Specifica dei Requisiti progetto Chat Server	**Application Note**

Descrizione Documento

Cliente	Scialpi, Benvenuti
Progetto	Chat Multi Utente
Titolo	Specifica dei Requisiti progetto Chat Multi Utente
Tipologia Documento	Application Note
Numero Documento	1
Versione	1
Data	15.10.20
Classificazione	Public Release
Autore(i)	Mathilde Patrissi
Approvato da	Mathilde Patrissi

Approvazione Documento

Data	**Nome**	**Titolo**	**Firma**
15/10/2020	Mathilde Patrissi		

Indice dei Contenuti

.1 Introduzione

1.1 Descrizione del documento

Il documento è strutturato in tre capitoli il cui contenuto può essere riassunto come segue.

Nel primo capitolo (il presente) vengono riportate le informazioni di carattere generale che permettono di identificare il progetto a cui questo documento si applica.

Nel secondo capitolo vengono riportati i requisiti funzionali del progetto in questione.

Nel terzo capitolo vengono riportati ulteriori requisiti funzionali del progetto da implementare in una successiva fase di sviluppo.

1.2 Descrizione del contesto

Utilizzando il linguaggio di programmazione java è stata realizzata un'applicazione client-server che permette lo scambio di messaggi di testo tra due host.

All'avvio dell'applicazione server viene chiesto il numero di porta su cui deve ricevere i messaggi dai client.

All'avvio dell'applicazione client viene chiesto il nome del server, la porta porta su cui è in ascolto il server e a cui devono essere inviati i messaggi dai client e infine il nome del client.

.2 Requisiti del progetto

Lo sviluppo del progetto è stato affrontato prendendo come riferimento le informazioni e le richieste fornite dai professori, di seguito riportate:

Realizzare un server in grado di accettare le connessioni da 2 client:

1) il client dopo la connessione al server deve inviare come prima informazione il proprio nome

2) il client, dopo aver inviato il nome, può invia messaggi

3) il server deve inoltrare i messaggi all'altro client

4) il server deve gestire la situazione in cui ci sia solo un solo client collegato (impossibile inoltrare il messaggio)

5) il client deve gestire attraverso un messaggio speciale la chiusura della connessione verso il server

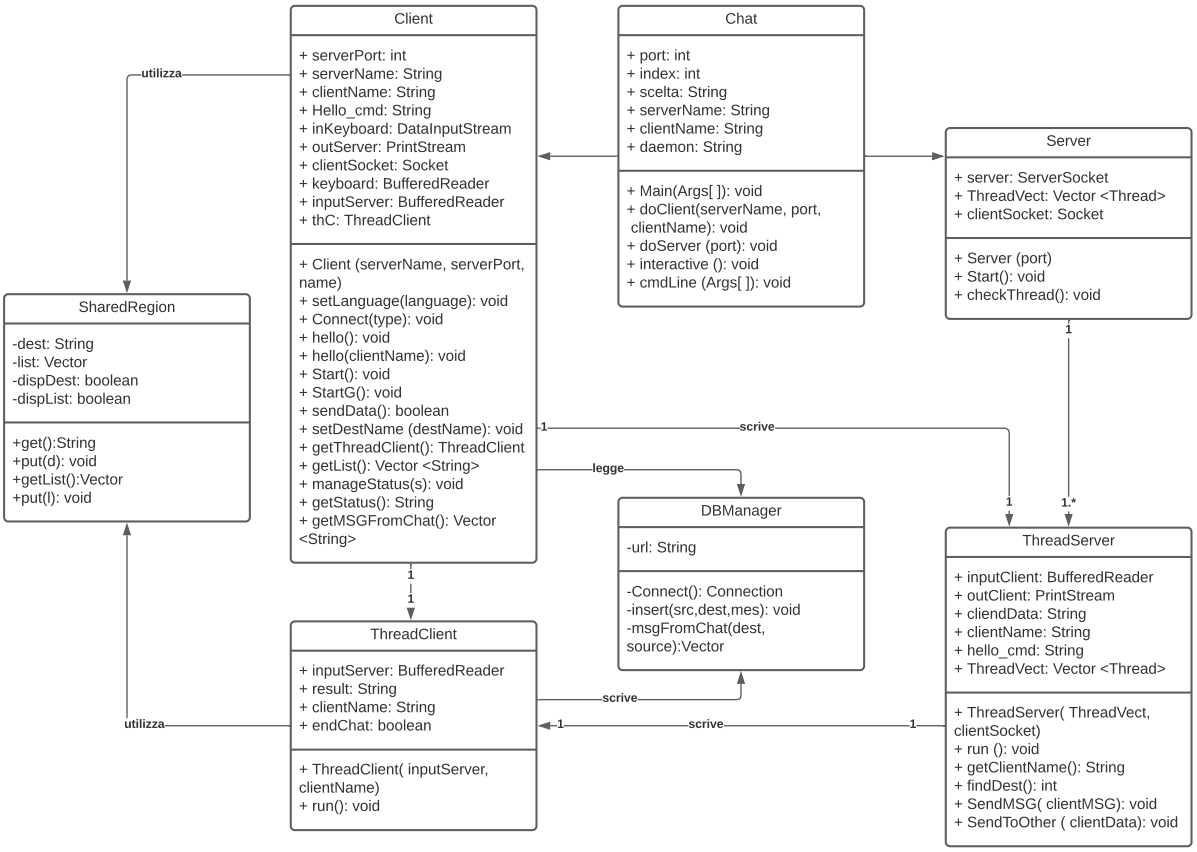
6) gestione della disconnessione del client

