UNIVERSITÀ DEGLI STUDI DI TRENTO
FACOLTÀ DI SCIENZE MATEMATICHE, FISICHE E NATURALI

Corso di Laurea (triennale) in Informatica

Elaborato Finale

SISTEMI SOFTWARE COOPERATIVI PER DISPOSITIVI MOBILI: IL CASO DEI GRUPPI D'ACQUISTO SOLIDALI

Relatore: prof. Paolo Giorgini

Laureando: Andrea Detassis

ANNO ACCADEMICO 2008 - 2009

Indice

Introduzione

Il mondo economico in cui viviamo, in continua e rapida mutazione, influenza le nostre azioni spesso più di quanto vorremmo. La sua conformazione concorre in larga misura ad indirizzare le nostre scelte ed, in presenza di particolari configurazioni, può indurci a stimare in maniera errata quelle componenti che concorrono a determinare il valore di un bene.
Anche lo sviluppo tecnologico, in modo particolare la diffusione di dispositivi mobili come telefoni cellulari e palmari, si muove in fretta e, vista la sua attuale disponibilità, offre uno strumento in grado di agevolare l'attività di realtà economiche "indipendenti".
Sempre più spesso piccoli gruppi di consumatori riescono a soddisfare le loro esigenze con l'ausilio anche di strumenti tecnologici.
Un esempio concreto di questo modello può essere trovato nella crescente diffusione dei così detti GAS:gruppi di acquisto solidale.

Queste organizzazioni attuano una forma di commercio cooperativo, e applicano valori condivisi, quali la provenienza locale di un prodotto, genuinità e sostenibilità.

Esse possono avere dimensioni e forme organizzative differenti ma, soprattutto nei piccoli gruppi, esistono delle concrete difficoltà nell'amministrazione interna dei prodotti, dettate da vari fattori. Uno di tali problemi è la comunicazione, ed è qui che la tecnologia può intervenire, agevolando la gestione di tali realtà.

Il diffuso utilizzo di telefoni cellulari e PDA, con tutte le potenzialità della comunicazione mobile, ci invita a pensare a dei sistemi di supporto per questo tipo di problema, offrendo una soluzione sempre a portata di mano.

La soluzione che viene studiata in questa tesi si pone quindi due obiettivi principali: dapprima quello di studiare un principio funzionale che riesca a conciliare al meglio la vendita dei prodotti all'interno del GAS. con il legame cooperativo che ha portato gli individui ad effettuare questa scelta commerciale.

In secondo luogo, riuscire a creare un meccanismo di transazione chiaro e facilmente accessibile, che non richieda particolari accorgimenti od operazioni da svolgere da parte

dell'utilizzatore.

Tale applicazione deve quindi riuscire ad agevolare e semplificare le operazioni necessarie al corretto funzionamento del meccanismo di vendite, permettendo una consultazione aggiornata e intuitiva dei possibili acquisti ed una facile formulazione delle richieste, mantenendo lo spirito cooperativo che caratterizza questa realtà.

La tesi è quindi strutturata in tre capitoli.

Nel primo viene effettuata un'analisi del contesto, relativa inizialmente al concetto di cooperazione e, successivamente, ai Gruppi di Acquisto Solidale, fornendo una panoramica sulle caratteristiche che li definiscono.

Viene inoltre aperta una parentesi descrittiva sullo sviluppo delle applicazioni su dispositivi mobili, introducendo la tecnologia che verrà utilizzata per l'implementazione del sistema: Java Micro Edition.

Il secondo capitolo tratta invece i requisiti fondamentali, che caratterizzeranno il sistema, definendo i vincoli ai quali esso dovrà sottostare.

Definiti questi ultimi, che possono avere un'origine non unicamente dettata da esigenze tecniche, viene offerta una panoramica della struttura progettata, analizzando i componenti fondamentali e sottolineando le difficoltà incontrate.

L'ultima parte descrive invece, alla luce delle riflessioni e della progettazione effettuate, le scelte implementative intraprese per la realizzazione del software.

Verranno infine analizzati tutti gli artifici utilizzati per la costruzione delle componenti necessarie al corretto funzionamento dell'applicazione, analizzando in seguito alcuni possibili sviluppi.

Capitolo 1: Cooperazione e Gruppi di Acquisto Solidale

In questo capitolo verranno delineati i concetti fondamentali che ci permetteranno di affrontare gli argomenti in questione con maggiore semplicità e accuratezza.
Risulta quindi necessario, come primo passo, un piccolo accenno al concetto di cooperazione, che sta alla base di tutto il sistema studiato.

1.1 Introduzione al concetto di cooperazione

Per cooperazione si intende un'"organizzazione di lavoro in comune, per il conseguimento, senza intermediari, di un dato fine di produzione, di consumo o di credito."[1]
Inoltre è opportuno sottolineare come il concetto di cooperazione inglobi necessariamente quello di beneficio comunitario.
La cooperazione difatti non ha come scopo il raggiungimento di un profitto economico ma bensì la tutela economica dei soci che ne fanno parte, garantendo loro un beneficio esteso all'intera comunità.
Il concetto di cooperazione è strettamente legato a quello di società cooperativa.
"*Capisaldi del sistema cooperativo sono i principi di mutualità, solidarietà, democrazia*."[2]
Difatti, citando la Costituzione Italiana: "*la Repubblica riconosce la funzione sociale della cooperazione a carattere di mutualità e senza fini di speculazione privata*."[3]
Pertanto, è possibile notare come la valenza sociale della mutualità all'interno delle cooperative sia fondamentalmente legata con l'assenza di speculazione privata, come avviene invece nelle società di capitali.

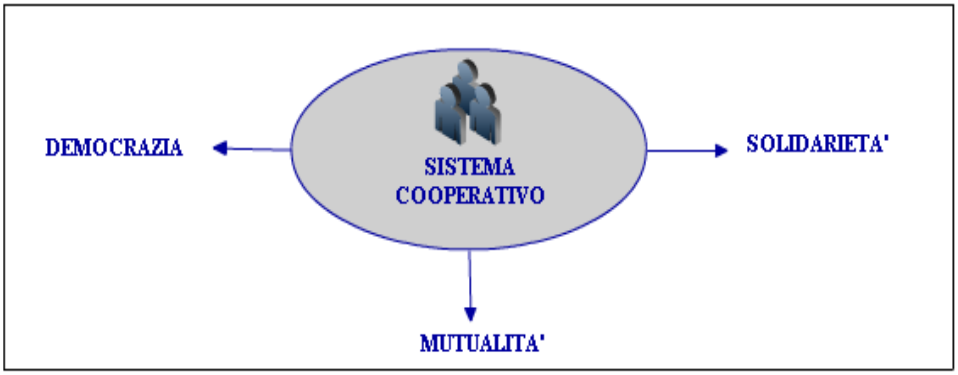

Figura 1: i valori di un sistema cooperativo

La cooperativa si distingue dalle società di capitali in quanto a differenza di queste, poste a scopo di lucro, essa si basa sul concetto di mutualità tra i lavoratori, consumatori e operatori che ne fanno parte.

L'aggettivo "mutuo" significa *"definito dall'esistenza di una partecipazione reciproca, comune, reciproco, scambievole*"[4], attributo che, nel caso delle società cooperative, consiste nella libera collaborazione tra più individui al fine del raggiungimento di uno scopo comune.

Un'ulteriore differenza tra società di capitali e cooperativa risiede nella destinazione del reddito conseguito: mentre infatti nella prima gli utili vengono divisi tra gli azionisti, nella seconda i redditi vengono solitamente reinvestiti nello sviluppo e nel rafforzamento della cooperativa stessa, secondo il principio delle così dette "risorse indivisibili".

Tale diversificazione comporta che, mentre gli azionisti di un'impresa sono i veri e propri proprietari dell'azienda, i soci della società cooperativa sono unicamente gestori di un patrimonio comune, in genere fortemente legato al territorio con il fine di trasmetterlo alle generazioni future. Fine proprio della cooperativa è dunque quello di assicurare a tutti soci che ne fanno parte condizioni migliori rispetto a quelle presenti sul mercato.

La definizione di cooperativa qui portata si mostra inoltre in pieno accordo con la definizione fornita dalla "*Dichiarazione di identità cooperativ*a"[5] approvata dal XXXI Congresso dell'Alleanza Cooperativa Internazionale, la quale definisce la stessa come "*un'associazione autonoma di persone che si uniscono volontariamente per soddisfare i*

propri bisogni economici, sociali e culturali e le proprie aspirazioni attraverso la creazione di una società di proprietà comune e democraticamente controllata." Inoltre tale documento include alcuni dei valori sui quali la cooperativa di fonda: "l*e cooperative si fondono sui valori dell'autosufficienza (il fare da sé), dell'autoresponsabilità, della democrazia, dell'uguaglianza, dell'equità e solidarietà. Fedeli allo spirito dei propri padri fondatori, i soci delle cooperative credono nei valori etici dell'onestà, della trasparenza, della responsabilità sociale e dell'attenzione verso gli altri."* Infine, tale dichiarazione individua i 7 principi che ne stanno alla base, i quali devono essere intesi sia come punti fermi per la sviluppo del movimento cooperativo che come linee guida per lo sviluppo, il giudizio dei comportamenti da adottare e la formulazione delle decisioni da intraprendere.

"I principi cooperativi sono linee guida con cui le cooperative mettono in pratica i propri valori.

- *1° Principio: Adesione libera e volontaria*

Le cooperative sono organizzate volontarie aperte a tutte le persone in grado di utilizzarne i servizi offerti e desiderose di accettare le responsabilità connesse all'adesione, senza alcuna discriminazione sessuale, sociale, razziale, politica o religiosa.

- *2° Principio: Controllo democratico da parte dei soci.*

Le cooperative sono organizzazioni democratiche, controllate dai propri soci che partecipano attivamente alla definizione delle politiche e all'assunzione delle relative decisioni. Gli uomini e le donne eletti come rappresentanti sono responsabili nei confronti dei soci. Nelle cooperative di primo grado, i soci hanno gli stesso diritti di voto (una testa, un voto), e anche le cooperative di altro grado sono ugualmente organizzate in modo democratico.

- *3° Principio: Partecipazione economica dei soci*

I soci contribuiscono equamente al capitale delle proprie cooperative e lo controllano democraticamente. Almeno una parte di questo capitale è, di norma, proprietà comune della cooperativa. I soci, di norma, percepiscono un compenso limitato sul capitale sottoscritto come condizione per l'adesione. I soci destinano gli utili ad alcuni o a tutti gli scopi: sviluppo della cooperativa, possibilmente creando delle riserve, parte delle

quali almeno dovrebbero essere indivisibili; erogazione di benefici per i soci in proporzione alle loro transazioni con la cooperativa stessa, e sostegno ad altre attività approvate dalla base sociale.

- *4° Principio: Autonomia ed indipendenza*

Le cooperative sono organizzazioni autonome, di mutua assistenza, controllate dai soci. Nel caso in cui esse sottoscrivano accordi con altre organizzazioni (incluso i governi) o ottengano capitale da fonti esterne, le cooperative sono tenute ad assicurare sempre il controllo democratico da parte dei soci e mantenere l'autonomia della cooperativa stessa.

- *5° Principio: Educazione, formazione ed informazione*

Le cooperative s'impegnano ad educare ed a formare i propri soci, i rappresentanti eletti, i managers e il personale, in modo che questi siano in grado di contribuire con efficienza allo sviluppo delle proprie società cooperative. Le cooperative devono attuare campagne di informazione allo scopo di sensibilizzare l'opinione pubblica, particolarmente i giovani e gli opinionisti di maggiore fama, sulla natura e i beneficie della cooperazione.

- *6° Principio: Cooperazione tra cooperative*

Le cooperative servono i propri soci nel modo più efficiente e rafforzano il movimento cooperativo lavorando insieme, attraverso le strutture locali e nazionali, regionali ed internazionali.

- *7° Principio: Interesse verso la comunità*

Le cooperative lavorano per uno sviluppo durevole e sostenibile delle proprie comunità attraverso politiche approvate dai propri soci."[6]

La cooperativa considera in maniera paritaria tutti i soci che ne fanno parte facendosi dunque garante di quello che il concetto di democrazia economica. Ma non solo, crede inoltre che le procedure democratiche applicate alle attività economiche siano inoltre fattibili, auspicabili ed efficienti. E ritiene infine che organizzazioni economiche democraticamente controllate contribuiscano al bene comune. Nella cooperazione difatti vige un principio che assegna uguale potere decisionale a tutti i soci ("una testa un voto") dividendo pertanto in maniera paritaria diritti e doveri.

1.2 I Gruppi d'Acquisto Solidali

Una volta introdotto il concetto di cooperazione, è necessaria un'altra definizione, propria dell'organizzazione aziendale, che ci permetta di introdurre il caso di studio.

Il clan

"*Il clan è un assetto in cui le relazioni (transazioni) vengono mediate con un feedback elevato usando il mutuo adattamento, e con un'alta collaborazione che implica fiducia e scambi su un periodo indefinito [Butler 1982]. Il clan implica un accordo sociale su un'ampia gamma di valori e convinzioni [...] e basa la sua capacità di controllo dei comportamenti organizzativi su un livello profondo di accordo tra i membri riguardo ciò che costituisce un comportamento corretto.[...] [Ouchi 1979].*"[7]

La combinazione di questi due elementi, cooperazione e clan, costituisce lo spirito che sta alla base della costituzione di un gruppo di acquisto solidale: esse sono delle realtà organizzative, cresciute "dal basso", che si prefiggono come scopo il consumo critico.

Questo prevede, una struttura organizzativa che tratteremo a breve, e dei solidi principi condivisi da parte di tutti coloro che fanno parte del gruppo. Il concetto di "solidale", infatti, possiede molte sfaccettature, soprattutto sul piano economico, che influenzano e distinguono un G.A.S. da un ordinario gruppo d'acquisto, il quale cerca unicamente di ridurre le spese centralizzando gli acquisti.

I capisaldi

"*Essere un GAS perciò non vuole dire soltanto risparmiare acquistando in grandi quantitativi, ma soprattutto chiedersi che cosa c'è dietro a un determinato bene di consumo: se chi lo ha prodotto ha rispettato le risorse naturali e le persone che le hanno trasformate; quanto del costo finale serve a pagare il lavoro e quanto invece la pubblicità e la distribuzione; qual è l'impatto sull'ambiente in termini di inquinamento, imballaggio, trasporto... fino a mettere in discussione il concetto stesso di consumo ed il modello di sviluppo che lo sorregge.*"[8]

Questo tipo di organizzazione tiene quindi in considerazione un ampio numero di fattori che influiranno notevolmente sulla scelta dei contatti e dei prodotti che il GAS tratterà.

