



CLAUDIO ROBERTO SCHEER JUNIOR

GUSTAVO HENRIQUE MOTTA ROSA

LARISSA DAIANE CANEPPELE GUDER

CRIPTOGRAFIA EM UM CHAT DE MENSAGENS PRIVADAS

Três de Maio

2015

CLAUDIO ROBERTO SCHEER JUNIOR
GUSTAVO HENRIQUE MOTTA ROSA
LARISSA DAIANE CANEPPELE GUDER

CRIPTOGRAFIA EM UM CHAT DE MENSAGENS PRIVADAS

Projeto do Trabalho Interdisciplinar I
do Curso Bacharelado em Sistemas
de Informação, da Sociedade
Educacional Três De Maio- SETREM

Professores Orientadores:

Msc. Aldalberto Lovato

Mestranda Denise da Luz Siqueira

Msc. Keila Kleveston Schneider

Três de Maio

2015

RESUMO

Quando se fala em segurança da informação, o que vem à cabeça é a vulnerabilidade dos dados. Consciente ou inconscientemente há a necessidade de segurança para a troca de informações. Tendo isto em mente, a criptografia tem um papel de extrema importância para manter as informações seguras. Como a maioria das informações trafega pela rede mundial de computadores, a internet, estes dados acabam por se tornar vulneráveis a eventuais ataques de pessoas mal-intencionadas, sendo que as informações transmitidas podem ser interceptadas e utilizadas de formas indevidas. Neste contexto, o trabalho tem como objetivo principal, o desenvolvimento de uma solução que permita a comunicação entre usuários de forma segura. Juntamente com conceitos de matrizes, o estudo acerca da criptografia e utilização da linguagem Java, foi desenvolvido um algoritmo que possibilita a criptografia de informações. Ao término do projeto foi possível validar a segurança oferecida pelo algoritmo, através da aplicação de testes de captura de dados na rede, que demonstraram que o algoritmo compartilha as informações criptografadas.

Palavras-chave: Sistemas de Informação, Segurança e Privacidade Criptografia, Funções da Criptografia e suas Chaves Simétricas, Fundamentos Matemáticos da Criptografia.

ABSTRACT

When it comes to information security, what comes to mind is the vulnerability of data. Consciously or unconsciously there is the need for security for the exchange of information. With this in mind, the encryption has a role of utmost importance to keep information secure. Like most information travels through the World Wide Web, the Internet, the data end up becoming vulnerable to possible ill-intentioned people from attacks, and the transmitted information can be intercepted and misused. In this context, the work has as main objective the development of a solution that will allow communication between users securely. Along with concepts matrices study about the encryption and the use of the Java language was developed an algorithm that allows encryption information. At the end of the project was to validate the security offered by the algorithm, by applying data capture tests on the network, which showed that the algorithm shares the encrypted information.

Keywords: Information systems, Security and privacy, Cryptography, Symmetric cryptography and hash functions, Mathematical foundations of cryptography.

LISTA DE ILUSTRAÇÕES

Quadro 1 - Cronograma	18
Quadro 2 - Orçamento	19
Quadro 3 – Valores para conversão de letras	31
Quadro 4 – Exemplo de mensagem convertida para números	31

