PariGUI 2012: user side

RELATORE: Ch.mo Prof. Enoch Peserico Stecchini Negri De Salvi
CORRELATORE: Ing. Michele Bonazza
LAUREANDO: Lorenzo Fabris

Corso di laurea in Ingegneria dell'Informazione

A.A. 2011-2012

UNIVERSITÀ DEGLI STUDI DI PADOVA
DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE
CORSO DI LAUREA IN INGEGNERIA DELL'INFORMAZIONE

TESI DI LAUREA

PariGUI 2012: user side

RELATORE: Prof. Enoch Peserico Stecchini Negri De Salvi

CORRELATORE: Ing. Michele Bonazza

LAUREANDO: Lorenzo Fabris

A.A. 2011-2012

Indice

Sommario

Questo elaborato analizzerà il processo di progettazione e realizzazione dell'interfaccia grafica del client di PariPari.
Verranno esaminate le problematiche derivanti dall'integrazione nella struttura del programma e dalla rappresentazione coerente delle funzionalità offerte, arrivando ad un progetto incentrato sull'espandibilità e sull'usabilità dell'interfaccia.
Ne sarà poi illustrata la realizzazione, esponendo le difficoltà incontrate nello sviluppo e le soluzioni adottate.

Introduzione

Il ruolo occupato da Internet nello stile di vita moderno è sempre più preponderante: la pervasività dell'accesso e la differenziazione dei dispositivi connessi alla rete stanno determinando un'evoluzione nelle modalità d'interazione con essa. La percezione comune è ad oggi quella di una piattaforma distribuita di servizi interconnessi.
L'utente non è più un fruitore passivo dei contenuti inseriti da terzi, ma un elemento attivo che richiede la possibilità di contribuire in prima persona all'arricchimento della rete.
Questa necessità ha portato al potenziamento del concetto di condivisione, sia di informazioni e risorse, sia di applicazioni complete, rese disponibili come servizi.
Il progetto PariPari nasce in questo contesto, proponendo un modello di condivisione in grado di integrare servizi differenti portando l'esperienza, da frammentata in più applicazioni, ad omogenea ed accentrata in un'unica piattaforma.
La diffusione ed il successo di PariPari dipenderanno dalla sua capacità di mettere a disposizione degli utenti strumenti d'interazione efficaci ed al contempo versatili, senza inficiarne l'usabilità.
La GUI (Graphical User Interface) assume dunque un'importanza fondamentale non solo nel fornire un volto attraente da presentare all'utente, ma anche nel creare l'esperienza di utilizzo completa ed ergonomicamente omogenea essenziale per la fruizione contemporanea dei diversi servizi.

Nel primo capitolo di questo elaborato verrà presentato il progetto PariPari, analizzando le problematiche che la sua struttura modulare pone allo sviluppo di un'interfaccia grafica.
Nel secondo capitolo si esaminerà il problema della rappresentazione delle funzionalità offerte dalla piattaforma, analizzando le caratteristiche di applicazioni simili e l'utilizzo dei componenti grafici nell'ottica di future espansioni.

Nel terzo capitolo si esaminerà la realizzazione vera e propria della GUI con l'esposizione degli strumenti software utilizzati, della struttura adottata per l'integrazione con la piattaforma e delle funzionalità prodotte.

Capitolo 1

Il progetto PariPari

Nel contesto di una crescente diffusione della cultura di Internet come ambiente di contenuti da e per gli utenti, viene proposto il progetto di PariPari, una rete innovativa che interpreta questa visione per superare i limiti degli approcci attuali.
La particolare struttura del programma impone un'attenta definizione del comportamento di un'interfaccia grafica.

1.1 PariPari

PariPari è un progetto software dell'Università di Padova che si pone come obbiettivo la realizzazione di una rete *peer-to-peer* multifunzionale [2], basata su una variante del protocollo Kademlia[1] per la condivisione di risorse e servizi quali *instant messaging*, *file sharing*, *dbms* (database management system) distribuito e altri.

Pur esistendo numerosi programmi in grado di offrire solamente una delle funzionalità sopra citate, un utilizzo in parallelo di più applicazioni può determinare l'inefficienza della concorrenza per le risorse del sistema o peggio errori per incompatibilità.
La proposta di PariPari consiste invece in una piattaforma in grado di far coesistere servizi di natura differente, gestendo in maniera efficiente la distribuzione delle risorse tra di essi e offrendo allo stesso tempo la possibilità di espanderne le funzionalità con un meccanismo semplice e veloce.
Non solo: tale coesistenza permette l'integrazione dei singoli servizi, per ottenere funzionalità inedite e non realizzabili con l'utilizzo di programmi separati.

[1] http://www.cs.rice.edu/Conferences/IPTPS02/109.pdf

Un elemento importante della rete PariPari sarà poi l'impostazione di un sistema di crediti dei singoli utenti, che garantisca un accesso equo alle risorse della rete.
La sicurezza viene gestita sia a livello del singolo servizio, monitorando l'accesso alle risorse macchina sensibili, che a livello di rete: essendo un sistema distribuito, è possibile garantire l'anonimato degli utenti e gestire i casi di nodi mal funzionanti o malevoli.
Il client di PariPari si propone come programma aperto: dovrà essere compatibile non solo con i servizi esistenti, ma anche con il maggior numero di configurazioni hardware e software disponibili all'utente.
Inoltre si vuole permettere l'espansione delle funzionalità mediante API (Application Programming Interface) liberamente disponibili agli sviluppatori esterni e di facile utilizzo.
Queste condizioni ne permetteranno la diffusione e si potrà giungere alla formazione di una comunità di utenti e sviluppatori in grado di migliorare sia il programma stesso che il suo utilizzo.

1.2 Il client

Il client di PariPari è stato progettato con un'architettura a plugin, in cui ogni componente fornisce agli altri delle funzionalità più o meno avanzate.
Vi è un modulo principale, che prende il nome di Core, con il compito di regolare le attività dell'intero programma.
Ogni richiesta e offerta di servizi viene fatta dai plugin al Core; quest'ultimo ne permette la trasmissione ai plugin che le eseguiranno.
Viene controllato anche l'utilizzo di risorse come la creazione di thread; il Core garantisce che i plugin ricevano una quantità adeguata di risorse macchina, e controlla che funzionalità sensibili per la sicurezza non vengano utilizzate in modo malevolo.
I servizi fondamentali sono forniti da:

- il plugin LocalStorage: permette l'accesso alla memoria in lettura e scrittura
- il plugin Connectivity: fornisce le funzionalità di connessione alla rete
- il plugin DHT (Distributed HashTable): permette di immagazzinare dati in maniera distribuita tra i nodi della rete e rende possibile l'accesso agli stessi

- il modulo Credits: garantisce un accesso equo alle funzionalità della rete, impedendo che utenti poco "generosi" nella condivisione di risorse sfruttino quelle a disposizione

Dunque il gruppo formato da Core, LocalStorage, Connectivity, DHT e Credits è la base su cui devono funzionare tutti gli altri plugin.

Il supporto ai vari servizi è realizzato da plugin, utilizzando le API della piattaforma: sono in sviluppo plugin per l'utilizzo di protocolli esistenti di *instant messaging*, Jabber[2]/xmpp[3], Windows Live Messenger[4] e IRC[5] (Internet Relay Chat), *file sharing*, bitTorrent[6] ed eD2K[7].
Lo stesso avviene per funzionalità più avanzate come Distributed Storage, Distributed dbms e Web Server HTTP (Hyper Text Transfer Protocol).
Per ogni famiglia di servizi è presente un modulo, denominato "accentratore", che si pone al di sopra dei plugin e ne coordina il funzionamento all'interno del gruppo.
Il compito degli accentratori è fornire al resto del framework una "black box" che realizza ad alto livello le funzionalità dei plugin della famiglia, gestendo internamente lo smistamento dei compiti comuni a più plugin, la configurazione dei singoli e l'uniformazione dei risultati.
Questa strategia migliora l'utilizzo interno dei plugin, perché le funzionalità disponibili non sono più direttamente dipendenti dai plugin specifici e l'intercomunicazione può avvenire su dati comuni a tutto il sistema; un aspetto delicato è l'uniformazione delle funzionalità dei plugin, che non deve avvenire in maniera eccessivamente semplicistica.
Se non fossero presenti gli accentratori, ogni plugin che facesse uso di più servizi si troverebbe a dover realizzare un analogo sistema di gestione, con spreco di risorse e difficoltà nella manutenibilità del codice.
Espandere le funzionalità di PariPari equivale a produrre un plugin che offra i nuovi servizi: è sufficiente rispettare le API fornite dai plugin interni e dall'accentratore di riferimento per ottenere una perfetta integrazione con il framework.

[2] http://www.jabber.org/
[3] http://xmpp.org/
[4] http://www.msn.com
[5] http://en.wikipedia.org/wiki/Internet_Relay_Chat
[6] http://en.wikipedia.org/wiki/BitTorrent
[7] http://en.wikipedia.org/wiki/EDonkey2000

L'applicazione è realizzata nel linguaggio Java[8]; si ha così a disposizione una tecnologia matura e diffusa, che garantisce un'ottima portabilità al programma (l'interprete del linguaggio, la Java Virtual Machine, è disponibile per la quasi totalità di sistemi operativi).
Lo sviluppo è facilitato da strumenti gratuiti di ottimo livello, una grande disponibilità di framework con le funzionalità più disparate e il supporto dato da una comunità di sviluppatori esperti.

1.3 La GUI

L'interfaccia deve combinare in un'esperienza omogenea le diverse funzionalità offerte dai plugin, offrendo all'utente qualcosa di più che il semplice affiancamento dei servizi. Per soddisfare questa necessità non si può semplicemente delegare agli accentratori la propria visualizzazione, in quanto non sarebbe possibile un'integrazione di servizi di natura differente; la GUI deve quindi porsi come entità separata e superiore ai singoli accentratori, che ne organizza le funzionalità secondo l'esperienza da offrire all'utente.

In un sistema così strutturato, la comunicazione deve avvenire delegando alla GUI la rappresentazione dei dati risultanti dalle operazioni degli accentratori (ad esempio comunicando il risultato di una ricerca di file e aggiornando i trasferimenti degli stessi), richiedendo una comunicazione diretta con l'utente (per la richiesta di configurazioni, o la comunicazione di errori) solo attraverso un sistema rigidamente gestito dalla GUI. La natura modulare di PariPari impone però che gli accentratori siano noti alla GUI solo attraverso i servizi offerti: dovranno essere formalizzate in interfacce Java le famiglie di funzionalità utilizzate dalla GUI, imponendo l'implementazione delle stesse agli accentratori interessati.
La GUI di PariPari entrerà a far parte del framework come plugin, interpretabile come un fornitore di servizi di visualizzazione.

Si sono analizzate le caratteristiche del progetto PariPari e del suo client, traendone una possibile struttura per l'integrazione di una GUI.
La natura modulare di PariPari rende difficoltosa la definizione del sistema come

[8] http://www.oracle.com/technetwork/java/index.html

un insieme statico di funzionalità, ma grazie agli accentratori è possibile gestirne la complessità con un approccio non direttamente dipendente dai componenti.

Capitolo 2

Progettazione dell'interfaccia grafica

Nel precedente capitolo è stato illustrato il progetto PariPari, e come la struttura a plugin del client definisca alcune caratteristiche di realizzazione e delle funzionalità della GUI; sulla base delle considerazioni fatte, si progetterà un'interfaccia grafica che riesca a combinare un linguaggio visivo chiaro ed accessibile con la variegata offerta di funzionalità del programma.