7) i client devono conoscere i nomi del client connessi

8) possibilità di avviare chat one to one oppure one to all

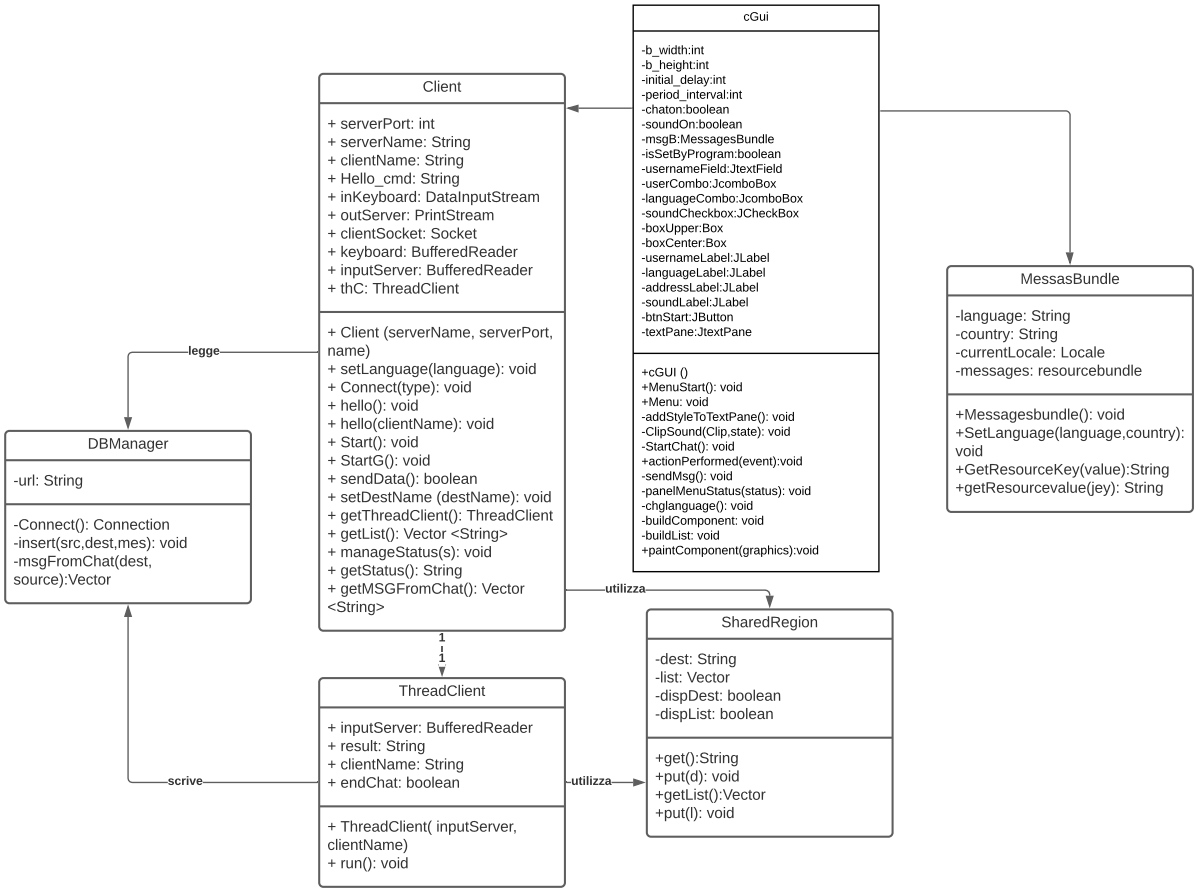
2.1 Diagramma della classe ChatServer

Di seguito viene riportato il diagramma UML relativo alla classe ChatServer



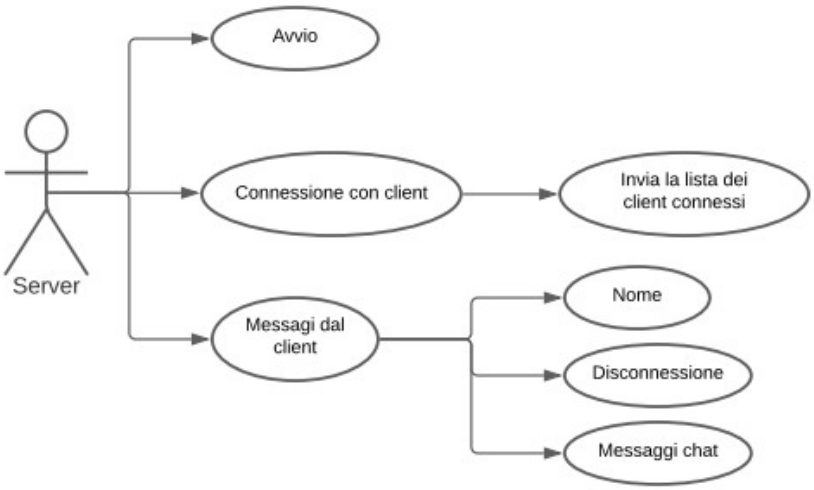