LISTA DE FIGURAS

Figura 1 - Tela inicial para servidor	38
Figura 2 - Opção sair do servidor	39
Figura 3 - Configurar número conexões permitidas no servidor.....	40
Figura 4 - Opção criar do servidor.....	40
Figura 5 - Tela do servidor criado.....	41
Figura 6 - Local onde é mostrado o nome do usuário servidor	42
Figura 7 - Local onde é mostrado o IP do servidor.....	43
Figura 8 - Número de usuários conectados ao servidor	44
Figura 9 - Opções número de conexões do servidor.....	45
Figura 10 - Função exportar mensagens servidor	46
Figura 11 - Esquema de conexões ao servidor	47
Figura 12 - Tela configurações para cliente conectar.....	47
Figura 13 - Opção sair do cliente	48
Figura 14 - Configurando o IP do servidor	48
Figura 15 - Opção conectar do cliente	49
Figura 16 - Tela para cliente conectado	50
Figura 17 - Local onde o mostra nome do cliente	51
Figura 18 - Função exportar do cliente.....	52
Figura 19 - Inversão de matriz no Excel.....	57
Figura 20 - Inversão de matriz no algoritmo desenvolvido	57
Figura 21 - Multiplicação de matriz no Excel.....	58
Figura 22 - Multiplicação de matriz no algoritmo desenvolvido	59
Figura 23 - Transposição de matriz no Excel	59
Figura 24 - Transposição de matriz no algoritmo desenvolvido	60
Figura 25 - Softwate sniffer com a mensagem criptografada capturada	60

SIGLAS

a.C. – antes de Cristo

det – Determinante

SETREM – Sociedade Educacional Três de Maio

SQL – Structure Query Language

ASCII – American Standart Code for Information Interchange

IP – Internet Protocol

API – Aplicação Programmer Interface

TXT – Text

SUMÁRIO

RESUMO.....	3
ABSTRACT.....	4
LISTA DE ILUSTRAÇÕES	5
LISTA DE FIGURAS	6
SIGLAS	7
SUMÁRIO	8
INTRODUÇÃO	12
1 PROJETO PESQUISA.....	14
1.1 Tema	14
1.1.1 <i>Delimitação do Tema</i>	<i>14</i>
1.2 Objetivos	14
1.2.1 <i>Objetivo geral.....</i>	<i>14</i>
1.2.2 <i>Objetivos específicos.....</i>	<i>14</i>
1.3 Justificativa	15
1.4 Problemas da pesquisa	16
1.5 Hipóteses.....	16
1.6 Variáveis	16
1.7 Metodologia.....	17
1.7.1 <i>Métodos de Abordagem.....</i>	<i>17</i>
1.7.2 <i>Métodos de Procedimento.....</i>	<i>17</i>

1.7.3 Técnicas	17
1.8 Cronograma.....	18
1.9 Orçamento	19
2 REFERENCIAL TEÓRICO	20
2.1 Matrizes	20
2.1.1 Propriedades de matrizes	20
2.1.2 Definições	21
2.1.2.1 Matriz Linha	21
2.1.2.2 Matriz Coluna	21
2.1.2.3 Matriz Quadrada	21
2.1.2.4 Matriz Nula	21
2.1.2.5 Matriz Diagonal.....	22
2.1.2.6 Matriz Identidade	22
2.1.2.7 Matriz Transposta	22
2.1.2.8 Matriz Triangular Superior	22
2.1.2.9 Matriz Triangular Inferior.....	23
2.1.2.10 Matriz Escalar	23
2.1.3 Operações com Matrizes	23
2.1.3.1 Soma de Matrizes.....	23
2.1.3.1.1 Associação	23
2.1.3.1.2 Comutação	24
2.1.3.1.3 Elemento Neutro ou Matriz Neutra	24
2.1.3.1.4 Matriz Negativa.....	24
2.1.3.2 Igualdade de Matrizes	24
2.1.3.3 Multiplicação por um escalar	24
2.1.3.4 Multiplicação de Matrizes	24
2.1.3.5 Potências de Matriz	25

