REALIZZAZIONE DI UN'APPLICAZIONE DI MESSAGGISTICA ISTANTANEA CON ARCHITETTURA DISTRIBUITA

Xhensila Doda (Project Leader), Giulia Grisendi

Sommario

Questo progetto ha avuto come obiettivo la creazione di un applicazione desktop di tipo social che prevede la comunicazione tra utenti in modo diretto tramite chat oppure indiretto tramite l'inserimento di commenti nella propria pagina principale. La tecnologia principale utilizzata è RMI, perciò l'applicazione prevede un server centrale dove sono implementati gli oggetti remoti e un lato client che invoca questi oggetti usufruendo delle loro funzionalità. Oltre a RMI, sono stati utilizzati anche JavaFX per la creazione dell'interfaccia utente, le socket per la realizzazione della chat e MySQL e SQLite come software per il database.

Index Terms

RMI, MySQL, SQLite, Socket, JavaFX, Eclipse

Indice

1 INTRODUZIONE

1.1 Obiettivo

Il progetto ha previsto la realizzazione di un'applicazione desktop in ambiente Java usufruendo di varie tecnologie sia prer la creazione del lato client che per il lato server.

Quest'ultimo ospita gli oggetti remoti che una volta implementati, con opportune modalità, possono essere chiamati dal client ed essere utilizzati come semplici oggetti locali.

Lo scopo principale dell'applicazione è quello di permettere a due o più persone di poter comunicare in vari modi. Dopo l'avvenuta registrazione, l'utente può accedere alla propria pagina principale e comunicare con i propri amici. Infatti, l'applicazione prevede sia l'interazione diretta tramite chat che quella indiretta tramite pubblicazione di post nella propria home di riferimento.

Il requisito fondamentale di quest'applicazione è la possibilità di far comunicare gli amici nonostante il server centrale non sia raggiungibile.

1.2 Definizioni, Acronimi ed Abbreviazioni

Di seguito vengono riportate le definizioni di alcuni termini o abbreviazioni utilizzate in questo documento e che possono risultare poco chiare.

Java RMI (Java Remote Method Invocation) – tecnologia che permette l'implementazione del modello a oggetti distribuiti e consente ad un oggetto su una macchina virtuale Java di invocare metodi di oggetti residenti in altre macchine virtuali Java[1].

Socket – è concettualmente un canale che permette la comunicazione tra processi o thread, residenti o meno sulla stessa macchina. È identificata dal numero di porta (port number) e dall'indirizzo IP della macchina (nodo o host)[2].

JavaFX – comprende un insieme di pacchetti grafici e multimediali che consentono agli sviluppatori di realizzare applicazioni desktop e web che operano in diverse piattaforme[3].

MySQL – è un sistema di gestione di database relazionale (DBMS) open-source largamente utilizzato nelle applicazioni web[4].

SQLite – una libreria software scritta in linguaggio C che implementa un DBMS SQL incorporabile all'interno di applicazioni[5].

Eclipse – è un ambiente di sviluppo integrato multi-linguaggio e multi-piattaforma utilizzato per lo sviluppo di software di vario tipo[6].

1.3 Descrizione generale

L'applicazione è stata realizzata secondo il modello Client-Server. La comunicazione via chat è stata creata tramite socket, quindi avviene tramite collegamento diretto agli amici senza doversi appoggiare ad un server centrale. Invece, per quanto riguarda la pubblicazione di commenti nella propria bacheca personale, essa avviene tramite l'invocazione di oggetti remoti e di conseguenza prevede l'utilizzo della tecnologia Java RMI. Inoltre, sono previste altre funzionalità come ad esempio la ricerca di amici e quindi la possibilità di poterne aggiungere degli altri oppure modifica dei propri dati personali.

Per quanto riguarda l'interfaccia utente è stata usata la tecnologia JavaFX che permette di arricchire l'applicazione con elementi che da una parte ottimizzano l'aspetto grafico e dall'altra parte la rendono user-friendly. Il progetto ha visto il coinvolgimento di due database: centrale e locale. Il database del server centrale che ospita i dati degli oggetti remoti è stato realizzato

tramite il software MySQL. Per quello locale, e cioè il database creato per ogni utente e che è installato automaticamente con l'applicazione, è stato utilizzato SQLite.

Per lo sviluppo dell'applicazione è stato utilizzato il software Eclipse su un sistema operativo Windows. Come è ovvio dalla descrizione, il vincolo per l'utilizzo di questo software è la presenza di una piattaforma che supporti Java e che ci sia collegamento internet.

Nei prossimi capitoli saranno descritti i dettagli implementativi del progetto sia a livello di requisiti che a livello tecnico e architetturale. In particolare nel capitolo 2 verranno riportati i requisiti fondamentali dell'applicazione, nel capitolo 3 verrà mostrata l'interfaccia grafica dell'applicazione e infine, nel capitolo 4, verrà spiegato nel dettaglio l'architettura e l'implementazione delle funzioni principali dell'applicazione.

2 SPECIFICHE DEL SOFTWARE

2.1 Requisiti Funzionali

In questa sezione verranno riportati tutti i requisiti funzionali del progetto realizzato e per ogni requisito viene indicata una breve descrizione, i dati di input/output ed eventuali errori.

Nome	Registrazione
Descrizione	Registrazione di un utente.
Input	Username, Password, Email , Immagine profilo
Processo	Inserimento di un nuovo utente e quindi dei rispettivi dati di profilo nel database del server centrale.
Output	Registrazione effettuata con successo.
Errori	Inserimento di dati non corretti (es. username già in uso, immagine di profilo troppo grande, le password non coincidono nella fase di ripetizione).

Nome	Login
Descrizione	Autenticazione di un utente.
Input	Username, Password
Processo	L'utente accede alla propria pagina principale tramite l'inserimento delle proprie credenziali. Se le credenziali sono corrette allora viene visualizzato il rispettivo profilo. Inoltre, se l'operazione ha avuto successo viene creata una cartella sul computer dell'utente con un database che effettua un salvataggio in locale dei propri dati e di quelli dell'amico e che è sincronizzata ogni 10 minuti con il server centrale.
Output	Accesso alla propria home principale.
Errori	Inserimento di dati non corretti (es. username o password errate).

Nome	Chatta
Descrizione	Possibilità di chattare con un selezionato amico.
Input	Selezione del amico con cui si vuole comunicare.
Processo	Nella lista degli amici che si trova a sinistra della propria pagina principale è selezionato l'amico con cui si desidera comunicare e poi si preme il tasto destro e si seleziona in automatico l'icona Chatta. Se l'amico è online allora è aperta una finestra per avviare la comunicazione. In modo automatico è creato un file database nella propria cartella locale che permette il salvataggio della cronologia della chat e quindi il ripristino di essa di volta in volta. Se l'amico non è online è visualizzato un messaggio che informa l'utente che l'amico non è collegato.
Output	Apertura delle finestra di comunicazione.
Errori	Messaggio che informa che l'amico non è momentaneamente collegato.

Nome	File
Descrizione	Invio di un file ad un amico
Input	Cliccare File.
Processo	L'operazione permette di inviare un file ad un amico. Se la connessione dovesse venire a mancare allora viene visualizzato un messaggio di errore.
Output	Visualizzazione del messaggio di invio del file se l'operazione è andata a buon fine oppure di un messaggio di errore.
Errori	Messaggio che informa l'utente per i problemi di connessione.

Nome	Invia
Descrizione	Invio del messaggio testuale ad un amico.
Input	Cliccare Invia.
Processo	L'operazione permette di inviare un messaggio alfanumerico ad un amico. Se la connessione dovesse venire a mancare allora viene visualizzato un messaggio di errore.
Output	Visualizzazione del messaggio se l'operazione è andata a buon fine oppure di un messaggio di errore.
Errori	Messaggio che informa l'utente per i problemi di connessione.

Nome	Pubblica
Descrizione	Inserimento di un post (messaggio personale).
Input	Testo del messaggio.
Processo	L'utente può scrivere un messaggio con un determinato numero di caratteri e premendo il tasto pubblica esso è visualizzato nella propria bacheca personale così come anche in quelle degli amici. Se il server RMI è online allora il messaggio viene inserito direttamente nel database centrale. In caso contrario, esso è salvato nel database locale dal quale poi sarà prelevato e inserito nel database centrale quando il server RMI è online. Gli amici vedranno le modifiche dopo che ci sarà la corretta sincronizzazione tra i due database.
Output	Pubblicazione del messaggio ed eventualmente un messaggio che informa l'utente che il server RMI non è online nel caso si verifichi questa situazione.
Errori	Non previsti.