2.2 Diagramma della classe ChatClientGUI

Di seguito viene riportato il diagramma UML relativo alla classe ChatClientGUI



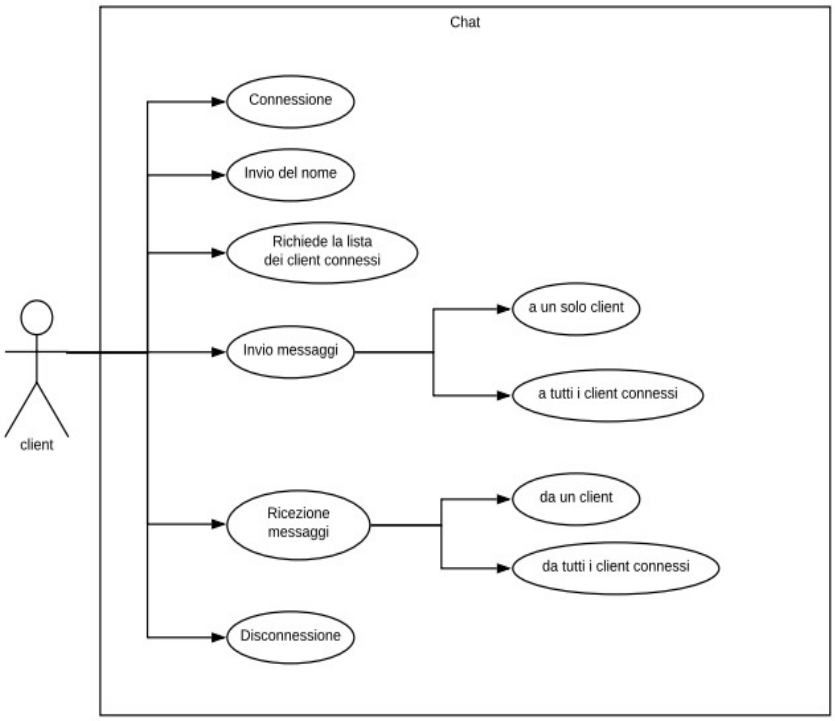
2.3 Casi d'uso server

Di seguito viene riportato il diagramma dei casi d'uso relativo alla componente server:



2.4 Casi d'uso Client

Di seguito viene riportato il diagramma dei casi d'uso relativo alla componente client:



.3 Requisiti funzionali del progetto

L'applicativo in questione è stato realizzato in forma modulare, ovvero è costituito da una componente server e una componente client:

- la parte client, che si occupa di interagire con l'utente, è disponibile sia in modalità grafica che da riga di comando;
- la componente server si occupa della gestione dei messaggi inviati, inoltrando agli opportuni destinatari.

Poiché durante il consueto utilizzo dell'applicativo le due componenti si troveranno su host differenti, la loro comunicazione è stata realizzata mediante l'utilizzo dei socket.

I messaggi ricevuti ed inviati lato client sono stati inoltre memorizzati in un database SQLITE: ad un successivo riavvio dell'applicativo, sarà quindi possibile visualizzare i messaggi precedentemente inviati.

L'applicativo è stato predisposto per il supporto alle principali lingue tra cui italiano, inglese, francese, spagnolo e tedesco.

3.1 Inserimento dati e messaggi di informazione

All'avvio dell'applicazione server viene chiesto il numero di porta su cui deve ricevere i messaggi dai client.

All'avvio dell'applicazione client viene chiesto il nome del server, la porta porta su cui è in ascolto il server e a cui devono essere inviati i messaggi dai client e infine il nome del client.

Per entrambi è possibile utilizzare i parametri a riga di comando invece di utilizzare la modalità interattiva.

3.2 Caratteristiche della chat

Tra i client è possibile solo lo scambio di messaggi di testo.

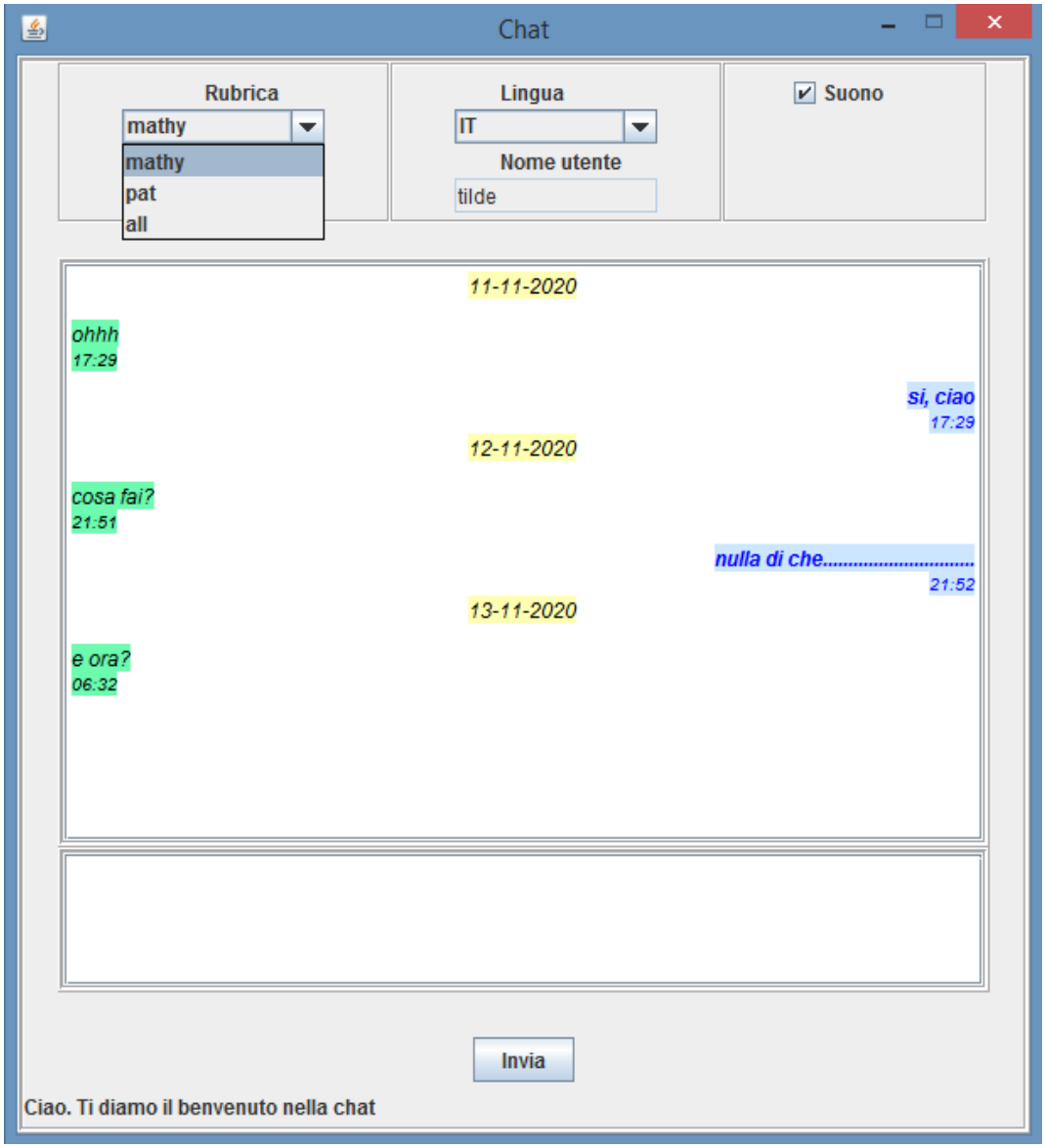
Il server utilizza il meccanismo dei Thread per la gestione e l'invio dei messaggi, quindi ogni volta che un client effettua una connessione al server viene creato un thread. Questo si occuperà della comunicazione con il client in modo da permettere al server di gestire altre richieste di connessione.

