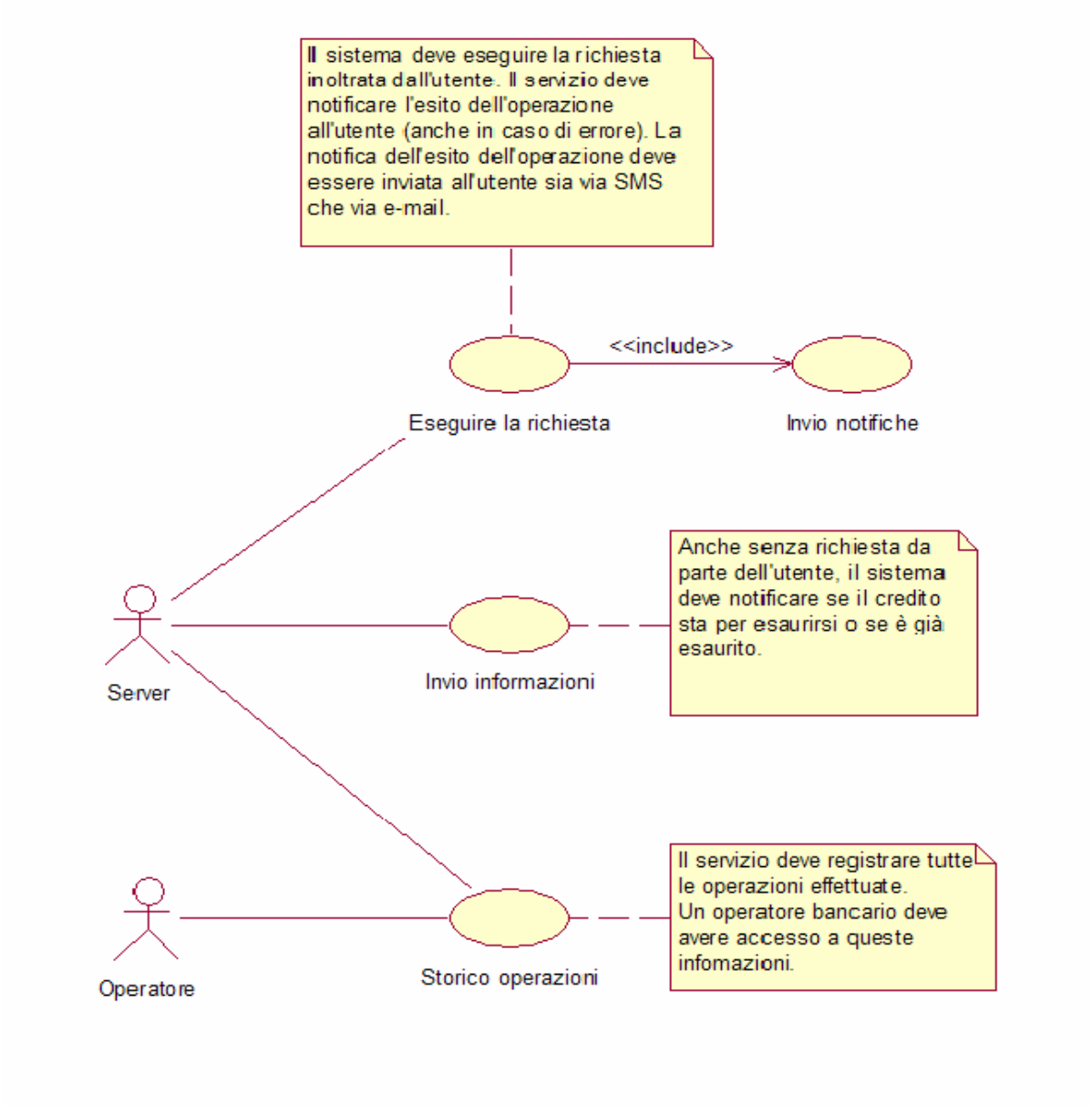

Figura 4

Come si può notare nello Use Case diagram in Figura 4 è stato inserito l'attore relativo all'operatore bancario: a sistema a regime dovrà avere la possibilità sia di visionare lo storico dei movimenti che di monitorare lo stato del sistema per controllarne il corretto funzionamento.

2.4 Robustezza e affidabilità

Prima di spiegare gli accorgimenti studiati affinché il sistema sia affidabile è necessario introdurre dei concetti base che aiutano a capire meglio questo importante aspetto.

Single Point of Failure: Ogni computer o device che esegue una singola operazione ed è connesso al sistema costituisce un single point of failure nel caso dovesse fallire e non ci siano alternative per rimpiazzarlo.
Fault Tolerant: Capacità di continuare il lavoro senza interruzioni quando un componente fallisce.
Fault Management: Il monitoraggio delle indicazioni degli errori che occorrono in un sistema e l'invio dei relativi messaggi d'allarme all'amministratore viene definito fault management.
Fault Detection: Scoperta che qualcosa nell'hardware o nel software è fallito. Tipicamente vi è la registrazione dell'istante di occorrenza dell'errore e viene mandato una avviso amministrativo.
Fault Isolation: Determinazione della causa del problema e isolamento di esso. Rispetto a Fault detection indica quale componente ha fallito.

In sostanza il sistema deve essere Fault Tolerant e, nel caso di fallimenti o errori, bisogna che si possa risalire alla causa.
Ovviamente questo riguarda principalmente il lato Back-End essendo composto da più componenti che svolgono compiti specifici.
Si è proceduto allora ad individuare i possibili Single Point of Failure individuando nei server ma soprattutto nei moduli GSM i punti più critici.

Nella quasi totalità dei casi i moduli GSM hanno programmi di diagnostica dedicati e c'è quindi la possibilità di monitorare il loro stato; questo non evita però che i moduli possano smettere di funzionare. Lo stesso discorso è valido anche per i server sui quali vengono compiute le operazioni di transazione.
Per poter garantire un servizio continuo a fronte anche di eventuali fallimenti di hardware o software è necessario che il sistema sia ridondante: dovremo quindi disporre di almeno due moduli GSM con uguali caratteristiche in modo tale che uno possa sopperire agli eventuali problemi dell'altro.
Stesso ragionamento lo si deve fare per i server, dove i dati e l'applicativo che gestisce la transazione saranno replicati. Questo approccio torna anche utile in un'ottica di manutenzione del sistema: nel caso di aggiornamenti e modifiche si potrà così agire su uno dei server, o riprogrammare uno dei moduli GSM mantenendo attivo il servizio agli utenti con il secondo, per poi riattivare il primo e apportare le nuove modifiche all'altro.
L'aspetto più importante è dunque che il processo nel suo insieme (dall'invio della richiesta alla notifica del compimento della transazione) non venga interrotto per alcun motivo.
Nella Figura 5 viene riportata una possibile struttura del sottosistema Back-End che sarebbe in grado di supportare un eventuale crash di uno dei componenti. Come si può notare la struttura con hardware ridondante assicura più robustezza al sistema, e quindi più affidabilità.
I moduli GSM potrebbero lavorare in maniera esclusiva facendo sì che magari il #2 subentri al #1 quando quest'ultimo non è attivo, oppure in modo parallelo, inoltrando ai server (in maniera concorrente calcolando il tempo di notifica del messaggio) le richieste che arrivano e poi inviando la risposta, in maniera esclusiva, all'utente.
I moduli GSM dovranno perciò essere identici, ovvero avere le stesse caratteristiche, in particolare il numero telefonico. Essi potranno essere anche programmati in modo tale che rispondano in maniera automatica all'utente nel caso i server non fossero raggiungibili, non lasciando il dubbio all'utente sull'esito della sua richiesta.

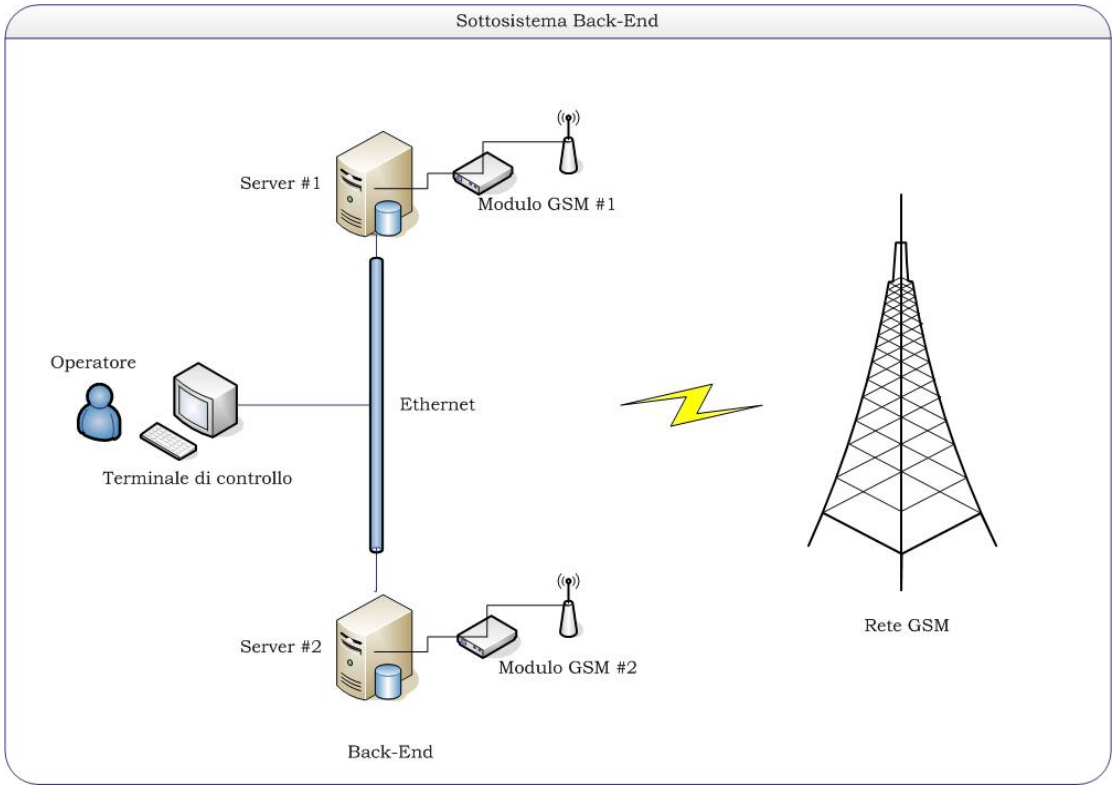
Sottosistema Back-End
Server #1
Modulo GSM #1
Operatore
Ethernet
Terminale di controllo
Rete GSM
Server #2
Modulo GSM #2
Back-End