2.1.4	<i>Determinantes</i>	25
2.1.4.1	Determinante de matriz quadrada de segunda ordem	25
2.1.4.2	Determinante de matriz quadrada com ordem igual a três	26
2.1.4.2.1	Regra de <i>Sarrus</i>	26
2.1.4.3	Determinante de matriz com ordem superior a três	26
2.1.4.3.1	Teorema de <i>Laplace</i>	26
2.1.5	<i>Matriz Inversa</i>	26
2.2	Criptografia	27
2.2.1	<i>Algoritmos Simétricos</i>	28
2.2.2	<i>Algoritmos Assimétricos</i>	30
2.3	Linguagens de programação	32
2.3.1	<i>Primeiras linguagens de programação</i>	32
2.3.1.1	Plankalkul	32
2.3.1.2	Fortran	33
2.3.1.3	Lisp	33
2.3.1.4	Algol	34
2.3.1.5	Java	34
2.3.2	<i>Níveis</i>	35
2.3.2.1	Linguagem de Máquina	35
2.3.2.2	Linguagens de baixo nível	35
2.3.2.3	Linguagem de Alto Nível	35
2.3.3	<i>Tipos de Linguagens de Programação</i>	36
2.3.3.1	Linguagens de Script	36
2.3.3.2	Linguagens Interpretadas	36
2.3.3.3	Linguagens compiladas	37
2.3.3.3.1	Compilador	37
3	CAPÍTULO 3	38

3.1 Interface do SISTEMA.....	38
3.1.1 <i>Servidor</i>	38
3.1.2 <i>Cliente.....</i>	47
3.2 Criptografia de mensagens.....	52
3.2.1 <i>Criptografia de mensagem.....</i>	53
3.2.2 <i>Descriptografia de mensagem</i>	55
3.3 Testes e validações	56
3.3.1 <i>Operações com matrizes</i>	56
3.3.2 <i>Criptografia das mensagens</i>	60
CONCLUSÃO	62
REFERÊNCIAS.....	64
APÊNDICE A – CLASSES PROJETO	65
APÊNDICE B – INTERFACE USUÁRIO	75
APÊNDICE C – MANUAL PROJETO.....	102

INTRODUÇÃO

Desde os primórdios das civilizações, o homem se preocupa em proteger suas informações. Sejam elas informações sobre sua família, identidade ou sua nação. E juntamente com essa vontade de proteger suas informações, existem aqueles que as tentam descobrir.

O ato de proteger as informações ficou conhecido como criptografar. Em grego: *kryptós*, significa "escondido", e *gráphein*, "escrita". Já o ato de desvendar o que estava protegido ficou conhecido como descriptografar.

Segundo Cavalcante (2004), a criptografia é a ciência que estuda formas de escrever mensagens codificadas. Trata-se de uma forma de fazer com que somente o receptor da mensagem seja capaz de compreendê-la e decifrá-la.

De acordo com pesquisadores, a criptografia teve início por volta de 1900 A.C., no antigo Egito. Criada pelo arquiteto Khnumhotep II, para proteger de ladrões informações sobre tesouros, que estavam em seus documentos. Porém, a cifra de César é a primeira forma de criptografia de informações que se tem registro. Ela foi desenvolvida pelo imperador Júlio César, com o objetivo de proteger as informações passadas a seus generais em guerra.

Antes do surgimento dos computadores, a criptografia era feita de forma manual ou mecânica. Uma das máquinas de criptografar que se pode tomar por exemplo é a máquina Lorenz SZ 40/42, que foi utilizada pelos nazistas durante a segunda guerra mundial. Ela permaneceu absoluta até o desenvolvimento do computador Colossus projetado por Turing, especialmente para descriptografar as suas mensagens.

Hoje, com os recursos disponíveis, os sistemas clássicos de criptografia seriam facilmente descobertos. Por isso se fez necessário o desenvolvimento de novas

técnicas de criptografia. Inúmeras técnicas já foram empregadas. Em 1929, Lester S. Hill utilizou pela primeira vez, conceitos de Álgebra Linear para codificar mensagens. Sua técnica futuramente ficou conhecida como cifra de Hill.

A cifra de Hill utiliza o método de substituição, baseado em transformações matriciais. Ela é uma classe de sistemas poligráficos, onde a mensagem é dividida por um conjunto n de letras, após, cada sistema é substituído por um conjunto n de letras já cifradas.