Nome	Visualizza Commenti.
Descrizione	Visualizzazione della cronologia dei commenti relativi ad un determinato messaggio.
Input	Selezione del messaggio nella propria bacheca.
Processo	Selezionando l'icona Visualizza Commenti tramite il tasto destro, dopo aver cliccato un messaggio nella propria bacheca, è possibile visualizzare l'intera cronologia dei commenti inseriti con anche la possibilità di poterne inserire degli nuovi. L'inserimento di un nuovo commento si effettua in modo identico a quello della pubblicazione di un messaggio nella propria home. É possibile visualizzare i commenti solo se il server RMI è online.
Output	Visualizzazione e/o pubblicazione di un nuovo messaggio.
Errori	Messaggio che informa che il server RMI non è online.

Nome	Rimuovi Post
Descrizione	Eliminazione di un messaggio.
Input	Selezione del messaggio da eliminare.
Processo	Selezionando un messaggio della bacheca si ha la possibilità di rimuovere il post solamente se questo appartiene all'utente autenticato. Quest'operazione ha buon esito solo se il server RMI è online. In questo caso il messaggio viene eliminato dal database. Altrimenti viene visualizzato un messaggio che informa l'utente di riprovare più tardi.
Output	Eliminazione del messaggio.
Errori	Eliminazione del messaggio. Messaggio che informa che il server RMI non è online.

Nome	Ricerca
Descrizione	Ricerca di utenti nel database centrale per poterli aggiungere come amici.
Input	Stringa alfanumerica.
Processo	Effettua una ricerca nel database centrale e visualizza la lista di persone che hanno almeno un carattere tra quelli presenti. Ovviamente la ricerca è ristrettiva e quindi più caratteri si inseriscono più essa si specifica. L'operazione funziona solamente se il server RMI è online.
Output	Elenco degli utenti che corrispondo ai parametri di ricerca.
Errori	Messaggio che informa che il server RMI non è online.

Nome	Accetta richieste di amicizia
Descrizione	Numero di richieste di amicizia.
Input	Cliccare Accetta richieste di amicizia.
Processo	L'utente cliccando su quest'icona può visualizzare la lista di persone che gli hanno richiesto l'amicizia e ha la possibilità di poterle accettare o lasciarle sospese.
Output	Lista degli amici.
Errori	Messaggio che informa che il server RMI non è online.

Nome	Aggiungi come amico
Descrizione	Aggiungere un utente nella propria lista di amici.
Input	Cliccare Aggiungi come amico.
Processo	Dopo aver effettuato un opportuna ricerca e aver ottenuta la lista di amici che corrispondono ai parametri desiderati, l'utente può aggiungere un amico richiedendoli l'amicizia tramite Aggiungi come amico. Ovviamente la richiesta di amicizia andrà a buon fine solamente se il server RMI è online.
Output	Richiesta di amicizia effettuata.
Errori	Messaggio che informa che il server RMI non è online.

Nome	Accetta Richiesta di Amicizia
Descrizione	Accettare la richiesta di amicizia da parte di un amico.
Input	Cliccare Accetta Richiesta di Amicizia.
Processo	Dopo aver cliccato l'icona che visualizza la lista delle richieste di amicizia in sospeso, ognuno di questi può essere aggiunto alla propria lista di amici.
Output	Aggiunta dell'amico nella propria lista amici.
Errori	Non previsti.

Nome	Rimuovi Amico
Descrizione	Eliminazione di un amico dalla propria lista amici.
Input	Cliccare Rimuovi Amico.
Processo	La funzione permette di eliminare un amico cliccando il tasto destro dopo aver selezionato uno degli amici. La funzione ha buon esito solamente se il server RMI è online.
Output	Eliminazione dell'amico dalla lista amici.
Errori	Messaggio che informa che il server RMI non è online.

Nome	Mio Profilo
Descrizione	Modifica dei dati del proprio profilo.
Input	Cliccare File>Mio Profilo.
Processo	L'operazione permette la modifica dei propri dati personali come lo username, la password, l'email e anche l'immagine di profilo. Le modifiche possono essere effettuate soltanto se il server RMI è online.
Output	Modifiche effettuate.
Errori	Messaggio che informa che il server RMI non è online.

Nome	Apri Bacheca
Descrizione	Apertura della bacheca.
Input	Cliccare File> Apri Bacheca.
Processo	L'operazione permette la visualizzazione della propria pagina principale.
Output	Visualizzazione dei dati della home.
Errori	Non previsti.

Nome	About
Descrizione	Informazioni sui programmatori.
Input	Cliccare Help> About.
Processo	Visualizzazione di informazioni sugli sviluppatori del progetto.
Output	Informazioni sui programmatori.
Errori	Non previsti.

Nome	Esci
Descrizione	Chiusura dell'applicazione.
Input	Cliccare File> Esci oppure X.
Processo	L'operazione permette di chiudere l'applicazione con opportuni salvataggi dei dati.
Output	Chiusura dell'applicazione.
Errori	Non previsti.

2.2 Rappresentazione UML

In questa sezione saranno mostrati il class-diagram e alcuni activity diagram per capire la sequenza logico temporale con cui avvengono gli eventi legati ad alcune funzioni principali che sono implementate nel software.

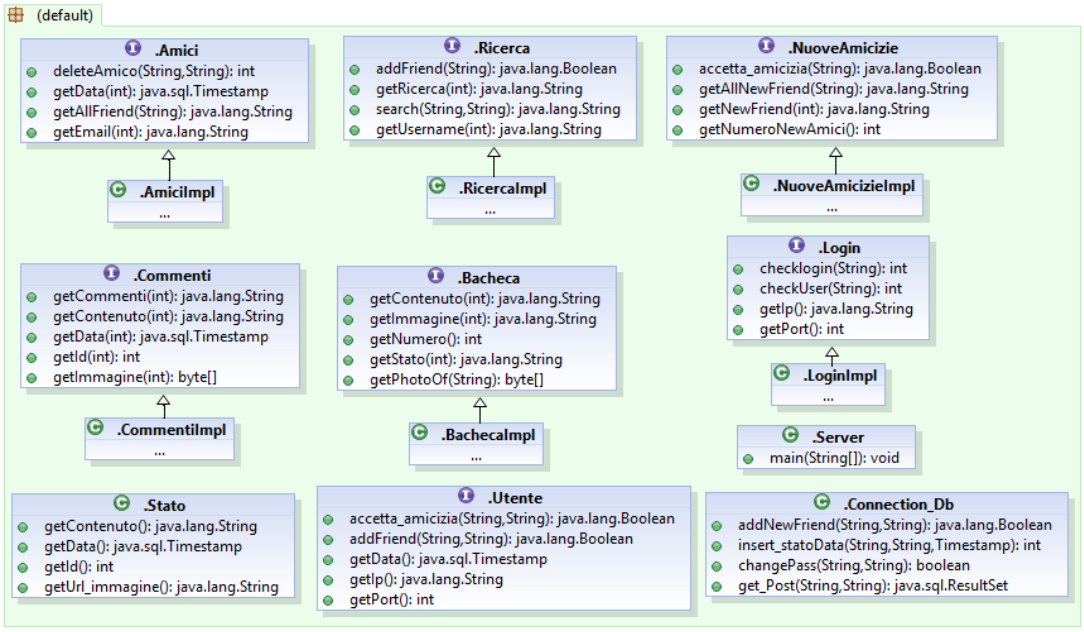

Figura 1. Class Diagram del server RMI dove sono mostrati anche alcuni dei metodi presenti.