2.1 Principi di progettazione

Il cuore della realizzazione di una GUI risiede nella creazione di immagini formate da elementi (pulsanti, aree testuali, tabelle etc.) che permettono all'utente di interagire con le funzioni di un programma.
L'obbiettivo è offrire un ambiente accogliente, auto-esplicativo e il più possibile conforme al modello mentale che un utente può avere dei servizi disponibili; questo deve riflettersi sia nell'estetica dei componenti, che in una disposizione degli stessi coerente con le azioni offerte e con le aree funzionali che si vengono a creare.
La violazione di questi principi può portare ad interfacce in cui il numero di azioni (click del mouse, inserimento di testo, scorciatoie da tastiera) o la loro complessità sono maggiori rispetto a quanto l'utente ritenga lecito, oppure schermate in cui gli elementi grafici rappresentanti dati che l'utente vuole combinare si trovano separati; il programma risulta di difficile fruizione e si costringe ad imparare a interpretare, piuttosto che leggere, l'interfaccia.

Sebbene in linea di principio non sia lecito fare assunzioni su ciò che un utente si aspetti dalla GUI, la realtà è che avrà sempre almeno una conoscenza approssimativa delle funzionalità del programma, e una minima esperienza nell'utilizzo di software simile.
Quindi analizzando le interfacce di software che offrono funzioni analoghe è possibile ricavare un linguaggio di elementi visivi ed azioni con cui realizzare un'interfaccia immediatamente utilizzabile anche da utenti con minima preparazione.

PariPari si propone non solo di offrire un certo numero di servizi differenti, ma anche di rendere possibile una loro interconnessione: dovranno essere individuati gli elementi fondamentali per la descrizione dei diversi ambiti, garantendo la possibilità di combinare gli stessi elementi per ottenere funzionalità più generali.
La strutturazione dell'interfaccia dovrà tenere conto della natura espandibile di PariPari, e della possibilità di integrare nuovi servizi con quelli preesistenti in maniera organica e coerente.
Verranno ora analizzate le GUI di alcune popolari applicazioni di *file sharing* e *instant messaging*, per ricavare le caratteristiche essenziali a cui gli utenti sono stati abituati.

2.2 Applicazioni di *file sharing*

2.2.1 μTorrent

μTorrent[1] è un client bitTorrent, un protocollo di *file sharing* supportato anche da PariPari; si contraddistingue per il ridotto utilizzo di memoria, sia su disco che in RAM. Pur essendo un programma giovane, è in poco tempo diventato l'applicativo di riferimento per l'utilizzo di bitTorrent.

L'interfaccia presenta uno stile spoglio, ma solidamente strutturato: la schermata è divisa in più parti, relative ad aree funzionali differenti.
L'aspetto estetico è piuttosto trascurato: d'altronde si tratta di un programma che per portare a termine le sue operazioni non necessita della continua attenzione dell'utente. In particolare, alcune icone sono realizzate con uno stile rozzo e squadrato, mentre altre presentano linee morbide e piacevoli.

[1] http://www.utorrent.com/

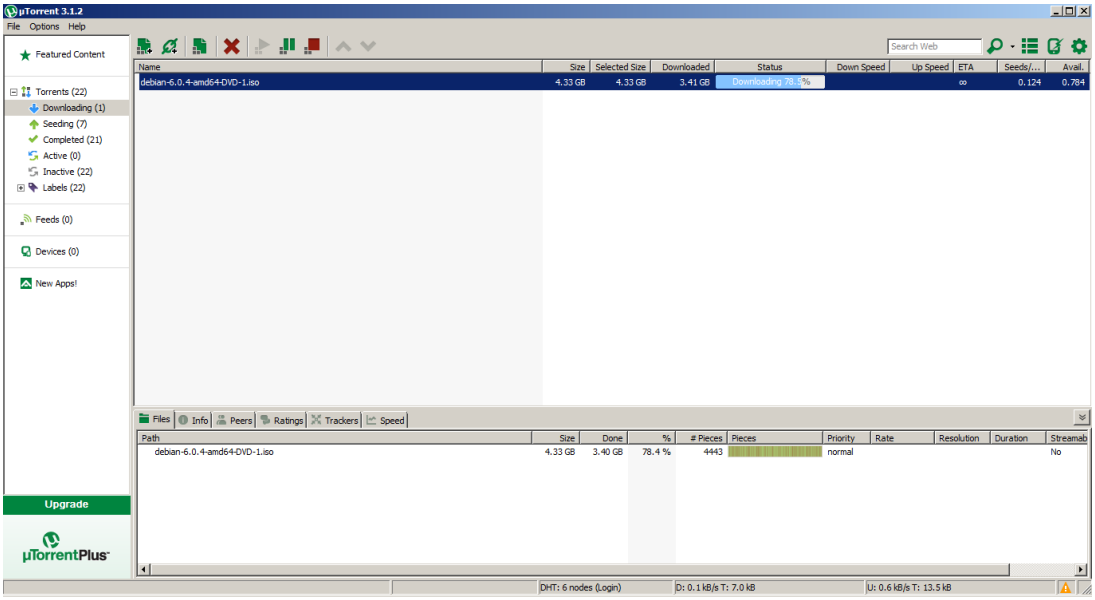

Figura 2.1: L'interfaccia di μTorrent versione 3.1.2

Il senso di lettura è da sinistra a destra, dall'alto verso il basso; gli elementi che si individuano sono:

- una barra dei menu, contenente solo tre voci, si pone come elemento di controllo ad alto livello dell'applicazione. Dai controlli qui disponibili si possono aggiungere nuovi torrent, uscire dall'applicazione e curare la configurazione del programma fin nei minimi dettagli.

- l'area sottostante è divisa in due parti affiancate orizzontalmente: quella a sinistra, più stretta, si identifica come elemento di controllo rispetto all'area di destra, più grande.
  Vi si trovano dei gruppi di pulsanti che agiscono sulla visualizzazione globale del contenuto della seconda area; il gruppo più importante è quello Torrent che permette di visualizzare i torrent a seconda di alcune caratteristiche (in scaricamento, completati, inattivi...).

- la parte di destra è a sua volta suddivisa in due sezioni, una sovrastante l'altra. Coerentemente con il senso di lettura, la sezione in alto ha precedenza su quella inferiore: qui vengono gestiti i trasferimenti dei torrent, rappresentati in una tabella che occupa la maggior parte dello spazio.
  Sopra la tabella c'è una barra degli strumenti che, oltre a replicare in forma

di pulsante alcune funzionalità del menu File, permette di fermare/far ripartire un trasferimento, cancellarlo, cercare nuovi torrent da siti internet (funzionalità avanzata che presuppone una buona comprensione del funzionamento dei siti da utilizzare), cambiare visualizzazione nella tabella e due accessi alle configurazione avanzate del programma.
Bisogna notare che alcune di queste funzioni erano disponibili in altri punti dell'interfaccia, ma qui sono presentate all'utente tramite icone e soprattutto senza la costrizione di navigare attraverso dei menu: ciò permette un'immediata comprensione dell'azione.
La tabella permette di selezionare le voci componenti le colonne, di modo che lo stato dei trasferimenti sia comunicato solo attraverso i dati più significativi per l'utente.

- l'area sottostante è dedicata alla rappresentazione di dettagli che non trovano posto nella tabella: è possibile visualizzare l'elenco di file di cui è composto il trasferimento, un insieme di statistiche, l'elenco dei *peers* disponibili, un riassunto delle valutazioni del torrent, un elenco dei *tracker* utilizzati e la rappresentazione grafica dell'andamento temporale dei flussi di download e upload.

- la schermata è conclusa da un *footer*: un'area alta pochi pixel e larga tutto lo schermo che ospita poche concise informazioni sull'andamento generale del programma.
  Costituisce un riassunto dell'esecuzione, nel caso di μTorrent vengono mostrati il numero di nodi della rete in contatto, l'entità del flusso di download e upload totale e un'icona comunica lo stato di salute della connessione.

L'interfaccia di μTorrent rappresenta l'ambiente ideale per un utente esperto del protocollo bitTorrent: ogni possibile azione è raggiungibile con pochi click da menu contestuali degli elementi che ne vengono influenzati, oppure è disponibile il relativo pulsante in una barra degli strumenti. Per un utente alle prime armi la moltitudine di opzioni offerte potrebbe invece risultare fuorviante e addirittura dannosa nel caso di configurazioni azzardate.
L'elemento focale dell'interfaccia è la tabella dei trasferimenti: i menu contestuali danno accesso a tutte le operazioni principali sui torrent e le colonne sono personalizzabili.

2.2.2 eMule

Celebre client per le reti eD2k e Kad (basata sul protocollo Kademlia), eMule[2] è uno dei volti più conosciuti del *file sharing*.
Disponibile solo per Windows, grazie alla licenza GPL[3] la base di codice ha dato origine a diverse versioni modificate come aMule[4] ed eMule AdunanzA[5].
Un aspetto centrale dell'utilizzo di eMule è il sistema di crediti alla base dei protocolli supportati, che premia gli utenti fornitori di file maggiormente richiesti.

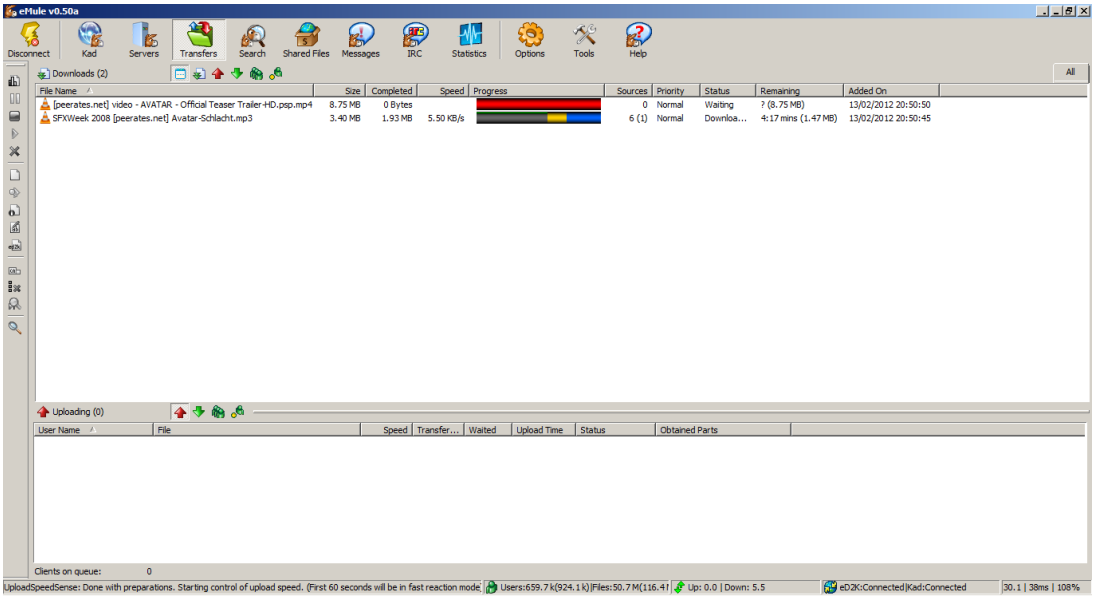

Figura 2.2: La schermata dei trasferimenti di eMule versione 0.50a

L'interfaccia di eMule è suddivisa in tre sezioni orizzontali:

- una barra di pulsanti, divisi in tre gruppi:
  - l'unico pulsante del primo gruppo connette il client alle reti secondo le configurazioni di default
  - i pulsanti del secondo gruppo visualizzano nell'area sottostante la barra una schermata per la gestione dettagliata dei vari ambiti del programma; sono

[2] http://www.emule-project.net/home/perl/general.cgi?l=1
[3] http://www.gnu.org/copyleft/gpl.html
[4] http://www.amule.org/
[5] http://www.adunanza.net/

disponibili la configurazione delle reti Kad ed eD2k, la visualizzazione dei trasferimenti in corso, uno strumento per la ricerca dei file, la gestione dei file condivisi, un client di chat, un client IRC (Internet Relay Chat) e una serie di statistiche sull'esecuzione del programma.