Uno degli aspetti centrali è il rapporto diretto con il produttore: questa caratteristica prevede la possibilità di vedere la zona di produzione del bene, in modo da garantire un riscontro vantaggioso sulla qualità e la tipologia di prodotto che viene proposto. Questo produce inoltre il grande vantaggio di poter acquistare anche da piccoli produttori locali, permettendo una crescita maggiormente ancorata sul territorio.

Un altro punto cardine, strettamente correlato al precedente, è il rispetto dell'uomo e dell'ambiente.Questo concorre a indirizzare gli acquisti del gruppo verso materiali e prodotti che siano stati generati da realtà conosciute e controllate. Questo permettedi valutare se il ciclo di lavorazione del materiale è produttivo e sostenibile per coloro che lo praticano, garantendo che le modalità di produzione di esso risultino compatibili con l'intero ciclo di vita del prodotto (inteso come una spirale che inizia con la produzione del bene e termina con il consumo di esso). A questo riguardo il riferimento alla biologicità del prodotto è sicuramente obbligatorio: esso infatti comporta necessariamente una diversa attenzione per l'ambiente nel particolare e per il mondo naturale in generale, considerando dunque come fondamentali i principi che consentono la riduzione degli agenti inquinanti.

Esistono inoltre dei vantaggi sia economici che sociali che nascono dall'acquisto in gruppo: ad esempio, per quanto concerne il trasporto, il fatto di eseguire un ordine comune e da un unico produttore e di effettuare ordini all'interno di una ristretta area territoriale, comporta numerosi vantaggi. Dal punto di vista economico tale scelta porta a non frammentare l'acquisto candidando un unico e piccolo produttore responsabile dell'ordine. Ciò riduce i costi di trasporto relativi al bene in quanto l'offerta non risulta più parzializzata (come se ciascun individuo si recasse da questi) e il costo si riduce. Inoltre tale scelta comporta anche dei vantaggi in termini ambientali e sociali. Difatti riducendo il trasporto cala, al pari del prezzo, anche l'inquinamento prodotto, mentre la scelta di produttori locali, legati al territorio di appartenenzam incrementa l'economia sul territorio e ne promuove i prodotti. In più ciò comporta a sua volta il consumo di

prodotti non solo locali ma anche stagionali, riducendo ulteriormente i costi e incentivando una cultura critica dell'acquisto e del alimentazione.

Organizzazione di un GAS

Esistono vari modelli organizzativi per la gestione di un gruppo di acquisto solidale: una prima forma prevede la creazione di un'Associazione, che richiede venga quindi di redirigere un Atto Costitutivo, la presenza di un Codice Fiscale, ed una serie di ulteriori accorgimenti burocratici, che consentono il gruppo ad essere effettivamente e legalmente riconosciuto.

Un'altra forma, più semplice, prevede invece una libera associazione di individui, un gruppo di persone che mosso da ideali e necessità comuni, decide di organizzarsi con la finalità di acquistare dei prodotti insieme, sempre nel rispetto dei criteri di scelta prima identificati.

Il meccanismo ed il principio economico che ne stanno alla base, alla luce delle motivazioni ideologiche sopra definite, è comunque quello di creare un gruppo che permetta l'acquisto combinato di risorse.

Esistono quindi diverse realtà e diverse modalità organizzative che permettono di raggiungere questo scopo: quella più comune, che sarà anche quella individuata per il caso di studio, è una modalità operativa che prevede la presenza di un referente, a "capo" del GAS.

Questa figura,responsabile, è quella che si onera di organizzare le vendite all'interno del gruppo: egli ha il compito di prendere contatti con i produttori, raccogliere gli ordini, e successivamente portare a termine la transazione.

Attraverso questa persona viene formalizzato tutto il principio di funzionamento che permetterà agli utenti di compiere i loro acquisti, in base alle proposte che egli effettuerà. Essa è, comunque, una sola figura di riferimento: è previsto, infatti, che ogni individuo che ne abbia la possibilità procuri nuovi contatti e nuovi produttori, in base ad una logica di scambio con altri gruppi o individui che condividono gli stessi interessi.

1.3 Amministrazione e gestione dei GAS

Esistono differenti maniere, all'interno di un GAS, di amministrare e gestire internamente le vendite dei prodotti e dello scambio di informazioni.

Nella maggior parte dei casi non vengono utilizzati strumenti specifici per l'amministrazione, vengono quindi utilizzati metodi e strumenti, quali l'organizzazione di riunioni e la comunicazione telematica, ad esempio attraverso l'uso della posta elettronica.

Una delle modalità più comuni è quella di effettuare una riunione periodica, alla quale partecipano tutti gli utenti interessati, nella quale vengono vagliate tutte le proposte di vendita derivanti dai fornitori, ed, eventualmente, vengono raccolti gli ordini che gli i componenti effettuano, cercando di far convergere le necessità di ognuno.

Questo metodo, però, è sconveniente per diverse ragioni: prima fra tutte la disponibilità di tutti gli individui nella partecipazione alla riunione, non è infatti semplice riuscire a soddisfare le esigenze di tutti, soprattutto nel momento in cui il gruppo cresce di numero. In secondo luogo non permette una gestione regolare ed efficiente dei prodotti: alcune tipologie di merce, come ad esempio alcuni cibi “freschi”, necessitano di tempi di acquisto, al fine di essere conservati, solitamente inferiori alla frequenza di queste riunioni.

Un altro metodo in uso è quello della comunicazione digitale, attraverso le email.

Anche questa modalità, però, presenta evidenti limiti: una volta infatti che il referente invia una email agli interessati, risulta molto difficoltoso riuscire a scindere le reali esigenze e necessità di coloro che stanno effettuando un ordinazione, rispetto alla mole di posta “di contorno” che viene generata dagli utenti che, solitamente, inseriscono nel thread di discussione domande e richieste che esulano dall'oggetto in questione.

Si deve inoltre valutare il fatto che molte persone non effettuano un controllo della mail quotidiano, alle volte nemmeno settimanale, per cui risulta difficile rispettare le scadenze.

Tuttavia esistono delle soluzioni specifiche, più innovative, nate con l'intento di agevolare il funzionamento di queste realtà.

Sono stati realizzati alcuni prodotti che, in maniere differenti, fungono da supporto alle attività dei GAS.

Equalway [9] è una piattaforma online autofinanziata che si pone come obiettivo la realizzazione di un framework collaborativo a favore dei gruppi di acquisto e di piccoli produttori. Attraverso questa piattaforma è quindi creare delle relazioni, anche dirette, con altri gruppo di acquisto e fornitori registrati nel sistema, in modo da creare una rete di contatti che agevoli le relazioni tra produttori e consumatori. Esso, quindi, prevede unicamente la "messa in condivisione" delle informazioni, incentivando gli utenti a compiere le transazioni direttamente, suggerendo come metodo di transazione l'incontro di persona.

Questo tipo di applicativo, quindi, non affronta il problema secondo l'ottica inquadrata da questa tesi in termini di obiettivi, ma evidenzia come il problema sia reale, e come, attraverso l'utilizzo della tecnologia, si possano agevolare tali realtà. Esso infatti è stato scelto perché, come nel nostro caso, si pone come fine quello di rendere l'utilizzo della tecnologia uno strumento messo a disposizione della causa, non snaturandone i principi che ne stanno alla base.

Gasdotto [10] è invece un applicativo che più si avvicina alla tipologia di lavoro che stiamo affrontando in questa tesi. Attraverso di esso, infatti, è possibile effettuare una gestione interna delle transazioni tra utenti di un GAS, utilizzando un'interfaccia web molto intuitiva. Viene offerta la possibilità di gestione dei fornitori, delle vendite e degli acquisti, differenziando le possibili azioni in base ai ruoli all'interno del GAS. È prevista quindi una duplice interfaccia: da una parte quella amministrativa che consenta l'organizzazione di tutte le informazioni necessarie al compimento delle vendite, dall'altra quella per l'utente finale, che gli consenta di effettuare gli acquisti.

La prima problematica riscontrata è la rigidità del meccanismo di vendita: viene a mancare la componente del gruppo, vista come condivisione delle risorse, portando più verso una visione di e-commerce, limitando molto la componente umana che porta ad effettuare tale scelta di acquisto.

Un'altro problema che si pone è l'utilizzo del sistema attraverso un applicativo per PC desktop: questo tipo di approccio, infatti, oltre a non rendere fruibile il servizio a chi poco incline all'uso del computer, non permette una consultazione costante ed aggiornata delle vendite. Una delle possibili soluzioni a questo tipo di problematica potrebbe essere l'implementazione di una parte client progettata per i dispositivi mobili.

1.4 Gestionale GAS per dispositivi mobili

Alla luce di quanto descritto finora, risaltano alcune problematiche pratiche che possono influire sulla qualità del servizio di gestione interna di un gruppo di acquisto solidale.

Il problema della comunicazione inerente agli acquisti, infatti, è presente in tutte le soluzioni sopra proposte, in maniera più o meno evidente a seconda dei casi.

Viene quindi studiato un sistema che, attraverso l'utilizzo di dispositivi mobili, permetta la gestione interna al GAS delle vendite tra gli utenti in maniera semplice ed efficace, offrendo la possibilità, attraverso pochi chiari passaggi, di portare a termine una transazione.

La finalità è quella di rendere immediata la gestione delle transazioni all'interno del GAS utilizzando uno strumento diffuso e pratico come il telefono cellulare: attraverso questo sistema, infatti, si è in grado di eliminare parte dei problemi legati alle metodologie precedentemente descritte.

Questo strumento permette di eliminare le attese dovute e alla consultazione del mezzo utilizzato per lo scambio di dati: come ad esempio l'utilizzo di un computer per consultazione dell'email o utilizzo di un software gestionale. Inoltre permetterebbe una semplice, immediata ed aggiornata consultazione delle sole informazioni legate alle transazioni a cui si è interessati, evitando di introdurre nel processo di vendita altre componenti che ne possano inficiare lo svolgimento.

Introduzione alla programmazione di dispositivi mobili

"*I programmi scritti per cellulari e palmari non sono semplicemente "software per piccoli computer". Non è "la taglia" dell'oggetto, né la sua potenza di calcolo a marcare la differenza rispetto alle applicazioni per PC desktop. Perlomeno questi non sono gli unici parametri da tenere in considerazione. A cambiare è il contesto di utilizzo e soprattutto i destinatari del software.[...] L'immediatezza e la rapidità di utilizzo devono essere assoluti, come avviene con il telefono di casa, la TV o il player DVD. Il telefono deve essere infallibile, come lo sono il frigorifero o il decoder per la TV digitale: non sono ammessi blocchi, rallentamenti, malfunzionamenti. Soprattutto non sono ammesse perdite di dati. […] Il telefono deve funzionare sempre e garantire velocità ed efficacia. Senza bloccarsi, senza rallentare l'esecuzione delle applicazioni, senza perdere dati. Soprattutto senza far consumare denaro senza che l'utente abbia esplicitamente dato consenso.*"[11]

Quest'introduzione, tratta dal Libro di Stefano Sanna Java Micro Edition, racchiude brevemente i concetti fondamentali che devono guidare uno sviluppatore nella progettazione di un software destinato ad essere eseguito su dispositivi mobili.

Esistono però una serie di restrizioni tecniche determinate dalla limitatezza dell'hardware che ospiterà il programma, che obbligano il progettista ad alcune fondamentali scelte: progettuali ed implementative.

La piattaforma Java Micro Edition

Uno dei linguaggi di programmazione più diffusi per lo sviluppo di software per dispositivi mobili è Java Micro Edition (noto anche come Java ME o J2ME), sviluppato da Sun Microsystems.

Questo linguaggio non è semplicemente un riadattamento della piattaforma Java" Standard Edition (J2SE), ma è la specializzazione del linguaggio Java, in tutte le sue componenti, per l'ambiente mobile: è una piattaforma disegnata ex novo, un'architettura appositamente progettata per soddisfare i particolari requisiti di questo settore applicativo. Un altro determinante fattore che scollega Java ME dalle sue sorelle

maggiori è la sua modularità: viene infatti concepita come una struttura fortemente personalizzabile in base alle necessità.

Questa sua proprietà intrinseca nasce dalla necessità di adattare il linguaggio alla moltitudine di tipologie differenti di dispositivi che il mercato offre: in base al tipo di apparecchiatura che dovrà ospitare l'applicazione sono necessarie, infatti, diverse caratteristiche tecniche da soddisfare.

Il concetto di modularità può venire definito attraverso l'individuazione di 3 componenti:

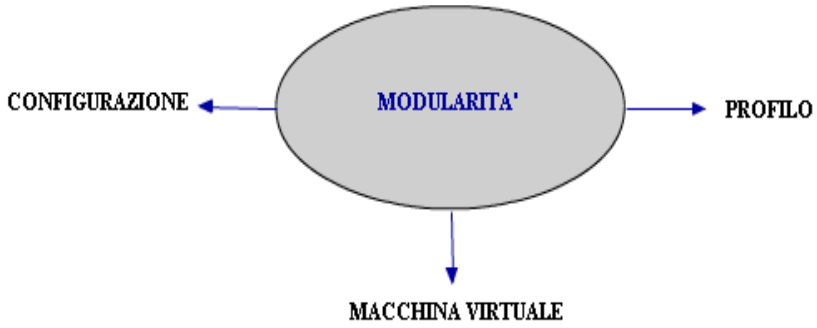

Figura 2: le componenti che definiscono la modularità

- *La macchina virtuale*

  È la parte incaricata di eseguire le applicazioni e tradurle in codice leggibile al processore del dispositivo interessato.

  "*La virtual machine è responsabile del caricamento del codice, dell'isolamento del runtime Java dal resto del sistema operativo, della gestione della memoria, dei thread e delle altre risorse fondamentali per l'esecuzione delle applicazioni*."[13]

  Questo tipo di architettura ha inoltre l'enorme vantaggio di essere altamente portabile, la virtual machine infatti può essere installata su diversi sistemi operativi, coprendo così una grande fascia di prodotti anche molto diversi fra loro.

- *La configurazione*

  La configurazione è un elemento inscindibile dalla macchina virtuale: essa infatti ha il compito di informare le applicazioni riguardo i servizi che la macchina virtuale può offrire, in base alla tipologia di hardware su cui sta girando, come ad esempio la presenza del supporto all'aritmetica in virgola mobile o di un eventuale connettività di rete.

