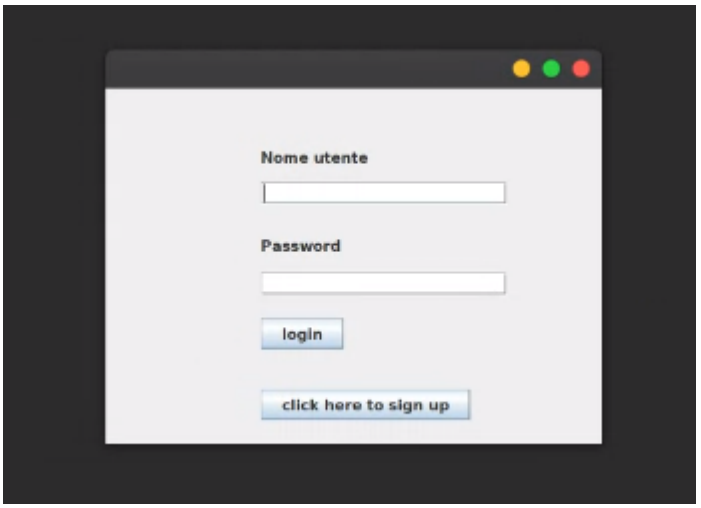
MARTINO MANIERO, ALESSANDRO BERGANTIN

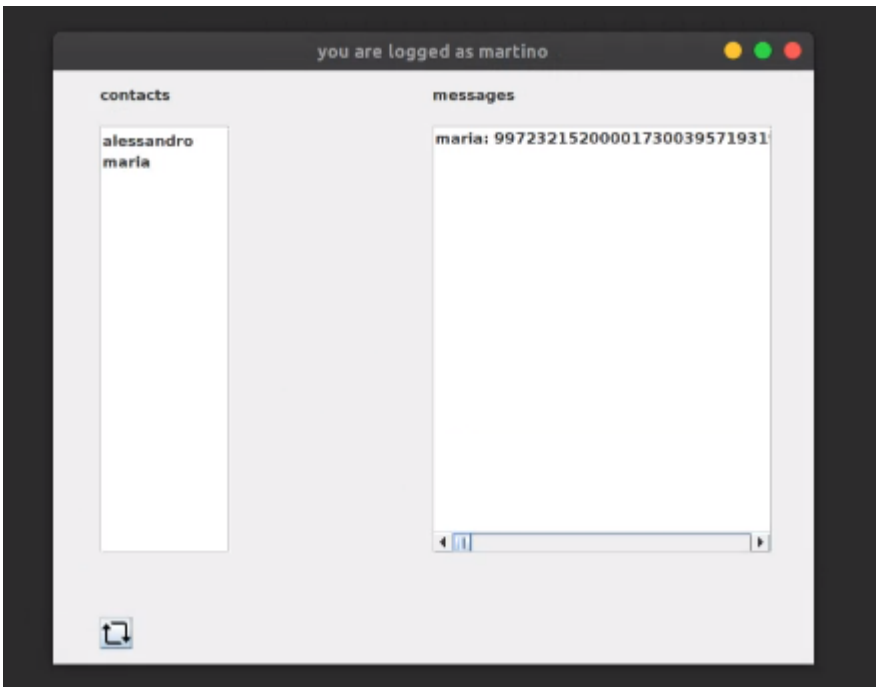
PROGETTO DI INGEGNERIA DEL SOFTWARE AVANZATA - CRITTOCHAT

Il progetto consiste in sistema di messaggistica stile mail-box, nel quale ogni iscritto può scambiare messaggi cifrati con algoritmo RSA da noi implementato.

I messaggi verranno visualizzati nelle apposite sezioni in maniera cifrata e si potrà leggere il contenuto in chiaro solamente inserendo la propria chiave privata che è stata ricevuta al momento della registrazione utente.

L'applicazione è basata su un database SQL.





PIANO DI LAVORO

Il termine di scadenza è stato fissato al 17/07/2022 mentre la consegna il 20/07/2022 si è quindi decido di stilare una tabella di marcia stile “metodo agile” dandosi delle scadenze precise per ogni task..

Data	Attività
05/07	Definizione dei focus point del progetto
06/07	Decisione software e fondamentali del
07/07	Inizio realizzazioni funzionalità base (login/signup/ grafica base)
08/07	Test prima parte. Inizio stesura relazione
09/07	Inizio realizzazione rubrica con lista utenti registatri con i quali si potranno scambiare messaggi. Realizzazione DB SQL.
10/07	Sistema di chat semi real time. + grafica visualizzazione dei messaggi ricevuti da tutti (in chiaro)
11/07	Test seconda parte. Integrazione relazione
12/07	Sviluppo algoritmo di cifratura e decifratura
13/07	Test terza parte. Integrazione relazione
14/07	Integrazio algoritmo di cif/decif versione base con una unica decriptyon key
15/07	Integrazione fornitura utenti registrati di key public e private (generatore pseudo-casuake)
16/07	Test quarta parte.
17/07	
18/07	
19/07	

I tempi sono stati rispettati ad eccezione della seconda parte dove la fase di testing delle query ha richiesto più del previsto.