  - l'ultimo gruppo di pulsanti permette la gestione del programma ad un livello più astratto, rispettivamente: visualizzando un menu per la configurazione, un menu per l'accesso veloce ad alcune aree centrali dell'applicazione e accedendo al sito internet di eMule

- un footer che mostra il server della rete eD2k, il numero file e di utenti, il flusso di download e upload, lo stato di connessione alle reti e il ritardo nella trasmissione

Anche eMule è un programma polifunzionale, e la scelta di dedicare ad ogni singolo servizio una vista differente ne permette una gestione estremamente dettagliata; l'utilizzo dell'applicazione risulta però frammentato, e un utente non esperto difficilmente apprezzerà il livello di personalizzazione offerto.
La ricchezza di contenuti simili porta ad un affollamento di elementi ripetitivi e anche inutili ai fini dell'utente; questi sono fattori che PariPari vuole eliminare, offrendo invece un sistema omogeneo e coerente in cui i dettagli delle azioni sono gestiti senza un intervento diretto.

Il cuore di eMule è costituito dalle schermate dei trasferimenti, di ricerca e di gestione dei file condivisi; la prima è un elemento comune anche a μTorrent, mentre la ricerca di file viene delegata in quest'ultimo a siti esterni.
L'organizzazione dei file condivisi, che in eMule è un aspetto fondamentale per l'utilizzo efficiente del programma, in μTorrent è relegata al menu delle configurazioni.

2.3 Applicazioni di *instant messaging*

2.3.1 Skype

Skype[6] è un applicativo per la comunicazione su protocollo VoIP[7] (Voice over IP), che offre la possibilità di effettuare chiamate anche verso numeri telefonici; in seguito alla

[6] http://www.skype.com/intl/en/home?intcmp=wlogo
[7] http://en.wikipedia.org/wiki/Voice_over_Internet_Protocol

sua diffusione è stato integrato anche un servizio di chat, facendone uno dei programmi di riferimento per la comunicazione istantanea.

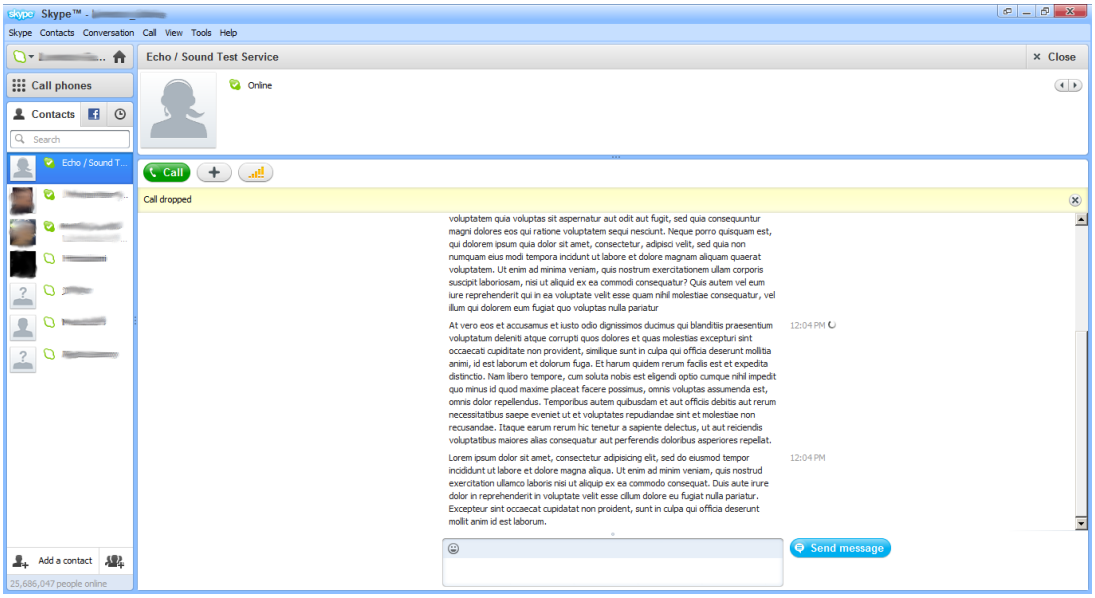

Figura 2.3: L'interfaccia di Skype versione 5.8.0.154

L'interfaccia di default di Skype ripone particolare cura nell'estetica: lo stile degli elementi grafici dà l'impressione che siano mattonelle dai bordi arrotondati adagiate su uno sfondo e lo schema di colori scelto è basato sul leggero contrasto tra bianco e tonalità pastello di azzurro e in quantità minore verde e giallo.
L'interfaccia è composta da tre sezioni:

- la fascia più alta della schermata è occupata dalla barra dei menu, contenente i controlli per l'applicazione, la gestione di contatti, le conversazioni scritte e vocali, la configurazione della GUI e del programma

- l'area sottostante è divisa in maniera simile all'interfaccia di μTorrent: a sinistra si trova un'area che, dall'alto verso il basso, contiene l'accesso ai dati del proprio profilo e un pulsante per cominciare una chiamata telefonica; il resto dell'area è dedicato alla rappresentazione dei contatti come lista verticale di immagine profilo, nome e stato di connessione.
  Sono disponibili dei pulsanti per visualizzare rispettivamente i contatti Skype, i contatti Facebook e i contatti con cui si hanno avuto delle conversazioni di

recente, ed è disponibile un campo per effettuare ricerche testuali tra i contatti; in fondo si trovano due pulsanti per aggiungere un contatto e creare gruppi

- l'area principale cambia contenuto a seconda del componente selezionato dal menu a sinistra: scegliendo il proprio profilo si possono navigare la Home di Skype, le configurazioni dell'account e la pagina del supporto utenti; la schermata per iniziare una chiamata telefonica mostra solo un tastierino numerico; infine selezionando un contatto vengono visualizzate nella parte alta l'immagine e alcune informazioni del profilo, mentre la parte inferiore è dedicata a visualizzare la conversazione.
  In caso di chat sono disponibili un'area per l'inserimento di testo e figure, e il contenuto dei messaggi viene mostrato al centro della schermata; per le conversazioni vocali invece del testo viene mostrata l'immagine del contatto , controlli per invitare altri utenti e per configurare le periferiche di acquisizione dei suoni

La GUI di Skype risulta fresca ed accogliente grazie ai colori tenui e alla morbidezza delle linee; inoltre la ricchezza di icone e di descrizioni sui pulsanti ne permette una veloce identificazione. L'accoglienza dell'interfaccia è particolarmente importante, perché l'utente si trova a passare anche molto tempo scrivendo messaggi.
Gli elementi principali dell'applicazione sono l'area verticale a sinistra, in cui vengono mostrati l'account dell'utente e i contatti, e l'area dedicata alla visualizzazione delle conversazioni.
La prima offre le principali opzioni di connettività tramite menu contestuali dei contatti.

2.3.2 Gmail

La proposta di Google nel campo dei servizi di posta elettronica è Gmail[8]; fin dalla sua apertura si è distinto per la ricchezza di funzioni e l'ampia quantità di memoria disponibile per il salvataggio delle mail.
Abbinati al servizio di posta si trovano un sistema di chat, collegamenti alle altre applicazioni web di Google e una stretta integrazione con il social network Google+[9].

L'interfaccia proposta è minimale, con pochi comandi immediatamente disponibili:

[8] https://gmail.com/
[9] https://plus.google.com/

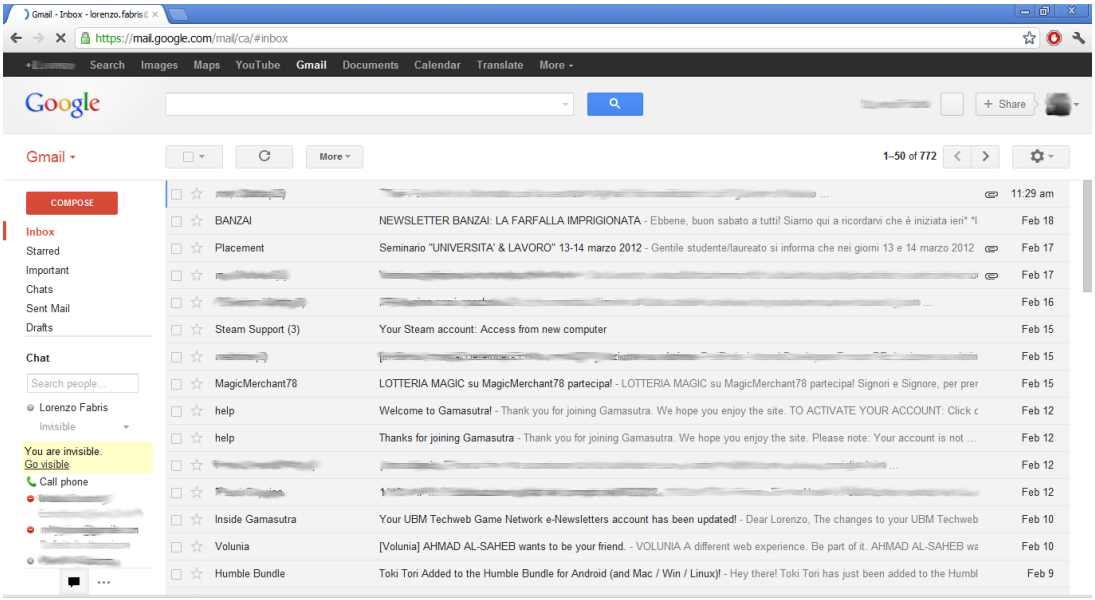

Figura 2.4: L'interfaccia di Gmail con il tema di default, visualizzata nel browser Chrome

- la parte superiore della schermata è un menu per l'accesso a vari servizi di Google; questo elemento accompagna nella navigazione i titolari di un Google Account.

- subito sotto si trova l'elemento più innovativo per un servizio di posta elettronica: una barra di ricerca che permette il filtraggio delle mail con la stessa sintassi utilizzata nel celebre motore di ricerca.
  Il compito è facilitato all'occorrenza da un form per l'inserimento dei termini di ricerca.

- proseguendo si incontra un menu che offre alcuni controlli per cambiare il contesto dell'applicazione (Mail, Contacts, Tasks), dei pulsanti per azioni di filtraggio e selezione delle mail, per l'aggiornamento della visuale, per navigare tra gli elenchi delle mail passate e per accedere alle configurazioni del client.
  Quando viene selezionata una conversazione, questo menu viene popolato di comandi aggiuntivi, che permettono l'eliminazione, l'archiviazione, lo spostamento ad altra cartella e la gestione delle etichette della conversazione corrente

- la sezione a sinistra ospita il pulsante per la creazione di una nuova mail, l'accesso ai gruppi definiti per la posta, e una sezione dedicata ai contatti: qui è possibile

iniziare una conversazione di chat (che viene archiviata come se fosse uno scambio di posta elettronica) e filtrare testualmente i contatti.

- l'area maggiore è dedicata alla visualizzazione delle mail: di default viene mostrata una lista in ordine cronologico delle conversazioni, cui viene sostituita la vista dell'intero scambio di messaggi quando ne viene selezionata una.
  Le mail in una conversazione sono rappresentate in successione cronologica, come i post di un forum

Lo stile adottato per il client di Gmail non è perfettamente intuitivo: viene privilegiata la descrizione testuale rispetto all'uso di immagini e la gestione dei contatti di chat è minimale.
La maggior parte dei controlli mostra un menu contestuale ricco di azioni, oppure agisce direttamente sull'area delle conversazioni cambiandone il contenuto; tutti questi elementi sono funzionali alla visualizzazione sui browser anche di dispositivi mobili, in cui a causa delle ridotte dimensioni dello schermo la visualizzazione di immagini *raster* è carente rispetto al testo, che invece è descritto in forma vettoriale.