  Sono previste due tipo di configurazioni: CDC (Connected Device Configuration):destinata ai prodotti hi-end, prevede dei requisiti abbastanza elevati per funzionare, come ad esempio 2MB di memoria e processori a 32 bit, e CLDC (Connected Limited Device Configuration): destinata invece ai prodotti con caratteristiche molto modeste, è la configurazione più semplice prevista.

- *Il profilo:*

  Il profilo invece è quella parte che permette di definire il dispositivo vero e proprio: "il profilo completa la configurazione definendo il modello applicativo, le classi per l'interfaccia utente (non necessariamente basata su componenti grafici) e le altre funzionalità che rendono l'ambiente Java ME pronto per l'esecuzione di un'applicazione."[14]

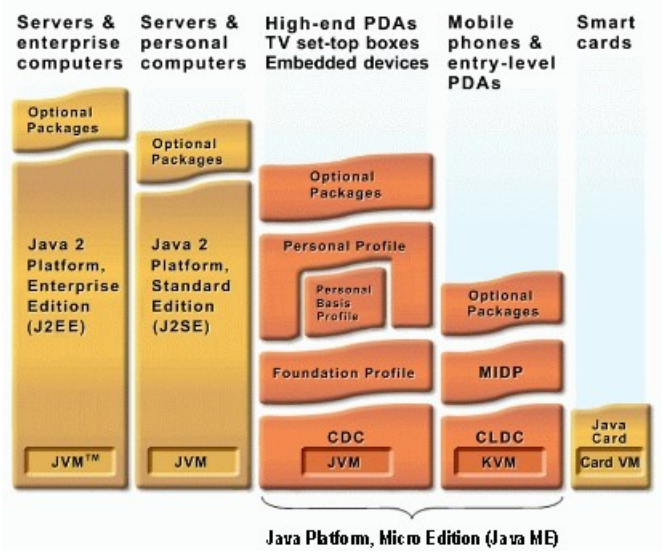

Figura 3: struttura dell'ambiente Java

Capitolo 2: Analisi e progettazione

Si vuole offrire la possibilità di amministrare le vendite interne ad un GAS, attraverso l'utilizzo di un telefono cellulare o di un PDA, in modo da poter utilizzare uno strumento che sia disponibile in ogni momento.

Definiremo 'vendita' la merce che il referente propone ai componenti del gruppo d'acquisto, e 'offerta' le richieste per un prodotto che gli utenti effettueranno.

Con il termine 'transazione', invece, verrà indicato lo svolgimento di una vendita, e dell'eventuale presentazione delle offerte, dal momento in cui essa viene posta in essere dal referente fino al termine temporale prestabilito.

2.1 Introduzione

Il sistema permette ai componenti del gruppo di effettuare un'offerta per uno stoccaggio di prodotti che il loro referente propone. In seguito, a transazione conclusa, organizza le offerte ricevute e comunica alle varie parti la quantità di "prodotti" ottenuta.

Più nel particolare la proposta di vendita posta in essere dal referente del gruppo viene inoltrata ai componenti potenzialmente interessati. Essa prevede un quantitativo massimo disponibile, una soglia minima affinché la vendita sia sostenibile e una scadenza per la presentazione delle offerte. Dall'altra parte, gli utenti potranno fornire un range di quantitativo a cui sono interessati, nell'ottica di una gestione cooperativa delle risorse disponibili all'interno del GAS in questione.

Ciò significa che, a differenza di un modello competitivo quale può essere un'asta, tutti gli offerenti potranno vedere soddisfatte le proprie richieste secondo una logica ridistributiva delle risorse possedute. Raggiunta la scadenza individuata dal referente, l'applicazione provvederà a informare gli offerenti circa la quantità di prodotto ottenuta, cercando di garantire sempre l'offerta minima da essi proposta.

2.2 Requisiti fondamentali

Vengono ora analizzati i requisiti che l'applicazione dovrà avere per essere funzionale al suo scopo:

Architettura client – server: viene definita una struttura centrale, scritta in Java, che, attraverso l'utilizzo di un archivio elettronico, funge da supporto ai client implementati invece in Java ME.

Connettività: Uno dei requisiti vincolanti per il funzionamento basilare di tutto il programma è il networking. Al fine di garantire il corretto funzionamento è indspensabile che il sistema si connetta ad Internet attraverso una connessione dati.

Comunicazione attraverso socket: la comunicazione tra le due componenti che formano l'applicazione deve avvenire tramite socket TCP, in modo da essere il più agevole possibile.

Archiviazione dati: La possibilità di archiviare dati in locale sul dispositivo in maniera ordinata e coerente è altresì fondamentale per il funzionamento del sistema. Questa infatti permette di agevolare la rielaborazione delle informazioni sul dispositivo e di consentirne una successiva consultazione offline

Pubblicazione di vendite: Il sistema deve consentire ad un referente di proporre delle vendite agli utenti del gruppo, permettendo la gestione di tutti i parametri necessari al suo corretto svolgimento. Esse, in particolare, saranno caratterizzate da due valori che indicheranno rispettivamente il quantitativo massimo di prodotto ordinabile, ed il quantitativo minimo necessario affinché la transazione possa essere portata a termine correttamente, entro una determinata scadenza imposta.

Interazione con i componenti del GAS: essi, dopo aver ricevuto una proposta a cui sono potenzialmente interessati, devono poter effettuare un'offerta per tale prodotto: anch'essa è caratterizzata da due valori, che indicano rispettivamente il quantitativo massimo di prodotto che si vuole ricevere, ed una soglia minima sotto la quale non considerare la propria offerta valida.

Gestione cooperativa delle transazioni: la gestione delle transazioni deve avvenire nel rispetto delle regole cooperative che stanno alla base della filosofia del GAS: la gestione

delle vendite deve quindi essere pensata in modo da garantire che ogni utente che effettua un'offerta per un determinato prodotto riceva almeno il quantitativo minimo da lui richiesto nel caso in cui la vendita vada a buon fine. Ciò avviene in virtù del fatto che l'ottenimento dei prodotti non avviene tramite un'asta, ma bensì attraverso una logica di distribuzione dei prodotti, che vengono visti non come merce ma come risorse all'interno dei GAS.

Facilità di utilizzo: una delle caratteristiche fondamentali è che l'applicazione sia progettata in modo da poter essere gestita in maniera intuitiva e pratica: le scelte possibili all'interno del software devono essere chiare e definite, in modo da poter garantire un buon impiego fin dalla prima esecuzione. Il programma deve essere progettato in modo da garantire una gestione semplice ed intuitiva dei dati e delle informazioni relative al commercio all'interno dei Gruppi di Acquisto Solidale. Questo requisito potrà essere soddisfatto anche da scelte grafiche che ne veicolino una immediata lettura tramite un percorso pre-impostato, che faciliti l'utente nelle scelte da effettuarsi.

2.3 Progettazione

Alla luce delle riflessioni sulle caratteristiche che il sistema dovrà avere, verranno ora valutati gli strumenti che ci permetteranno di portarlo a termine.
Verrà dapprima valutata l'architettura generale, che sarà di tipo client – server.

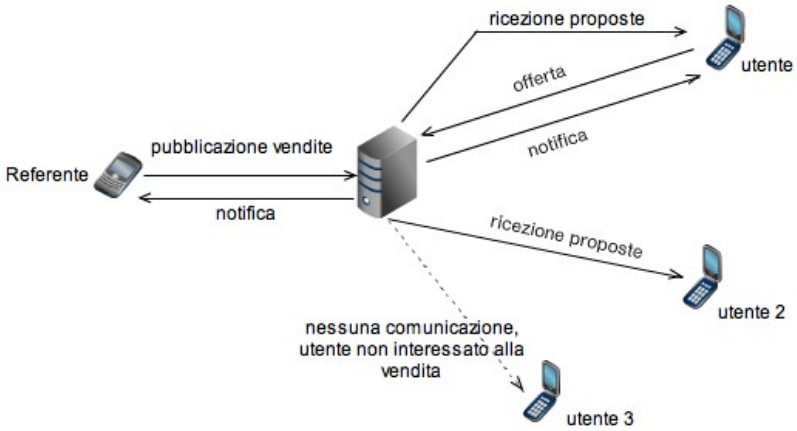

Figura 4: architettura logica del sistema

Il nocciolo dell'applicazione è costituito dalla parte server: essa infatti rielabora e organizza i dati ricevuti dai client, comunicando poi loro l'esito della transazione.
Il client d'altra parte ha la funzione di permettere all'utente di inserire i dati necessari alle transazioni, senza farsi carico di come questi verrano poi elaborati. La scelta di un'architettura così definita è determinata dal fatto che, con la presenza di un server centrale, si riduce notevolmente il carico di lavoro della parte client, che si trova su dispositivi mobili, migliorandone conseguentemente le prestazioni e l'usabilità.
Di seguito si riportano nello specifico le scelte di progettazione intraprese per le due macro-aree client e server.

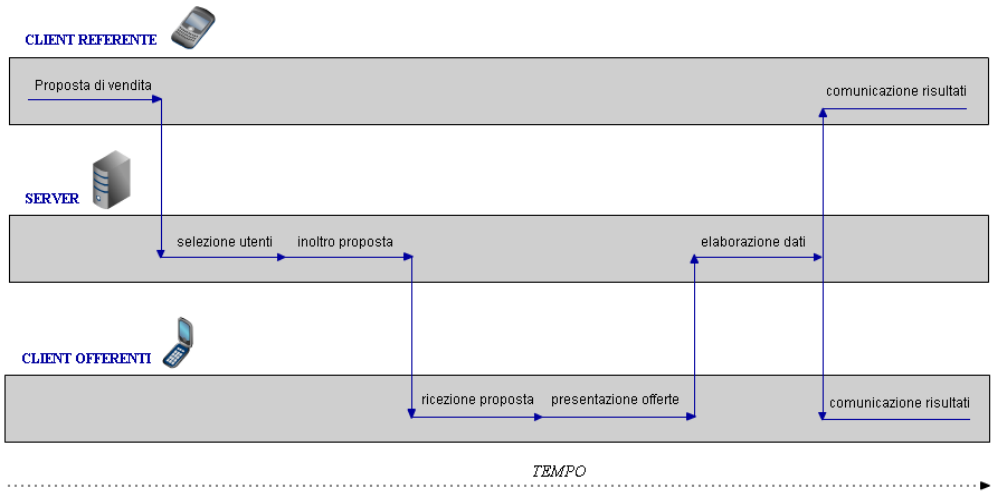

Figura 5: il protocollo

Il server

La parte lato server del sistema che stiamo progettando ha un ruolo fondamentale per la funzionalità dell'intero sistema.

La presenza di una componente server, infatti, permette di compiere numerose operazioni in maniera centralizzata, comportando numerosi vantaggi rispetto alla sola elaborazione su dispositivo mobile.

Esso si occupa sia della gestione e dell'immagazzinamento dei dati relativi agli utenti e alle loro transazioni, viste come vendite nel caso dei referenti e come offerte nel caso degli utenti, sia dell'elaborazione di questi dati per la conclusione della transazione, attraverso l'appoggio di altre strutture dati.

Analizziamo ora nel dettaglio quali saranno le funzionalità richieste:

- Gestione e immagazzinamento dei dati trasmessi dagli utenti: attraverso un database vengono qui archiviate tutte le informazioni utili al sistema per il funzionamento: dati di login per gli utenti, dettagli di vendite e offerte necessari alle transazioni.
- Matching dei potenziali utenti interessati ad una vendita con successivo inoltro della proposta ad ognuno di essi: l'applicazione, in base ai dati inseriti, deve

autonomamente informare gli utenti potenzialmente interessati e successivamente raccogliere le loro offerte nel caso vengano effettuate.

- Controllo delle scadenze delle vendite: quando le vendite poste in essere dai vari referenti raggiungono la scadenza temporale fissata per la raccolta delle offerte, esse devono essere evidenziate e predisposte per la conclusione della transazione.
- Gestione delle transazioni concluse: dopo esser stata riconosciuta, una vendita che ha raggiunto la scadenza prefissata per il termine delle offerte deve essere analizzata per determinare se, ed eventualmente con che risultati, essa è andata a buon fine, informando quindi gli utenti coinvolti.

I client

E' la parte visibile all'utente finale, progettata per essere installata ed eseguita su dispositivi mobili: è l'interfaccia che permette ai vari attori di effettuare le loro operazioni.

Nel nostro caso specifico il lato client è stato diviso in due applicazioni distinte per le due diverse tipologie di utenti presenti nel sistema: i referenti ed i compratori.

Questa scelta viene fatta con la finalità di rendere ancora più semplice l'utilizzo del software, fornendo all'utente che lo utilizza solo le parti e le funzionalità che possono essere utili al suo scopo, sia esso quello di vendere dei prodotti o di acquistarne.

Vediamo ora nel dettaglio quali sono i compiti che i client devono svolgere:

- *Client per referente:* è la parte client che viene utilizzata dal referente del GAS per poter proporre dei prodotti all'interno della comunità e gestirne la vendita.
  Deve quindi permettere la creazione di una nuova proposta di vendita in maniera facile ed intuitiva, permettendo l'introduzione di tutti i parametri utili alla pubblicazione della stessa attraverso l'uso di modelli preimpostati.
  Deve inoltre permettere una rapida visione d'insieme delle vendite in essere e di quelle terminate, offrendo la possibilità di consultare le eventuali offerte che gli utenti hanno fatto per ogni propria proposta di vendita, in corso o conclusa.

Viene inoltre richiesta la possibilità di vedere i dettagli, inseriti in fase di registrazione, degli utenti che appartengono al proprio gruppo di acquirenti, al fine di garantire un eventuale comunicazione diretta.

- *Client per Compratore:* è la parte client che viene utilizzata dagli utenti finali del GAS, che permette loro di ricevere le proposte di vendita a cui sono potenzialmente interessati.
  Deve garantire una facile e pratica consultazione delle varie proposte di vendita che il proprio referente pubblica, mantenendole aggiornate e mostrandone, all'occorrenza, tutti i dettagli. Deve inoltre fornire la possibilità di effettuare delle offerte per i prodotti a cui si è interessati, passando attraverso dei moduli preconfigurati che ne assicurino una semplice fruibilità.

Entrambe le applicazioni devono inoltre essere progettate per informare l'utente nel caso in cui avvenga qualche evento che lo possa direttamente interessare, come ad esempio la conclusione di una vendita o la presentazione di un'offerta per una proposta di vendita.