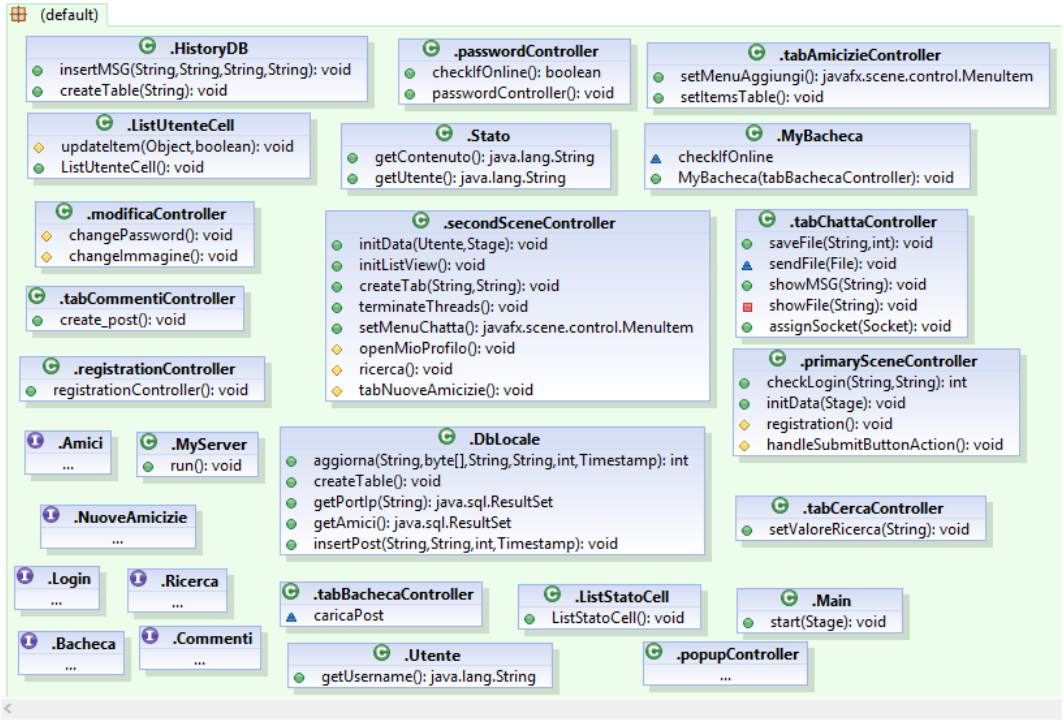

Figura 2. Class Diagram del client RMI dove sono mostrati anche alcuni dei metodi presenti.

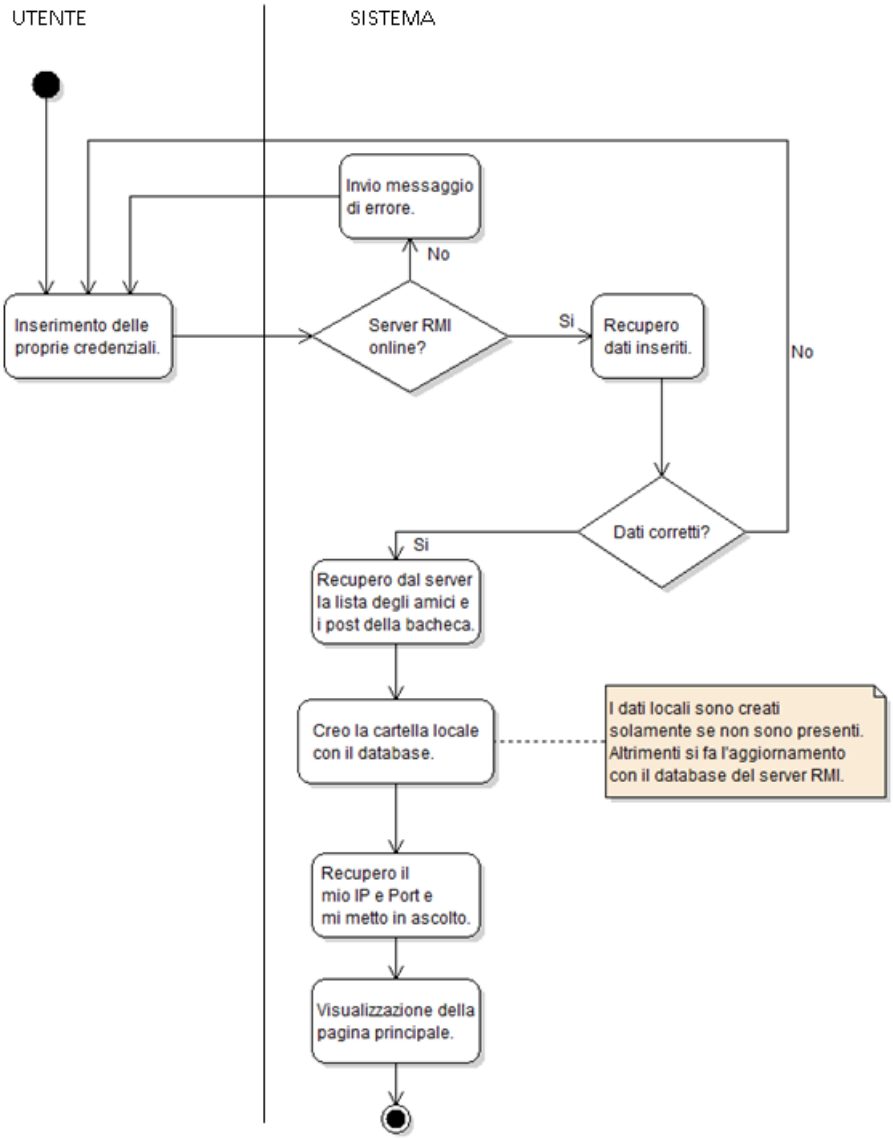

Figura 3. Diagram Activity della funzione Login.

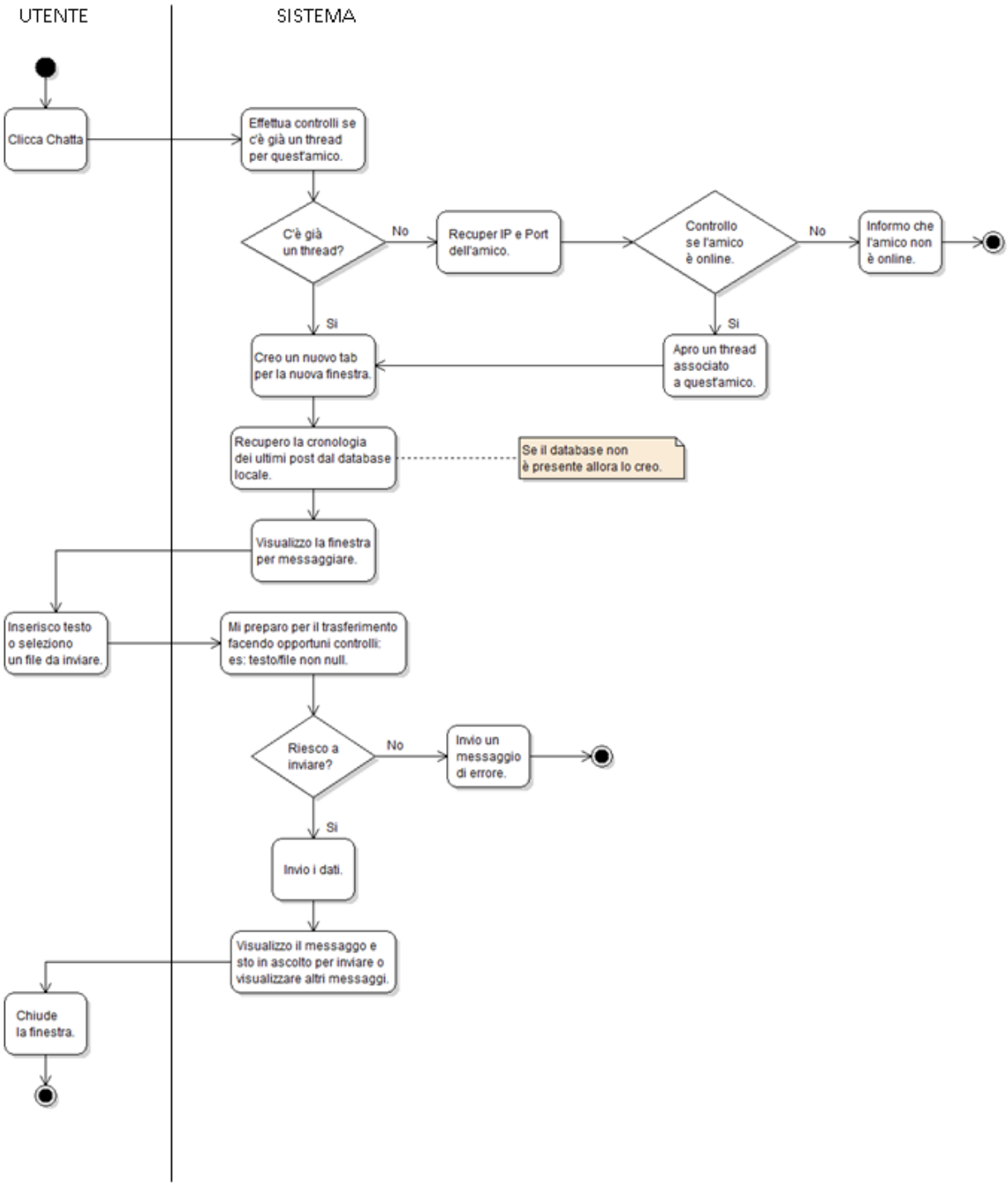

Figura 4. Diagram Activity della funzione Chatta.