2.4 PariGUI

L'accesso a PariPari avverrà tramite il suo sito ufficiale, sfruttando la tecnologia *Java WebStart*[10]; per accentuare l'idea di esperienza di connessione alla rete, si vuole che l'interfaccia del client sia visualizzata all'interno del browser e permetta anche il controllo dell'applicazione da remoto.
Si vuole rendere irrilevante per l'utente la modalità di esecuzione del programma: PariPari deve risultare come un servizio sempre accessibile, che come qualsiasi web application necessita solo di un browser per essere eseguita.
Queste richieste hanno importanti ripercussioni sulla realizzazione, e verranno discusse nel prossimo capitolo.

Poiché l'interfaccia è oggetto di utilizzi prolungati nel tempo, lo stile grafico deve essere fresco e riposante alla vista: è stata scelta una tavolozza basata sul verde chiaro e il bianco, con particolari in nero/grigio.
L'utilizzo di gradienti di intensità e di angoli arrotondati per gli elementi di maggior risalto (finestre, pulsanti), allo stesso modo che nell'interfaccia di Skype, permettono

[10] http://www.java.com/en/download/faq/java_webstart.xml

un'esperienza morbida e riposante.

In quanto applicazione polifunzionale, PariPari può far buon uso di un sistema a schermate analogo a quello di eMule; a differenza di questo bisognerà però garantire degli elementi di intercomunicazione tra le viste, posti graficamente al di fuori di esse, per identificare la loro appartenenza all'applicazione piuttosto che ad un singolo ambito.
Per garantire coerenza ed uniformità nell'utilizzo dell'interfaccia, le modalità di interazione con gli elementi grafici dovranno essere il più possibile standardizzate tra schermate differenti.
Allo stato attuale, PariPari può contare sulla presenza degli accentratori di *file sharing* e di *instant messaging*. Le funzionalità da offrire all'utente sono rispettivamente:

- ricerca di file da scaricare in rete con la possibilità di selezionarli per le loro caratteristiche, visualizzazione e organizzazione dei file attualmente in trasferimento e condivisi e controllo dei trasferimenti

- visualizzazione e organizzazione di contatti e gruppi di contatti, possibilità di comunicare via chat (prevedendo VoIP, videochat e posta elettronica)

Inoltre si vuole offrire la possibilità di inviare file ai singoli contatti anche al di fuori di una sessione di chat; tra gli sviluppi futuri ci saranno il supporto allo storage e backup distribuiti e la ricerca testuale tra le conversazioni.
L'elemento della ricerca sarà comune ai due contesti: per mantenere la coerenza dell'interfaccia, deve essere realizzato con lo stesso aspetto e presentare un funzionamento il più possibile uniforme tra le schermate.

L'area dello schermo è divisa in tre zone funzionali, similmente alle interfacce delle applicazioni analizzate.
A sinistra viene visualizzata una lista di contatti e gruppi: divengono così un elemento focale dell'applicazione, oggetto dell'interazione con tutte le altre funzionalità del programma.
La rappresentazione consiste di una lista verticale contenente i nomi dei gruppi disponibili, espandibili a mostrare i contatti che ne fanno parte; è previsto un raggruppamento dei contatti senza appartenenze, ed è ammesso che un contatto appartenga a più gruppi.

Una barra di ricerca posta in cima alla lista permette di filtrarne gli elementi per nome. La zona a destra è adibita alla visualizzazione delle schermate dedicate agli accentratori, con il meccanismo a tab ripreso dai moderni web browser: delle etichette poste nella parte alta permettono di selezionare il contesto, e nella parte inferiore viene visualizzata la schermata corrispondente.

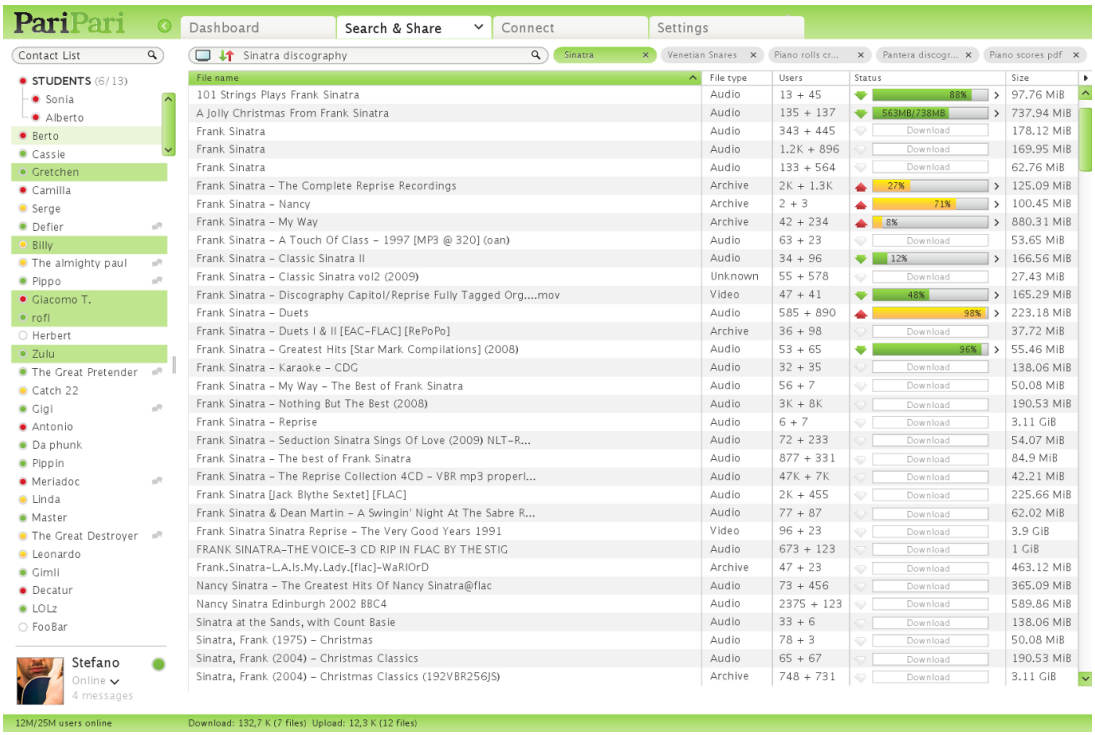

Figura 2.5: Prototipo della schermata di *Search&Share* ad opera di Stefano Calgaro

La schermata dedicata al *file sharing*, denominata *Search&Share*, è divisa in due parti: in alto viene posta l'area di ricerca, che mostra le ricerche effettuate, mentre al di sotto sono visualizzati in una tabella i file relativi alla ricerca selezionata.
I file sono accessibili solo a partire da una ricerca: sono attivabili dei filtri per selezionarne la tipologia, lo stato di trasferimento e/o la locazione (presenza sulla macchina o nella rete).
A differenza di quanto avviene in eMule, non vi è una separazione netta tra la gestione dei file in trasferimento e quelli condivisi: tutti i file a disposizione possono essere visualizzati nello stesso luogo e contemporaneamente.
Le colonne della tabella riportano le principali informazioni del file e dei trasferimenti

cui è sottoposto; elemento inedito rispetto alle interfacce degli altri client è l'accesso ai controlli del trasferimento tramite un menu richiamabile con un pulsante.
Inoltre si introduce la possibilità di inviare un file ad un contatto (o ad un gruppo) tramite *drag and drop* del file dalla tabella di *Search&Share* sul contatto.

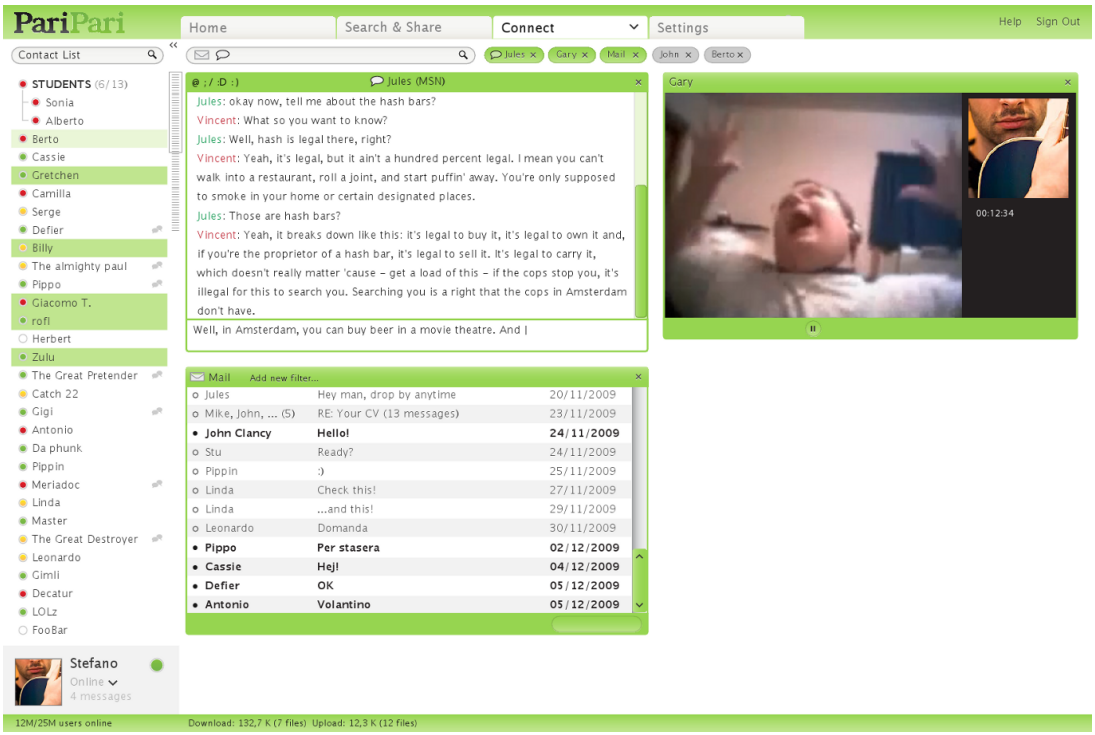

Figura 2.6: Prototipo della schermata di *Connect* ad opera di Stefano Calgaro

L'interfaccia di *Connect* è invece simile all'ambiente desktop di un sistema operativo: l'area di ricerca, posizionata in alto per coerenza con *Search&Share*, mostra le conversazioni esistenti.
L'area sottostante visualizza (in numero massimo di quattro, per evitare una eccessiva riduzione delle dimensioni su schermo) le conversazioni selezionate in forma di finestre. È possibile agire sulle finestre chiudendole (con effetto di terminare la conversazione) o nascondendole (azione analoga alla minimizzazione nei sistemi operativi, la conversazione continua ad essere attiva); al loro interno si trovano i controlli relativi al tipo di conversazione cui si riferiscono.
Nel caso di una finestra di chat, l'unica modalità di comunicazione al momento disponibile, sono presenti un'area per l'inserimento del testo, i pulsanti di invio e modifica,

e un'area per visualizzare il contenuto della conversazione.
L'area di ricerca deve essere abbastanza flessibile da adattarsi all'uso che ne farà ogni tab; deve poter utilizzare dei filtri attinenti al contesto, i cui effetti possono applicarsi alle ricerche selezionate, in tempo reale, o alla ricerca prodotta all'attivazione del pulsante.
I filtri sono posizionati a sinistra rispetta all'area di inserimento del testo, mentre i risultati sono visualizzati come "bolle" che si espandono verso destra.
Queste bolle riportano un breve testo che funge da identificativo, possono essere selezionate con un click sulla loro area centrale o eliminate con un click sulla croce alla loro destra; il loro colore riflette lo stato dell'oggetto cui sono legate.
Al momento sono disponibili tre stati, identificati da colori diversi: selezionato/attivo (verde), inattivo (grigio) e modificato (arancione).
Nel caso che il numero di bolle visualizzate cresca troppo, quelle più vecchie vengono raggruppate in un'unica verde riportante il testo "..."
La schermata di *Search&Share* permette una sola bolla attiva in ogni momento e utilizza lo stato modificato per segnalare le ricerche a cui sono stati aggiunti risultati mentre erano nascoste.
Similmente *Connect* utilizza lo stato attivo per identificare le conversazioni visualizzate nell'area sottostante e lo stato modificato per le conversazioni in cui sono stati aggiunti messaggi non ancora letti.
Anche l'ordinamento delle bolle deve essere controllato dalle tab; in linea di principio, le bolle attive dovrebbero essere poste a sinistra nell'area di immagazzinamento.