O principal objetivo da criptografia é permitir a transmissão de mensagens entre pessoas, não importando a rede que se está utilizando, podendo ela ser segura ou não.

Assim neste trabalho é abordada a técnica de criptografia com chaves simétricas. Para tal, será desenvolvido um algoritmo, na linguagem JAVA, que será capaz de proporcionar a troca segura de mensagens entre usuários de um chat. Para proporcionar essa segurança, serão utilizados princípios de Álgebra Linear para realizar a criptografia.

1 PROJETO PESQUISA

1.1 TEMA

Criptografia de um chat de mensagens privadas

1.1.1 Delimitação do Tema

Aplicar conceitos de cálculos de matrizes no desenvolvimento de algoritmo em Java, que permita a criptografia de mensagens compartilhadas entre usuários, de um chat online. O trabalho será realizado no período de agosto a dezembro de 2015, pelos acadêmicos Claudio Scheer, Gustavo Motta e Larissa Guder, do 2º semestre do curso de Bacharelado de Sistemas de Informação – SETREM, como Trabalho Interdisciplinar I, envolvendo as disciplinas de Álgebra Linear e Geometria Analítica, Algoritmo II e Metodologia da Pesquisa.

1.2 OBJETIVOS

1.2.1 Objetivo geral

Desenvolver um algoritmo com a linguagem de programação Java, aplicando conceitos de matrizes, para a criptografia de informações compartilhadas entre usuários de um chat privado online.

1.2.2 Objetivos específicos

Estudar os conceitos de matrizes e criptografia.

Desenvolver o algoritmo de criptografia.

Realizar testes do algoritmo na procura de eventuais falhas.

Elaborar um manual para utilização do mesmo.

Verificar se a criptografia realmente funciona.

1.3 JUSTIFICATIVA

Hoje em dia existem várias aplicações no mercado que permitem o compartilhamento de informações e dados online. Devido à facilidade e rapidez no envio e no recebimento de informações, cada vez mais pessoas utilizam esses serviços. Devido a essa grande procura, os dados, tanto de pessoas físicas como jurídicas, podem ser comprometidos.

Os usuários que compartilham as informações online, prezam por segurança e privacidade. Caso algum destes requisitos falhe, ou até mesmo a aplicação utilizada não leve em consideração algum destes requisitos, o prejuízo pode ser incalculável e até mesmo irreversível.

Se, por exemplo, os dados de uma determinada empresa que tem relação com o governo forem expostos ao público em geral, além do prejuízo financeiro para a empresa – que poderá ser incalculável, a exposição dos dados, pode afetar a relação empresa-governo, já que a mesma não tem condições de armazenar dados confidenciais em segurança.

Pessoas físicas também podem ser seriamente afetadas com a exposição de seus dados pessoais. Frequentemente pode ser visto na mídia notícias sobre pessoas que são vítimas desse problema. Um dos casos mais frequentes é o vazamento de dados de cartões de crédito, o que, dependendo do caso, acaba causado grandes problemas financeiros para a pessoa.

O uso da criptografia permite que as informações trocadas pelos usuários sejam completamente indecifráveis enquanto trafegam pela rede. Sendo assim, as más intenções de algumas pessoas serão barradas quando tiverem que descriptografar os dados captados. Por isso, o uso da criptografia se torna indispensável quando o assunto é o compartilhamento de informações online.

A criptografia, não apenas auxilia na parte social, econômica ou política da vida dos envolvidos na troca de informações, mas também contribui para um grande avanço científico e tecnológico da sociedade.

Com a atenção cada vez mais voltada para a segurança e privacidade dos usuários, resulta que cada vez mais pessoas utilizem os serviços disponibilizados para a troca de informações, devido a facilidade e rapidez.

Em Álgebra Linear e Geometria Analítica, com relação a matrizes, James Joseph Sylvester foi o primeiro a demonstrar a importância das matrizes em 1850. Alan Turing, em meados de 1940, durante a segunda guerra mundial, estudou formas de compartilhar mensagens secretas com segurança para seus aliados.

