



Università
degli Studi
di Ferrara

CRITTO_CHAT

Progetto per il corso di “Ingegneria del Software Avanzata” dell’università di Ferrara.

Il team



Alessandro
Bergantin



Martino
Maniero

MISSION:

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

TABELLA DI MARCIA

Data la possibilità di lavorare congiuntamente, e la stretta relazione tra i componenti del team si è provveduto a procedere in maniera agile. Di fatto il “piano” è stato definito inizialmente solamente sui punti salienti della nostra applicazione e in corso d’opera è stato aggiornato per aggiungere le funzionalità richieste.

Come scadenza di realizzazione SW ci si è prefissati il 17/07 mentre come data di consegna del progetto 20/07.

Data	Attività
05/07	Definizione dei focus point del progetto
06/07	Decisione software e ricerca informazioni su progetti simili
07/07	Inizio realizzazioni funzionalità base (login/signup/ grafica base)
08/07	Realizzazione DB SQL. Connessione all’applicazione
09/07	Inizio realizzazione rubrica con lista utenti registrati con i quali si potranno scambiare messaggi.
10/07	Sistema di chat semi real time. + grafica visualizzazione dei messaggi ricevuti da tutti (in chiaro)
11/07	Test prima parte. Integrazione relazione
12/07	Sviluppo algoritmo di cifratura e decifratura
13/07	Test seconda parte. Integrazione relazione
14/07	Integrazione algoritmo di cif/decif versione base con una unica decryption key e public key
15/07	Integrazione fornitura utenti registrati di key public e private (generatore pseudo-casuale)
16/07	Test terza parte. Inizio Docker
17/07	Docker + Presentazione
18/07	CONCLUSIONE PRESENTAZIONE + RIPASSO + Docker
19/07	RIPASSO PER PRESENTAZIONE

The background of the slide is a dark blue image featuring a circuit board pattern. In the center, there is a metallic padlock with its shackle open, positioned next to a small, translucent globe showing the Earth's continents. The text 'L'algoritmo RSA' is overlaid in a large, white, sans-serif font.

L'algoritmo RSA

CARATTERISTICHE

CHIAVE PUBBLICA:

- $e \rightarrow$ encryption-key
- $n = p \cdot q$

CHIAVE PRIVATA:

- $p, q \rightarrow$ due primi grandi
- $\phi = (p-1) \cdot (q-1)$
- $d \rightarrow$ decryption-key

PROCEDIMENTO CIFRATURA/DECIFRATURA

Processo di cifratura prende in ingresso il messaggio (M) -> viene trasformato in byte secondo la conversione string to byte -> viene passato il risultato al metodo "encryptMessage" il quale va a calcolarsi il cifrato $C \equiv M^e \bmod(n)$.

Processo di decifratura riceve in ingresso il cifrato C -> viene chiesto all'utente di inserire la chiave privata (PERSONALE) -> conversione ASCII TO STRING.



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.

<https://github.com/MarMannnn/ProgettoIsa>



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

mvn compile

mvn exec:java

mvn site

PROCESSO DI SVILUPPO

Come IDE si è scelto 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.

0-DATABASE

userName	password	pubblica
alessandro	psw	496155981585083281636...
maria	psw	351125771583289707004...
martino	psw	725151050008967873161...

E' stato creato un DB SQL con due tabelle
(utenti, messaggi)

testo	mittente	destinatario	iD
1882238890311021256647054795...	alessandro	maria	55
8003942131494420694212843293...	maria	alessandro	56
4475142995771733046564600961...	martino	maria	57
4812771902631215893904176069...	martino	maria	58
4812771902631215893904176069...	martino	maria	59
9972321520000173003957193193...	maria	martino	60
3850181297867082510393602303...	martino	maria	61
8239440064531279625078163081...	martino	maria	62