Figura 5

3 CAPITOLO 3: Implementazione del prototipo

In questo capitolo viene trattata l'implementazione di un prototipo del servizio che deve soddisfare i requisiti analizzati e rispettare la logica di funzionamento studiata nel capitolo precedente.
La parte più innovativa è quella relativa all'applicativo mobile, ovvero la MIDlet destinata all'utente, comprendente le interfacce grafiche studiate per rendere la navigazione più semplice; la sezione più delicata dal punto di vista del funzionamento logico è invece quella del sottosistema bancario.

3.1 Premessa

Si è scelto di strutturare la parte Back-End del prototipo in maniera diversa da quella presentata in fase di analisi in quanto:

- Avendo preso in considerazione un sistema bancario non si potevano avere informazioni relative ai server e a come i dati sono organizzati: si è quindi solo ipotizzata la fase di transazione dell'importo.
- Non potendo disporre di moduli GSM per le prove pratiche, si è scelto di adibire uno smartphone a modulo GSM connesso tramite una connessione Bluetooth con un computer che emulava le funzioni del server.

Questo non implica che la logica del sistema sia stata sconvolta, cambiano soltanto le parti relative al metodo di ricezione. Valgono perciò tutte le decisioni prese in fase di analisi sapendo che un'architettura come quella del prototipo non può essere presa in seria considerazione in quanto poco conveniente dal punto di vista delle prestazioni e soprattutto soggetta a manutenzione più attenta. Questa è solamente una scelta dettata dalla praticità e dall'impossibilità di reperire apparecchiature idonee.
L'applicativo sviluppato è pensato per essere adattato in maniera semplice all'idea originale della struttura del sistema.

L'architettura del prototipo ha quindi assunto la logica descritta nell'immagine rappresentata in Figura 6.

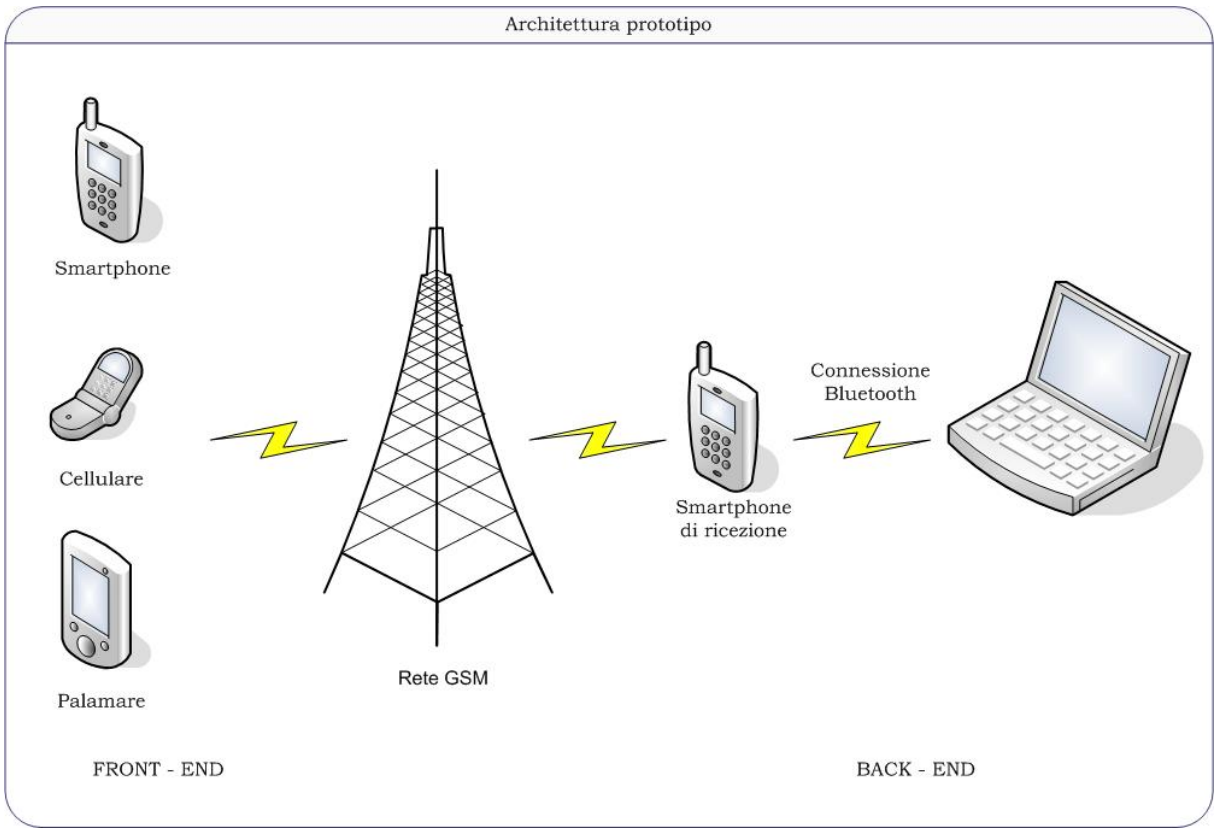

Figura 6

Per quello che riguarda l'aspetto tecnico, si è sviluppato il software con l'ambiente NetBeans 5.5 aggiornato con il Mobility Pack e testato con il Java Wireless Toolkit for CLDC entrambi di SUN Microsystem.

3.2 L'applicativo mobile

Come già fatto per la fase di analisi, viene preso in considerazione inizialmente il lato Front-End. La J2ME mette a disposizione una serie di profili, tra cui il MIDP 1.0 - Mobile Information Device Profile – che è il pacchetto di librerie necessarie per sviluppare le prime applicazioni J2ME per i nostri cellulari. In esso è presente la libreria javax.microedition.lcdui la quale contiene le classi per creare delle interfacce utente sul display.

Con le classi appartenenti a questo package è infatti possibile gestire dei Form per l'inserimento dei dati con campi di testo, selezioni a scelta multipla, date o ore in formato grafico, immagini e così via.
L'applicativo sviluppato è strutturato su due classi: `SMSSender` e `MyCanvas`.