Durante la progettazione del sistema e lo studio di Java ME, sono state riscontrate una serie di problematiche e di limitazioni del linguaggio che hanno richiesto alcune scelte specifiche di tipo strutturale.

- *Indirizzo IP dinamico*: Il primo problema è legato alla connettività. Un dispositivo mobile dotato di una connessione dati, ottiene al momento dell'instaurazione della connessione un indirizzo IP dinamico dal proprio gestore del servizio. Il dispositivo diventa quindi irraggiungibile sulla rete a chiunque non conosca il suo indirizzo, non permettendo così la ricezione di dati da una fonte esterna. Questa dinamica precludeva quindi, nel nostro caso, la possibilità di ricevere dei dati dal server, se la comunicazione verso quest'ultimo non era stata instaurata dal client, limitando quindi la possibilità degli aggiornamenti necessari al corretto funzionamento del sistema di transazioni.
- *Assenza di serializzazione:* La seconda problematica riscontrata è anch'essa nell'ambito della comunicazione, non più in termini di indirizzamento dei dati, ma nella maniera di trasmetterli.

Una volta instaurato il canale di comunicazione tra il client ed il server viene subito riscontrata una prima difficoltà nella trasmissione dei dati tra i due sistemi, in quanto Java ME non prevede un metodo per la serializzazione e la deserializzazione dei dati, al contrario della piattaforma Java2 Standard Edition che prevede dei metodi appositi.

"*La serializzazione è un processo per salvare un oggetto in un supporto di memorizzazione lineare (ad esempio, un file o un'area di memoria), o per trasmetterlo su una connessione di rete. La serializzazione può essere in forma binaria o può utilizzare codifiche testuali (ad esempio il formato XML) direttamente leggibili dagli esseri umani. Lo scopo della serializzazione è di trasmettere l'intero stato dell'oggetto in modo che esso possa essere successivamente ricreato nello stesso identico stato dal processo inverso, chiamato deserializzazione.*"[15]

Questa mancanza è dovuta al grande lavoro che questi metodi producono per essere eseguiti, che non sarebbe sopportabile dalla capacità computazionale di un dispositivo mobile.

- *Gestione del Multithreading:* Un'ulteriore difficoltà incontrata durante la progettazione del sistema è stata quella di trovare una maniera efficace per gestire la concorrenza dei thread nell'applicazione.

  L'utilizzo dei thread è molto importante per il corretto funzionamento di un software, poiché permette di suddividere programmaticamente diverse operazioni in flussi separati ed indipendenti: la presenza del multithreading permette di amministrare questi flussi in maniera ottimale.

  J2ME offre una basilare gestione del multithreading, semplificata e ridotta rispetto alle sofisticate versioni presenti nelle piattaforme desktop ed enterprise, ma non sufficiente ad ottenere una gestione dei thread ottimale per l'esecuzione.

2.4 I protocolli

Una delle caratteristiche essenziali che il sistema deve avere, vista la sua natura architetturale, è un protocollo di comunicazione efficiente.

Le parti che compongono il sistema, infatti, per poter essere funzionali, richiedono uno scambio di informazioni tra di loro.

Gli utenti del GAS devono avere infatti la possibilità di ricevere le proposte di vendita che il loro referente pubblica, e devono essere messi nella condizione di formulare delle offerte per i prodotti a cui sono interessati.

Deve inoltre essere studiata una struttura che permetta l'aggiornamento costante di entrambe le parti client, rispetto alle operazioni che vengono compiute sui prodotti a cui sono direttamente interessati.

E' compito del server, dunque, amministrare le comunicazioni tra le parti coinvolte nel sistema, seguendo il seguente modello:

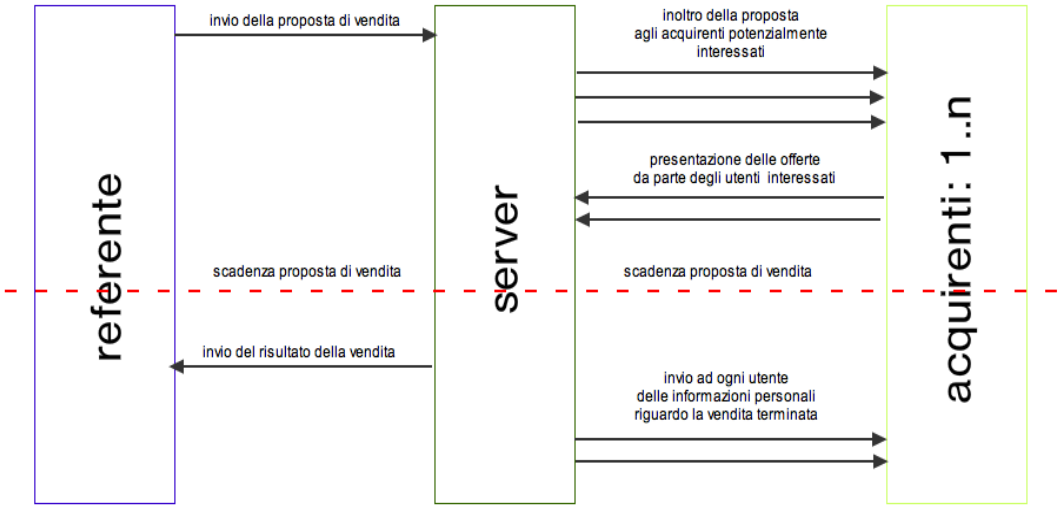

Figura 6: il protocollo di comunicazione

2.5 Le transazioni

In questo paragrafo verranno descritte le scelte progettuali che porteranno l'applicazione a portare a termine le contrattazioni all'interno del GAS, in maniera da rispettare il "vincolo di cooperatività".

Quest'ultimo infatti prevede che ogni offerta che viene effettuata da un utente in merito al prodotto posto in vendita da un referente debba essere potenzialmente soddisfatta.

Analizziamo ora le due componenti che formano una transazione: la vendita di un prodotto e le offerte per questo.

Proposta di vendita

Vediamo ora quali sono le caratteristiche ed i dati necessari affinché la vendita possa essere pubblicata.

Al momento della creazione della proposta di vendita vengono richiesti al referente i seguenti dati relativi al prodotto:

- *nome del prodotto*: per identificare cosa si sta vendendo
- *tipo di prodotto*: viene effettuata una divisione in 4 macro categrie (carne, pesce, verdura, frutta), in modo da restringere poi l'inoltro della vendita unicamente agli utenti interessati a quella tipologia di merce
- *prezzo*: definisce il prezzo unitario
- *scadenza*: indica il termine (data, ora) entro cui possono pervenire le offerte per il prodotto
- *quantitativo massimo*: indica la disponibilità massima che può essere venduta del prodotto
- *quantitativo minimo*: rappresenta una soglia minima di acquisto da raggiungere affinché la vendita sia conveniente anche per il produttore. Questa soglia minima può essere, ad esempio, legata alle necessità di trasporto o di immagazzinamento della merce

Offerta

Nel momento in cui un'utente riceve una proposta di vendita, egli può effettuare un'offerta per essa.

Al fine però di garantire il "vincolo cooperativo", l'offerta avrà una particolare caratteristica: il quantitativo di merce richiesta, infatti, non sarà formato da un unico valore (ad esempio 2 kg), ma da un range di valori (ad esempio da 1,5 kg a 2,5 kg).

L'utente definisce un valore minimo di prodotto che vuole ricevere, ed un valore massimo che è disposto ad acquistare.

Questo permette di avere un sistema più flessibile, che possa quindi cercare di soddisfare tutte le offerte degli utenti per una determinata proposta di vendita.

Una volta raggiunto la scadenza per la presentazione delle offerte, si deve analizzare la procedura per l'assegnazione delle parti destinate ad i vari utenti.

Questa ripartizione deve rispettare il principio di gestione cooperativa definito nella sezione dedicata all'analisi.

Assegnazione dei quantitativi

Il primo passo è quello di controllare se la vendita è garantita.

Esistono infatti due casistiche che portano la proposta di vendita a rimanere insoluta:

1) la somma del quantitativo massimo di ogni offerta non raggiunge la soglia minima necessaria per la vendita imposta dal referente.
2) la somma del quantitativo minimo di ogni offerta è superiore al quantitativo massimo disponibile.

Una volta accertatosi di non trovarsi in nessuna delle due precedenti situazioni, l'algoritmo cerca di soddisfare tutte le offerte con i valori massimi, in modo da fornire ad ogni utente il quantitativo massimo che ha richiesto al momento della formulazione dell'offerta.

Nel caso in cui la sommatoria dei valori massimi superi la soglia massima imposta dalla vendita, si prosegue effettuando la stessa procedura utilizzando invece i valori minimi per ogni offerta: questo è il requisito minimo per la conclusione della vendita con successo.

Quest'ultima situazione, infatti, garantisce ad ogni offerente il minimo che egli ha richiesto. Si deve ora trovare la maniera di gestire la parte rimanente, cioè la differenza tra il quantitativo massimo disponibile definito nella vendita, e la sommatoria del minimo di ogni offerta. Difatti, una volta verificato che la vendita è garantita, e che non ci si trova nel caso ottimale (cioè non si possono soddisfare tutte le offerte utilizzando il valore massimo inserito dall'utente), si utilizza il valore minimo come base di assegnazione delle risorse per ogni offerente.
Vengono definite due tipologie di ridistribuzione del rimanente: una di tipo "uniforme" ed una di tipo "proporzionale"; il referente, al momento della pubblicazione della proposta di vendita, sceglierà quale delle due vuole utilizzare.

- *La ripartizione uniforme*

Questo tipo di ridistribuzione è quella che, in termini teorici, più si avvicina al concetto di cooperatività in senso "solidale": essa infatti prevede che il rimanente venga equamente diviso tra tutti gli offerenti.
La parte in disavanzo viene quindi assegnata un unità alla volta agli offerenti che non abbiano già raggiunto il limite massimo che sono disposti a ricevere.

```
//dati che possiedo
Vendita ven;                // vendita
Offerte[] off;              // offerte riferite a vend
int numerOfferte;           // numero delle offerte per vend

int sommaMin    ;                           // sommo il valore minimo di ogni
   for(int i=0;i<=numerOfferte;i++){        // offerta in un int, in modo da calcolare
    sommaMin += off[i].getMin();            // la somma dei valori
     }
//calcolo ora la parte rimanente da assegnare
//sottraendo dal valore massimo della vendita la sommatoria prima calcolata
int rimanente = ven.max – sommaMin;

//effettuo ora l'assegnazione del rimanente agli offerenti
if(ven.max > sommaMin >= ven.min){              // condizione necessaria per la vendita
   while(rimanente > 0){                        // finché il rimanente non arriva a 0,
      for( int j = 0 ; j =< numerOfferte; j++ ){    // per ogni offerta
          if( off[j].min <= off[j].max ){           // se il quantitativo, partendo dal minimo
          off[min] ++;                              // non supera il massimo, incrementalo di uno
          }
      }
}}
```

- La ripartizione proporzionale

Questo tipo di ridistribuzione, invece, premia coloro che richiedono un quantitativo maggiore di merce.

Il rimanente viene infatti ridistribuito in maniera direttamente proporzionale al valore massimo dell'offerta effettuata.

```
// dati che possiedo
Vendita ven;                // vendita
Offerte[] off;              // offerte riferite a vend
int numerOfferte;           // numero delle offerte per vend

int sommaMin;                               // sommo il valore minimo e massimo di
int sommaMax;                               // ogni offerta in due int, in modo da calcolare
for(int i=0;i<=numerOfferte;i++){           // il valore della somma dei massimi e dei
   sommaMin += off[i].getMin();             // minimi
   sommaMax+= off[i].getMax();

// calcolo ora la parte rimanente da assegnare sottraendo dal valore massimo della
// vendita il valore della somma dei minimi prima calcolata
int rimanente = ven.max – sommaMin;

//effettuo ora l'assegnazione del rimanente agli offerenti
if(ven.max > sommaMin >= ven.min){              //condizione necessaria per la vendita
   for( int j = 0; j <= numerOfferte; j++ ){    // per ogni offerta
                  //calcolo la percentuale da assegnale in base al valore massimo
      int app_percentuale = ((off[j].getMax()*100)/sommaMax);
                  //ed infine incremento il valore minimo con la percentuale corrispondente
      off[j].min += (rimanente/100) * app_percentuale;
   }
}
```

Capitolo 3: Implementazione dell'applicazione

3.1 Introduzione

Alla luce delle scelte progettuali individuate nel capitolo precedente, viene ora affrontata la parte di implementazione del sistema. In questo capitolo verranno descritte le scelte implementative e le tecniche utilizzate per ultimare il lavoro nelle sue due componenti: client e server. In seguito verranno poi descritti i principi ed i meccanismi che regolamentano le transazioni ed i protocolli di comunicazione tra le due parti del sistema.

Si effettuerà inoltre un'analisi degli strumenti e delle strategie utilizzate per aggirare i vincoli a cui l'applicazione deve necessariamente sottostare, mostrando, dove necessario, porzioni di codice esemplificative.

3.2 I client

La parte client, come definito in precedenza, sarà la parte che permetterà l'interazione con l'utente attraverso un dispositivo mobile sviluppato utilizzando la piattaforma Java ME.

Per l'implementazione del client viene utilizzato NetBeans.

NetBeans è un progetto open source con l'obiettivo di sviluppare una piattaforma IDE (Integrated Development Environment), costituisce quindi uno strumento che aiuta i programmatori nella realizzazione di un software.

L'ambiente

Il programma è scritto interamente in linguaggio Java,e per questo motivo è disponibile su tutti i tipi di piattaforma. Il suo sviluppo è stato incentrato sull'uso di plug-in, in modo tale da permettere a chiunque di crearne uno nuovo o modificarne uno esistente.

Tra questi plug-in ce n'è uno di particolare interesse per lo sviluppo dei client mobile, il "NetBeans Mobility Pack", che permette una completa integrazione tra l'IDE e le funzionalità che Sun mette a disposizione, per lo sviluppo di progetti in J2ME, nel relativo Software Development Kit.[16] Si specifica che recentemente l'SDK ha raggiunto la versione 3.0, portando inoltre questo ambiente di sviluppo anche su piattaforma Mac OSX e offrendo anche un proprio IDE che comprende tutte le funzionalità necessarie.

L'integrazione di queste due componenti permette quindi di sfruttare la comodità e la potenza di un IDE come NetBeans utilizzando tutti gli utili strumenti messi a punto da Sun per la programmazione mobile: dalla possibilità di gestire comodamente le interfacce grafiche ed il flusso di esecuzione, al testing dell'applicazione su vari emulatori.