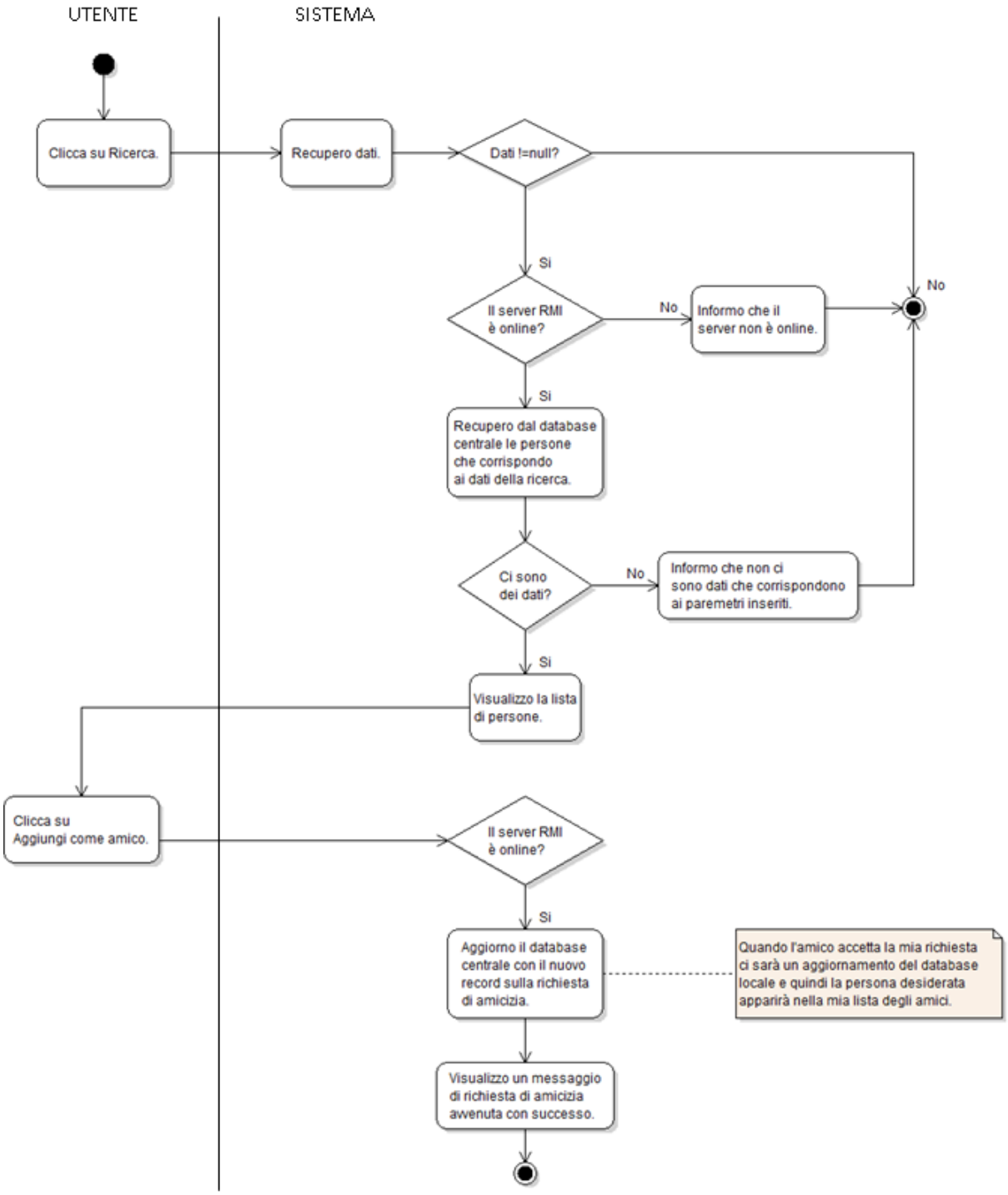

Figura 5. Diagram Activity della funzione Ricerca e Aggiungi come amico.

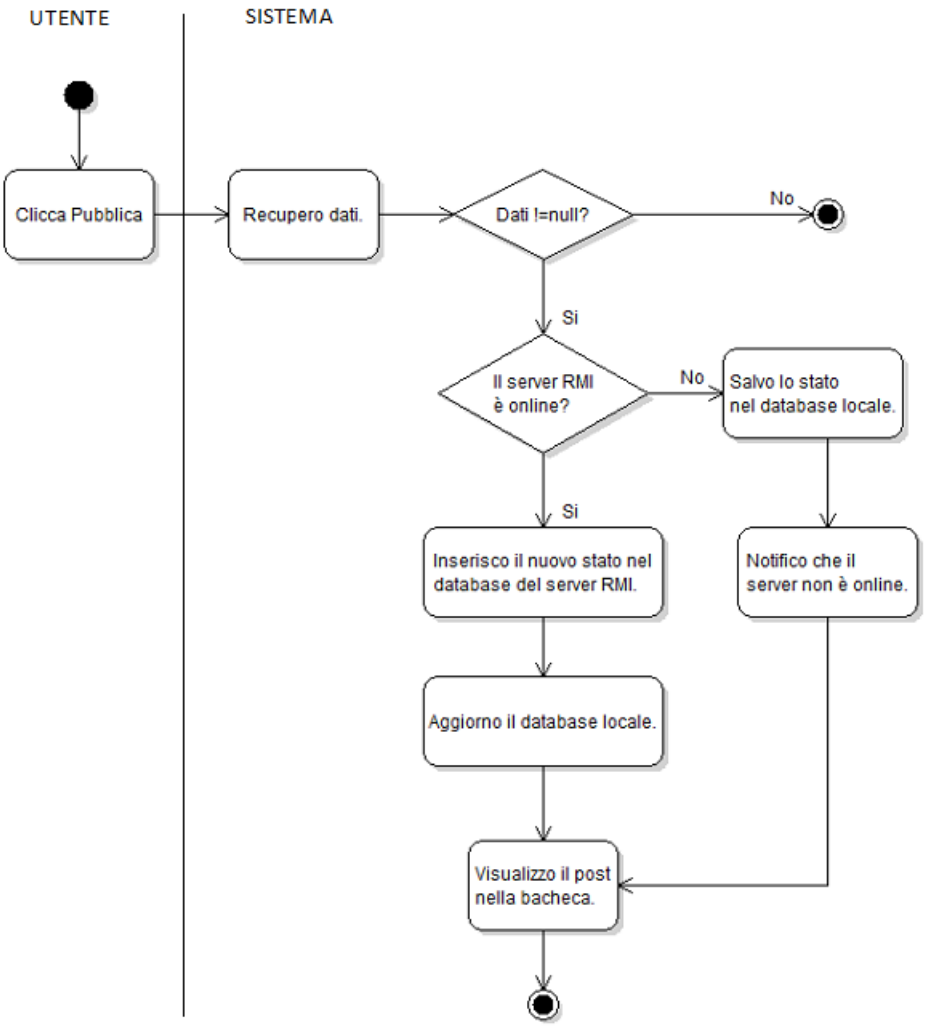

Figura 6. Diagram Activity della funzione Pubblica.