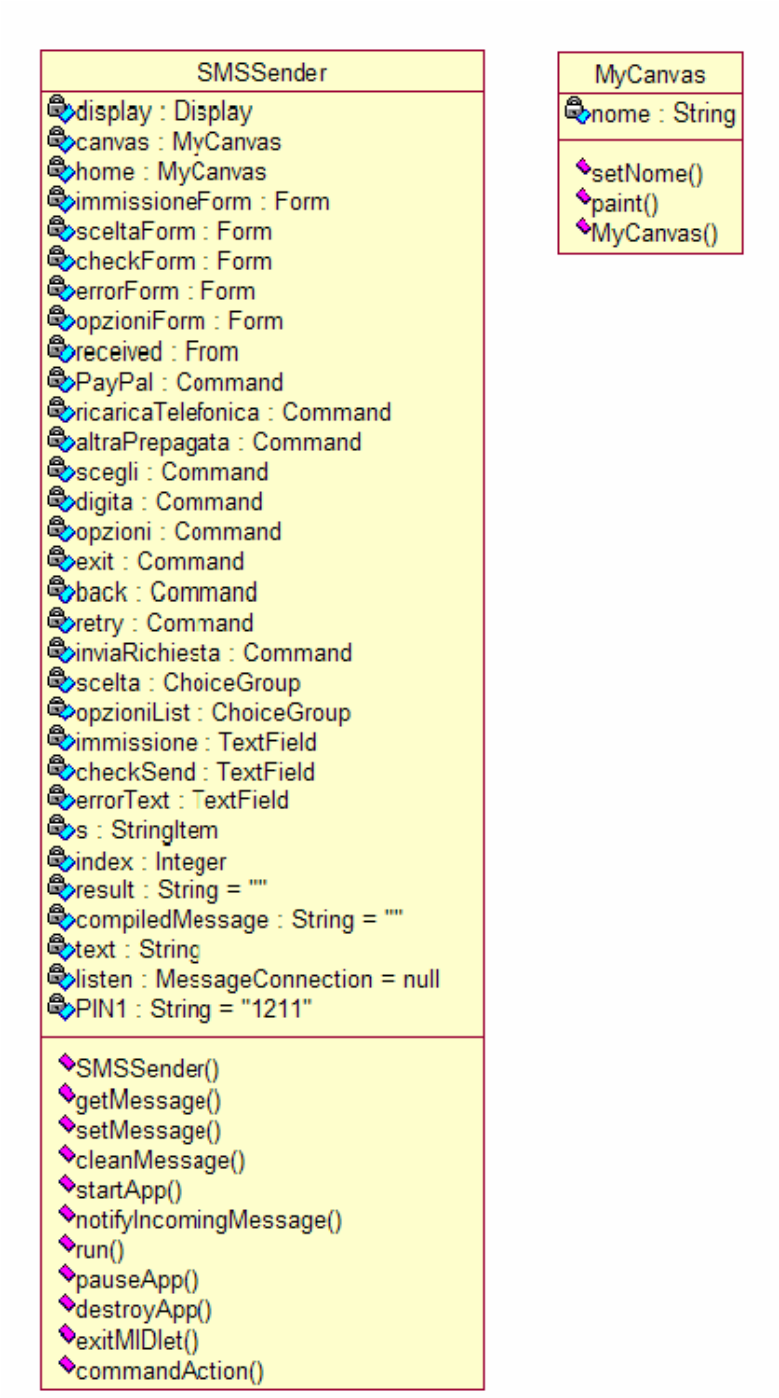

Figura 7 – Vengono riportati solo i nomi dei metodi per questione di leggibilità e non la firma intera

Mentre la logica di utilizzo è la seguente:

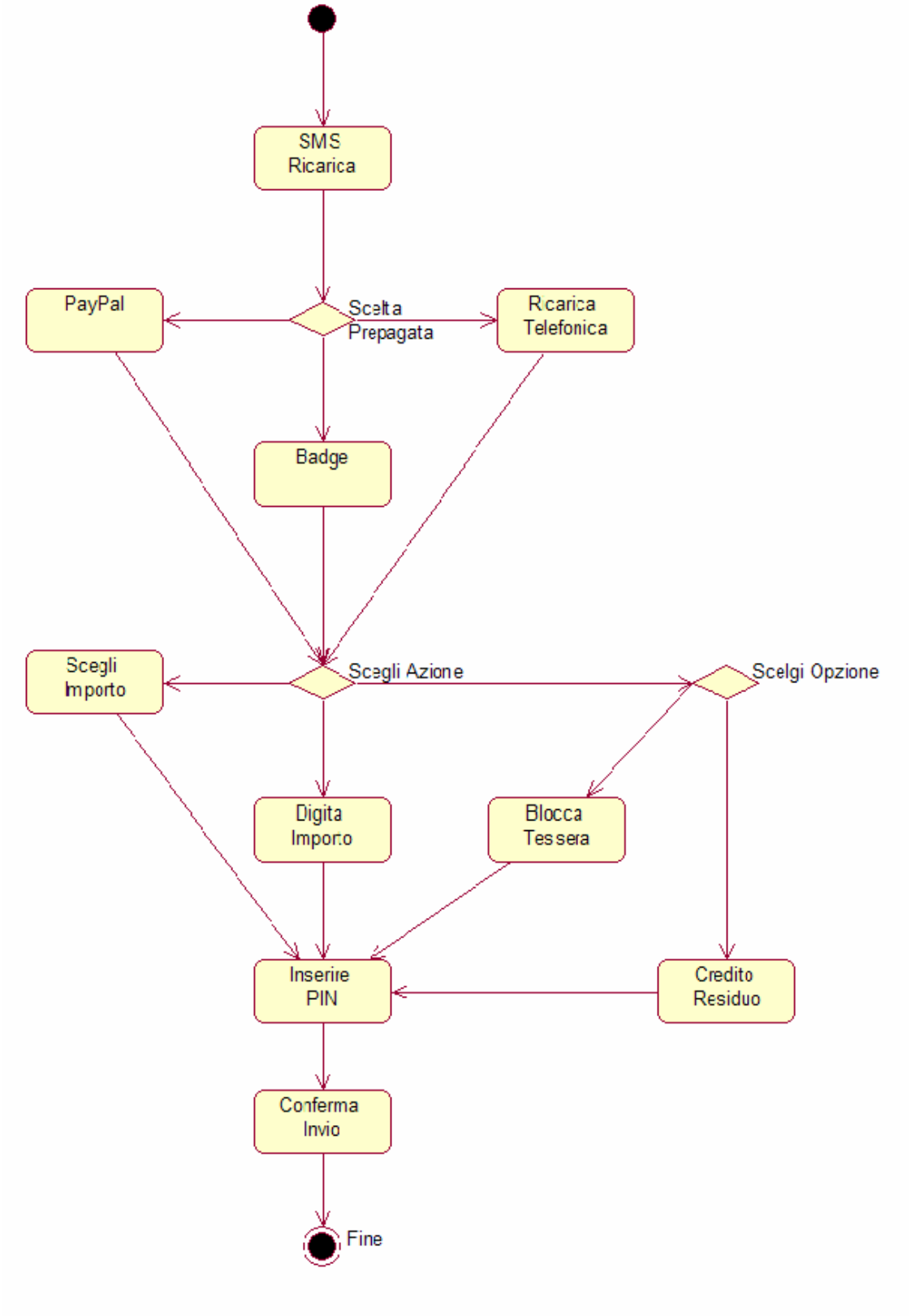

Figura 8

La classe SMSSender è la classe principale, inizializza con il metodo `startApp()` la MIDlet e gestisce operazioni che l'utente può compiere.
La seconda, MyCanvas, è stata implementata estendendo `Canvas`, oggetto definito nella libreria javax.microedition.lcdui, che implementa il metodo `paint(Graphics g)`:

```
class MyCanvas extends Canvas {

  String nome;

  public MyCanvas(String nome)
  {
      setNome(nome);
  }

  public void setNome(String nome)
  {
    this.nome = nome;
  }

  public void paint(Graphics g)
  {
    g.setColor(255, 255, 255);
    g.fillRect(0, 0, getWidth(), getHeight());

    g.setColor(0, 0, 0);
    g.drawString("",
                 getWidth()/2,
                 getHeight()/2,
                 Graphics.TOP | Graphics.HCENTER);

    g.setColor(255, 0, 0);
    g.drawRect(0, 0, getWidth()-1, getHeight()-1);

    Font font = Font.getFont(Font.FACE_PROPORTIONAL,
                             Font.STYLE_UNDERLINED,
                             Font.SIZE_LARGE);
    g.setFont(font);
    g.drawString(nome,
                 getWidth()/2,
                 getHeight()/2,
                 Graphics.BOTTOM | Graphics.HCENTER);

    g.setColor(255, 0, 0);
    g.fillRect(10, 10, getWidth()-20, 10);
    g.fillRect(10, getHeight()-20, getWidth()-20, 10);
  }
```

MyCanvas viene istanziata da SMSSender, e viene utilizzata come oggetto contenitore per gli altri tipi di elementi, per esempio gli oggetti di tipo `Command`:

```
home = new MyCanvas("SMS Ricarica");
home.addCommand(exit);
home.addCommand(PayPal);
home.addCommand(ricaricaTelefonica);
home.addCommand(altraPrepagata);
home.setCommandListener(this);
```

In questo estratto di codice si nota come gli oggetti di tipo Command vengano aggiunti al Canvas "`home`".