Specifiche

La parte client è formata da due distinte applicazioni, ma possiamo definire dei tratti che le accomunano. Entrambe, infatti, appartengono alla stessa configurazione ed allo stesso profilo tra quelli definiti da J2ME.

La scelta della configurazione cade obbligatoriamente su CLDC (Connected Limited Device Configuration): è la configurazione più piccola disponibile, progettata per dispositivi con una memoria complessiva di 512KB, con processori a 16 o 32 bit e connettività ridotta. Essa dispone inoltre di una macchina virtuale opportunamente modificata (KVM, Kilobyte Virtual Machine) per l'esecuzione su processori con limitata capacità di elaborazione.

Anche la scelta del profilo, necessariamente vincolata alla configurazione,è obbligata: MIPD. MIPD (Mobile Information Device Profile) è infatti il profilo applicativo di Java ME più diffuso nel mercato dei telefoni cellulari programmabili e PDA. Esso fornisce le funzionalità di base per il funzionamento delle applicazioni dei dispositivi mobili, come la gestione dell'interfaccia, della connettività e della memorizzazione dei dati.

Per la stesura del progetto è stata scelta la versione di MIDP 2.0 Questo ambiente fornisce tre classi fondamentali java:java.lang, java.io, java.utils, ereditate direttamente dalla piattaforma standard, ed un package *javax.microedition* che contiene invece le API necessarie per i basilari funzionamenti del programma (MIDlet), dell'interfaccia grafica, delle operazioni di I/O, della memorizzazione persistente dei dati. Attraverso l'uso pacchetti aggiuntivi ed opzionali è possibile però estendere le funzionalità di sistema: le Wireless Messaging API 2.0 (WMA – JSR 205), ad esempio, forniscono al sistema tutte le funzionalità necessarie all'invio ed alla ricezione di SMS ed MMS, oltre che di messaggi binari.

Si introduce quindi il concetto di MIDlet, che è il nome che viene dato alle applicazioni scritte con il profilo MIDP. Questa tipologia di applicazione prevede al suo interno la presenza di una o più classi che estendono la classe *javax.microedition.midlet.MIDlet*, che a sua volta comprende i meccanismi per la gestione degli “stati” in cui la midlet può trovarsi nella sua esecuzione.

Struttura

Entrambe le MIDlet che compongono il client fanno parte della medesima *MIDlet suite,* una struttura che permette la presenza di più midlet all'interno dello stesso ambiente.

Questo permette quindi l'utilizzo di classi condivise ad entrambe le applicazioni, riducendo così il carico di lavoro. Queste strutture possono essere quindi scritte una volta sola, garantendo una maggiore compatibilità: vengono infatti, come nel nostro caso ad esempio, utilizzati gli stessi oggetti per manipolare i dati. La struttura della midlet suite prevede la presenza di packages comuni per la gestione delle informazioni, e di due applicativi distinti che garantiscono due flussi di esecuzioni differenti.

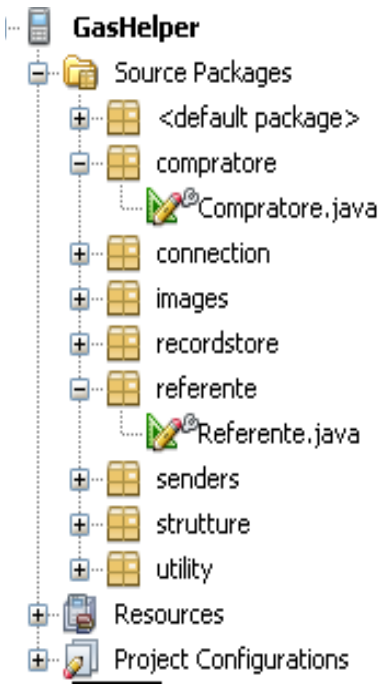

Figura 7: la struttura della midlet suite

- Il package "*compratore*" contiene la midlet che verrà utilizzata dall'acquirente, mentre il package "referente" contiene la midlet che verrà utilizzata dal referente del GAS.
- "*connection*" e "*senders*" contengono le classi che permettono la connettività di rete e la comunicazione con il server, sia in entrata sia in uscita.
- "*images*", invece, contiene tutte le immagini che vengono utilizzate dalle midlet durante l'esecuzione all'interno dell'interfaccia grafica.
- "*recordstore*" ed "*utility*" invece implementano le classi necessarie alla memorizzazione permanente sul dispositivo e per la gestione della funzionalità multithread.
- "*strutture*" contiene invece tutte le classi che compongono gli oggetti utilizzati dl sistema.

3.3 Il Client per il referente

Analizziamo ora nel dettaglio il client che verrà utilizzato unicamente dai referenti dei GAS, ossia da coloro che porranno in vendita i prodotti per proporli ai potenziali acquirenti, i quali utilizzeranno invece la prossima midlet che analizzeremo.

Lo scopo principale della midlet per il referente è quello di agevolare la messa in atto di vendite di prodotti all'interno del GAS, controllare lo stato delle vendite già proposte ed infine, una volta raggiunto il termine per la vendita, raccogliere ed organizzare le offerte ricevute dai vari utenti, nel caso ve ne siano state.

Si è cercato di rendere queste operazioni il più semplici possibile, anche utilizzando delle componenti grafiche proprie dell'ambiente Java ME (contenute nel package javax.microedition.lcdui) quali liste e form, cercando di riproporre una struttura il più possibile simile in tutte le sezioni della midlet e attraverso l'accorpamento di tutte le operazioni eseguibili su un determinato elemento (come ad esempio l'elemento di una lista) in un unico menù, in modo da ridurre i passaggi che l'utente deve compiere, offrendo così una visione d'insieme dei comandi che si possono impartire.

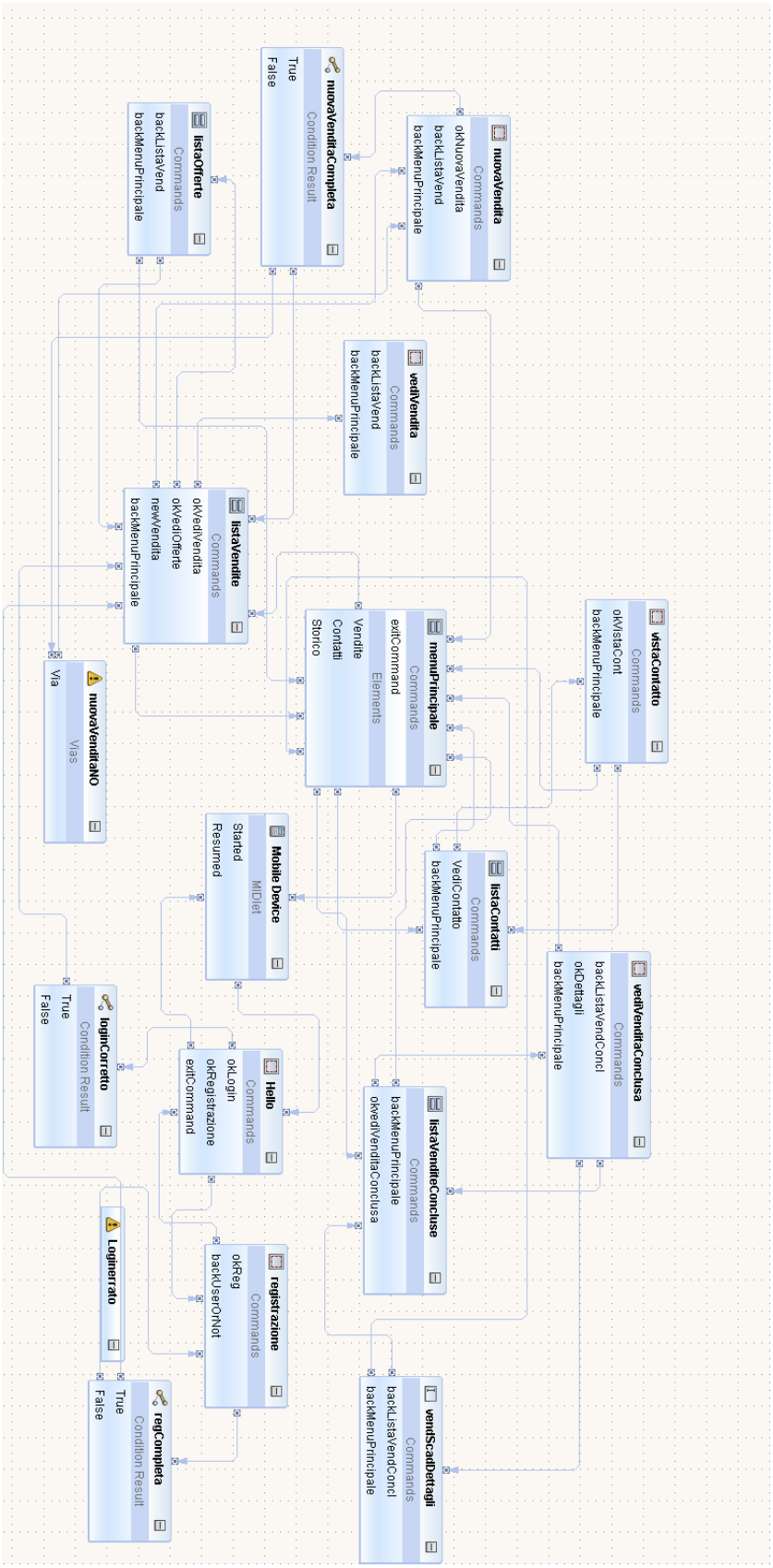

Figura 8: struttura della midlet per il referente

Ecco, nel dettaglio, tutte le componenti che formano questa midlet, con un immagine estratta direttamente dal diagramma che l'ambiente NetBeans mette a disposizione degli sviluppatori.All'avvio l'applicazione mostra una schermata per l'accesso al sistema o per un'eventuale nuova registrazione, attraverso due distinti moduli da compilare.

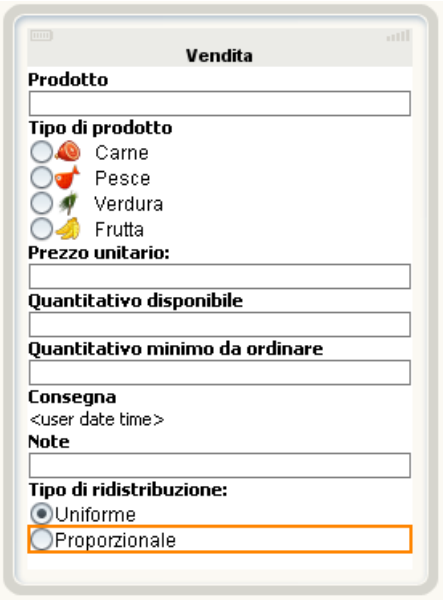

Figura 9: form per la vendita

Una volta effettuato l'accesso, previa eventuale registrazione, la midlet ci condurrà alla schermata da cui potremo porre in essere una nuova vendita e vedere, sotto forma di lista, lo stato di quelle pubblicate da noi in precedenza. Da qui, quindi, sarà possibile accedere alla funzione caratterizzante dell'applicazione: il modulo per pubblicare la vendita di un prodotto.

Questo form prevede l'inserimento di tutti i dati necessari alla pubblicazione della vendita (spiegata nel dettaglio nei prossimi paragrafi), oltre alla tipologia di ripartizione in caso di disavanzo del prodotto. La stessa schermata è inoltre disponibile nel caso si vogliano consultare le vendite precedentemente proposte

Il completamento di quest'operazione conduce successivamente al menu principale della midlet, che permette di amministrare le 3 seguenti componenti, attraverso un'interfaccia intuitiva

- *Vendite*: porta alla lista delle vendite effettuate, consentendo di visionarle e di controllare lo stato delle offerte per essa, conduce inoltre al modulo per creare una nuova vendita.
- *Contatti:* conduce ad una lista contenente i propri utenti e permette di visualizzare facilmente i dettagli inseriti da questi ultimi in fase di registrazione.

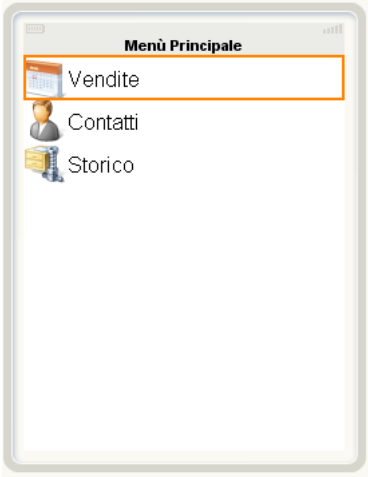

Figura 10: menu principale

- *Storico:* presenta la lista delle proprie proposte d'offerta che hanno raggiunto il termine temporale, mostrando il quantitativo totale acquistato e il prezzo da pagare al fornitore, offendo inoltre la possibilità di una vista dettagliata del quantitativo che ogni utente ha acquistato, con il relativo prezzo.

Il sistema viene strutturato in maniera da raggruppare tutti i possibili comandi che possono essere impartiti da quella posizione, e per l'oggetto selezionato, in un unico menù, in modo da rendere chiara e semplice la scelta dell'operazione che si vuole effettuare.

3.4 Il client per il compratore

Analizziamo ora l'altra midlet che compone il client, quella che permetterà all'utente finale di effettuare delle offerte riferite ai prodotti che il proprio referente pubblicherà.

Anche in questo caso viene proposto il diagramma sviluppato con NetBeans.