3 INTERFACCIA GRAFICA

L'interfaccia grafica dell'applicazione è stata realizzata tramite JavaFX. JavaFX consente agli sviluppatori di utilizzare le librerie Java all'interno delle applicazioni JavaFX.

Gli sviluppatori hanno la possibilità di espandere la conoscenza di Java e usufruire della tecnologia di presentazione fornita da JavaFX per la creazione di un interfaccia utente flessibile con notevoli funzionalità. JavaFX può essere scaricata gratuitamente dal sito di Oracle e per eseguire le applicazioni JavaFX, è necessario che siano installati sul PC sia Java Runtime Environment (JRE) che il runtime JavaFX.

Una caratteristiche peculiare di JavaFX è la sua natura multipiattaforma consentendo il passaggio da un dispositivo all'altro. Inoltre, JavaFX permette anche di interagire senza alcuno sforzo con classi Java preesistenti. È possibile fare anche l'opposto, e cioè si può includere un'applicazione JavaFX all'interno di un normale programma scritto in Java e Swing. Tramite JavaFX è possibile lavorare secondo una logica MVC (Model View Controller) che permette agli sviluppatori di separare la parte grafica con l'implementazione del codice logico funzionante. Infatti, per la realizzazione dell'interfaccia utente JavaFX introduce FXML e Scene Builder. FXML è un linguaggio di markup dichiarativo basato su XML e Scene Builder è un software avanzato che permette di creare l'interfaccia grafica in modo interattivo con semplici operazioni drag and drop. In aggiunta, JavaFX fornisce tutti i maggiori controlli dell'interfaccia utente necessari per sviluppare un'applicazione con tecnologie Web standard come CSS[3].

JavaFX offre anche supporto per operazioni di multitouching e per l'accelerazione hardware sfruttando al massimo la piattaforma sottostante.

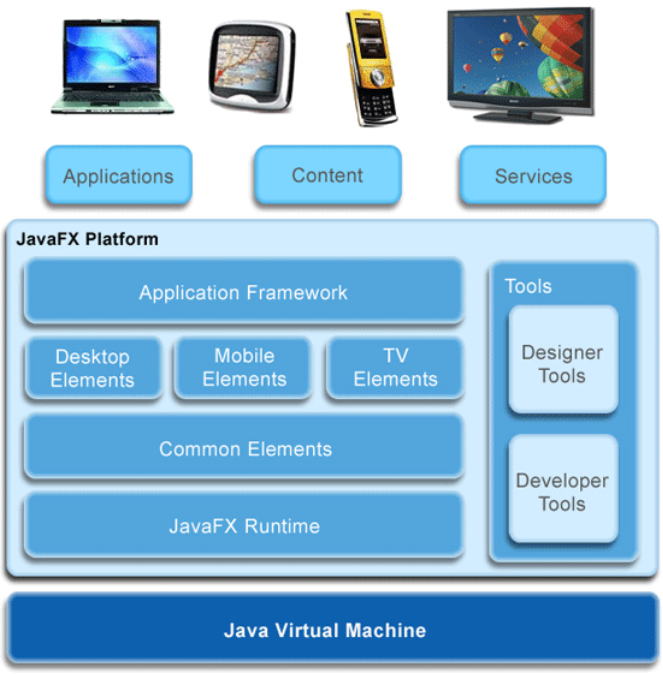

Figura 7. L'immagine sopra mostra l'architettura della piattaforma JavaFX.

Ritornando al progetto realizzato, all'apertura dell'applicazione verrà mostrata la pagina di Login (vedi Figura 8 sinistra) nella quale l'utente dovrà inserire le proprie credenziali, cioè username e password, per accedere alla propria homepage. Nel caso in cui le credenziali siano errate o nel caso in cui il server centrale non sia raggiungibile verrà visualizzato un messaggio di errore. L'utente da questa pagina potrà accedere anche alla pagina di Registrazione (vedi Figura 8 destra), nella quale potrà inserire, oltre all'username, password e e-mail, anche una propria immagine profilo.

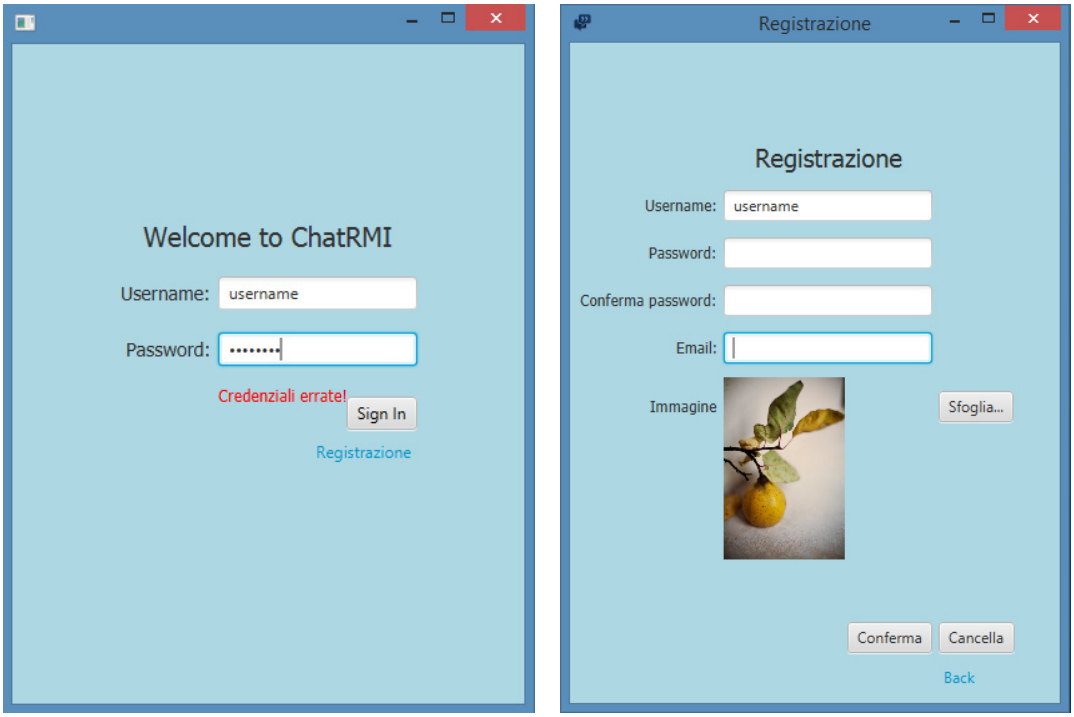

Figura 8. A sinistra è mostrata la pagina di Login con un messaggio di errore. In quella a destra è visualizzata la pagina di registrazione.

Una volta che l'utente ha fatto il Login, potrà accedere alla propria pagina principale. In figura 9 ne vediamo un esempio: in alto si trovano la barra di ricerca, per la ricerca di nuovi amici, e il pulsante che notifica il numero di nuove amicizie da accettare; a sinistra è mostrata la lista degli amici dell'utente e a destra la bacheca. Nella bacheca verranno mostrati i post dell'utente e dei propri amici in ordine temporale. Ogni post della bacheca può essere visualizzato con i corrispondenti commenti o, nel caso in cui sia dell'utente, può essere rimosso.

In figura 10, invece, è mostrato il menù per ogni elemento della lista degli amici. La funzione principale di questa applicazione è la chat. Quando l'utente preme 'Chatta' e l'amico selezionato è online viene mostrata una finestra per lo scambio di messaggi. All'apertura della finestra della chat viene recuperata la cronologia degli ultimi messaggi scambiati tra i due amici e viene visualizzata. Oltre ai messaggi testuali e possibile inviare anche file, come mostrato nella figura 10 a destra.

Nell'eventualità che l'amico con cui si vuole chattare non sia online, verrà mostrato un messaggio di errore (vedi Figura 11).