La comunicazione è gestita mediante l'utilizzo di un particolare protocollo che permette l'identificazione e la distinzione dei messaggi da inviare sulla chat (preceduti dal comando DATA_CMD) e dei comandi speciali (HELLO_CMD, QUIT_CMD, SETDEST_CMD, LIST_CMD).

Dalla cli vi è quindi la possibilità di utilizzare alcuni comandi come mostrato nello SplashScreen iniziale:

1) “q!” per lasciare la chat.
2) “dest:” seguito dal nome del destinatario, per comunicare unicamente con tale utente, oppure da “all” per comunicare con tutti gli utenti connessi
3) “list” per conoscere il nome degli utenti connessi alla chat

Tutto ciò è possibile anche mediante l'utilizzo dell'interfaccia grafica, gestita tramite swing. In questa modalità la lista, chiamata Rubrica, degli utenti connessi è presente nella list box situata in alto a sinistra, come mostrato dall'immagine seguente:



Cliccando sul nome dell'utente desiderato verrà effettuato l'inoltro all'opportuno destinatario.
Ogni volta che viene cambiato il destinatario vengono mostrati i messaggi scambiati con quest'ultimo, compresa la relativa data e ora di invio. Ciò è possibile mediante l'utilizzo di un database SQLite dove vengono inseriti e letti i messaggi nei momenti opportuni. Le comunicazioni da parte del server, ad esempio in caso di errori, vengono mostrate nella parte inferiore della chat.
Vi è infine la possibilità di attivare e disattivare la musica di sottofondo e di settare la lingua desiderata.
Quest'ultima funzionalità è stata realizzata tramite l'utilizzo della classe java.util.ResourceBundle che ci permette di predisporre il supporto alle principali lingue tra cui italiano, inglese, francese, spagnolo e tedesco.
E' stata pertanto realizzata la classe MessagesBundle che, alla scelta della lingua desiderata, seleziona l'opportuno bundle di risorse per la gestione di tale lingua. Un bundle di risorse è un file .properties Java che contiene dati specifici della lingua. È un modo di internazionalizzare l'applicazione Java rendendo il codice indipendente dalla linuga locale.
Di seguito un esempio del file *MessagesBundle_it_IT.properties* per la gestione della lingua italiana:

```
btn_send= Invia
btn_start= Inizia
btn_soundOn = Acceso
label_place = Sfondo
label_username = Nome utente
label_address= Rubrica
label_language = Lingua
label_sound = Suono
warn_msg=Attenzione
warn_noname_msg= Per favore, inserisci il nome utente
err_msg= Errore
hello_msg=  Ciao. Ti diamo il benvenuto nella chat
client_connectOK = Connessione stabilita
errorsrv_exit_msg= Errore dal server. Bye Bye.
errorsrv_cmd_msg=Il server risponde: errore comando
errorsrv_readfromclient_msg=Il server risponde: errore lettura da client
server_name=localhost
server_port=9876
```