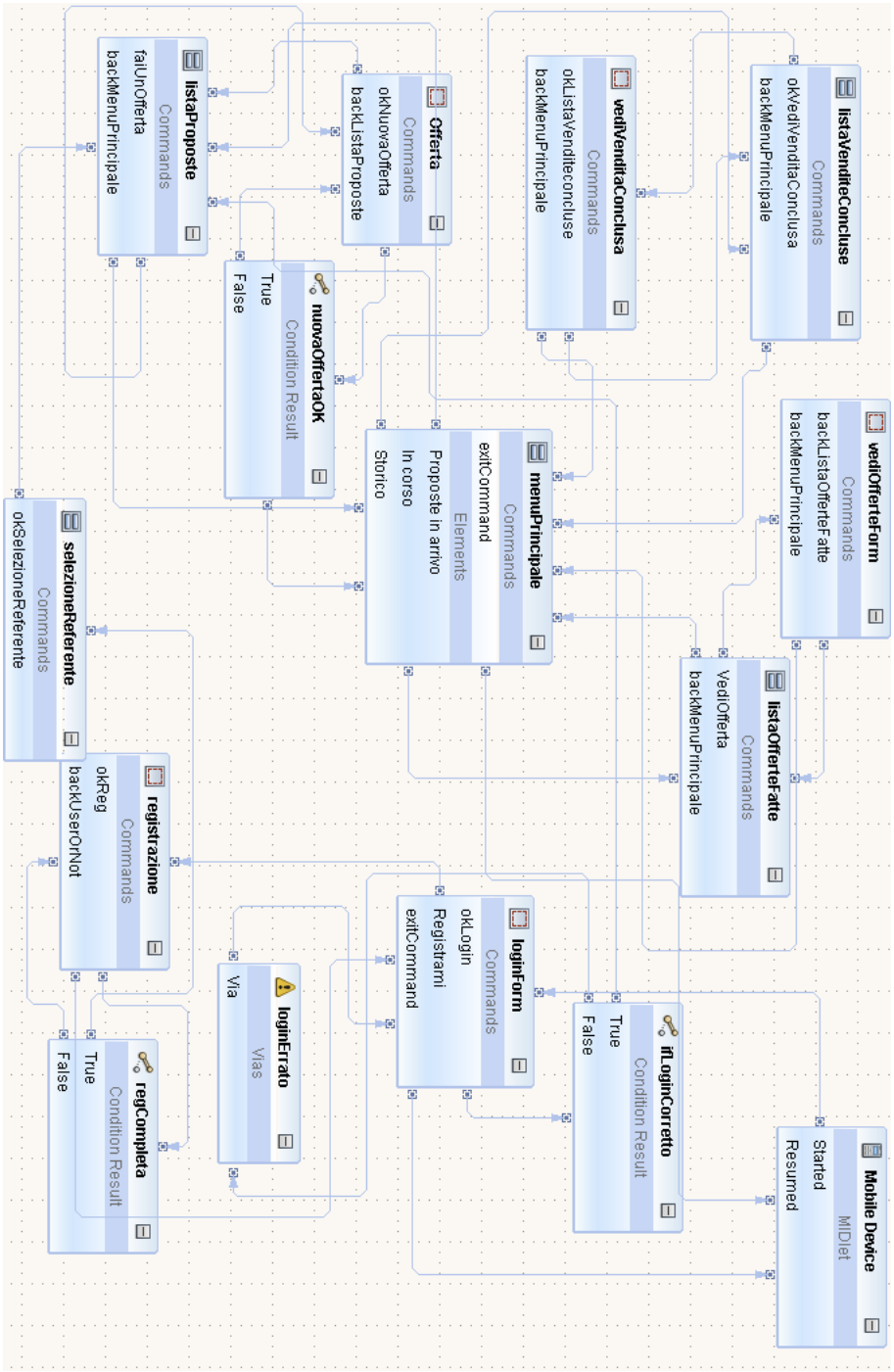

Figura 11: struttura della midlet per l'acquirente

Si è cercato di rendere la procedura il più semplice possibile: si mantiene infatti una struttura grafica uniforme nelle varie componenti che formano l'interfaccia grafica proposta all'utilizzatore, limitando e raggruppando le operazioni possibili per ogni elemento in un unico menu con i comandi disponibili.

Figura 13: modulo di registrazione

Come la parte client per il referente, anche questa, all'avvio, presenta una schermata per il login o la registrazione di un nuovo utente.

Al momento della registrazione, in questo caso, viene però richiesto di selezionare un referente tra quelli già registrati nel sistema, in modo da avere un riferimento univoco al GAS di riferimento.

Verrà inoltre richiesto di selezionare la tipologia di prodotti di cui si è interessati ricevere le proposte, attraverso la divisione in 4 macrocategorie: carne, pesce, verdura, frutta.

Una volta effettuato l'accesso, l'applicazione conduce alla prima interfaccia, che illustra le proposte di vendita pubblicate dal proprio referente: verranno mostrate unicamente le vendite che corrispondono alle categorie inseriti in fase di registrazione.

Attraverso il menu sarà quindi possibile visionare tutti i dettagli inerenti alla proposta selezionata, ed eventualmente effettuare un'offerta per essa.

L'esecuzione conduce conseguentemente al menu principale dell'applicazione, che permette di interagire con le 3 parti che compongono il client per questa tipologia di utente.

- *Proposte in arrivo*: mostra la lista delle proposte che il proprio referente pubblica, offrendo la possibilità di effettuare un'offerta per i prodotti a cui si è interessati.
- *In corso*: mostra le offerte che sono state effettuate dall'utente per le vendite che sono ancora in essere, mostrandone all'occorrenza i dettagli.
- *Storico*: conduce ad una lista di offerte effettuate per vendite già terminate, permettendo la visione dei dei quantitativi ottenuti se richiesto.

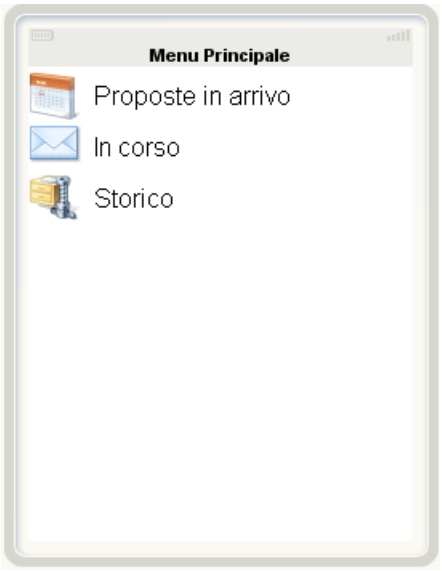

Figura 14: menu principale

3.5 Organizzazione dei dati: il package strutture

Vengono qui definite le 5 classi che conterranno la struttura dei principali oggetti utilizzati in entrambe le midlet (e successivamente anche nel server), con i relativi metodi necessari ad estrapolarne le informazioni.

Figura 15: il package strutture

- *Utente.java:* contiene le informazioni utili alla creazione dell'oggetto che permette la gestione degli utenti.

```
public class Utente {
    String nome;
    String cognome;
    String email;
    String password;
    String telefono;
    boolean[] tipo;                    // tipologia di merce a cui l'utente è interessato
    boolean referente;                 // determina se l'utente è un referente
    String mio_referente;              // email del referente di riferimento
                                       // se sono io stesso è uguale ad “email”
….
}
```

- *Vendita.java*: contiene tutte le informazioni che caratterizzano la vendita.

```
public class Vendita {

    int idVendita;                     // numero identificativo della vendita
    String prodotto;                   // nome del prodotto proposto
    float prezzo;                      // prezzo
    boolean[] tipo;                    // categoria a cui la merce appartiene
    int max;                           // massimo disponibile per la vendita
    int min;                           // quantitativo minimo da acquistare
    long consegna;                     // termine massimo per la presentazione delle
                                       // offerte
    String note;                       // eventuali note aggiuntive
    String autore;                     // proprietario dell'inserzione

….
}
```

• *Offerta.java*: in questa classe sono invece definite le proprietà dell'oggetto offerta.

```
public class Offerta {

    int idOfferta;              // identificativo dell'offerta
    int idVendita;              // id della vendita di riferimento
    String prodotto;            // nome del prodotto
    int max;                    // quantitativo massimo che si è disposti a ricevere
    int min;                    // quantitativo minimo richiesto
    long oraOfferta;            // data e ora dell'effettuazione dell'offerta
    String offerente;           // identificativo dell'utente che effettua l'offerta

….
}
```

• *Contatto.java*: questa classe, formata solamente una parte della classe Utente.java, serve per la gestione dei contatti da parte del referente. Essa quindi non conterrà tutte le informazioni dell'utente, ma solo quelle parti indispensabili e necessarie al referente per identificare l'offerente e le sue preferenze.

```
public class Contatto {

    String nome;
    String email;
    String telefono;
    boolean[] tipo;

….
}
```

• *Notifica.java*: attraverso questa classe vengono creati gli oggetti che comunicheranno sia ai referenti, sia agli acquirenti, gli esiti di una vendita.

In questo caso, quindi, si dovrà avere la possibilità di differenziare l'oggetto da creare in base alla tipologia di utente a cui inviare la notifica.

```
public class Notifica {

    boolean referente;                  // definisce il tipo di utente
    int idvendita, numeroOff;           // identificativo della vendita e num di offerte ad essa
                                        // correlate
    float quantitativo, prezzo;
    String offerente;                   // identificativo dell'offerente
    String prodotto;
    String dettagli;




   //costruttore per il referente
   public Notifica (boolean referente, int idvendita, String prodotto,
                    float quantitativo, float prezzo, int numeroOff, String dettagli) {

      this.referente = referente;
      this.idvendita = idvendita;
      this.prodotto = prodotto;
      this.quantitativo = quantitativo;          // quantitativo totale, ottenuto dalla somma delle
                                                 // offerte
      this.prezzo = prezzo;                      // prezzo totale, legato al quantitativo totale
      this.numeroOff = numeroOff;                // numero di offerte relative alla vendita
      this.dettagli = dettagli;                  // qui vengono inseriti, per il referente, i
                                                 // resoconti dettagliati per ogni utente,
                                                 // in termini di quantitativo e prezzo

   }

   //costruttore per gli offerenti
   public Notifica (int idvendita, String prodotto, float quantitativo, float prezzo, String offerente) {

      referente = false;
      this.idvendita = idvendita;
      this.prodotto = prodotto;
      this.quantitativo = quantitativo;          // quantitativo assegnato ad ogni offerente
      this.prezzo = prezzo;                      // prezzo per il quantitativo da ricevere
      this.offerente = offerente;

   }
```

Vengono quindi creati due distinti costruttori, uno per il referente e uno per gli offerenti.Queste saranno dunque le strutture che verrano utilizzate all'interno delle midlet per manipolare i dati necessari alle transizioni.

Prima di proseguire con l'analisi degli altri packages che compongono la midlet suite, è necessaria un'introduzione riguardo ad alcune scelte implementative che sono state effettuate.

Tra le problematiche riscontrate in fase di progettazione vi è, infatti, quella della serializzazione dei dati.

Questa problematica, che nasce da una "carenza" del linguaggio, è di grande importanza, in quanto influirà sia sulla connettività di rete, che sul salvataggio dei dati in locale.

3.6 Il problema della serializzazione

La letteratura propone diverse alternative, in ambiente Java ME, per raggiungere questo scopo: prima tra tutte la serializzazione "leggibile dall'uomo". Essa prevede di codificare i dati da serializzare in un file XML, in modo da garantirne un'eventuale lettura.

La modalità invece utilizzata nel progetto è quella di una *serializzazione binaria:* questa tipologia prevede quindi che l'oggetto venga trasformato in un array di byte per poi essere gestito.

Le ragioni che portano all'utilizzo di questo tipo di approccio sono legate alla struttura stessa dell'ambiente J2ME. Esso infatti offre classi per la manipolazione dei dati, che possono essere utili sia per la trasmissione di informazioni sulla rete che per il salvataggio di esse in memoria locale.

L'oggetto "DataOutputStream", ad esempio, permette di inserire in un buffer di memoria stringhe ed altri dati primitivi, mentre "ByteArrayOutputStream" trasforma il contenuto di uno stream di dati in un array di byte. L'utilizzo combinato quindi di questi due oggetti con "DataInputStream" e "ByteArrayInputStream", che forniscono i servizi opposti, formano quindi una base per la serializzazione degli oggetti. Utilizzando questi particolari servizi per la manipolazione dei dati viene quindi implementato un metodo per poter incapsulare un intero oggetto all'interno di un flusso di byte.

La soluzione per questo problema si trova nell'implementare, all'interno di ogni classe che definisce un oggetto, presenti nel package strutture, due metodi: persist e resurrect. [17] Essi forniscono, essenzialmente il servizio di serializzazione e deserializzazione.

persist()

Il metodo persist (la cui firma: `public byte[] persist() throws IOException`) è quello che permette di trasformare ogni dato primitivo di un oggetto in un flusso di byte, immagazzinando temporaneamente le informazioni in un "data stream". La funzione non presenta parametri in ingresso, in quanto è specifica dell'oggetto.

```
public byte[] persist() throws IOException {
        ByteArrayOutputStream bout = new ByteArrayOutputStream();
        DataOutputStream dout = new DataOutputStream(bout);

        dout.writeUTF(nome);
        dout.writeUTF(cognome);
        dout.writeUTF(email);
        dout.writeUTF(password);
        dout.writeUTF(telefono);
        dout.writeBoolean(tipo[0]);
        dout.writeBoolean(tipo[1]);
        dout.writeBoolean(tipo[2]);
        dout.writeBoolean(tipo[3]);
        dout.writeBoolean(referente);
        dout.writeUTF(mio_referente);
         dout.flush();

        return bout.toByteArray();
    }
```

Portiamo come esempio la funzione presente nella classe utente.java, che quindi servirà a serializzare l'oggetto che gestisce gli utenti.La funzione, che restituisce quindi un array di byte, inserisce nel flusso di dati ogni elemento che compone l'oggetto, attraverso la chiamata alla funzione specifica relativa al tipo di dato primitivo che si vuole aggiungere.

resurrect(byte[] b)

Una volta trasformato l'oggetto in questione in un array di byte e manipolato secondo le esigenze, vi è la necessità di ricreare la struttura originaria, deve essere deserializzato.

Il metodo resurrect (public void resurrect(byte[] b) throws IOException), prende in ingresso tale array di byte e ricompone l'oggetto in questione

```
public void resurrect(byte[] b) throws IOException {
        ByteArrayInputStream bin =  new ByteArrayInputStream(b);
        DataInputStream din =  new DataInputStream(bin);

        nome = din.readUTF();
        cognome = din.readUTF();
        email = din.readUTF();
        password = din.readUTF();
        telefono = din.readUTF();
        tipo[0] = din.readBoolean();
        tipo[1] = din.readBoolean();
```

```
byte [] utenteInByte;              // utente serializzato in un array di byte
Utente user = null;                // struttura che definisce l'utente

user = new Utente();               // inizializzo l'oggetto
user.persist(utenteInByte);        // compilo i campi dell'oggetto prendendo
                                   // le informazioni dal flusso di byte.
```

E' quindi strettamente necessario che ogni oggetto che deve essere manipolato implementi i propri due metodi in questione, al fine di, come sarà spiegato nei prossimi paragrafi, essere salvato in memoria locale o trasmesso sulla rete.