No entanto, existem algumas lacunas a serem preenchidas quanto ao desenvolvimento de algoritmos que considerem a segurança e a privacidade dos usuários, em relação a criptografia de informações na troca de mensagens.

1.4 PROBLEMAS DA PESQUISA

É possível desenvolver um algoritmo, em Java, utilizando conceito de criptografia com matrizes que seja capaz de proporcionar troca de informações com segurança e privacidade?

1.5 HIPÓTESES

- O uso da criptografia impossibilita o uso indevido dos dados compartilhados entre usuários.
- O algoritmo permitirá que os usuários tenham segurança nas suas mensagens.
- É possível criptografar mensagens com até 140 caracteres ou mais.

1.6 VARIÁVEIS

- O algoritmo.
- A linguagem *Java*.
- A plataforma *NetBeans*.
- Tamanho máximo de caracteres nas mensagens.

- Envio e recebimento de mensagens criptografadas.

1.7 METODOLOGIA

1.7.1 Métodos de Abordagem

Segundo (LOVATO, 2013) existem duas dimensões dos métodos de abordagem, a primeira se refere ao tipo de raciocínio que será utilizado na conclusão do trabalho, e a segunda faz referência ao uso ou não uso de números e de estatística.

No presente trabalho será feito uma análise sobre os métodos de criptografia já existentes, implementando o método mais confiável no trabalho, assim como aplicações que oferecem a troca de mensagens pela internet com segurança.

Os métodos de abordagem existentes são os métodos dedutivo, indutivo, qualitativo e quantitativo. No presente trabalho será usado os métodos dedutivo e qualitativo, no qual será analisado a segurança apresentada pelo algoritmo.

1.7.2 Métodos de Procedimento

Os métodos de buscas utilizados para o desenvolvimento do trabalho serão tanto de livros, como de consultas a internet, que se referenciam ao assunto do projeto.

A linguagem de programação utilizada foi Java. Usando como ferramenta a plataforma de desenvolvimento *NetBeans*.

Para realizar a criptografia dos dados serão utilizadas duas operações com matrizes: multiplicação de matrizes e transposição de matrizes. Para a inversão de matrizes será utilizado o método de Gauss.

A criptografia será realizada com princípios matemáticos de Álgebra Linear. Após o desenvolvimento, o algoritmo será testado pelo Wireshark, *software* utilizado para a validação de segurança e privacidade do algoritmo.

1.7.3 Técnicas

Para criptografia das mensagens será utilizada as chaves simétricas. A chave simétrica consiste em guardar a chave utilizada para a criptografia no próprio sistema,

ou seja, todas as chaves são privadas, sem a necessidade de compartilhamento de uma chave para descriptografar os dados compartilhados.

Para análise da segurança da criptografia foi usado o *software* Wireshark. Já para a validação dos cálculos utilizados no processo de criptografia, foi utilizado a ferramenta Microsoft Excel.

1.8 CRONOGRAMA

O cronograma apresenta um quadro com as datas pré-determinadas para início, meio e fim do projeto interdisciplinar.

Quadro 1 - Cronograma

Atividades	Meses				
	Ago	Set	Out	Nov	Dez
Obter conhecimentos sobre os assuntos relacionados a criptografia					
Tema do trabalho					
Desenvolvimento do Capítulo 1					
Desenvolvimento do Capítulo 2					
Entender os problemas do algoritmo					
Desenvolver o algoritmo					
Resolver os possíveis problemas					
Entregar Trabalho Interdisciplinar					
Apresentar Trabalho Interdisciplinar					

Legenda:

Previsto:  Realizado: 

1.9 ORÇAMENTO

O Orçamento envolve os gastos com a execução do projeto, entre eles estão gastos com cópias e encadernações.