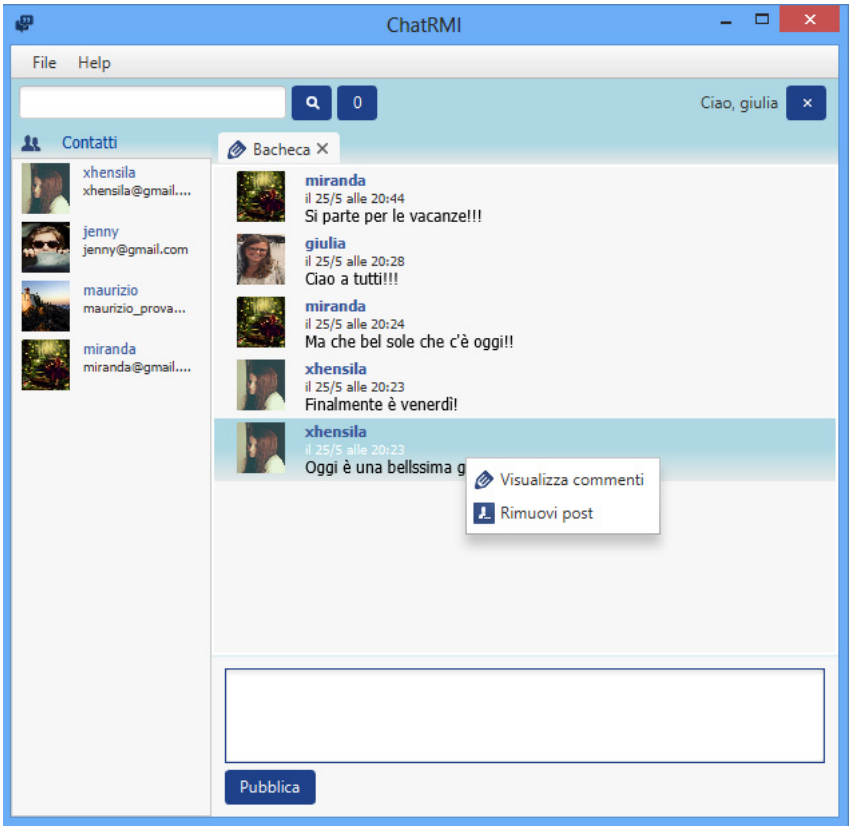

Figura 9. Pagina principale dell'utente.

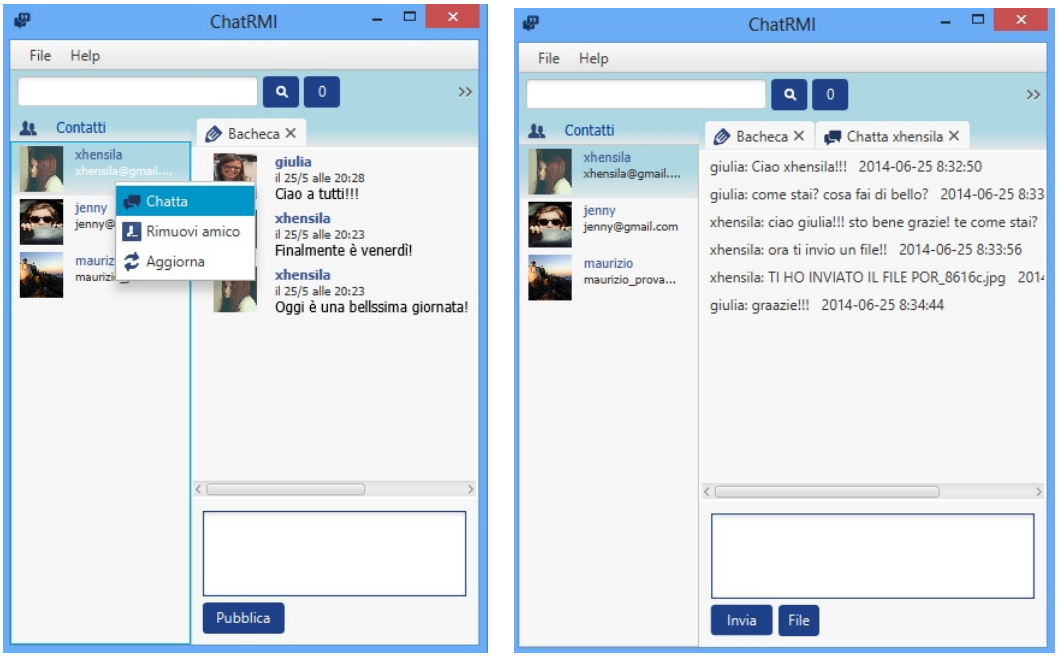

Figura 10. Nell'immagine a sinistra è mostrato il menu per ogni elemento della lista degli amici. In quella a destra la finestra per la chat.

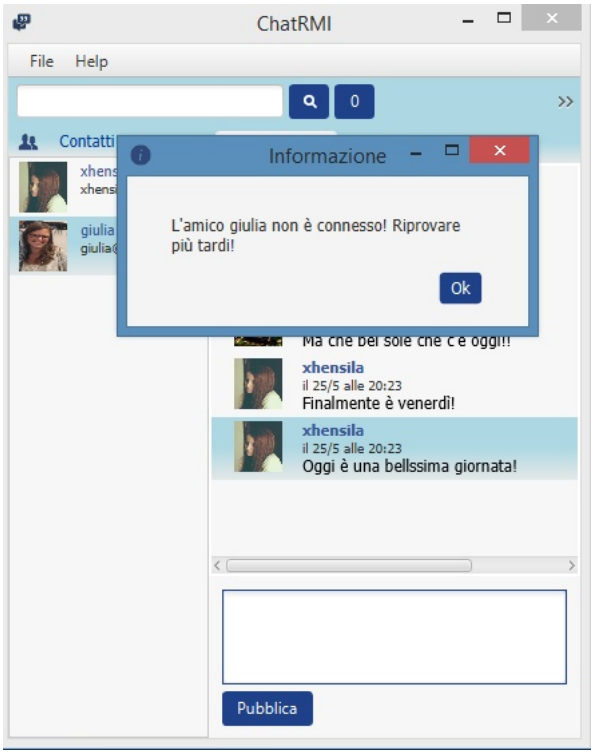

Figura 11. Messaggio di errore quando l'amico non è online.

Tramite la funzione di ricerca è possibile cercare nuovi amici in base alla stringa inserita. Si aprirà una nuova pagina nella quale verranno elencati tutti gli utenti corrispondenti. Per ognuno di questi, come si può vedere in figura 12, si potrà richiedere l'amicizia. Questa nuova amicizia non sarà definitiva finchè l'altro utente non accetterà la richiesta. Infatti, nella figura a destra, si può notare che il pulsante 'nuove amicizie' ha come valore 1: questo significa che l'utente ha un'amicizia da confermare. Quindi, premendo su questo pulsante verrà visualizzata la relativa pagina Nuove Amicizie, nella quale si potrà confermare o meno l'amicizia di un utente.

Infine, come detto precedentemente, per ogni post della bacheca è possibile visualizzare i relativi commenti. Verrà quindi visualizzata una nuova pagina con tutti i commenti del post selezionato, sia che essi siano dei propri amici sia che appartengano ad altri utenti. Sarà possibile rimuovere solo i propri commenti.

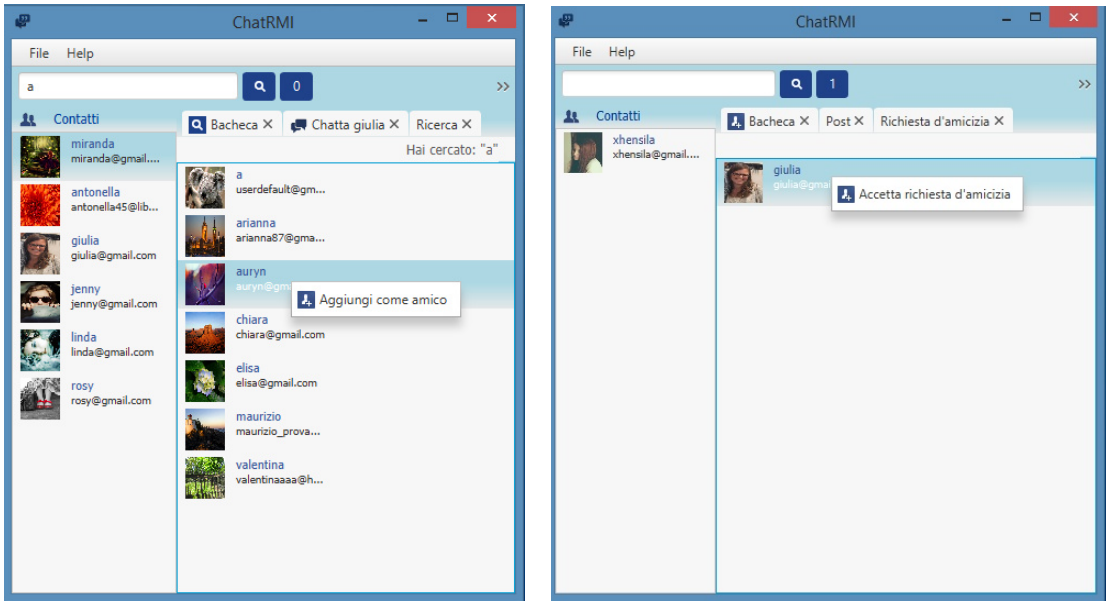

Figura 12. Nell'immagine sinistra è rappresentato la ricerca di nuovi amici, in quella a destra l'accettazione di una richiesta d'amicizia.