3.7 Gestione e salvataggio dei dati: il package recordstore

Una delle abilità che vengono richieste all'applicazione è quella di salvare i dati in maniera persistente, in modo da garantire anche una temporanea consultazione offline.

L'ambiente Java ME offre un sistema di persistenza semplificato, così creato a causa delle modeste risorse a sua disposizione: il Record Management System (RMS).

Questa struttura, implementata dalle classi del package *javax.microedition.rms* permette la creazione di semplici archivi, attraverso la scrittura di array di byte in un area riservata della memoria.

Hashtable

Al fine di permettere un'organizzazione migliore e più funzionale dell'archiviazione dei dati, viene implementata un'hashtable: questa struttura dati, infatti, non solo migliora il tempo di ricerca all'interno del recordstore (che si può stimare in tempo circa costante $T(n) = O(1)$) [18], ma permette una consultazione più veloce e meno dispendiosa, in termini computazionali, delle informazioni: di ogni elemento vengono, infatti, tenute in memoria solamente la chiave d'accesso e l'indice di riferimento alla memoria dove sono conservati i dati ad essa relativi.

Serializzazione per l'archiviazione locale

Questa struttura implementata permette quindi di associare una chiave ad un elemento salvato in memoria, in modo da poterla facilmente richiamare.

Vengono dunque create, al momento dell'inizializzazione del programma, le tabelle necessarie all'immagazzinamento dei dati, cioè degli oggetti, usati per compiere le transazioni.

Il metodo di inserimento nella tabella (`public void put(String key, byte[] data) throws Exception`), richiede come parametri in ingresso la chiave di identificazione dell'oggetto da salvare e il corrispondente array di byte che lo rappresenta.

Il processo di serializzazione studiato nel paragrafo precedente, se utilizzato in questo contesto, ci permette quindi di inserire agevolmente qualsiasi oggetto che disponga dei

due metodi “persist()” e “resurrect()” all'interno della tabella di memorizzazione, previa scelta sistema di identificazione delle informazioni.

Viene quindi creata una tabella per ognuno degli oggetti da noi utilizzati (alcune di esse, come ad esempio la voce ”contatti”, non sarà presente in entrambe le midlet, ma unicamente in quella in cui è necessaria la presenza, in questo caso nella midlet “Referente”).

La scelta delle chiavi verrà descritta in seguito, in quanto strettamente collegata al metodo di memorizzazione delle informazioni da parte del server, al fine di garantire una compatibilità assoluta tra le due componenti del sistema.

3.8 Gestione della concorrenza dei thread: il package utility

Un altro fattore chiave, che influenza in maniera forte le prestazioni e l'affidabilità del sistema è la gestione dei thread. Essa, infatti, permette un'esecuzione fluida e lineare dell'applicativo, eliminando quasi completamente le situazioni di stallo che si possono creare, alle volte, in situazioni di comunicazione asincrona o attesa delle risorse condivise. Java ME offre un'architettura basilare ma sufficiente, per le esigenze dei dispositivi mobili, per la gestione flussi di lavoro concorrenti, il multithreading appunto. Essa però manca di una "gestione centralizzata", che permetta di sfruttarne a pieno le potenzialità, garantendo sempre un uso moderato e conforme delle risorse del sistema.Un modello risolutivo a questo tipo di problematica, che verrà utilizzato nel sistema, è presentato e spiegato da S.Sanna nel libro *Java Micro Edition.*

Egli sviluppa un meccanismo di gestione dei thread come un modello di esecuzione di Task: il TaskEngine.

Questa struttura non è altro che una classe che, attraverso un'intelligente sistema di code e priorità, permette di eseguire i thread in maniera sequenziale, mantenendo contenuto l'utilizzo delle risorse.

Essa si appoggia alle già presenti funzionalità dell'ambiente, utilizzando quindi le funzioni già esistenti per la creazione e l'esecuzione di nuovi thread, creando però un modello che si occupi della sequenza di esecuzione di questi ultimi, e delle risorse da essi utilizzati: un vettore, infatti, determina la sequenza dei flussi che devono essere eseguiti, bloccando le risorse da essi richieste (utilizzando il meccanismo rappresentato dalla keyword *synchronized* [19]) e occupandosi anche della successiva liberazione della memoria al termine dell'esecuzione.

Questo sistema è inoltre estremamente efficace nel caso in cui siano necessari più thread contemporanei in esecuzione: basta infatti creare più di un'istanza della classe per poter eseguire senza complicazioni più flussi di esecuzione concorrenti.

3.9 Protocolli di comunicazione

Dopo una prima analisi dei client, al fine di completare il quadro in maniera corretta, è necessaria un'introduzione alla comunicazione che avverrà tra client e server.

Questa, infatti, permetterà successivamente di trattare la spiegazione del server con maggiore semplicità. La comunicazione tra le parti del sistema, come introdotto nel capitolo precedente, è infatti basilare per il corretto funzionamento dell'intera struttura.

La comunicazione tra le due parti avviene tramite socket TCP: questa particolare struttura risulta molto congeniale alle nostre esigenze per diversi motivi:

1. Entrambi i linguaggi utilizzati nello sviluppo dell'applicazione (J2ME per il client, J2SE per il server), presentano dei metodi nativi per la gestione di questo tipo di connessione.

2. Questo modello comunicativo prevede un controllo automatico dell'ordine di trasmissione dei dati che, come vedremo in seguito, è basilare per il funzionamento del tipo di serializzazione da noi adottato.

3. Permette di identificare l'indirizzo IP sulla rete della sorgente che trasmette i dati.

Quest'ultimo punto ci rimanda ad una delle problematiche descritte in fase di progettazione: la dinamicità dell'indirizzo IP del client.

IP dinamico del client sulla rete mobile

Il client, sia esso per il referente o per il compratore, presenta un problema di identificazione sulla rete che può inficiare sulla comunicazione con il server.

Esso infatti non dispone di un indirizzo noto, in quanto viene assegnato dal provider ogni volta che viene instaurata una connessione, e tendenzialmente questo è sempre differente.

Si pone quindi il problema di come il server, all'occorrenza, possa comunicare con il client: esso infatti non dispone di un indirizzo per contattarlo.

Per ovviare a questa limitazione si utilizza il servizio mobyt.[20]

Mobyt

Mobyt è un servizio che permette l'invio (ed eventualmente anche la ricezione, funzionalità però non utilizzata dal progetto) di sms di vario tipo.

Figura 16: logo di Mobyt

Questa funzione, utilizzata dalla parte server, combinata con un servizio nativo del profilo MIDP 2.0 di J2ME (il push registry), permette agevolmente di risolvere il problema dell'IP dinamico prima descritto.

Il push registry

Il push registry è un servizio offerto dalla seconda versione del profilo MIDP.

Esso prevede che l'applicazione parta automaticamente all'accadere di un determinato evento, come ad esempio la scadenza di un timer o la ricezione di un sms.

La proprietà MIDlet-Push ha la seguente sintassi:

MIDlet-push-N: URL, NomeCompleto_MIDlet, mittenti autorizzati

Questo tipo di servizio permette quindi di differenziare che midlet debba essere avviata in base alla porta da cui proviene l'SMS. La combinazione dell'utilizzo del servizio mobyt con la funzione push registry permette quindi al server di poter avviare l'applicazione client secondo necessità, facendo in modo che quest'ultima comunichi con il server, rivelando il proprio indirizzo sulla rete.

Socket Listener

Entrambe le parti che compongono il nostro sistema, per poter comunicare, devono quindi implementare dei meccanismi che permettano loro di ricevere e instaurare una comunicazione di tipo socket TCP.

L'approccio utilizzato è pressoché equivalente in entrambe le parti: all'avvio del

software viene caricata in memoria un'istanza della classe "*SocketListener.java*", che pone il sistema in ascolto su una porta specifica (1234 per la parte client, 1235 per il server).Al momento dell'instaurazione della connessione il flusso di dati in arrivo viene quindi inoltrato a "*IncomingSocketSwitch.java*", anch'esso presente in entrambe le parti, con funzioni parallele, che si occupa della decodifica dei dati.

Questo file è composto da uno switch, che, all'arrivo del flusso di informazioni, in base ai primi 4 bit di dati (il valore di un intero nel nostro caso), devia il flusso verso la parte corretta che lo potrà decodificare, deserializzare: questo è possibile in quanto ogni tipologia di oggetto inviato tra le due componenti del sistema, prima di essere serializzato, viene preceduto da un intero che ne definirà la natura.

IncomingSocketSwitch per i client

Analizziamo ora, brevemente, il principio di funzionamento nel caso dei client, per poter giustificare le affermazioni precedenti riguardo a mobyt e push registry.

Quando il server, infatti, ha necessità di comunicare con il client, attraverso la procedura spiegata prima, invia un SMS al dispositivo mobile che ospita il software client per far partire l'esecuzione della midlet, che effettua una richiesta di login al server, rivelando il proprio indirizzo.

Vengono quindi inviate alla midlet tutte le informazioni in blocco, codificate secondo il principio illustrato prima: è proprio attraverso questo switch che possono essere correttamente individuate, deserializzate ed, eventualmente, salvate in memoria.

La scelta di spedire tutte le informazioni utili in un unica sessione nasce dalla necessità di inviare il numero minore possibile di sms utilizzando il servizio Mobyt.

3.10 Il server

Vediamo ora nel dettaglio come è strutturata la parte server, che è quella che permette l'amministrazione di tutte le componenti necessarie al sistema per la sua completa funzionalità. Ha il compito di ricevere, organizzare, rielaborare, immagazzinare ed eventualmente re-inviare tutti i dati necessari al completamento delle transazioni.

Essa è fondamentalmente composta di 2 parti inscindibili:

- il database per l'immagazzinamento delle informazioni
- l'applicazione per la manipolazione dei dati

Essa adopera, così come la midlet suite che compone la parte client, lo stesso package che definisce gli oggetti (Vendita, Offerta, Utente, Contatto, Notifica) per la gestione delle informazioni.

In questo caso, però, questo torna molto utile per l'interfacciamento con il database alla quale il sistema si appoggia.

Il Database

La parte server, come già detto, è quella che si occupa della gestione centralizzata di tutti gli elementi necessari al corretto svolgimento delle transazioni.

Essa necessita quindi di uno strumento di supporto per il salvataggio e l'archiviazione dei dati, in modo da poter essere poi rielaborati e maneggiati in maniera conveniente.

Viene quindi integrato con il sistema il database relazionale PostgreSQL[21]: la scelta di questo database relazionale è legata all'ambiente su cui il server dovrà essere posto in esecuzione.

Vengono quindi create, al suo interno, delle tabelle per gestire la memorizzazione di utenti, vendite, offerte e notifiche. I campi che compongono ciascuna tabella quasi totalmente corrispondenti alla struttura degli oggetti che sono destinati ad ospitare, agevolando quindi la procedura di inserimento dall'applicazione gestore.

In particolare, vengono utilizzate, per ogni oggetto da memorizzare, le stesse chiavi di

ricerca: esse infatti sono corrispondenti sia nel database, sia nella procedura di memorizzazione locale sul client attraverso l'RMS.

Un altro ruolo fondamentale del database è quello di assegnare, ad ogni nuova vendita, ed eventualmente ad ogni nuova offerta, un identificatore univoco, creato in maniera sequenziale, che permetta poi di riconoscerlo ed isolarlo all'interno dell'intera struttura, come appena descritto.

Vengono inoltre create due ulteriori tabelle: veditescadute ed offertescadute: esse servono per distinguere e gestire con maggiore chiarezza le transizioni che hanno già raggiunto il termine.

Il "gestore"

E' la parte sviluppata in linguaggio Java2SE, ed è quella che contiene tutte le classi e i metodi che permettono la gestione centralizzata dei dati.

L'applicazione si compone di due packages: uno contenente tutte le strutture scritte come supporto all'elaborazione dei dati, chiamato appunto "strutture"; ed uno contenente invece le classi in cui sono implementati i vari metodi per la corretta esecuzione:"gestore"

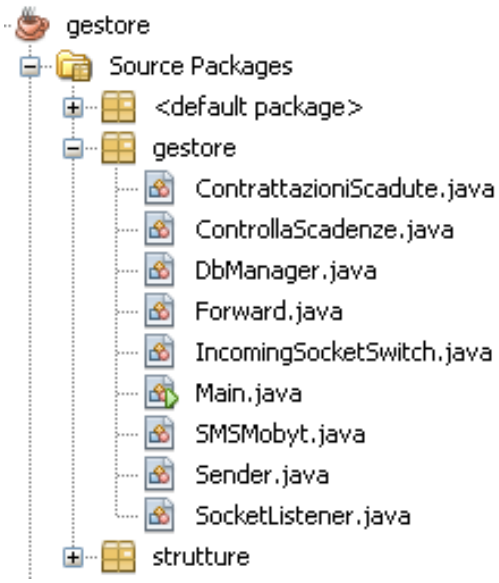

Figura 17: il package gestore

Il package strutture è esattamente identico a quello contenuto nella midlet suite, contiene le stesse classi che, anche in questo caso, definiscono gli oggetti che verranno utilizzati a tempo di esecuzione per manipolare le informazioni, appoggiandosi anche al database sopra descritto.

Il package gestore, invece, è la parte dell'applicazione che contiene tutte le classi che implementano i metodi necessari al funzionamento del sistema di transazioni e alla comunicazione con i client.

Al momento dell'inizio dell'esecuzione, che avviene creando un'istanza della classe "Main.java", il sistema avvia le due strutture fondamentali per il suo funzionamento: inizializza infatti le componenti necessarie per la connettività, rappresentate dalle classi "SocketListener" e "IncomingSocketSwitch", e quelle invece richieste per la gestione delle transazioni, identificate in "ControllaScadenze" e "ContrattazioniScadute".

La gestione delle connessioni

Al fine di essere raggiungibile sulla rete dai client, oltre alla necessità di un indirizzo IP statico, l'applicativo server deve essere fornito di un'ascoltatore per le chiamate di connessione in ingresso.

Viene quindi implementato, all'interno di "SocketListener.java", un listener di connessioni socket TCP sulla porta 1235. Quest'ultimo, nel momento in cui viene instaurata la connessione, inoltra il flusso di dati ai metodi della classe "IncomingSocketSwitch.java" che, attraverso l'esecuzione in un thread separato, decodifica i dati in arrivo: questa classe ha un comportamento equivalente alla classe che porta questo stesso nome nella midlet suite.

Ogni flusso di dati che le viene inoltrato è preceduto da un intero che identifica una tipologia differente di richiesta, e richiede conseguentemente una differente deserializzazione dei dati.