I messaggi ricevuti ed inviati lato client sono stati inoltre memorizzati in un database SQLite. Ad un successivo riavvio dell'applicativo, sarà quindi possibile visualizzare i messaggi precedentemente inviati.

Di seguito la tabella utilizzata:

	source	dest	mess	groupm	insert_at
	Filtro	Filtro	Filtro	Filtro	Filtro
1	pat	tilde	ciao a tutti	all	2020-11-11 17:01:39
2	mathy	tilde	ciao a tutti anche da mathy	all	2020-11-11 17:01:56
3	tilde	all	ciao benvenuti	all	2020-11-11 17:02:15
4	tilde	pat	ciao pat	pat	2020-11-11 17:19:25
5	pat	tilde	ciao tilde	all	2020-11-11 17:19:46
6	pat	tilde	provo ora tilde	tilde	2020-11-11 17:21:48
7	pat	tilde	ciao	tilde	2020-11-11 17:28:44
8	mathy	tilde	ed io?	all	2020-11-11 17:29:00
9	mathy	tilde	ohhh	tilde	2020-11-11 17:29:19
10	tilde	mathy	si, ciao	mathy	2020-11-11 17:29:30
11	tilde	all	Cosa fate?	all	2020-11-11 21:09:33
12	tilde	all	tutto bene?	all	2020-11-11 21:09:40
13	tilde	all	ciaooo	all	2020-11-11 21:15:51
14	pat	tilde	fatto?	tilde	2020-11-11 21:42:55
15	tilde	pat	forse	pat	2020-11-11 21:43:03
16	tilde	pat	vediamo come funziona.	pat	2020-11-11 21:43:19
17	pat	tilde	ve bene, proviamolo ancora un po'	tilde	2020-11-11 21:43:37
18	mathy	tilde	cosa fai?	tilde	2020-11-12 21:51:37

Messaggi inviati dal client al server

Di seguito ler tipologie di messaggi inviati dal client al server:

Messaggio	Mittente	Contenuto	Formato
HELLO_CMD	Client	Nome del client	“ciao sono”+nome client
QUIT_CMD	Client	Disconnessione del client	“q!”
DATA_CMD	Client	Messaggio del client	“data:”+mittente+”:”+destinatario+”:”+messaggio
SETDEST_CMD	Client	Nome del destinatario	“dest:”+ nome destinatario
LIST_CMD	Client	Lista client connessi	“list”

Messaggi di risposta del server

Di seguito ler tipologie di messaggi inviati dal client al server:

Messaggio	Mittente	Contenuto	Formato
STATUS_RESP	Client	Messaggio di stato	“status”+messaggio
QUIT_CMD	Client	Conferma disconnessione	“q!”+messaggio

.4 Requisiti non funzionali del progetto

Requisiti di prodotto

Il progetto deve essere realizzato tramite il linguaggio di programmazione java e tramite il meccanismo dei thread.
Deve essere realizzata un’applicazione client-server che permette lo scambio di messaggi di testo tra due host.

Requisiti di consegna

Il progetto, che deve essere terminato entro il 15 Novembre, deve comprendere i file sorgenti, con gli opportuni commenti, e i javadoc.
Il progetto deve essere inserito in un repository git il cui url dovrà essere consegnato entro e non oltre la data suddetta.

.5 Requisiti non funzionali del progetto

In una fase successiva del progetto si potrebbe ipotizzare la gestione di gruppi differenti e l'invito o la rimozione di utenti all'interno di tali gruppi.
Si potrebbe anche ipotizzare la gestione dello scambio di file multimediali, l'impostazione di una propria foto profilo e stato.
Infine si potrebbe implementare l'autenticazione dell'utente e la crittografia del canale di comunicazione.