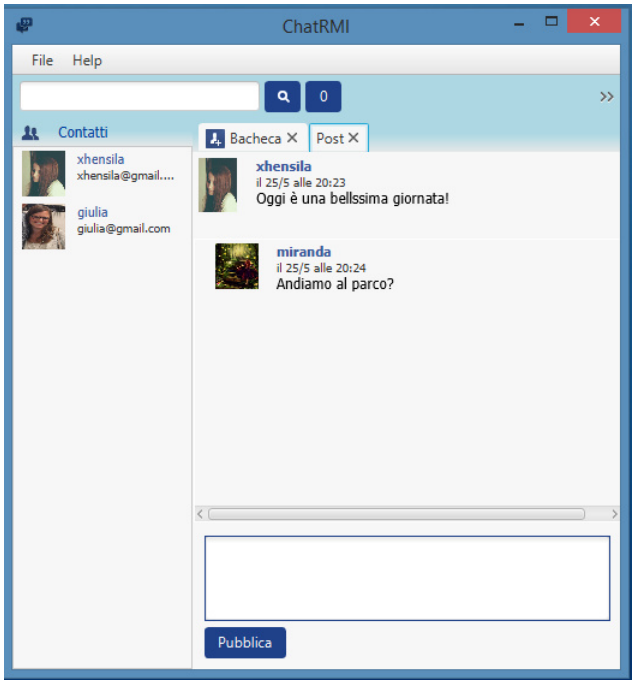

Figura 13. L'immagine mostra la pagina relativa ai commenti di un post.

4 ARCHITETTURA E PROTOCOLLI

L'architettura di questo progetto può essere vista come l'insieme di due parti, e cioè la parte che comprende la tecnologia RMI e quella che riguarda invece la connessione peer-to-peer. Come è stato anticipato tutto il progetto è stato realizzato in Java utilizzando come software di sviluppo Eclipse.

4.1 Java RMI e il protocollo Client-Server

La parte di RMI abbraccia l'approccio classico della struttura Client-Server. Il server ospita gli oggetti remoti che una volta pubblicati possono essere utilizzati dai client, che risiedono in diversi host, invocando i metodi di questi oggetti come se fossero semplici oggetti locali garantendo così trasparenza di programmazione. Siamo quindi di fronte ad un meccanismo simile a quello del RPC (Remote Pocedure Call) orientato però agli oggetti.

L'architettura RMI comprende tre elementi fondamentali: registry, stub e skeleton. Il registry serve per recuperare un riferimento ad un oggetto remoto. Infatti, tramite i metodi bind e rebind un servizio viene registrato in un host e successivamente, tramite il metodo lookup, il nome può essere risolto recuperando così il riferimento all'oggetto. Quindi il registry svolge la funzione di naming. Per pubblicare un oggetto si utilizza l'eseguibile rmiregistry.

Lo stub e lo skeleton agiscono da proxy e rendono trasparente la comunicazione tra gli host. Il client interagisce con lo stub e il server con lo skeleton. Per generare le classi dello stub e dello skeleton si utilizza il compilatore rmic.

Per non andare incontro a problemi di NAT (Network Address Translation), che non riguardavano l'obiettivo del nostro progetto, tutti i componenti sono stati disposti sotto la stessa rete. Sono state eseguite delle prove con indirizzi IP (Internet Protocol) diversi in modo da poter verificare il funzionamento del progetto su host diversi e quindi constatare la sua natura come applicazione distribuita.

Per recuperare le informazioni degli oggetti, il server RMI comunica con un database che nel nostro caso è stato implementato tramite MySQL.

Il client, invece, ha un suo database locale di tipo SQLite. Questo database è sincronizzato a determinati intervalli di tempo con il database del server RMI in modo da garantire la consistenza dei dati.

Quando l'utente accede all'applicazione per la prima volta, viene creata in automatico una cartella con i propri dati e le informazioni riguardanti gli amici, in modo da poter permettere il suo funzionamento anche se il server RMI non è raggiungibile. Ci sono però alcune funzioni per cui è strettamente necessaria la presenza del server RMI, come ad esempio la registrazione, il login, commentare i post della bacheca o aggiungere nuove persone alla propria lista di amici. Dall'altra parte si può inserire un nuovo post nella bacheca anche se il server RMI è offline. Questo è possibile dato che il client ha il suo database locale nel quale viene memorizzato il nuovo post. Il database centrale è aggiornato con i nuovi dati attraverso un thread (lato client) che si occupa di controllare a determinati intervalli di tempo se il server RMI è online e poi sincronizza il database locale per poter ricevere gli aggiornamenti degli amici.

Un'altra funzionalità che può essere svolta anche in assenza del server RMI è la possibilità di comunicare con un amico tramite chat siccome essi si collegano tra loro direttamente con il loro indirizzo IP. Nella prossima sezione sarà mostrato in maniera più dettagliata la comunicazione tra due client tramite socket.

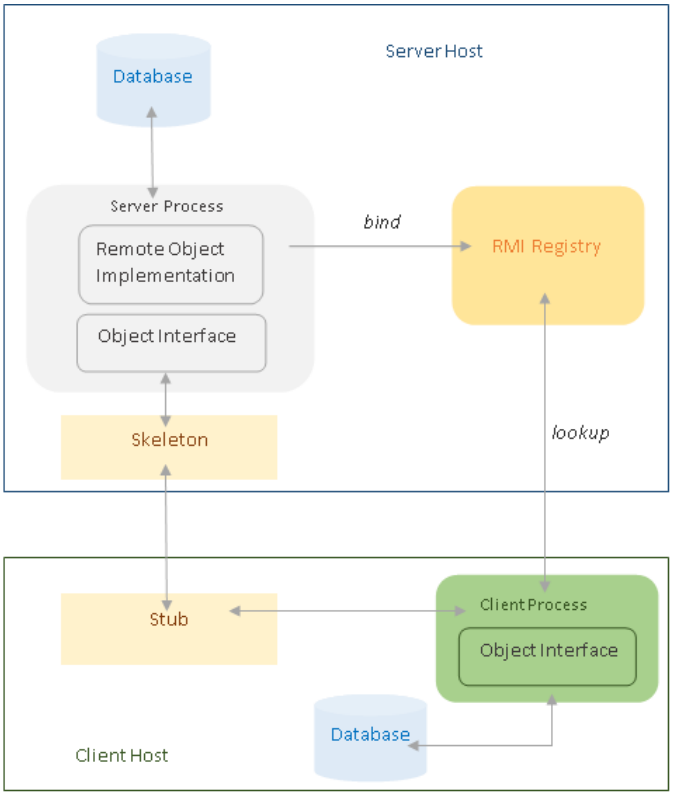

Figura 14. L'immagine mostrata la comunicazione tra i vari componenti secondo l'architettura RMI.

4.2 Socket e il protocollo peer-to-peer

Una socket è un astrazione software per la comunicazione tra processi o thread creata sfruttando l'API (Application Programming Interface) fornita dal sistema operativo per la trasmissione di dati attraverso la rete. Ogni socket è caratterizzata dal numero di porta e dall'IP dell'host in cui il processo è in esecuzione.

Nel progetto le socket sono state utilizzate per la comunicazione via chat, e quindi per la trasmissione sia di testo che di file di diverso tipo. Questa scelta è stata voluta per poter permettere ai client di comunicare tra loro anche in momentanea assenza del server RMI. Ogni volta che il database locale è sincronizzato con quello centrale del server RMI, per ogni amico sono salvati anche l'indirizzo IP e la porta.