ALGORITMO RSA

E' stato utilizzato un crittosistema RSA.
*L'algoritmo è basato su due primi grandi (p e q),n=p*q, phi=(p-1)(q-1), chiave di cifratura "e" t.c. (e,phi(n))=1, chiave di decifratura "d"-> ed=1 mod(phi).*
CHIAVE PUBBLICA= e,n
CHIAVE PRIVATA=d,phi,p,q

PROCEDIMENTO DI CIFRATURA
si vuole cifrare il messaggio M e C è il cifrato.
$C=M^e \bmod(n)$

PROCEDURA DI DECIFRATURA
si vuole ricondursi al messaggio M e solamente chi ha la chiave privata d riesce a ricondursi ad esso.
$C^d=M \bmod(n)$

Nel progetto si è sviluppato anche un generatore casuale di chiavi (privata e pubblica), in modo che ogni utente al momento della registrazione riceva la propria chiave private, e una chiave pubblica che viene associata al suo profilo. In questo modo, chiunque voglia inviare un messaggio può usarla e in maniera completamente trasparente cifrato.
Il generatore ha definito un BigInteger molto grande che sarà moltiplicato per un valore random, infine viene incrementato questo numero fino ad ottenerne uno relativamente primo con PHI=(p-1)*(q-1) ottenendo così "E" la chiave pubblica.
Per ottenere la "D" chiave privata si calcola l'inverso moltiplicativo ED=1 (mod(PHI)).

UTILIZZO GITHUB

come sistema di controllo di versione è stato usato github, essenziale per tener traccia dei vari avanzamenti del progetto, arrivando ad avere varie versioni del software.
E' stato aggiunto un file ".gitignore" dove vengono specificati i file che al momento del push devono essere ignorati.

Nel file readme si possono trovare alcune informazioni utili per un corretto utilizzo dell'applicazione.

https://github.com/MarMannnn/ProgettoIsa *(GUARDA)*

UTILIZZO MAVEN

Come build e management tool è stato usato MAVEN.
Attraverso un file pom.xml è stato possibile gestire le dipendenze andando per esempio ad aggiungere: JUnit, SQLconnector etc
Inoltre è stata fornito il percorso del nostro progetto in modo da poter eseguire il build e compile direttamente.
E' stata rilasciata documentazione attraverso il comando mvn site.

PROCESSO DI SVILUPPO SW

Come IDE si è deciso di utilizzare NetBeans 14, questo ci ha permesso di realizzare semplici interfacce grafiche fornendoci framework appositi.
Possiamo individuare 3 versioni principali del sw, una prima versione dove si è stato implementato un sistema di messaggistica in chiaro (no cifratura), una seconda versione dove i messaggi venivano cifrati tutti con la stessa chiave pubblica, e una versione finale dove ogni utente ha una chiave pubblica e privata.

COMPONENTI APP:
0-Database: è stato creato un database SQL con due tabelle (utenti, messaggi). Nella tabella utenti abbiamo: UserName, psw e chiave pubblica, mentre nella tabella messaggi: testo, mittente, destinatario ed un iD per tener ordine nei messaggi.
1-Login/sign up: questa è stata la prima parte sviluppata. Un nuovo utente può registrarsi andando ad inserire un UserName univoco e una psw. Questi verranno salvati nel database precedentemente creato attraverso un INSERT.
Il login avverrà inserendo le proprie credenziali che verranno verificate interrogando il database.
2-Rubrica: seconda componente sviluppata. Presenta 2 sezioni principali: lista utenti e lista messaggi ricevuti.
Nella lista utenti attraverso una select vengono riportati tutti gli utenti registrati; con un doppio clic sull'utente di interesse verrà aperta la schermata "Chat" dove sarà possibile chattare con l'utente.

Nella lista messaggi ricevuti si potranno visualizzare i messaggi cifrati ricevuti. Con un doppio clic su essi, verrà chiesto di inserire la propria chiave privata e se corretta verrà decifrato il messaggio.
3-Chat: In questa parte l'utente potrà inviare messaggi al destinatario selezionato. I messaggi verranno automaticamente cifrati secondo la chiave pubblica del destinatario desiderato.
Inoltre vi è una sezione dove si riescono a vedere i messaggi scambiati e con un doppio clic su essi si possono visualizzare in chiaro inserendo la chiave privata corrispondente.

FASI DI TESTING

E' stato utilizzato come framework per l'unit test JUNIT.
Si sono volute testare quelle componenti da noi realizzate e quindi con possibile presenza di errori.

Nel sistema RSA i casi di test eseguiti sono stati i seguenti:
- casi di test "limite": abbiamo verificato che il primo elemento (33=!) della tabella ASCII, l'ultimo (126=~), e un carattere intermedio (99=c) fossero cifrati e decifrati correttamente.
- casi di test generale: è stata inserita una parola (ciao) della quale si conosceva il cifrato, e si sono andati a confrontare il cifrato ottenuto dall'algoritmo e quello da noi conosciuto. Si è anche testato l'algoritmo di decifrazione quindi partendo da un cifrato conosciuto si è arrivati a verificare la correttezza della decifratura.

MyConnectionTest: test eseguito per verificare che la connessione al nostro DB vada a buon fine

GeneratoreChiaviTest: si sono eseguiti test per verificare la corretta generazione di chiavi (pubblica e privata), andando a verificare che la E generata fosse relativamente prima con PHI=(p-1)*(q-1).

RubricaTest: si sono eseguiti test per andare a verificare il corretto esito delle query eseguite per fornire "lista utenti" e "lista messaggi".