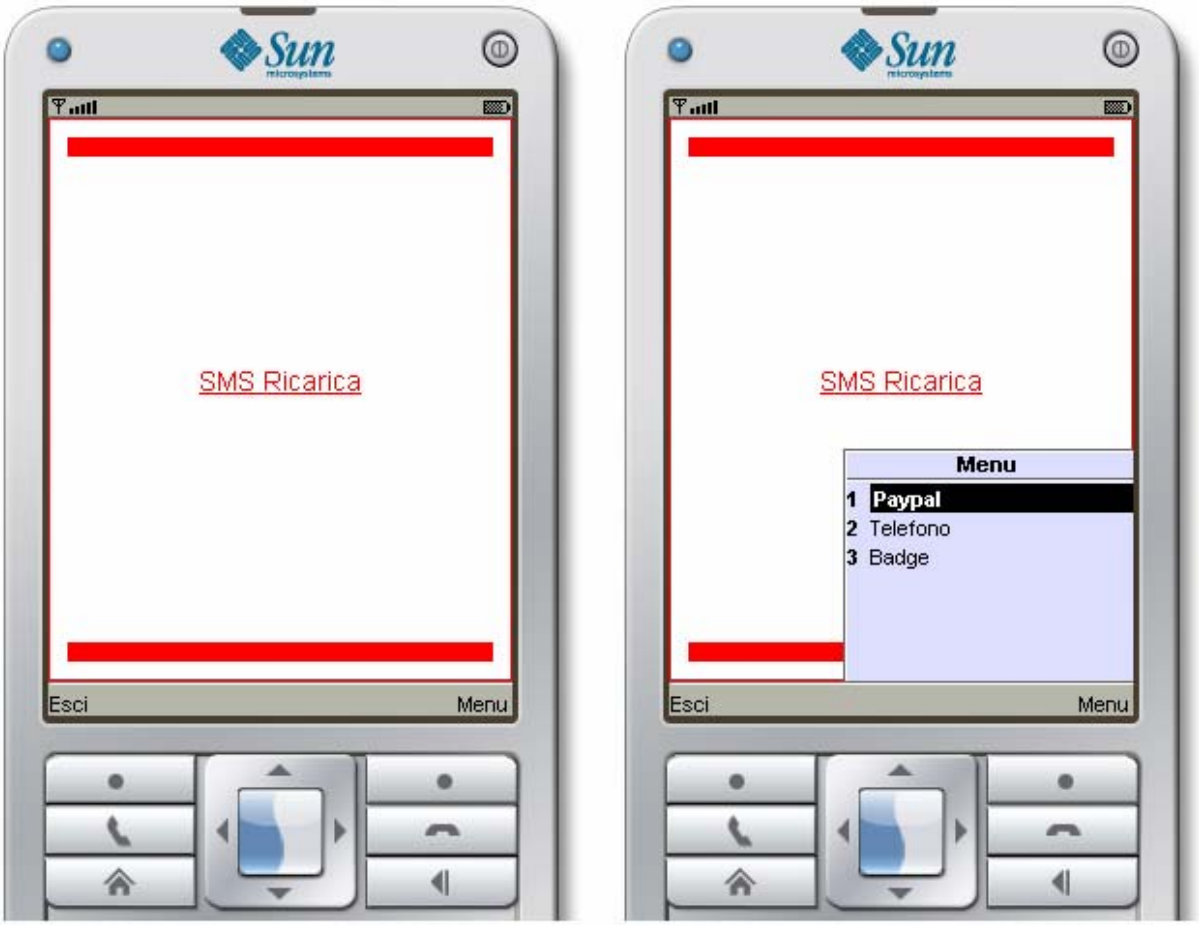

Figura 9

In Figura 9 vengono riportate due schermate iniziali dell'applicativo che l'utente andrà a utilizzare: la prima a sinistra mette in evidenza solamente l'oggetto Canvas che "accoglie" l'utente, mentre la seconda visualizza l'azione

del testo “Menu” una volta premuto, ovvero il primo menu che incontrerà l’utente, quello dove egli deciderà che carta prepagata vorrà ricaricare.
Tale menu è l’elenco di comandi (PayPal, Telefono e Badge) che sono stati inseriti all’interno del Canvas il quale rimane in sottofondo.
Utilizzare i Canvas è sicuramente più efficace dal punto di vista estetico, in quanto permettono di personalizzare il Layout del programma rendendolo più gradevole all’utente. Affinché il loro utilizzo sia fluido necessitano però una programmazione più verbosa e quindi complessa e questo si rivela uno svantaggio.
Si è scelto perciò di utilizzarli in combinazione con i Form, ovvero delle schermate standard, più austere ma più pratiche da utilizzare, più snelle e più intuitive. I Form, come già accennato, mettono a disposizione inoltre una serie di oggetti per poter affrontare le diverse tipologie di opzioni da inserire.

Figura 10

Nell'immagine riportata in Figura 10 si nota la differenza estetica tra Canvas, a sinistra, e Form, a destra: mentre nel primo caso per far sì che siano visibili le opzioni del menu è necessario agire sull'apposito pulsante, nel secondo il menu ad esclusione è già caricato nel Form stesso e vi si può agire direttamente. In questo preciso caso, la seconda schermata è figlia della scelta della prima opzione del menu Canvas ovvero "Scegli Importo" per la prepagata PayPal, come si nota sullo sfondo.
Nello specifico il Form sopra visualizzato è così strutturato:

```
sceltaForm = new Form("Inserire l'importo");
String choices[] = {"5€", "10€", "15€"};
scelta = new ChoiceGroup("Scegli l'importo",Choice.EXCLUSIVE,choices,null);
sceltaForm.addCommand(ok);
sceltaForm.addCommand(back);
sceltaForm.append(scelta);
sceltaForm.setCommandListener(this);
```

Come nel caso del Canvas, anche in questo esempio si nota che gli oggetti vengono inseriti all'interno del Form con il comando `addCommand()`, ma vi è il metodo `append()` che permette di visualizzare direttamente l'oggetto all'interno del form, in questo caso il `ChoiceGroup "scelta"`.
Inoltre anche per il Form vi è come ultima istruzione il metodo `setCommandListener(this)`: questo, come nel caso del Canvas, si occupa di richiamare la funzione `commandAction(Command c, Displayable s)` della MIDlet la quale si occupa di definire le azioni per ogni tipo di evento, quale può essere la pressione di un pulsante. Questa funzione è dunque molto importate perché in qualsiasi punto ci si trovi all'interno del programma e a seconda di quale input noi diamo deve gestire gli eventi in maniera corretta.
Qui sotto viene riportato un esempio di evento gestito da tale funzione:

```
public void commandAction(Command c, Displayable s){
…
else if (c == PayPal || c == ricaricaTelefonica || c == altraPrepagata) {
setMessage(result + c.getLabel());
canvas.setNome(c.getLabel());
display.setCurrent(canvas);
…}
```

Questo caso si riferisce agli eventi gestiti all'inizio della navigazione durante la scelta della carta prepagata dal menu del Canvas: in base al Command scelto verrà per prima cosa inizializzata una stringa la quale conterrà il messaggio da spedire successivamente con il Label relativo e verrà impostata la nuova schermata con un Canvas aggiornato con quel Label.
La stessa logica vale per tutti gli altri passaggi atti a completare il messaggio da inviare con le informazioni necessarie: ad ogni step viene aggiornata la stringa risultante che alla fine del processo avrà una struttura del tipo "Tipo_ricarica:Importo" oppure "Tipo_ricarica:Opzione".

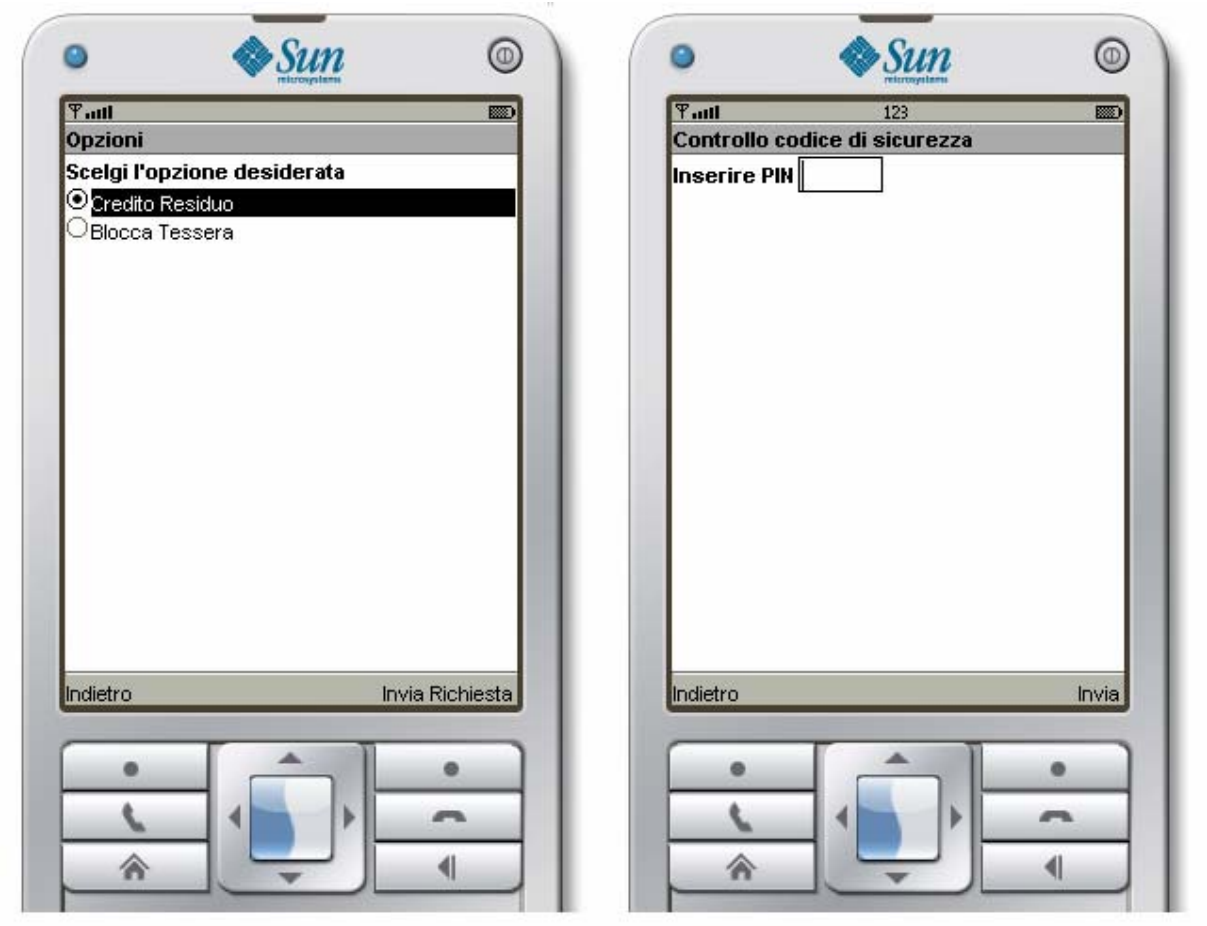

Figura 11

La Figura 11 mette in mostra altre due schermate dell'applicativo, sempre basate su Form: la prima alternativa a quella relativa all'inserimento dell'importo da ricaricare offre l'opportunità di richiedere un'informazione