La terza area dello schermo è occupata da un footer su cui visualizzare lo stato della rete, dei trasferimenti e un accesso alle notifiche del programma, analogamente a quanto visto nelle altre interfacce.

Il progetto cui si è giunti descrive un'interfaccia grafica espressiva e di facile fruizione; rispetto alle applicazioni concorrenti presenta un minor livello di dettaglio, ma ciò ne fa lo strumento più versatile per un approccio coerente alla multifunzionalità di PariPari.
Nel prossimo capitolo verranno analizzati gli strumenti e le scelte realizzative adottate per la realizzazione del progetto.

Capitolo 3

Realizzazione del plugin GUI

Avendo definito nel primo capitolo le funzionalità con cui una GUI deve integrarsi nella struttura a plugin di PariPari, e sulla base del progetto cui si è giunti nel secondo capitolo, verrà illustrato il processo di realizzazione dell'attuale interfaccia.

3.1 Gli strumenti

3.1.1 Eclipse

L'IDE (Integrated Development Environment) utilizzato è Eclipse[1]: scritto in e per Java, supporta ottimamente il linguaggio grazie ad un potente sistema di indicizzazione di package, classi e metodi e ad una vasta scelta di strumenti per la modifica del codice. Sono presenti un gran numero di scorciatoie da tastiera per velocizzare l'accesso agli strumenti di modifica, con la possibilità di configurarne di nuove; ciò permette di raggiungere una grande velocità di scrittura, che combinata con l'analisi lessicale riduce gli errori.
La struttura a plugin di Eclipse offre poi ampie possibilità di espansione: oltre al supporto a linguaggi diversi da Java, è possibile integrare le funzionalità dei client dei più diffusi programmi di controllo di versione come SVN[2] (Apache Subversion), utilizzato per lo sviluppo di PariPari, ma anche Git[3] e Mercurial[4].

[1] http://www.eclipse.org/
[2] http://en.wikipedia.org/wiki/Apache_Subversion
[3] http://git-scm.com/
[4] http://mercurial.selenic.com/

3.1.2 Vaadin

Vaadin[5][6] è un framework per la realizzazione di web application nel linguaggio Java. A differenza di soluzioni basate sul solo JavaScript[7] o su plugin per il browser, Vaadin è basato sull'esecuzione di una sezione server, scritta in Java, e di una sezione browser basata su AJAX[8] (Asynchronous JavaScript and XML) e realizzata con l'utilizzo di GWT[9] (Google Web Toolkit).
Lo sviluppatore ha a disposizione i comuni componenti utilizzati dai framework per UI (User Interface), come Button, TextField, Layout; la realizzazione di schermate avviene con la creazione di alberi di componenti e la definizione delle loro relazioni funzionali. Sono disponibili più modi per creare componenti personalizzati:

- la semplice composizione di più elementi basilari
- l'applicazione di uno stile CSS[10] (Cascading Style Sheet) al componente, funzionalità offerta dalla visualizzazione nel browser ed esposta da Vaadin al codice Java
- la scrittura delle classi server-side e client-side del nuovo componente; questa via è difficilmente percorribile, a causa della profonda conoscenza di GWT e Vaadin richiesta

Una particolarità di Vaadin è la presenza di un modello di dati, ampiamente utilizzato nei componenti per la visualizzazione di gruppi di oggetti.
Tale modello si compone di tre elementi:

- **Property**: rappresenta un campo di un oggetto, dandone accesso al valore e al tipo e permettendone la modifica
- **Item**: costituito da un insieme di Property, rappresenta la serializzazione dell'oggetto
- **Container**: è l'elemento che permette la visualizzazione di un insieme di Item

[5] https://vaadin.com/home
[6] il testo di riferimento per l'utilizzo è 4
[7] http://en.wikipedia.org/wiki/JavaScript
[8] http://en.wikipedia.org/wiki/Ajax_programming
[9] http://code.google.com/webtoolkit/
[10] http://www.w3.org/Style/CSS/

Le modifiche alle Property di un Item vengono propagate fino al Container in cui è contenuto, dando modo a quest'ultimo di aggiornare oculatamente la visualizzazione dell'elemento modificato; Vaadin permette l'interfacciamento automatico con database basati su SQL[11] (Structured Query Language) e una conversione da oggetti Bean[12] a Item.
L'utilizzo di oggetti non Bean è più difficoltoso, e richiede la modifica del Container che si desidera utilizzare; inoltre nell'aggiornamento di un oggetto generico non è sempre possibile modificare una singola Property, costringendo il Container ad aggiornare la visualizzazione più volte del necessario.

Vaadin è ben integrato in Eclipse grazie ad un plugin dedicato, che comprende la documentazione completa e un editor visuale per l'aggregazione di componenti.
Il supporto allo sviluppo è completato da un sito ricco di esempi e da una vasta community.

3.2 Situazione iniziale

Lo sviluppo della GUI è cominciato prima che fossero realizzati i plugin accentratori; ciò ha imposto che molte funzionalità di controllo dei plugin fossero inizialmente delegate alla GUI stessa, evidenziando ben presto come una tale gestione ben superasse le funzionalità di un'interfaccia grafica.
Il modello di dati sviluppato per descrivere ad alto livello i dati si è rivelato sufficientemente flessibile da essere stato adottato non solo nella comunicazione con GUI, ma anche all'interno degli stessi accentratori per la fusione e l'uniformazione dei plugin.
Molte soluzioni sviluppate per questo primo approccio sono poi state adottate nella realizzazione degli accentratori, a cominciare dalle astrazioni adottate per uniformare i dati dei plugin.
I moduli da supportare erano Mulo (supporto ad eD2k) e Torrent nell'ambito del *file sharing*, IRC e IM (supporto a Windows Live Messenger e Jabber/xmpp) per l'*instant messaging*.
La prima necessità è stata elaborare delle interfacce sufficientemente flessibili su cui i plugin potessero adattare i propri tipi, e che rendessero possibili le funzionalità richieste

[11] http://en.wikipedia.org/wiki/SQL
[12] http://docs.oracle.com/javase/tutorial/javabeans/

alla GUI.
Si è giunti alle seguenti definizioni (il prefisso IPG identifica un'interfaccia appartenente alla GUI):

- IPGFile: descrive un file come un insieme di nome, dimensione, tipo, un generico campo "availability", numero di utenti in grado di fornire il file, percentuale di scaricamento, direzione principale del trasferimento, un valore booleano per identificare i file posseduti dall'utente e il tempo previsto alla fine del trasferimento.
- IPGFileQuery: contiene solo una stringa, rappresenta una ricerca testuale
- IPGContact: rappresenta un generico contatto, contiene un nome, un insieme di stringhe che rappresentano i gruppi di cui il contatto fa parte, un booleano per stabilire se il contatto è conosciuto e/o bloccato, lo stato del contatto e un insieme di IPGContactLocation
- IPGUser: rappresenta l'utente come un contatto e le relative credenziali di accesso
- IPGContactLocation: rappresenta un profilo in un generico sistema di *instant messaging*, contiene un nickname, un identificativo (generalmente un indirizzo email), lo status di connessione e il protocollo utilizzato
- IPGConversation: una conversazione viene rappresentata dall'insieme dei contatti che vi prendono parte
- IPGMessage: un messaggio in una conversazione di *instant messaging*; contiene l'identificativo della conversazione di cui fa parte, un contatore, l'istante temporale in cui viene ricevuto, l'identificativo dell'autore e il contenuto testuale

I campi relativi al tipo di file, direzione di trasferimento, protocollo di *instant messaging* e status del contatto sono stati modellati come enumerazioni; nella versione definitiva i protocolli di *instant messaging* dovranno essere forniti dall'accentratore con un meccanismo differente, in quanto non noti alla GUI.
Queste interfacce non pretendono di essere perfettamente esaurienti, ma potranno essere modificate una volta che il resto di PariPari avrà raggiunto la maturità richiesta.

La mancanza di un immediato supporto dei plugin ha poi imposto la necessità di sviluppare degli strumenti alternativi per il test della GUI; gli unit-test normalmente

utilizzati per lo scopo non possono imitare i movimenti del mouse né verificare la corretta visualizzazione delle schermate.
Si è quindi realizzato un generatore di dati e di eventi di conversazione, basato sulla struttura del plugin GUI, che ha permesso l'interazione in tempo reale con le principali funzionalità dell'interfaccia.
L'interfacciamento è stato poi progressivamente convertito all'utilizzo degli strumenti di connessione ai plugin con poco sforzo.

3.3 La struttura del plugin

Le specifiche di supporto al controllo remoto, dell'utilizzo di Vaadin e dell'interfacciamento con il framework di PariPari hanno portato ad un plugin GUI diviso in due parti:

- la parte costituita dal plugin vero e proprio: si occupa dell'interfacciamento con il resto dei plugin e della comunicazione con il Core per la creazione del server su cui eseguire l'applicazione Vaadin.
  I componenti fondamentali sono:
  - ServerConnectionHandler e Synchronizer, classi incaricate della gestione di un canale di comunicazione tra plugin e server; tutta la comunicazione avviene tramite l'invio di pacchetti di dati serializzati con JSON[13] (JavaScript Object Notation)
  - PGDataWarehouseManager: questa classe agisce da magazzino per tutti i dati che i plugin producono in seguito alle proprie azioni e che la GUI deve visualizzare
  - LocalPluginInquirer, la classe incaricata della propagazione dei comandi provenienti dall'applicazione Vaadin al resto di PariPari
  - PluginManager: permette alla GUI di gestire i plugin con cui dovrà comunicare
  - GUIAPIImplementation: offre ai plugin l'interazione con PGDataWarehouseManager per quanto riguarda il flusso di dati, e con Synchronizer per la comunicazione diretta con l'applicazione Vaadin

[13] http://www.json.org/

- la parte che realizza l'interfaccia utente, basata sulla struttura di Application di Vaadin.
  Si compone di:
  - PGDataWarehouseManager, copia di quella creata dal plugin e con questa sincronizzata
  - ConnectionHandler, riferimento al medesimo oggetto creato dal plugin
  - RemotePluginInquirer, permette l'instradamento dei comandi della GUI nel canale; verranno ricevuti dal LocalPluginInquirer del plugin
  - PariGUI: estende la classe Application di Vaadin ed è il punto di entrata per l'esecuzione su server.
    Rende accessibile il RemotePluginInquirer al resto degli oggetti, e gestisce la creazione e rimozione delle finestre interne all'applicazione

Nel primo capitolo si era discussa la necessità di operare una separazione tra l'insieme dei dati e delle funzionalità dei plugin, e la loro visualizzazione: questa strategia di progetto è nota come design pattern MVC (Model View Controller)[14].
L'elemento centrale è il Controller, che gestisce la visualizzazione (View) e il modello di dati (Model); la View ha il compito di fornire una rappresentazione dei dati, ma non può modificarli direttamente. L'input dell'utente viene comunicato dalla View al Controller, che può agire di conseguenza sia sui dati che sulla View.
Nella GUI di PariPari il ruolo di Model è realizzato dalla data warehouse e la View è costituita dall'applicazione Vaadin; il ruolo di Controller è invece interpretato dalla combinazione delle classi Remote e LocalPluginInquirer con i plugin accentratori.