Questa classe, inoltre, ha un altro importante ruolo: quello di identificare l'indirizzo del client con cui comunica: instanziando un oggetto di tipo "InetAddress" è possibile, infatti, ricavare dalla socket in uso l'indirizzo IP e l'hostname della sorgente dei dati.

Attraverso questo tipo di codifica dei dati, è dunque possibile riconoscere tutti gli elementi necessari al corretto funzionamento del sistema: dalla registrazione di un nuovo utente al login, dalla messa in atto di una nuova vendita alla presentazione delle offerte, dalla gestione delle notifiche al termine di una transazione a tutte le minori funzionalità richieste per la gestione dei contatti.

Analizziamo il caso del login, associato al numero 0 della sequenza di codifica: esso, oltre al controllo dell'accesso degli utenti, ha anche il compito di mantenere aggiornati i

client che accedono al sistema.

Ogni volta che un utente effettua il login, il sistema, attraverso una serie di interrogazioni mirate al database utilizzando l'email che lo identifica, ricompone tutti gli elementi utili per quell'utente, facendo ovviamente distinzione tra referente e acquirente.

Quando un referente accede al sistema, esso provvede a reperire tutte le informazioni a lui inerenti: le vendite in essere, con i relativi aggiornamenti delle offerte ad esse riferite, le vendite che sono terminate, con conseguente notifica del successo o meno della transazione ed, eventualmente, i quantitativi assegnati ad i vari acquirenti, nuovi contatti che lo hanno scelto come referente di riferimento.

Nel caso dell'acquirente, invece, ci sono necessità differenti: esso infatti deve essere informato sulle nuove proposte che il proprio referente pubblica, sempre se corrispondenti alle proprie preferenze, espresse in fase di registrazione, o sull'esito di una transazione per la quale aveva effettuato un'offerta, eventualmente contenente anche il quantitativo conseguito e la cifra corrispondente da pagare.

Associato al numero 2, invece, troviamo un altro caso che vale la pena di analizzare: la ricezione di una nuova vendita proposta da parte di un referente.

Quando quest'ultimo compila il form sul dispositivo mobile con tutte le informazioni necessarie prima descritte e invia la vendita per pubblicarla ai propri contatti, il server che la riceve si deve occupare della sua gestione.

Dopo averla riconosciuta come tale, il software si occupa della memorizzazione nel database: ed è proprio quest'ultimo che, attraverso l'uso di un contatore, assegna un id alla vendita, in modo da renderla unica e consultabile. Successivamente inoltra questa chiave identificativa al referente, con lo scopo di confermargli il corretto svolgimento dell'attività di pubblicazione, oltre che consentirgli di archiviare tale informazione in modo coerente sulla memoria locale del dispositivo mobile.

E' compito di questo processo anche l'inoltro della richiesta agli utenti, relativi a quel referente, potenzialmente interessati alla vendita. Viene quindi nuovamente interrogato il database in modo da selezionare questi elementi e, una volta estratto il numero di telefono do ognuno di loro, viene inviato un sms in modo da avviare l'applicativo sul

telefonino, al fine di ricevere l'aggiornamento.

Lo stesso procedimento avviene, in maniera ridotta, per la presentazione di un offerta da parte di un acquirente, identificato nel numero 5 della sequenza.

Anche in questo caso, infatti, è compito del server assegnare all'offerta un id di riconoscimento e di reinviarlo all'acquirente che l'ha effettuata, notificando l'avvenuto al referente.

La gestione delle transazioni

La seconda parte che compone la struttura è quella che dovrà regolamentare le transazioni che vengono archiviate nel database del server.

Il compito principale che deve venir svolto è il controllo delle vendite: esse infatti sono definite da un parametro che ne determina la scadenza temporale, un limite entro il quale accettare delle offerte per essa. Al fine del corretto funzionamento del sistema, è stato quindi necessario implementare un meccanismo che monitori queste scadenze, gestendo successivamente le vendite terminate.

Per fare questo si è implementato uno strumento, nella veste di "ControllaScadenze.java", che, tra tutte le proposte di vendita presenti nel sistema seleziona e monitora quello con la scadenza più incombente, comparandone ciclicamente il termine con la data e l'ora correnti.

Quando una proposta di vendita, infine, raggiunge l'ora di scadenza, viene spostata nella tabella 'venditescadute' del database, e conseguentemente le offerte ad essa relative, se presenti, vengono spostate nella tabella 'offertescadute'; viene quindi selezionata la vendita successiva, lasciando il compito di completare le operazioni su quella appena terminata ad un'istanza della classe "ContrattazioniScadute.java".

E' compito dei metodi presenti in questa classe, infatti, prelevare le proposte di vendita appena terminate e valutare il loro esito: attraverso le interrogazioni alle corrette tabelle del database, i dati della vendita in questione e delle relative offerte, se presenti, vengono estratti ed strutturati per essere elaborati.

Dapprima viene controllato se la vendita è garantita, secondo i criteri definiti in fase di progettazione: si controlla quindi che la somma dei quantitativi richiesti dagli utenti sia conforme ai limiti imposti dal referente. Affinché la vendita sia garantita, infatti, è necessario che ogni persona che ha effettuato un'offerta riceva almeno il quantitativo minimo richiesto.Se questo requisito non dovesse essere soddisfatto, il sistema informa le parti interessate dell'insuccesso della transazione.

Nel caso, invece, i requisiti minimi per la vendita siano presenti, come prima cosa si cerca di valutare se ci si trova nel "caso ottimo", situazione in cui ogni ogni acquirente riceve il massimo dell'intervallo di prodotto che aveva richiesto: questo avviene quando la somma dei quantitativi massimi richiesti da ogni compratore si mantiene nel range di prodotto disponibile.Se questo dovesse accadere, il sistema provvederebbe quindi ad informare gli utenti del successo della transazione.

Qualora, invece, ci si trovi solamente con la condizione minima per portare a termine la transazione, il sistema proverà a ripartire l'eventuale disavanzo tra il massimo disponibile, definito al momento della pubblicazione della proposta di vendita, ed il quantitativo formato dalla somma del minimo di ogni offerta, secondo il criterio scelto dal referente.

Nel caso quest'ultimo sia di tipo "ridistribuzione uniforme", il rimanente verrà equamente distribuito tra gli offerenti, sempre nel rispetto dei quantitativi massimi che gli utenti hanno dichiarato essere disponibili a ricevere.

La "ridistribuzione proporzionale", invece, provvederà a ripartire la parte eccedente in maniera direttamente proporzionale al quantitativo massimo da ognuno richiesto, premiando quindi coloro che che ordinano molto prodotto.

Anche al termine di questa procedura, il sistema ha il compito di informare tutte le parti coinvolte della fine della transazione, comunicando quindi il quantitativo assegnato ed il relativo importo da pagare ad ogni acquirente,ed il quantitativo ed il prezzo totali al referente, fornendogli inoltre una panoramica della ripartizione per i vari compratori.

3.11 Scenari di utilizzo di GasHelper

Una volta terminata la fase di realizzazione del sistema GasHelper, è stato necessario controllarne la funzionalità: grazie agli strumento forniti dall'ambiente J2ME, è stato possibile eseguire la parte client del sistema su un apposito emulatore. Attraverso questo strumento, quindi, si è potuto valutare l'efficienza delle scelte architetturali in relazione ai dispositivi mobili, senza scendere nel dettaglio delle problematiche legate ad ogni specifico dispositivo.

La parte server, dopo una breve permanenza su uno spazio concessomi su un server dell'ateneo, è stata spostata su un server casalingo: questa scelta incarna uno scenario più reale, viste le dimensioni organizzative medie dei GAS considerati.

Basandomi sull'esperienza diretta, acquisita durante la partecipazione alle sedute del mio gruppo di acquisto solidale di appartenenza, il sistema è stato messo alla prova utilizzando dati riguardanti le ultime transazioni da noi effettuate. Queste, di durata indicativamente settimanale e con mediamente una decina di utenti attivi, sono state portate a termine correttamente secondo i criteri prima descritti. Il sistema, però, ha offerto una soluzione più flessibile rispetto ai risultati reali del GAS, permettendo di soddisfare più richieste, a causa dell'intervallo di offerta.

Possibili scenari di utilizzo:

Sebbene il sistema sia ancora da considerarsi prototipale, può essere utilizzato con successo in alcune realtà.

Esso, infatti, va inteso come mezzo di supporto alle mere transazioni, in grado quindi di automatizzare parte del processo commerciale, ma non è ancora in grado di proporsi come mezzo unico per la gestione di esse. Affiancando quindi questo strumento alle altre modalità di amministrazione precedentemente elencate, è possibile sfruttare le i suoi punti di forza per coordinare in maniera più pratica le attività di compravendita.

Il suo utilizzo risulta inoltre più adatto ad un contesto piccolo ed abitudinario: in presenza di tale situazione, infatti, l'accordo tra i partecipanti risulta semplificato, riducendo la possibilità di incomprensioni e diminuendo, conseguentemente, la

comunicazione, ausiliaria alle vendite, esterna al sistema.

Sviluppi futuri

Il sistema, allo stato attuale, presenta alcuni svantaggi che ne precludono l'utilizzo in determinate circostanze.

La prima osservazione riguarda la definizione del costo unitario di una vendita. Quest'ultimo, infatti, potrebbe venir ampliato differenziando le varie voci che lo compongono, come ad esempio il trasporto e l'imballaggio, in modo da offrire un confronto tra esse. L'integrazione con un sistema gestionale software come “Gasdotto”, presentato nel capitolo 1.3, permetterebbe inoltre una gestione completa di tutte le componenti necessarie alle transizioni, garantendo al contempo l'utilizzo delle funzioni basilari in qualunque momento tramite il client presente sul dispositivo mobile.

Per quel che riguarda invece la comunicazione esterna ai meccanismi di vendita veri e propri, uno degli sviluppi più interessanti è quello che riguarda una procedura di feedback interna al GAS: questo permetterebbe infatti di avere uno strumento di controllo sulla qualità del servizio offerto dai vari produttori, referenti ed utenti.

Conclusioni

Il lavoro svolto in questa tesi ha prodotto, come risultato finale, un'applicazione per la gestione delle transazioni all'interno dei GAS, che opera in modo da garantire il mantenimento del principio cooperativo. Essa è stata sviluppata in Java, utilizzando un'architettura client-server, che prevede l'esecuzione della parte client su dispositivi mobili. Quest'ultima è stata implementata utilizzando Java Micro Edition, a causa della sua portabilità su dispositivi con caratteristiche anche molto diverse. La parte client è formata da due differenti MIDlet, una per ogni tipologia di utente previsto dal sistema: il referente e l'acquirente, al fine di offrire loro un'interfaccia il più adeguata possibile alle loro necessità.

Attraverso il sistema è possibile coordinare le transazioni all'interno del GAS., cercando di soddisfare tutte le offerte che vengono effettuate utilizzando un meccanismo di vendita appositamente strutturato, e offrendo immediati e specifici aggiornamenti sulle proposte pubblicate. Questo metodo riduce quindi la comunicazione relativa alle vendite all'interno del gruppo, ed automatizza la raccolta degli ordini, agevolando la procedura. Il sistema sviluppato è da considerarsi solo a livello prototipale e, sebbene riesca nel suo intento di semplificare alcuni processi, presenta alcune limitazioni.

Ad esempio quella di non gestire separatamente alcune voci che influiscono direttamente sull'importo di vendita, come ad esempio il trasporto o l'imballaggio, costringendo così il referente a dover accorpare tutte le spese in un'unica voce, non offrendo quindi la possibilità di valutarne singolarmente l'incidenza sul prezzo totale.

Esso al momento è da considerare come uno strumento da affiancare alla normale amministrazione del GAS, in quanto non è presente attualmente uno strumento per la comunicazione di feedback o strumenti di controllo qualità.

Uno dei possibili sviluppi per rendere questo strumento più funzionale su larga scala è quello di integrarlo con uno scenario di gestione dei GAS Software presentato nel capitolo 1, in modo da poter sfruttare la praticità della compravendita attraverso un dispositivo mobile, e allo stesso tempo integrare le informazioni aggiuntive attraverso un'interfaccia web più completa.

Bibliografia

[1] Definizione di cooperazione: G. Devoto, G.C. Oli, *Dizionario della lingua italiana*, Le Monnier

[2] Definizione di Società cooperativa di wikipedia: http://it.wikipedia.org/wiki/Societ%C3%A0_cooperativa

[3] Articolo 45 della Costituzione Italiana

[4] Definizione di mutuo: G. Devoto, G.C. Oli, *Dizionario della lingua italiana*, Le Monnier

[5] *La dichiarazione di identità cooperativa,* approvata dal XXXI Congresso dell'Alleanza Cooperativa Internazionale,Manchester, 1995

[6] Principi cooperativi, descritti ne "*La dichiarazione di identità cooperativa*": http://www.modena.legacoop.it/updown/storia/storia-08.pdf

[7] Giovanni Costa, Paolo Gubitta: *Organizzazione Aziendale,* McGraw-Hill, seconda edizione, pag. 118

[8] Rete G.A.S., Documento base dei G.A.S.,1999, pag 5: http://www.retegas.org/

[9] Sito del progetto equalway: http://www.equalway.org/

[10] Sito dell'applicazione Gasdotto: http://gasdotto.barberaware.org/drupal/

[11] S. Sanna, *Java Micro Edition*, *Sviluppare applicazioni network-oriented per telefoni cellulari e PDA*, Hoepli, 2007, pag. IX – X

[12] Sito ufficiale dell'ambiente Java Micro Edition: http://java.sun.com/javame/index.jsp

[13][14] S. Sanna, *Java Micro Edition*, *Sviluppare applicazioni network-oriented per telefoni cellulari e PDA*, Hoepli, 2007, pag. 3

[15] Definizione di serializzazione di Wikipedia: http://it.wikipedia.org/wiki/Serializzazione

[16] Java Micro Edition Software Development Kit: http://java.sun.com/javame/sdk/index.jsp

[17] Eric Giguere, *Databases and MIPD*, part 2, 2004: http://developers.sun.com/mobility/midp/articles/databasemap/

[18] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein: *Introduzione agli algoritmi e strutture dati.* McGraw-Hill, 2005, pag. 187

[19] Eric Giguere, *Using Threads in J2ME Applications,* February 2003: http://developers.sun.com/mobility/midp/articles/threading2/index.html

[20] Sito del servizio Mobyt: http://www.mobyt.it/index.php?pg=2

[21] Sito di PostgreSQL: http://www.postgresql.org/