riguardante il credito residuo della carta scelta, la seconda richiede un codice personale di sicurezza prima che il messaggio venga inviato, in modo tale che nessuno se non il proprietario del terminale possa inoltrare richieste.
Una parte fondamentale del programma è la spedizione del messaggio. Essendo l'invio di un SMS una operazione delicata, è preferibile che esso venga eseguito in un thread differente. La MIDlet pertanto deve implementare l'interfaccia Runnable per poter fare l'overriding del metodo run() appositamente modificato in base alle nostre esigenze:

```
public void run() {
      MessageConnection send = null;
      String url = "sms://+5550000:5000";
      try {
      send=(MessageConnection)Connector.open(url);
      TextMessage msg = (TextMessage)send.newMessage(send.TEXT_MESSAGE);
      msg.setPayloadText(compiledMessage);
      send.send(msg);
      checkSend.setString("");
      cleanMessage();
      } catch(Exception e) { System.out.println("Invio Abortito");}
      finally{
      try{ if(send == null) send.close();} catch(Exception e)
      {
       System.out.println("Connessione KO (Server)");}
      }
}
```

La funzione run() in questo caso si occupa di aprire una connessione con URL "sms://+5550000:5000" che identifica l'indirizzo destinatario: nello specifico 5550000 è il numero di un terminale instanziato dal programma Wireless Toolkit mente 5000 è la porta su cui il terminale è in ascolto. Dopo aver instaurato la connessione vi è la spedizione del messaggio compilato durante la navigazione all'interno del programma, ovvero la stringa compiledMessage, che diventa il PayLoad di un campo TextMessage che verrà a sua volta spedito tramite il comando `send.send(msg)`. Questa funzione run() verrà richiamata al momento della conferma dell'invio, subito dopo l'inserimento del PIN di sicurezza, tramite il comando

`new Thread(this).start()`. Tale comando si troverà nella sezione del metodo `commandAction(Command c, Displayable s)` relativa all'evento associato al pulsante di invio.

3.3 Sicurezza trasmissione dati

Inizialmente lo standard GSM prevedeva un livello di sicurezza relativamente basso, utilizzando un sistema di crittografia parzialmente condiviso per autenticare l'utente. Gli algoritmi A5/1 e A5/2 *stream cipher* proteggono la comunicazione fra telefono e stazione radio-base mentre la comunicazione fra stazione radio-base e resto della rete non è protetta.

L'A5/1 è un algoritmo che garantisce un maggior livello di protezione, ed è quello usato prevalentemente in Europa, mentre l'A5/2, usato in molti altri paesi, permette un minor livello di protezione. Comunque entrambi i sistemi di crittografia si sono rivelati vulnerabili, tanto che sono stati previsti meccanismi automatici di cambio dell'algoritmo in caso di necessità.

Per proteggere ulteriormente l'informazione si potrebbe codificare il messaggio in modo tale che esso possa viaggiare crittato e, una volta a destinazione verrebbe decodificato e interpretato nelle sue varie parti.

Un algoritmo che sarebbe possibile utilizzare potrebbe essere l'MD5, Message Digest algorithm 5. Questo tipo di codifica prende in input una stringa di lunghezza arbitraria e produce in output una firma digitale sotto forma di stringa a 128 bit (ovvero con lunghezza fissa di 32 valori esadecimali, indipendentemente dalla stringa di input).

3.4 L'applicativo Back-End

Come già introdotto nella premessa del capitolo, il sottosistema Back-End è sviluppato in maniera differente rispetto a quanto affrontato in fase di analisi per quello che riguarda il metodo di ricezione. Il relativo applicativo è stato perciò suddiviso in due parti, quella relativa allo smartphone che nella logica sostituisce il modulo GSM chiamata PhoneServer e quella relativa al server, denominata per l'appunto Server.

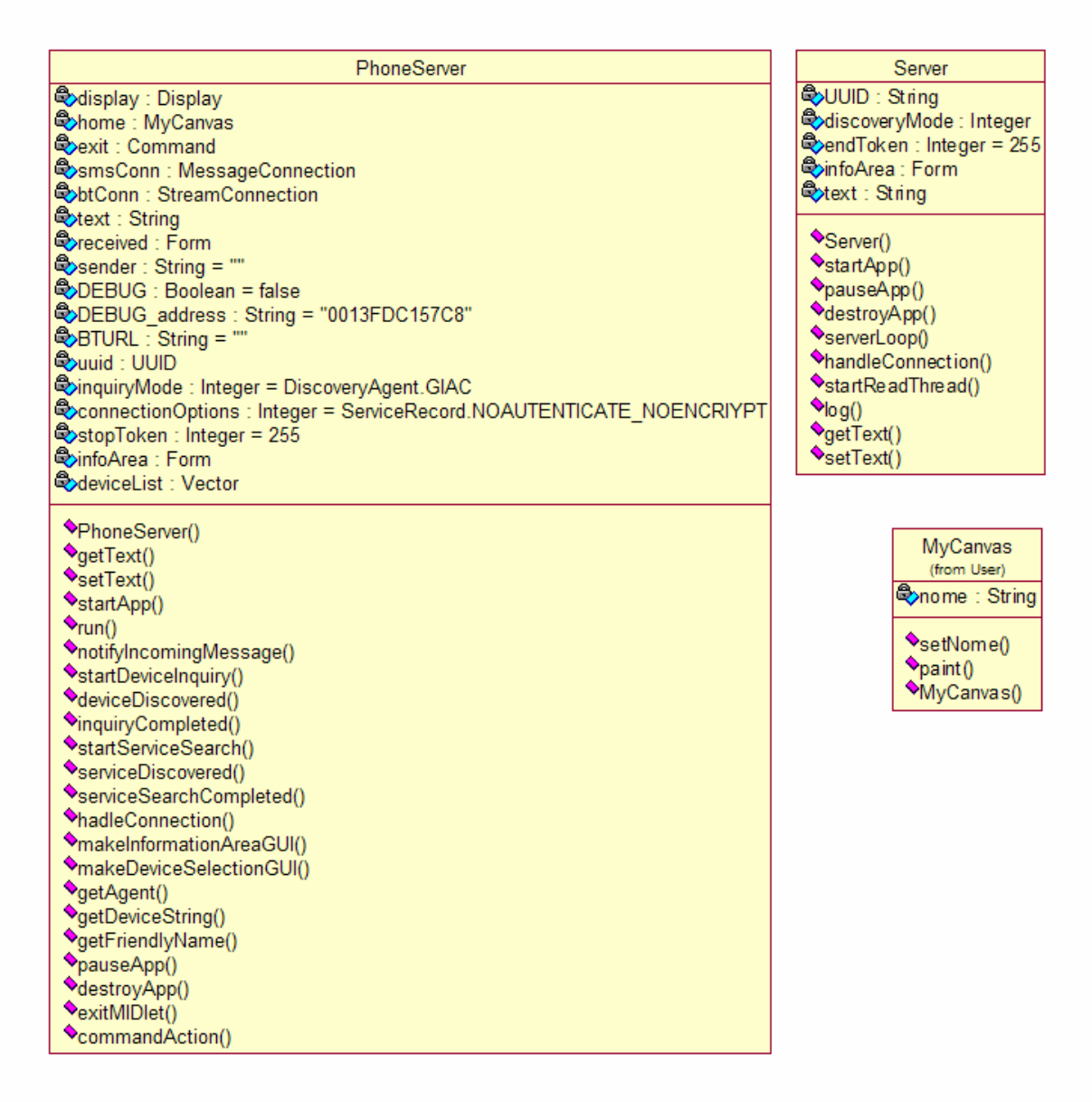

Figura 12 - Vengono riportati solo i nomi dei metodi per questione di leggibilità e non la firma intera

In Figura 12 vengono riportate le classi MIDlet relative al sottosistema Back-End. Come si può notare è ancora presente la classe MyCanvas in quanto la struttura grafica del PhoneServer è stata voluta mantenere coerente con la MIDlet utente SMSSender.