3.3.1 La DataWarehouse

La data warehouse mette a disposizione della GUI i dati dei plugin in maniera indiretta; i plugin hanno facoltà di aggiungere, eliminare e modificare i dati di propria competenza, che vengono organizzati in gruppi sulla base delle funzionalità della GUI.
Attualmente vengono creati e mantenuti:

- un insieme di tutti i contatti noti
- l'insieme di tutte le conversazioni disponibili e dei messaggi che ne fanno parte

[14] Cfr: [5]

- le ricerche di file effettuate e i gruppi di file risultanti

Gli oggetti memorizzati dalla datawarehouse sono le astrazioni definite precedentemente, cui è stato aggiunto come identificativo univoco un campo UUID (Universally Unique IDentifier).
Per mantenere la visualizzazione aggiornata è necessario comunicare agli elementi interessati le modifiche che intervengono sui dati e sui gruppi; questo meccanismo è realizzato con la possibilità di registrare dei listener sugli elementi della data warehouse, applicazione del diffuso pattern dell'*Observer*[15].
I componenti che visualizzano liste di dati, come la lista dei contatti, registrano un listener al gruppo stesso e ad ogni oggetto di loro interesse contenutovi; in questo modo è possibile anche mantenere sottoinsiemi dei gruppi nella data warehouse, selezionando prima gli oggetti desiderati e gestendo solo le azioni di aggiunta e rimozione che riguardano il sottoinsieme di interesse.
La GUI non ha facoltà di modificare nessun oggetto nella datawarehouse; la richiesta di qualsiasi azione viene mandata ai plugin e applicata quando possibile.
PGDataWarehouseManager è realizzata come *Singleton*[16], un design pattern che impone l'esistenza di un'unica istanza della classe, accessibile globalmente con un metodo statico.

3.4 La MainWindow

Questa classe fornisce l'intelaiatura in cui si inseriscono i vari elementi grafici: lo spazio viene diviso nelle tre zone illustrate nel capitolo 2 con l'utilizzo di diversi Layout e uno SplitPanel.
L'*header* ospita il logo di PariPari, l'area delle etichette delle Tab e dei pulsanti per eseguire il login alla rete e ricevere supporto nell'utilizzo del programma. Queste ultime due funzionalità non sono al momento supportate, e sono destinate a subire modifiche in futuro.
Lo Splitpanel, di dimensione fissa, contiene la visualizzazione della lista dei contatti, controllata dalla classe PGContactListController, e l'area destinata al contenuto delle Tab che verrà analizzata nella prossima sezione. È possibile nascondere l'area dei con-

[15] ampiamente trattato in [5]
[16] ampiamente trattato in [5]

tatti con un pulsante posto nello header.
Nella parte bassa dello schermo viene visualizzato il footer.

3.4.1 PGContactListController

Questa classe gestisce la visualizzazione dei contatti, delle loro funzionalità e la ricerca tra di essi. La parte superiore ospita un SuperImmediateTextField[17], con cui è possibile filtrare contatti e gruppi per nome; questo componente non standard è disponibile come espansione di Vaadin, ed aggiunge alle funzionalità del semplice TextField la notifica ad un listener del proprio cambiamento di stato poco dopo che l'utente ha finito di digitare il testo della ricerca.
Il resto dell'area è occupato da un PGContactTree, classe che estende il Tree di Vaadin per semplificare l'utilizzo degli oggetti in data warehouse e personalizzarne visualizzazione e gestione.
PGContactTree è un primo esempio di listener di datawarehouse: ascolta gli eventi provenienti sia dalla lista che dai singoli contatti.
Rispetto al prototipo del secondo capitolo, i gruppi visualizzano a fianco del nome la quantità di contatti disponibili (con cui è possibile cominciare una conversazione) e il numero di quelli che ne fanno parte.
Il singolo contatto possiede un'icona per segnalarne lo stato (disponibile, impegnato, non al computer, non connesso) con un codice di colori, e la stessa icona è modificata con l'aggiunta di un punto di domanda nel caso che il contatto non sia memorizzato in maniera permanente nell'account dell'utente.
È disponibile un menu contestuale sia per i gruppi che per i contatti: per i primi è possibile lo scioglimento (semplice rimozione dell'appartenenza al gruppo dai contatti che ne fanno parte) o la completa rimozione dei contatti contenuti; per gli altri vengono fornite la possibilità di iniziare una conversazione (sia specificando il protocollo tra quelli disponibili che lasciando questa scelta al programma), di modificare le proprietà del contatto e di rimuoverlo completamente dalla lista.
La classe PGContactTree è resa complessa dall'impossibilità del Tree di Vaadin di visualizzare lo stesso oggetto in più sottoalberi: per la realizzazione di questa funzionalità (che corrisponde all'appartenenza di un contatto a più gruppi) è stato necessario utilizzare strutture dati aggiuntive per memorizzare sia i riferimenti agli oggetti che riflettono

[17] https://vaadin.com/forum/-/message_boards/view_message/77093

lo stato del singolo contatto che quelli assegnati ai gruppi.
Inoltre, poichè non è stata prevista una classe per i gruppi, la gestione di questi avviene solo attraverso modifiche dei contatti: ciò impone che le relative funzionalità siano piuttosto complesse, e debbano essere realizzate con particolare attenzione.
Nel caso la lista risultasse vuota, o in seguito ad una ricerca senza risultati o per un'effettiva mancanza di contatti noti, al posto dell'albero viene mostrato un pulsante per l'aggiunta di un nuovo contatto, che avviene con lo stesso meccanismo usato per la modifica.

3.4.2 PGContactFormWindow

Questa classe realizza una *subwindow* contenente un form per modificare o creare un contatto.
I vari campi riflettono la composizione della classe PGContact: in particolare, l'aggiunta di location avviene solo dopo un controllo (basato su espressioni regolari) della validità dell'identificativo e le tabelle per la gestione dei gruppi permettono il drag and drop dai gruppi disponibili a quelli del contatto.
La finestra viene creata con i campi vuoti se l'utente vi è acceduto per la creazione di un nuovo contatto, o con i dati iniziali in caso di modifica di un contatto esistente.

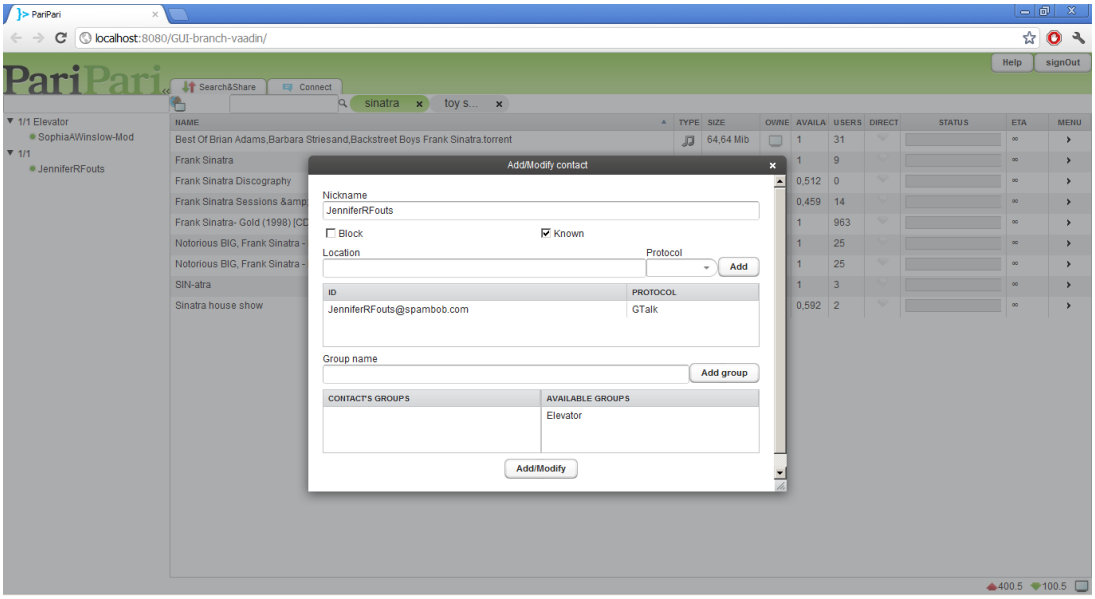

Figura 3.1: Aspetto di PGContactFormWindow nella gestione di un contatto

3.4.3 Le etichette delle tab

Costituita da un semplice HorizontalLayout, quest'area visualizza le Tab disponibili come pulsanti dotati di icona e nome.
La realizzazione di un meccanismo analogo sarebbe stata possibile semplicemente con il componente TabbedPanel di Vaadin, ma sarebbe stato impedito il supporto al drag and drop di elementi esterni sulle Tab; pur se al momento non realizzato, la soluzione adottata prevede almeno la possibilità di espansione in tal senso.
La corretta visualizzazione delle etichette è basata sull'utilizzo di uno stile css apposito e di immagini per lo sfondo dei pulsanti personalizzate; l'aspetto grafico è determinato dall'uso di due immagini distinte, una per il bordo sinistro e una per il resto. La modifica del comportamento predefinito ha richiesto un'approfondita analisi delle pagine web prodotte da Vaadin.
L'utilizzo di icone, non previste dal prototipo, è stato inserito per aumentare la leggibilità dell'interfaccia; manca invece un meccanismo per l'identificazione della Tab attiva, realizzabile producendo un altro stile css.

3.4.4 Il Footer

Questo componente non ha ancora delle funzionalità complete: è prevista la visualizzazione dei flussi di download e upload con l'utilizzo di icone e testo, e l'accesso ad una lista di generiche operazioni in sospeso.
Lo sviluppo procederà quando sarà fornito un effettivo supporto da parte del programma.

3.5 Le Tab

Le classi che realizzano la fornitura dei servizi specifici di PariPari sono le Tab: la loro visualizzazione avviene nell'area a destra della lista dei contatti, con cui si prevede una frequente interazione.
Ogni tab è definita da un nominativo e un'icona per quanto riguarda le etichette, mentre il contenuto da visualizzare è reso disponibile alla MainWindow come ComponentContainer.

3.5.1 La SearchBar

L'elemento della ricerca è comune alle tab finora previste; per mantenere l'omogeneità di rappresentazione, la barra di ricerca e i relativi risultati in forma di bolle sono gestiti dalla classe astratta PGSearchBar, le cui funzionalità possono essere personalizzate dalle tab combinando ereditarietà e pattern *Observer*.

La classe QueryEvent rappresenta un evento di ricerca: permette di risalire allo stato dei filtri al tempo della creazione, al testo della ricerca, alla modalità (permanente o temporanea) e alla relativa bolla.

I filtri sono realizzati dalla classe SearchBarFilter e visualizzati come pulsanti a più stati dalla classe interna FilterManager; i singoli stati vengono creati come oggetti FilterState, contengono una stringa come identificativo e un'icona per la rappresentazione dello stato nel pulsante.
La SearchBar prevede una distinzione per i filtri attivi e passivi: i primi generano un evento di ricerca temporaneo ad ogni variazione di stato, mentre i secondi non hanno effetto immediato.
L'immissione di testo per la ricerca avviene in un SuperImmediateTextField, che genera un evento temporaneo al termine della digitazione; un pulsante a fianco del campo permette invece la generazione di eventi permamenti.
La bolla di ricerca è realizzata dalla classe Bubble: rende disponibile gli elementi per ricostruire lo stato dei filtri e dell'area testuale, e dispone di uno stato attivo, inattivo o modificato.
I Bubble sono posizionati alla destra della SearchBar; quando sono in numero eccessivo (fissato arbitrariamente a sei nell'attuale versione), viene visualizzato un Bubble che raggruppa al suo interno le bolle eccedenti, accessibili con un DropDownMenu.
La rappresentazione è realizzata con un Layout cui viene applicato uno stile css per ottenere la colorazione e i bordi arrotondati, un pulsante che visualizza il nome della bolla e un pulsante per l'eliminazione della bolla.