Quadro 2 - Orçamento

Descrição	Valor Unitário	Unidades	Total (R\$)
Impressão Preto e Branco	0,12	500	R\$ 60,00
Impressão Colorida	1,20	75	90,00
Encadernação	2,40	10	R\$ 24,00
TOTAL			R\$ 174,00 ,

2 REFERENCIAL TEÓRICO

2.1 MATRIZES

Desde o surgimento das matrizes, no século II a.C, na China antiga, e em todo seu período de evolução, as matrizes se tornaram peça fundamental para a resolução de problemas que envolvem equações simultâneas lineares. Atualmente, ela deixou de ser usada somente nessa função. Ela passou a ser utilizada para imagens digitais, criptografia, por profissionais de engenharia, computação, físicos. Está presente em uma gama de profissões.

O termo MATRIZ surgiu em 1850 através do matemático James Joseph Sylvester. Porém, só se tornou de conhecimento comum graças a seu colega, Cayley, que através da publicação *Memoir on the Theory of Matrices*, realizada em 1858, apresentou o termo e demonstrou sua utilidade. Segundo ele, as matrizes eram um bloco retangular de valores, mas que não representam um determinante. Porém, é como se fosse uma matriz, a partir de qual é possível formar inúmeros sistemas de determinante, fixando um valor P , e escolhendo P linhas e P colunas. (CAYLEY, 1858)

2.1.1 Propriedades de matrizes

Segundo (KUERTEN, 2002), as matrizes são definidas como um agrupamento retangular de elementos, organizados por linhas e por colunas.

Os elementos de uma matriz podem ser números, expressões algébricas, como também outros caracteres, como de texto.

As matrizes geralmente são representadas por uma letra maiúscula e seus elementos por uma letra minúscula, acompanhados de um índice que representa a

sua localização do elemento na matriz através da sua linha e da sua coluna, respectivamente.

2.1.2 Definições

As matrizes são denominadas de acordo com suas características. A seguir estão citadas algumas, dentre as mais conhecidas.

2.1.2.1 Matriz Linha

É caracterizada por possuir apenas uma linha e n colunas.

$$\text{Exemplo: } A = [51 \ 43 \ 78] \quad (1)$$

2.1.2.2 Matriz Coluna

Possui apenas uma única coluna, e contém n linhas.

$$\text{Exemplo: } A = \begin{bmatrix} 3 \\ 5 \\ 7 \\ 8 \end{bmatrix} \quad (2)$$

2.1.2.3 Matriz Quadrada

Considera-se uma matriz quadrada quando a mesma possuir o mesmo número de linhas e de colunas. As matrizes quadradas são denominadas matrizes de ordem, onde o número de ordem será igual ao seu número de linhas e de colunas.

$$\text{Exemplo: } A = \begin{bmatrix} 9 & 6 \\ 4 & 5 \end{bmatrix} \quad (3)$$

A matriz acima será de ordem 2.

2.1.2.4 Matriz Nula

A matriz é considerada nula quando todos os seus elementos forem nulos (iguais a zero).

$$A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad (4)$$

2.1.2.5 Matriz Diagonal

Para ser considerada uma matriz diagonal, esta deve ser quadrada e necessita ter seus elementos nulos, salvo os presentes na diagonal principal, os quais não devem ser todos iguais ao número 1.

$$A = \begin{bmatrix} 4 & 0 & 0 \\ 0 & 12 & 0 \\ 0 & 0 & 2 \end{bmatrix} \quad (5)$$

2.1.2.6 Matriz Identidade

Matriz quadrada que possui todos os seus elementos nulos, salvo os seus elementos presentes na diagonal principal, os quais devem ser iguais a 1.

$$I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (6)$$

Assim, para uma matriz identidade segue-se a regra:

$$I_n[a_{ij}], \quad a_{ij} = \begin{cases} 1 & \text{se } i = j \\ 0 & \text{se } i \neq j \end{cases} \quad (7)$$