Per prima cosa analizziamo proprio la MIDlet PhoneServer in quanto, nella logica del processo completo, entra in gioco prima. Essa si occupa di eseguire lo *Store-and-Forward* dei messaggi che arrivano, tenendoli in memoria nel caso la connessione con il server non fosse momentaneamente

disponibile, e attivando una connessione Bluetooth con la MIDlet Server per poterli inoltrare. La funzione che si occupa di ricevere i messaggi è notifyIncomingMessage(MessageConnection listen) :

```
public void notifyIncomingMessage(MessageConnection listen) {
        Message msg = null;
        try {
            msg = listen.receive();
        }catch (Exception e) {e.printStackTrace();}
        if(msg instanceof TextMessage);
        TextMessage tmsg = (TextMessage)msg;
        sender = msg.getAddress()+ ":5000";
        text = tmsg.getPayloadText();
        try{
            received.append(msg.getAddress() + ":" + text);
            display.setCurrent(received);
        }catch(Exception e) {e.printStackTrace();}
        new Thread(this).start();
    }
```

richiamata dalla funzione principale startApp() della MIDlet tramite il blocco:

```
try {
    smsConn =(MessageConnection)Connector.open("sms://:5000");
    smsConn.setMessageListener(this);
    } catch (Exception e) {e.printStackTrace();}
```

Il suo compito è quello di rimanere in ascolto sull'indirizzo "sms://:5000" e ricevere i messaggi che giungono, azione eseguita dall'istruzione `msg = listen.receive()`. Dopo aver ricevuto il messaggio ne ottiene il PayLoad e, assieme all'indirizzo mittente ricavato tramite la funzione `msg.getAddress()`, aggiorna un'opportuna Form dedicata alla visualizzazione dei messaggi arrivati.

Come si può notare alla fine della funzione troviamo l'istruzione `new Thread(this).start()` la quale si occupa di inizializzare un nuovo thread che si provvederà ad attivare una comunicazione con il server tramite Bluetooth e una SMS per la risposta all'utente.

Nella Figura 13 si può notare la schermata del PhoneServer che evidenzia il messaggio arrivato dall'utente con indirizzo sms://+5550002 e PayLoad PayPal:5€.

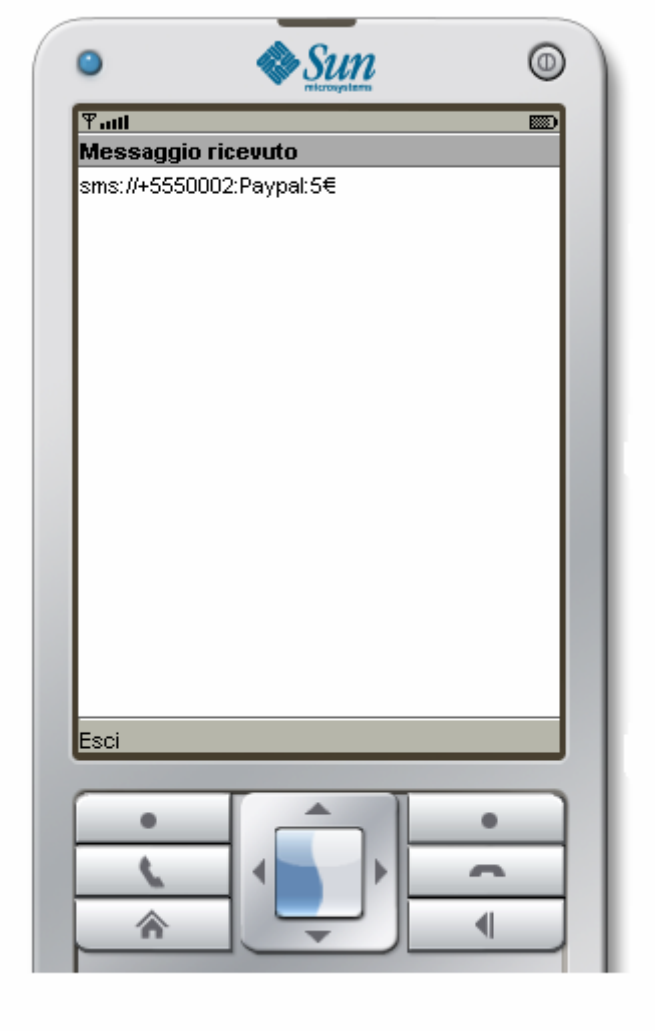

Figura 13

Da questo punto inizia la parte più delicata del processo Back-End, ovvero l'attivazione di una connessione Bluetooth con l'applicazione Server per poter inoltrare il messaggio in modo che ne venga attuata la richiesta, la generazione di una risposta e l'invio di quest'ultima all'utente.

La logica di funzionamento del processo è rappresentata dal Sequence Diagram in Figura 14 dove per prima cosa verrà negoziata una connessione Bluetooth tra i due dispositivi; dopodichè, come spiegato in precedenza, il PhoneServer rimane in attesa dell'arrivo di messaggi da parte dell'utente e, al momento opportuno, inizia a comunicare tramite la connessione stabilita inoltrando il messaggio ricevuto.

Il Server si occuperà di gestire la richiesta contenuta nel messaggio e notificherà l'esito al PhoneServer che provvederà a rispondere all'utente.

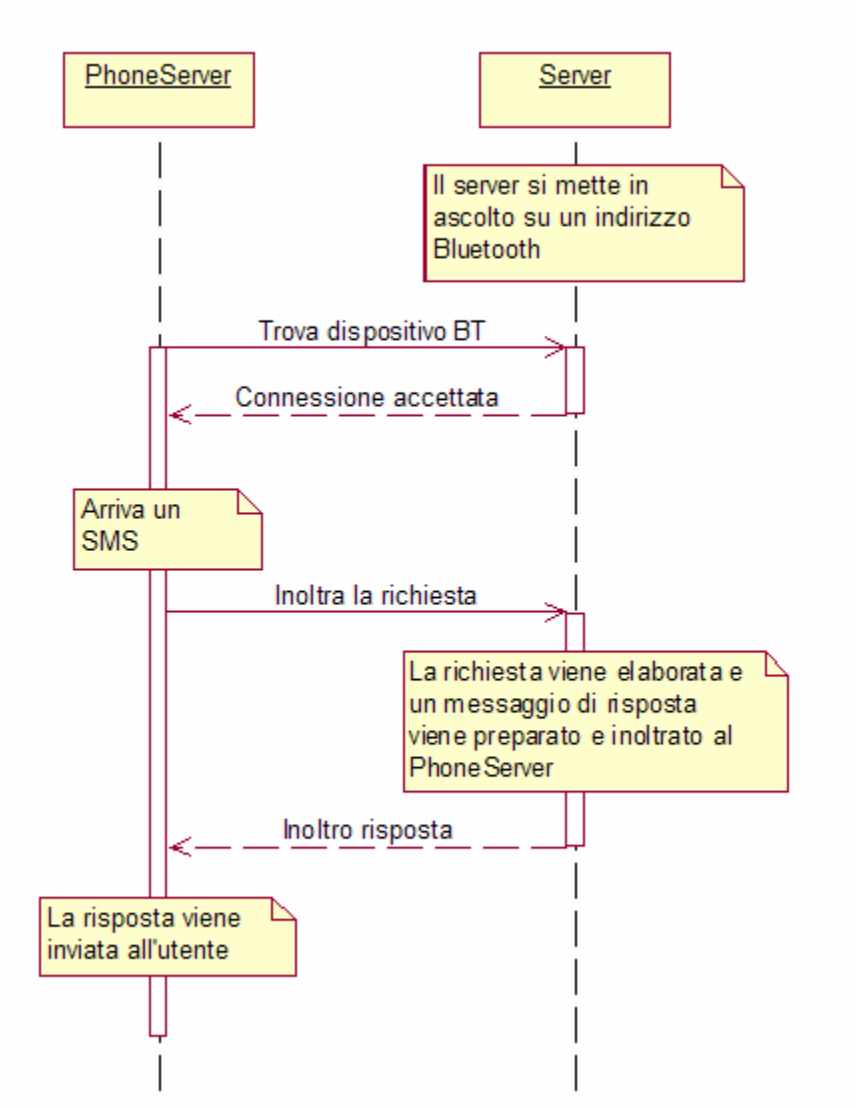

Figura 14

L'attivazione della connessione Bluetooth è affidata alla funzione `startDeviceInquiry()` così strutturata:

```
private void startDeviceInquiry() {
        try {
            log("Start inquiry method - this will take few seconds...");
            DiscoveryAgent agent = getAgent();
            agent.startInquiry(inquiryMode, this);
        } catch (Exception e) {
            log(e);
       }
 }
```

dove la funzione getAgent() è

```
private DiscoveryAgent getAgent() {
        try {
            return LocalDevice.getLocalDevice().getDiscoveryAgent();
        } catch (BluetoothStateException e) {
            throw new Error(e.getMessage());
        }
    }
```

In sostanza con questi due metodi il PhoneServer si mette alla ricerca di dispositivi Bluetooth disponibili. Nel nostro caso l'applicazione Server sarà in attesa di dispositivi con cui attivare una connessione tramite il metodo:

```
public void serverLoop(StreamConnectionNotifier notifier) {
        try {
            while (true) { // infinite loop to accept connections.
                log("Waiting for connection...");
                handleConnection(notifier.acceptAndOpen());
            }
        } catch (Exception e) {
            log(e);
        }
    }
```

e inizializzerà i canali di comunicazione tramite il metodo handleConnection(StreamConnection conn), definito in questo modo:

```
private void handleConnection(StreamConnection conn) throws IOException {
        DataOutputStream out = conn.openDataOutputStream();
        DataInputStream in = conn.openDataInputStream();
        startReadThread(in, out);
    }
```

Come possiamo notare, la connessione Bluetooth mette a disposizione due canali di comunicazione DataOutputStream out e DataInputStream in. Questo per permettere che la comunicazione sia bidirezionale e continua per ogni connessione attiva, a differenza di quella SMS per la quale vi è una connessione per ogni comunicazione: in sostanza, una volta che i dispositivi saranno connessi potranno attivare più comunicazioni Full-Duplex, senza che si debbano connettere ogni volta.