Applicando dei listener alla SearchBar e ai Bubble è possibile intercettare tutti gli eventi di ricerca generati e la selezione di bolle.

3.5.2 I Widget

Le tab di *Search&Share* e *Connect* hanno comportamenti assimilabili al funzionamento di un desktop manager nei moderni sistemi operativi: permettono di portare in primo piano, nascondere ed eliminare degli elementi grafici (uno solo per *Search&Share*, fino a quattro per *Connect*) nell'area ad esse assegnata.
Questa particolarità ha portato alla definizione delle interfacce IPGWidget e IPGWidgetManager, che specificano i metodi per visualizzare, nascondere, aggiungere ed eliminare gli elementi visualizzati (Widget) da parte delle tab (WidgetManager).
Le bolle di ricerca vengono utilizzate dalle tab per funzionare come le finestre minimizzate a cui gli utenti sono abituati: la selezione di un Bubble inattivo porta in primo piano il relativo Widget, e analogamente la chiusura di un widget attivo determina la rimozione del Bubble collegato.
In seguito a queste operazioni, l'ordinamento dei Bubble può cambiare: la variazione è gestita dalle classi private alle tab che ereditano da SearchBar, sovrascrivendo i relativi metodi.

3.5.3 SearchAndShareTab

L'attuale aspetto e funzionamento della tab di *Search&Share*, realizzata da PGSearchAndShareTab, è visibile in figura 3.2 .
La tab applica un listener al gruppo dei risultati di ricerca in data warehouse: in seguito ad eventi di aggiunta viene creato un nuovo Widget, che rappresenta l'insieme dei file del risultato in una PGSearchAndShareTab (derivante dalla più generica PGFileTable, una Table adattata alla visualizzazione di IPGFile).

La barra di ricerca utilizzata presenta tre filtri attivi: il primo seleziona la locazione, il secondo la direzione di trasferimento e il terzo il tipo di file da visualizzare nella tabella sottostante. La sola digitazione nel campo testuale determina un filtraggio sul nome dei risultati in tabella, mentre la creazione di nuove ricerche viene richiesta ai plugin all'attivazione del pulsante.
Per quanto riguarda l'ordinamento dei Bubble, si è deciso di mantenere nella posizione più a sinistra l'unica bolla attiva: la selezione di una inattiva ne determina lo spostamento in prima posizione, mentre le altre scorrono semplicemente a destra.

La tabella reagisce agli eventi del relativo risultato di ricerca e dei file che lo compongono; i campi dei file sono rappresentati sia con colonne testuali che con l'utilizzo di icone.
La percentuale di trasferimento è rappresentata con una semplice WaitBar, che a differenza del comportamento standard non permette l'aggiunta di un valore numerico sul componente; questa mancanza non può essere corretta con l'applicazione di uno stile css e non è stata ancora realizzata una soluzione definitiva.
Il menu disponibile nell'ultima colonna permette il controllo dei trasferimenti: sono disponibili le azioni per attivare, mettere in pausa e cancellare i trasferimenti attivi, oltre all'invio con destinatario e la rimozione del file dal disco.
È possibile il drag and drop di un file dalla tabella alla lista dei contatti, interpretato come la richiesta dell'invio del file al contatto o al gruppo su cui viene effettuato il drop.

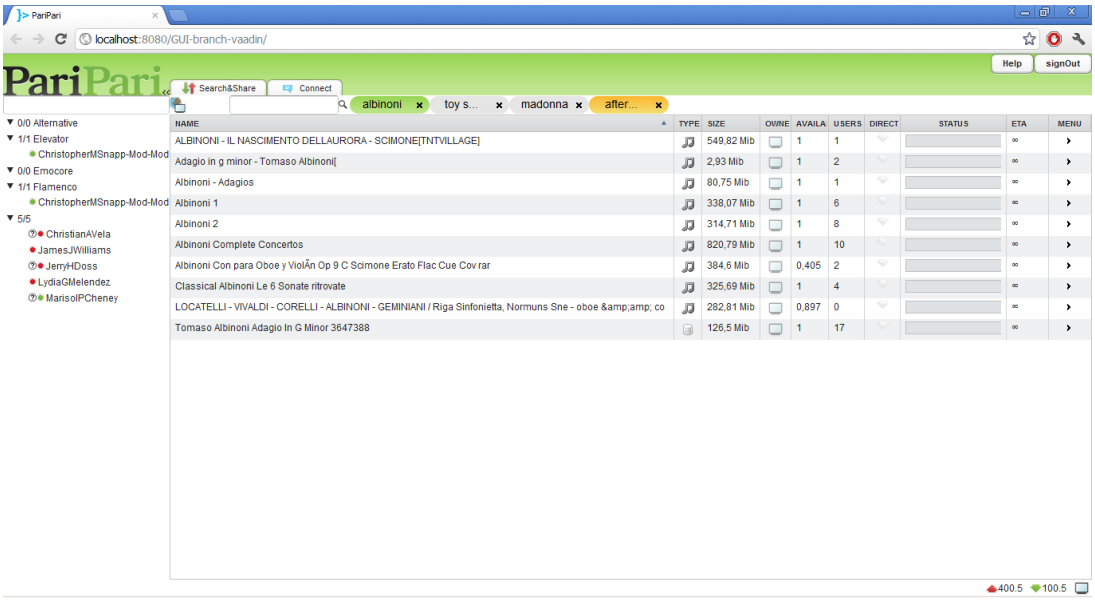

Figura 3.2: Attuale aspetto della tab *Search&Share*; i filtri sono tutti in stato inattivo

3.5.4 ConnectTab

La schermata di *Connect*, mostrata in figura 3.3, è realizzata dalla classe PGConnectTab.

I Widget vengono creati dalla tab in seguito ad eventi di aggiunta sul gruppo delle conversazioni in data warehouse; le bolle di ricerca rappresentano quelle esistenti,

vengono mantenuti nelle prime quattro posizioni i Bubble relativi ai Widget in primo piano.
Al momento l'unico tipo di conversazione è la chat testuale, rappresentata dalla classe PGChatWidget: l'aspetto del componente dà l'illusione che si tratti di una finestra, ma è realizzato con l'applicazione di uno stile css a un complesso di Layout e pulsanti. È possibile minimizzare o eliminare il Widget utilizzando i pulsanti in alto a destra, coerentemente con il loro aspetto; il posizionamento del componente è determinato dall'ordinamento dei Bubble.
In alto a sinistra è posto un pulsante che visualizza in una finestra due gruppi di contatti: a sinistra sono rappresentati i contatti che prendono parte alla conversazione e a destra sono indicati i contatti che dovranno essere cacciati.
La modifica di questi gruppi è possibile con il drag and drop dei contatti tra le due tabelle; l'invito di un nuovo partecipante avviene con il trasporto di questo dalla lista dei contatti alla prima tabella.

L'area che visualizza la conversazione è una semplice Label, il cui contenuto viene inserito sotto forma di XHTML[18] (eXtensible HyperText Markup Language) per ottenerne la colorazione o altri effetti grafici. I nomi degli autori dei messaggi vengono resi più facilmente distinguibili colorandoli casualmente.
Nel caso in cui arrivasse un nuovo messaggio in una conversazione che non avesse avuto attività da almeno cinque minuti, viene aggiunto l'istante di arrivo del messaggio.
Completano il Widget un campo per la digitazione del messaggio, un pulsante per l'invio e uno per l'inserimento facilitato di emoticon, imitando la finestra di chat integrata in Gmail.

La funzionalità di ricerca, attualmente non funzionante, può venire realizzata portando in primo piano le conversazioni che soddisfano i criteri inseriti; questa funzionalità è già disponibile alla tab.

3.6 Comunicazione diretta tra plugin e utente

Si è visto come il meccanismo della data warehouse permetta una comunicazione indiretta tra plugin e interfaccia grafica, impedendo a quest'ultima l'azione diretta sui dati.

[18] http://www.w3.org/TR/xhtml1/

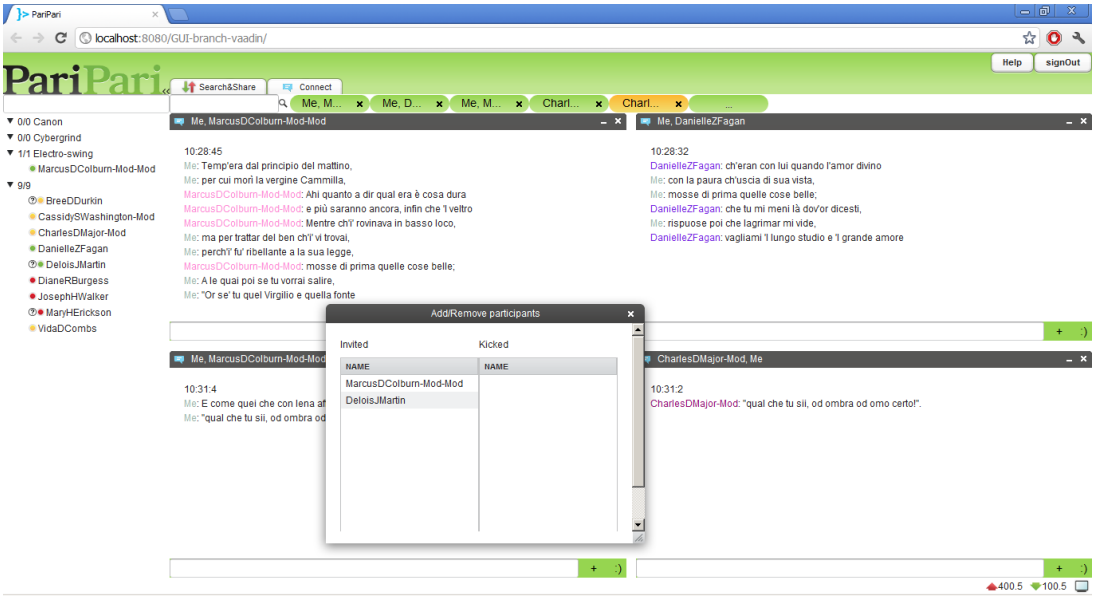

Figura 3.3: Attuale aspetto della tab *Connect*; nella terza conversazione è visibile la finestra per la gestione dei partecipanti

Nonostante questo approccio sia corretto per quanto riguarda oggetti con una lunga aspettativa di vita e utilizzo, nel caso della comunicazione di messaggi o di richiesta di configurazioni da parte dei plugin l'utilizzo della data warehouse risulta inutilmente macchinoso, poichè gli oggetti necessari alla comunicazione richiedono, a meno di errori, una sola visualizzazione.
Si è voluto offrire ai plugin un modello di dati indipendente dalla data warehouse e dedicato alla visualizzazione di elementi non permanenti, che non avrebbero posto nelle schermate delle tab.
Coerentemente con la separazione tra Model e View, la creazione degli elementi grafici avviene solo da parte dell'interfaccia, mentre gli oggetti creati dai plugin ne descrivono in maniera astratta il contenuto.
L'aspetto più importante è la flessibilità del sistema: nel caso della richiesta di configurazioni, i plugin devono poter descrivere schermate complesse, dotate di relazioni non semplici tra i componenti e in grado di operare una valutazione preventiva della validità dell'input dell'utente; un errore nella compilazione di un form verrebbe individuato solo quando questo fosse ricevuto dal plugin, costringendo alla ritrasmissione.

La sola definizione di un UIDL (User Interface Definition Language) avrebbe im-

pedito ai plugin di comporre dinamicamente le schermate; l'approccio adottato si basa sulla serializzazione di alberi di oggetti in una stringa XML (eXtensible Markup Language).
I plugin hanno a disposizione delle classi per definire campi di valori e controllori di validità, aggregabili secondo regole precise; una volta composto, l'albero viene esplorato ed è prodotta una stringa che lo descrive. Una volta arrivata alla GUI, dalla stringa vengono interpretati i campi e i controllori originari e viene assemblata una schermata coerente con il resto dell'interfaccia.
Dopo che l'utente ha interagito in maniera soddisfacente con la schermata, l'interfaccia produce una copia dell'albero contenente i valori utili e questa fa il percorso inverso; l'estrazione dei valori dall'albero che viene restituito completa il ciclo di comunicazione.