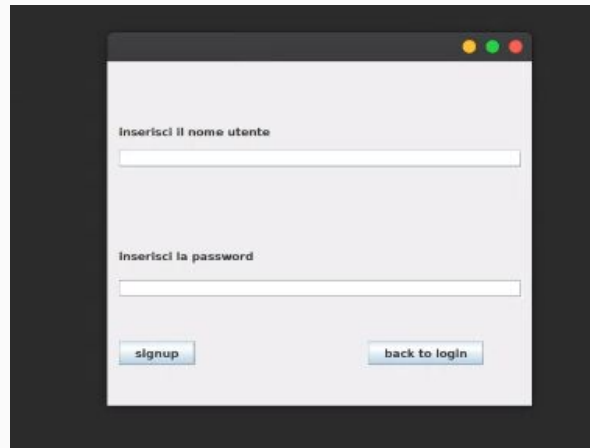
1-LOGIN/SIGNUP

Prima parte sviluppata. Un nuovo utente può registrarsi inserendo UserName (univoco) e PSW, i quali attraverso un INSERT verranno aggiunti al DB.

Nella fase di login, l'utente inserirà proprio User e psw che verranno verificati andando ad interrogare il DB.

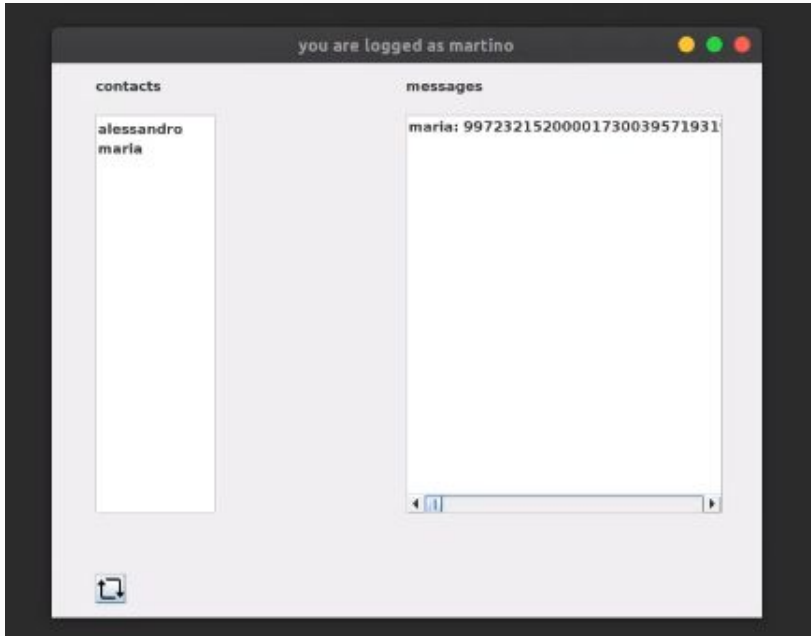


A screenshot of a web application login form. The form is light gray with a dark gray header bar containing three colored window control buttons (yellow, green, red). The form contains two text input fields: the first is labeled "Nome utente" and the second is labeled "Password". Below the password field is a blue "login" button. At the bottom of the form is a blue button labeled "click here to sign up".



A screenshot of a web application signup form. The form is light gray with a dark gray header bar containing three colored window control buttons (yellow, green, red). The form contains two text input fields: the first is labeled "Inserisci il nome utente" and the second is labeled "Inserisci la password". Below the password field are two blue buttons: "signup" on the left and "back to login" on the right.

2-RUBRICA



Seconda componente sviluppata. Presenta 2 sezioni principali: CONTATTI e MESSAGGI RICEVUTI.

Nella lista contatti attraverso una SELECT vengono visualizzati tutti gli utenti registrati. Possibilità attraverso un doppio clic di chattare con l'utente selezionato.