Figura 15

La Figura 15 sintetizza la situazione iniziale: nella schermata di destra viene visualizzato l'applicativo Server* in attesa di connessione, mentre in quella a sinistra l'applicativo PhoneServer che ha individuato il dispositivo a cui connettersi.

A connessione Bluetooth attiva, un messaggio che il PhoneServer riceve deve essere inoltrato al Server. Prima abbiamo notato che per ogni messaggio giunto viene istanziato un nuovo thread il quale per l'appunto, oltre al

* Per quanto riguarda l'applicativo Server si è deciso di implementare anch'esso come una MIDlet: questo è stato necessario in quanto, usando ambienti di sviluppo NetBeans e Wireless Toolkit, per poter eseguire dei test in maniera efficace ed esaustiva questa scelta era la più agevole: non cambia la logica di funzionamento e nel caso si volesse una classe Java standard il codice sarebbe sostanzialmente identico, fatta eccezione per la parte grafica.

compito di rispondere all'utente, deve inizialmente inoltrare il messaggio al Server. Tale comunicazione è gestita dalla funzione `handleConnection`:

```
private void handleConnection(final String url) {
        Thread echo = new Thread() {
            public void run() {
                try {
                    log("Connecting to server by url: " + url);
                    btConn = (StreamConnection) Connector.open(url);
                    DataInputStream in = btConn.openDataInputStream();
                    DataOutputStream out = btConn.openDataOutputStream();
                    out.writeUTF(text); //Inoltra il messaggio al server
                    String r = in.readUTF(); // risposta dal server
                    setText(r); //Setta la riposta per il client
                    log("Read " + getText());
                    out.flush();
                } catch (IOException e) {
                    log(e);
                }
            }
        };
        echo.start();
    }
```

Come si può notare in questa funzione del PhoneServer vengono utilizzati i canali di comunicazione citati in precedenza per poter inoltrare il messaggio e attenderne la risposta elaborata dal Server. Logicamente lato Server ci sarà una funzione analoga e speculare che invece che scrivere prima e leggere poi, prima legge la richiesta e poi ne scrive la risposta.
Dopodichè tale risposta verrà utilizzata dal PhoneServer per poter informare l'utente dell'avvenuta operazione.
La Figura 16 mostra per l'appunto la situazione al momento dell'inoltro del messaggio tramite la connessione Bluetooth. Nella schermata di sinistra notiamo il PhoneServer che, come in precedenza, visualizza il messaggio che gli è giunto da un utente, mentre nella schermata di destra, quella dell'applicativo Server, vi è una voce "read:Paypal:5€" riferita alla ricezione del messaggio e sotto una voce relativa alla conferma dell'avvenuta operazione.

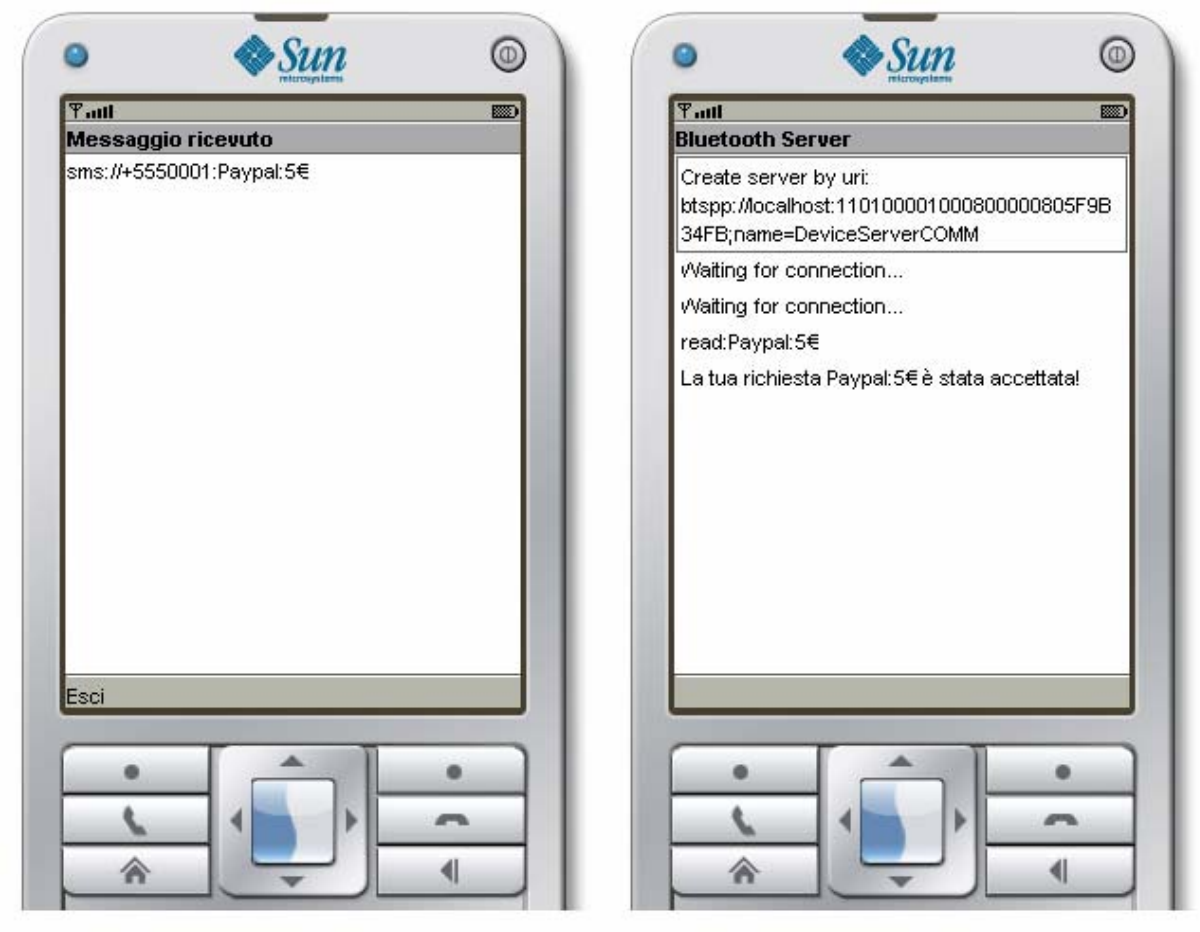

Figura 16

L'operazione di risposta esegue lo stesso medesimo percorso visto sin'ora ma in maniera speculare: tramite Bluetooth inoltra il messaggio dal Server al PhoneServer il quale, con lo stesso procedimento esposto per l'invio del messaggio utente, invia la risposta tramite un nuovo thread. L'utilizzo di thread per gestire le connessioni non è obbligatorio ma consigliato in quanto quest'ultime sono operazioni delicate e potenzialmente bloccanti.

Il solo aspetto differente tra l'invio della richiesta tramite SMS e la ricezione della risposta è che l'utente a sua insaputa invia il messaggio alla MIDlet PhoneServer su una porta specificata, mentre PhoneServer recapita la notifica nella casella Inbox del mittente.

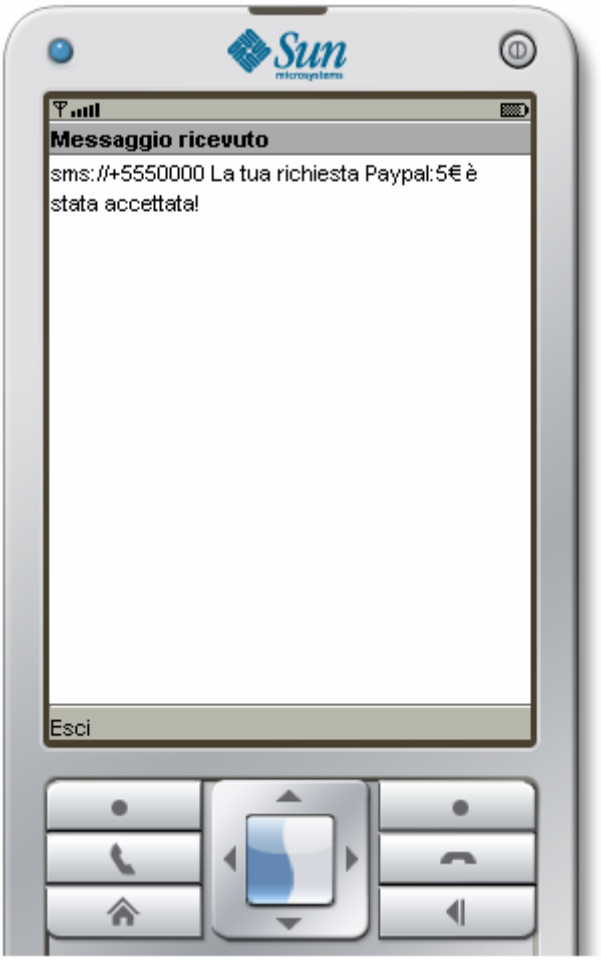

Figura 17

In Figura 17 viene visualizzata la casella Inbox del terminale utente con il messaggio di notifica prodotto dal Server.