3.6.1 Field, Validator e Control

Si è optato per un sistema asimmetrico: i plugin hanno a disposizione oggetti Field che definiscono i campi, mentre la GUI gestisce degli oggetti Control discendenti da Field, che integrano l'elemento di visualizzazione. Ai Field, e di conseguenza ai Control, vengono assegnati degli opportuni Validator, che la GUI utilizza per verificare il corretto inserimento di dati da parte dell'utente.

Il sistema non può semplicemente basarsi su interfacce, perchè la creazione di Field dal comportamento corretto non verrebbe garantita: si è scelto di adottare il design pattern *Builder*[19] per semplificare ai plugin l'utilizzo delle classi base dei Field.
La classe PGFieldFactory permette, attraverso la chiamata a metodi statici dal nome significativo, la generazione di oggetti Field nascondendo la complessità del costruttore ed eseguendo automaticamente dei controlli sulla validità dell'oggetto.
La serializzazione e l'estrazione dei dati di un albero di Field sono operazioni automatizzate dalla classe PGXMLSharedUtil, disponibile anche ai plugin, mentre l'interpretazione di una stringa XML come albero di Control è possibile solo all'interno della GUI, in quanto i Control sono basati su Vaadin.

L'invio di un albero di Field richiede la composizione di un messaggio, in cui vengono specificati un identificativo del mittente, il titolo da visualizzare nella schermata e il

[19]Cfr: [5]

tipo di comunicazione effettuata.
I tipi disponibili nell'enum MessageType sono:

- NOTIFY: un messaggio che non richiede una risposta da parte dell'utente
- QUESTION: un messaggio che invece necessita di risposta
- ANSWER: la risposta ad una QUESTION, inviata dalla GUI quando l'utente completa in maniera corretta la schermata
- UNANSWER: identifica un messaggio vuoto, segno che la comunicazione è fallita e il plugin deve rinviare il messaggio QUESTION

La visualizzazione del messaggio è automatizzata dalla classe PGMessageWindow, che a seconda del tipo di comunicazione mette a disposizione dell'utente un pulsante per la presa visione (nel caso di NOTIFY) o una coppia di controlli (nel caso di QUESTION) per posporre o eseguire l'invio della risposta.
L'operazione di invio non è possibile finchè l'input non viene verificato come corretto dai Validator sui campi del messaggio.

Ogni Field contiene una stringa identificativa e una di descrizione; inoltre è possibile definire il tipo e la modalità di inserimento (inserimento singolo o multiplo, scelta di uno o più tra un insieme di possibilità) del valore da memorizzare con degli enum.
I Validator assegnabili dipendono dal tipo di dato e dalla modalità di inserimento; nel caso di selezione da un insieme di valori non ha senso effettuare controlli.
I Field disponibili sono:

- PGGroupField: è l'unico Field che può essere radice dell'albero; il suo scopo è fornire un contesto all'insieme dei Field figli
- PGBooleanField: come il GroupField, ma con la possibilità di immagazzinare un valore booleano
- PGStringField: memorizza uno o più valori String e non può avere figli
- PGNumericField: memorizza uno o più valori numerici, non ammette figli
- PGFileField: memorizza l'identificativo di un file sulla macchina dell'utente come la stringa restituita da `java.io.File.getAbsolutePath()`, non ammette figli

- PGLocationField: permette la memorizzazione di una o più coppie formate dai compoenti di una PGContactLocation e una password in forma di stringa; anch'esso non può avere figli

Le controparti Control hanno una funzionalità in più: possono essere disabilitate (impedendo l'input dell'utente) quando un antenato BooleanControl memorizza lo stato `false`. È questa la motivazione di BooleanField, che permette di realizzare configurazioni incomplete a seconda della necessità o meno di parte dei dati.
L'elenco dei Control è identico a quello dei Field, ma la visualizzazione è diversa a seconda del tipo di inserimento del dato:

- PGGroupControl: visualizza la propria descrizione e i Control figli indentati sotto di essa; è un semplice controllo della formattazione
- PGBooleanControl: come il GroupControl, ma una checkbox a sinistra della descrizione permette la disabilitazione del sottoalbero di cui è radice
- PGStringControl: a seconda della modalità di inserimento, sotto la descrizione vengono visualizzati
  - un TextField per l'inserimento singolo
  - un TextField e una tabella contenente i valori salvati, in caso di inserimento multiplo
  - una serie di RadioButton o una lista (nel caso le scelte possibili siano più di sei) per la selezione singola
  - analogamente alla selezione singola, una serie di checkbox o una lista per la selezione multipla
- PGNumericControl: esattamente come lo StringControl
- PGFileControl: visualizza un TextField e un pulsante per la visualizzazione di una PGFileChooserWindow, che rappresenta il file system della macchina. È possibile una futura espansione per memorizzare più file, mentre la selezione è già realizzabile con l'utilizzo di uno StringField
- PGLocationControl: nel caso di singolo inserimento, vengono visualizzati un TextField per l'inserimento del nickname, uno per l'identificativo, un ComboBox per il protocollo e un PasswordField per nascondere la scrittura della password;

l'inserimento multiplo aggiunge sotto questi componenti una tabella che riassume i dati già inseriti

In ogni caso, il valore viene restituito come una stringa ed è compito dell'utilizzatore finale effettuare le dovute conversioni; nel caso di LocationField, il valore è rappresentato dalla serializzazione di un JSONArray contenente i valori come stringhe.
Particolare è il caso dei Field ad inserimento multiplo: la stringa restituita è una serializzazione dei risultati come JSONArray.

I Validator disponibili sono:

- PGStringRegexValidator: applicabile a StringField e ai campi nickname, identifier e password di LocationField, verifica che le stringhe proposte vengano accettate da un'espressione regolare
- PGIntegerValidator: applicabile ai soli NumericField, verifica che l'interpretazione come Double del valore (inserito in forma di stringa) sia all'interno di un intorno, di raggio specificabile come tolleranza, centrato nel più vicino intero
- PGNumericRangeValidator: anche questo applicabile ai soli NumericField, verifica che l'interpretazione come Double del valore sia in relazione (qualsiasi forma di minore, maggiore, uguaglianza o differenza) con un valore assegnato
- PGNumericGroupValidator: verifica NumericField soddisfi una combinazione (secondo le operazioni logiche) di due Validator adatti ai NumericField; questo permette l'uso di altri NumericGroupValidator per incrementare il numero di vincoli
- PGFileExtensionValidator: controlla che l'estensione di un file appartenga ad un insieme di stringhe specificato, o che sia una cartella oppure entrambe le condizioni

Il sistema illustrato presenta certamente delle limitazioni, ad esempio non è possibile mettere in relazioni totalmente arbitrarie i Field, ma tutte le normali opzioni di configurazione sono ben supportate. La composizione di notifiche per l'utente richiede solamente l'utilizzo di un GroupField la cui descrizione è il testo da comunicare.

Le differenze intrinseche dei plugin che componevano la realtà da rappresentare hanno spinto dunque a definire una struttura ad alto livello di astrazione, basata sui

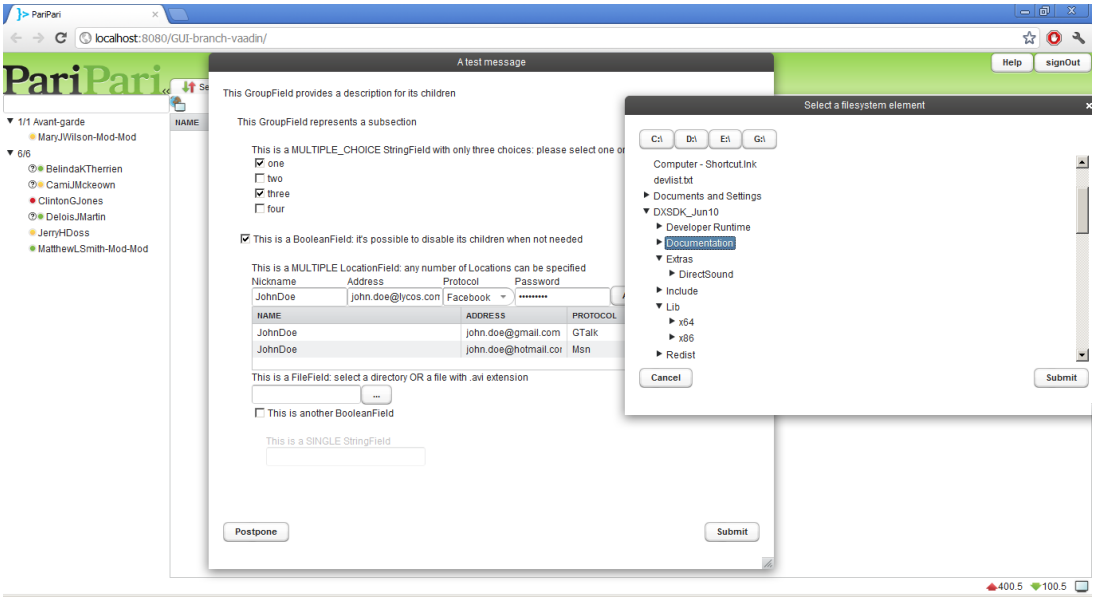

Figura 3.4: Un esempio di PGMessageWindow; è visibile la finestra per la selezione del file di PGFileControl

pattern di funzionamento e sul modello di dati presentati in 3.2 ed in 3.3.
Nonostante le difficoltà implementative legate al framework di sviluppo scelto, si è giunti quindi ad una realizzazione leggermente adattata e di pari espressività rispetto al progetto iniziale.

Conclusioni

La realizzazione dell'interfaccia grafica di PariPari ha richiesto l'analisi di aspetti del programma molto diversi tra loro.
La struttura modulare dell'applicazione permette di espanderne le funzionalità oltre i limiti degli attuali concorrenti, e ne impedisce una rappresentazione statica: la GUI deve essere un componente in grado di evolvere assieme alla piattaforma stessa, mantenendo una struttura chiara e coerente.
È stato necessario analizzare i prodotti concorrenti e le direzioni di espansione offerte dalla multifunzionalità del programma per arrivare alla sintesi di un linguaggio visuale che fosse naturale e flessibile, tale da integrare l'utilizzo di elementi nuovi senza l'alienazione di quelli esistenti.
Si è dovuto ricercare una struttura del codice che riflettesse le separazioni tra i diversi componenti, ma garantisse la possibilità di combinare gli stessi in maniera omogenea, applicando tecniche di progetto volte al riutilizzo e all'espandibilità del prodotto a partire dagli strumenti software a disposizione.
Con la crescita del progetto PariPari e il cambiamento delle funzionalità da offrire si renderà necessaria la modifica di quanto fatto finora, ma si ritiene di aver realizzato una solida fondazione per gli sviluppi futuri.

Bibliografia

[1] Paolo Bertasi, *Progettazione e realizzazione in Java di una rete peer to peer anonima e multifunzionale*, Dipartimento di Ingegneria dell'Informazione, Università di Padova, 2004

[2] Alan Cooper, Robert Reimann, David Cronin, *About Face 3: the essentials of interaction design*, Wiley, 2007

[3] Mattia Samory, *PariGUI 2010*, Dipartimento di Ingegneria dell'Informazione, Università di Padova, 2010

[4] Marko Grönroos, *Book of Vaadin*, Vaadin Ltd, 2012

[5] E. Gamma, R. Helm, R. Johnson, J. Vlissides, *Design Patterns - Elements of Reusable Object-Oriented Software*, Addison Wesley, 1995

Elenco delle figure