Quando un client accede all'applicazione, in automatico un particolare thread, chiamato MyServer, è mandato in esecuzione. Il suo scopo è quello di aprire una socket in una determinata porta e di rimanere in ascolto nel caso in cui arrivassero richieste di connessioni da parte di altri client.

Quando un client si vuole connettere ad un altro client allora all'interno di un blocco try/catch si crea una socket con il suo indirizzo IP e porta. Se l'amico ha accettato la connessione allora mi registro nel suo server tramite la funzione *addNick*. Questo perché ogni client mantiene la lista degli amici che si connettono ad esso. Quando un client accetta una richiesta di connessione, allora crea per l'amico che ha richiesto la connessione un oggetto di tipo ClientConn che implementa la

classe Runnable di Java. In una hashtable sono mantenuti il nome e il corrispondente thread di tipo ClientConn per ogni client che si connette.

Dopo che un client si registra nel server dell'amico tramite *addNick*, esso crea dal suo lato un thread di tipo ServerConn che implementa anch'esso Runnable e il cui scopo è quello di ricevere i messaggi dal output-stream della socket associata e successivamente, dopo averli interpretati correttamente, visualizzarli. Inoltre, un altro compito di questo oggetto è quello di salvare nella cartella locale i file inviati dagli altri client.

Lo scambio dei messaggio segue un particolare protocollo di comunicazione, chiamato ChatServerProtocol. Lo scopo di ChatServerProtocol è la trasmissione corretta dei messaggi tra due client. In particolare, quando un client invia un messaggio al proprio amico, prima che questo messaggio sia trasferito viene verificata la sua correttezza. La funzione responsabile per l'elaborazione del messaggio si chiama *process* e ha lo scopo di leggere lo stream di dati e di individuare il destinatario per poi inoltrare il messaggio. Il destinatario ovviamente sarà un oggetto di tipo ClientConn che si occuperà di trasmettere il messaggio al corrispondente ServerConn che dovrà visualizzarlo.

Dunque per ogni richiesta di connessione si creano due thread, uno da parte del client che ha richiesto la connessione (ServerConn) e l'altro da chi accettato la connessione (ClientConn). Il client scrive nell'output-stream della socket del amico selezionato. Il messaggio è letto dall'oggetto ClientConn che gli era stato associato. Il messaggio è elaborato secondo le regole del protocollo ChatServerProtocol che poi invia il messaggio al thread ClientConn del destinatario che lo inoltra al suo ServerConn, il quale a sua volta dovrà visualizzarlo.

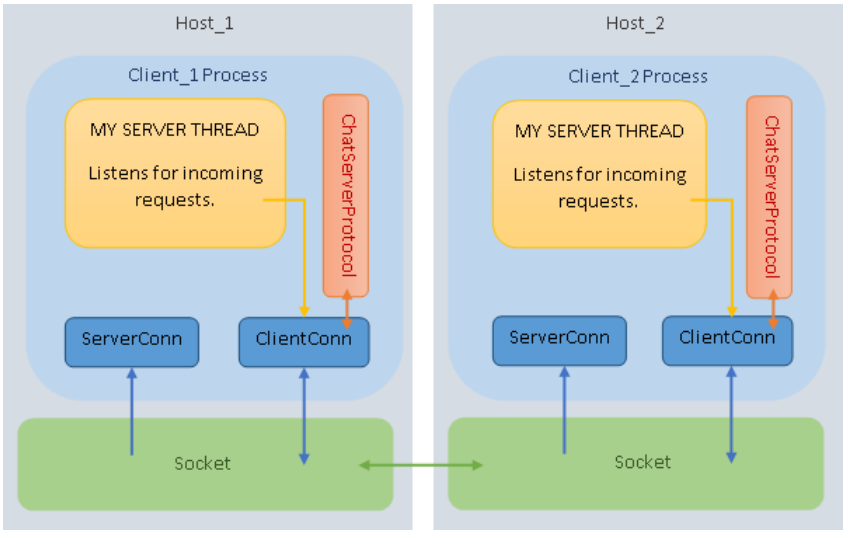

Figura 15. Comunicazione peer-to-peer con socket.

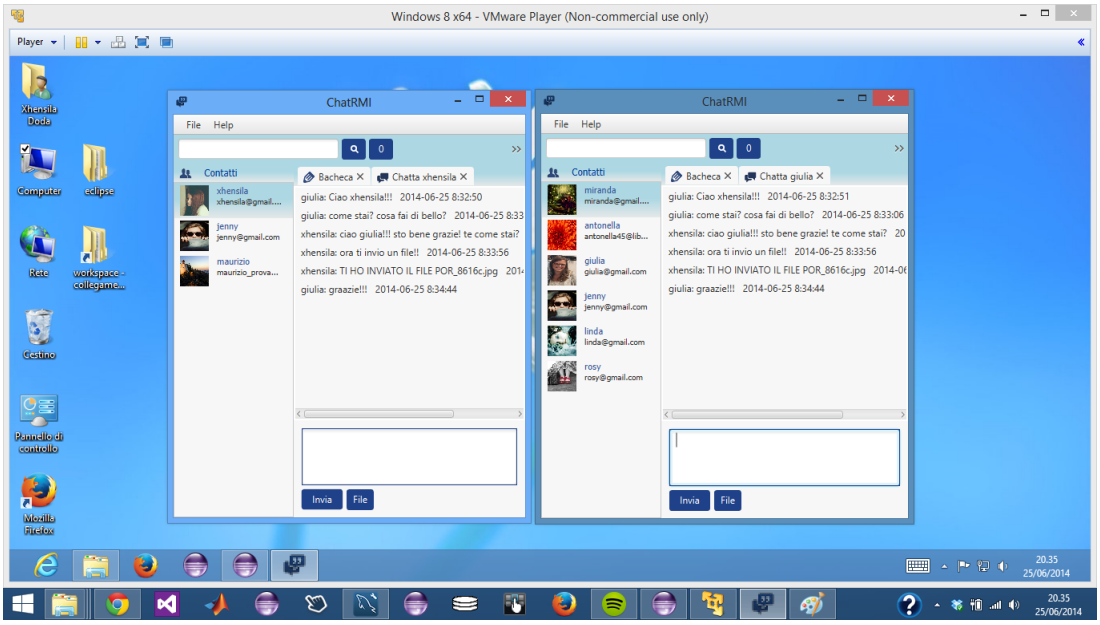

Figura 16. L'immagine mostra un momento di scambio di messaggi tra due client. Uno dei due risiede in una macchina virtuale quindi ha indirizzo IP diverso dall'altro.

5 CONCLUSIONI

Il progetto realizzato adempie alle funzionalità prestabilite durante la stesura dei requisiti del software ed ha permesso di sperimentare in maniera pratica le conoscenze acquisite durante il corso di Sistemi del Software Distribuiti. Infatti, tramite la creazione di quest'applicazione è stato possibile conoscere la tecnologia Java RMI e la comunicazione peer-to-peer in modo più dettagliato e pratico. Inoltre, è stato studiato anche il pacchetto di sviluppo JavaFX creato da Oracle per l'implementazione dell'interfaccia grafica.

Il progetto può essere arricchito con ulteriori nuove funzionalità nel futuro come ad esempio la possibilità di rifiutare una richiesta d'amicizia, di pubblicare post con contenuti multimediali e non solo testuali, di commentare i post nonostante il server RMI non sia online, creare la comunicazione via chat anche attraverso RMI e non solo tramite socket, inviare messaggi ad un client, senza che esso sia online, sotto forma di posta privata.

Riferimenti bibliografici

[1] An Overview of RMI Applications:http://docs.oracle.com/javase/tutorial/rmi/overview.html

[2] Class Socket:http://docs.oracle.com/javase/7/docs/api/java/net/Socket.html

[3] What Is JavaFX?:http://docs.oracle.com/javafx/2/overview/jfxpub-overview.htm

[4] MySQL:http://en.wikipedia.org/wiki/MySQL

[5] SQLite: http://en.wikipedia.org/wiki/Sqlite

[6] Eclipse:http://en.wikipedia.org/wiki/Eclipse_(software)