4 CAPITOLO 4: Considerazioni sul prototipo

Con lo sviluppo del prototipo si è cercato di fornire una soluzione al problema posto che riuscisse a soddisfare i requisiti iniziali e che fosse fedele all'analisi svolta.
Questo breve capitolo delinea ciò che ci si aspettava di ottenere dal sistema, in particolare per verificare che le decisioni prese fossero corrette.

4.1 Comportamento ipotizzato

Il prototipo si proponeva di fornire una soluzione valida basandosi sulle caratteristiche elaborate in fase di progettazione.
In base al funzionamento dell'applicativo prodotto si poteva capire quindi se la struttura ipotizzata in partenza fosse adatta o meno alla soluzione del problema.
Per quando riguarda l'applicativo lato utente, a fronte delle scelte fatte ci si aspettava di riuscire a sviluppare un sistema semplice da utilizzare, che guidasse l'utente attraverso un percorso chiaro e logico. L'utilizzo di Form in combinazione con Canvas sembrava il miglior compromesso tra usabilità e fattore estetico e quindi la migliore delle soluzioni possibili per gestire l'interfaccia. Si è cercato di minimizzare i passaggi necessari per terminare l'operazione facendo sì che ogni schermata che l'utente visualizzava fosse comunque completa e che le voci relative all'azione dei pulsanti fossero esplicative di quello che sarebbe stato l'evento ad essi associato.
Con l'applicativo Back-End invece si cercava ottenere un sistema di ricezione messaggi che fornisse una risposta alla richiesta in maniera efficiente e sicura: anche in questo caso pochi passaggi per portare a termine il processo.
Con la modifica della struttura di ricezione si era previsto un passaggio extra che avrebbe complicato il meccanismo di comunicazione al server e che probabilmente avrebbe rallentato il processo, ma che comunque fosse in grado di sostituire la soluzione originale. Si è pertanto cercato di ottimizzare

la sequenza di operazioni relative all'inoltro dei messaggi attraverso i diversi dispositivi.

4.2 Osservazioni

Il software è stato sviluppato e testato con ambienti che emulano il funzionamento dei dispositivi mobili. Questo aspetto è di fatto un limite, in quanto non significa che un programma correttamente funzionante su un emulatore possa in realtà esserlo anche su device "fisici". Nel caso lo fosse, non c'è comunque la certezza che lo sia su tutti i dispositivi: esistono infatti sostanziali differenze tra i cellulari dei diversi produttori e a volte anche tra i diversi modelli dello stesso produttore.

Disposizione dei pulsanti, dimensioni e risoluzione del display sono per esempio dei fattori che vincolano non poco la programmazione della MIDlet che dovrà essere ospitata dal dispositivo, tant'è che per quasi ogni singolo modello è necessaria una propria versione.

Tale problema è stato constatato anche nei test che sono stati eseguiti per verificare che il software non fosse solo funzionante all'interno dell'ambiente di sviluppo. Sono stati infatti utilizzati due dispositivi Nokia, il primo uno smartphone e60 e il secondo un più comune 6020. Si è potuto notare infatti che le caratteristiche del display influenzavano non poco la navigazione e in più, mentre l'e60 si comportava come era stato previsto e testato in fase di programmazione con l'emulatore, il 6020 aveva un problema per quanto riguardava la disposizione dei tasti per accedere ai menu.

Si è perciò aggiunta una versione idonea in modo tale che l'applicativo funzionasse nella stessa maniera su entrambi i telefoni.

5 CONCLUSIONI

L'obiettivo principale di questa tesi è proporre un sistema grazie al quale il cittadino possa usufruire dei vantaggi offerti dalla tecnologia mobile per compiere azioni che era solito compiere in determinati luoghi e momenti.
Si è dunque cercato di individuare un servizio che potesse essere considerato sotto questo nuovo punto di vista, e tra le diverse tipologie messe a disposizione dal Comune di Bolzano ci si è concentrati su quello relativo alla ricarica di una carta prepagata per la ristorazione proposta agli studenti. Questo servizio ha fornito un'ottima base di partenza per il lavoro da svolgere, dando la possibilità di aggiornare l'aspetto della ricarica dal punto di vista tecnologico e di estendere il concetto ad altre tipologie di prepagate.
Si è perciò studiato il problema a fondo, cercando di capire quali potevano essere gli altri ambiti d'utilizzo, gli utenti ai quali tale sistema poteva essere rivolto e soprattutto a come doveva essere concepita la soluzione migliore.
In fase d'analisi ci si è concentrati ad individuare l'architettura idonea per poter strutturare un sistema che rispondesse agli obiettivi posti inizialmente e a concepire la logica di funzionamento che tale sistema doveva seguire. Dopodichè si è passati a sviluppare da un lato un applicativo che permettesse all'utente di effettuare vari tipi di ricariche soltanto inviando un messaggio alla propria banca e dall'altro un sistema che ricevesse tali richieste e ne producesse una risposta.
Si è dunque affrontato il processo nel suo insieme, di come cioè la comunicazione parta dall'utente, arrivi alla banca e torni al mittente sotto forma di notifica.
Il risultato ottenuto è quindi duplice: un software Java che offre la possibilità al cittadino di scegliere la carta prepagata e l'importo da ricaricare (o la possibilità di bloccarla piuttosto di richiederne il credito) e un sistema che ascolta le richieste e provvede a fornire gli esiti all'utente.
Tutto questo rispettando i requisiti elaborati in fase d'analisi e garantendo l'affidabilità del processo.

Proprio il fattore affidabilità è stato al centro dell'analisi del sistema, dovendo garantire una comunicazione robusta tra utente e banca senza interruzioni di servizio.
Dal punto di vista della sicurezza dei dati presenti nel messaggio ci si è affidati alla garanzia offerta dalle reti GSM, ma non per questo non si è pensato ad ipotizzare, per gli sviluppi futuri, un ulteriore livello di sicurezza: anche se per come è strutturato il software utente non vi è modo di conoscere in che modo viene composto il messaggio, quali informazioni esso contenga e a chi è destinato, potrà essere interessante in futuro poter fare in modo che il messaggio venga crittografato per garantire al processo maggiore affidabilità. Un algoritmo di codifica come MD5 può semplicemente essere introdotto in qualsiasi MIDlet per eseguire la codifica dei dati prima dell'invio, dati che saranno decodificati al momento della elaborazione da parte del software bancario.
Per questo e altri tipi di sviluppi futuri la logica dell'applicativo mobile è studiata appunto in modo modulare per essere facilmente aggiornabile. Tecnicamente si potrà per esempio rivedere il sistema per permettere pagamenti "sul luogo" tramite una transazione *prepagata – prepagata* solamente facendo comunicare tra due dispositivi Bluetooth gli estremi della transazione

Tali sistemi potrebbero quindi agevolare ulteriormente la vita quotidiana del cittadino, permettendogli di compiere operazioni per le quali prima era costretto a spostarsi in luoghi e in tempi prestabiliti.
Lo sviluppo delle tecnologie mobile andrà sicuramente avanti ma solo se le persone perderanno il loro scetticismo verso di esse, sistemi del genere potrebbero diventare un nuovo standard.

6 Bibliografia

Introduzione:

- *Principi Generali delle Reti Cellulari* – Renato Lo Cigno http://dit.unitn.it/locigno/didattica/wn/03-04/Cell-GSM-GPRS_H.pdf

Capitolo 1:

- *An Introduction to GSM* – Siegmund M. Redl, Matthias K. Weber, Malcolm W. Oliphant
- *Introduzione a J2ME e profili MIDlet* – Matteo Zinato http://www.wmlscript.it/j2me/
- *Introduction to SMS Messaging* – Developers' Home Team http://www.developershome.com/sms/smsIntro.asp
- *SMS - Short but Sweet* – Tom Clements http://developers.sun.com/mobility/midp/articles/sms/
- *School-Card - Sistema di pagamento elettronico* http://www.comune.bolzano.it/servizi_context02.jsp?area=51&ID_LINK=2627&page=1

Capitolo 3:

- *J2ME: Introduction, Configurations and Profiles* – Urs Steiner
- *A MIDlet Example Using the Wireless Messaging API and the Nokia SMS API* – Forum Nokia – Nokia
- *The Java APIs for Bluetooth Wireless Technology* – Qusay H. Mahmoud http://developers.sun.com/mobility/midp/articles/bluetooth2/
- *Introduction To Developing Networked MIDlets Using Bluetooth* – Forum Nokia – Nokia
- *Sicurezza delle comunicazioni Wireless – GSM Security* – Iskra Matto ftp://ftp.ge.cnr.it/pub/Sicurezza2004-05/gsm_iskra1.pdf