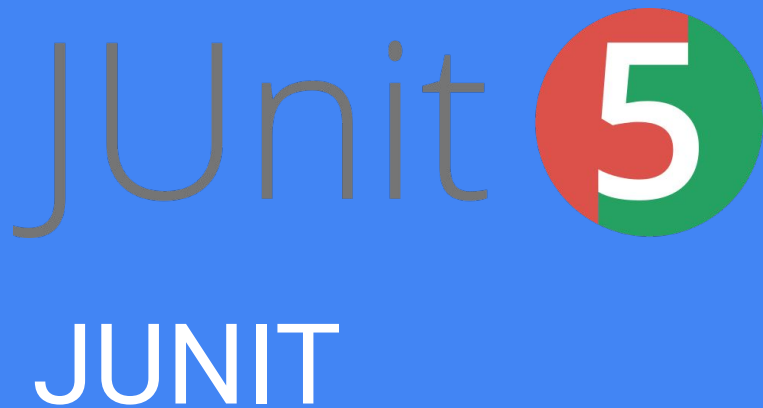
Nella lista messaggi ricevuti si vedono tutti i messaggi ricevuti dai vari utenti ed è possibile decifrarli andando ad eseguire un doppio clic ed inserendo la propria chiave privata.

3-CHAT



Terza parte sviluppata, 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.



JUnit è un framework open source, usato per scrivere ed eseguire unit testing per Java. Si tratta di un framework assai rilevante in ambito QA & test, che ha promosso il ruolo fondamentale dei test nelle attività di programmazione.

L'attività di testing si è suddivisa in varie parti:

- testing delle componenti di rubrica*
- testing delle componenti di chat*
- testing connessione al database*
- testing delle componenti dell'algoritmo*
- testing generatore chiavi pseudo-casuale*

TESTING COMPONENTI ALGORITMO RSA e GENERATORE PSEUDO-CASUALE.

Si è provveduto ad eseguire come prima cosa il testing dei casi limite, e anche stato testato il codificatore ASCII-toString.

<https://github.com/MarMannnn/Progettolsa/blob/main/testMex/src/test/java/com/mycompany/testmex/RSANumericoTest.java>

Per quanto riguarda il generatore pseudo-casuale si è testato che la chiave generata fosse relativamente prima con PHI

<https://github.com/MarMannnn/Progettolsa/blob/main/testMex/src/test/java/com/mycompany/testmex/GeneratorEchiaviTest.java>

```

@Test
public void testCifratura() {
    System.out.println("test metodo cifratura");
    String msgDaCifrare = "ciao";
    RSANumerico instance = new RSANumerico("7251510500089678731610942856...");
    String expResult = "237558993459994918204834051968540503098142685243...";
    String result = instance.cifratura(msgDaCifrare);
    assertEquals(expResult, result);
}

/**
 * Test of decifratura method, of class RSANumerico.
 */
@Test
public void testDecifratura() {
    System.out.println("test metodo decifratura");
    String msgDaDecifrare = "2375589934599949182048340519685405030981426...";
    BigInteger d = new BigInteger("8274961930261298473418953057409006767...");
    RSANumerico instance = new RSANumerico(d);
    String expResult = "ciao";
    String result = instance.decifratura(msgDaDecifrare);
    assertEquals(expResult, result);
}
```


LIMITI DELL'APP:

codice di conversione ASCII -> è, à,
ù non disponibili.

DOCKER



docker

Creazione di un Container, si è provveduto ad utilizzare "Docker", si è però riscontrato un problema!

-> La nostra app possiede un GUI.

Primo errore riscontrato mancanza di display:

```
java.awt.HeadlessException:
```

```
No X11 DISPLAY variable was set, but this program performed an operation which requires it.
```

Andando ad implementare DISPLAY X11 segnalazione di un errore che non si è riusciti a risolvere:

```
An exception occurred while executing the Java class.  
/usr/local/openjdk-11/lib/libawt_xawt.so: libXext.so.6: cannot open  
shared object file: No such file or directory
```