2.1.2.7 Matriz Transposta

Para obter a transposta de uma matriz, que fica conhecida com \mathbf{A}^t , é necessário trocar de posição as linhas com as colunas.

$$A = \begin{bmatrix} 5 & 2 \\ 9 & 15 \\ 3 & 1 \end{bmatrix} \quad \text{então} \quad A^t = \begin{bmatrix} 5 & 9 & 3 \\ 2 & 15 & 1 \end{bmatrix} \quad (8)$$

Assim sendo, uma matriz A que antes era de ordem $c \times v$, terá, com \mathbf{A}^t , uma ordem $v \times c$.

2.1.2.8 Matriz Triangular Superior

Para ser considerada uma matriz triangular superior, a matriz necessita ser quadrada, e todos os elementos que estiverem abaixo da diagonal principal precisam ser nulos. Para tal, segue-se a seguinte regra:

$$a_{ij} = 0 \text{ se } \{i > j\},$$

$$A = \begin{bmatrix} 14 & 56 & 5 \\ 0 & 9 & 23 \\ 0 & 0 & 7 \end{bmatrix} \quad (9)$$

2.1.2.9 Matriz Triangular Inferior

Ao contrário da matriz triangular superior, a matriz triangular inferior tem nulos todos os elementos acima da diagonal principal.

Segue a seguinte regra:

$$a_{ij} = 0 \text{ se } \{i < j\}$$

$$A = \begin{bmatrix} 14 & 0 & 0 \\ 6 & 12 & 0 \\ 3 & 8 & 9 \end{bmatrix} \quad (10)$$

2.1.2.10 Matriz Escalar

Em uma matriz escalar, todos os elementos da diagonal principal possuem o mesmo valor e os elementos acima e abaixo da diagonal principal são nulos, seguindo a seguinte regra:

$$a_{ij} = n \text{ e se } a_{ij} = 0 \text{ se } i \neq j$$

$$A = \begin{bmatrix} 6 & 0 & 0 \\ 0 & 6 & 0 \\ 0 & 0 & 6 \end{bmatrix} \quad (11)$$

2.1.3 Operações com Matrizes

As matrizes possibilitam realizar operações entre elas, gerando assim uma nova matriz com o resultado final.

2.1.3.1 Soma de Matrizes

Teorema: Sejam **A**, **B**, **C** e **D** matrizes de ordem $m \times n$

Duas ou mais matrizes somente poderão ser somadas, se tiverem a mesma ordem, ou seja, o número de linhas e colunas das duas matrizes deverá obrigatoriamente ser igual. Para realizar a soma de duas ou matrizes, existem algumas propriedades que deverão ser observadas.

2.1.3.1.1 Associação

Um exemplo de associação de matrizes é:

$$a_{ij} + (b_{ij} + c_{ij}) = (a_{ij} + b_{ij}) + c_{ij} \rightarrow A + (B + C) = (A + B) + C \quad (12)$$

2.1.3.1.2 Comutação

Na comutação, independente da ordem das matrizes, os valores não serão alterados, ou seja

$$a_{ij} + b_{ij} = b_{ij} + a_{ij} \rightarrow A + B = B + A \quad (13)$$

2.1.3.1.3 Elemento Neutro ou Matriz Neutra

A matriz em que todos os elementos são 0s é chamada de matriz nula, ou elemento neutro $m \times n$.

Seja $U = u_{ij}$. Então $A + U = A$, somente se $a_{ij} + u_{ij} = a_{ij}$ o que verifica se e somente se $u_{ij} = 0$. Logo $U = 0$.

2.1.3.1.4 Matriz Negativa

Para cada matriz A , $m \times n$, existe uma única matriz D , $m \times n$, tal que $A + D = 0$. Denotaremos D por $-A$. Esta matriz recebe o nome de inversa aditiva ou negativa de A . Seja $D = (d_{ij})$, então $A + D = 0$, somente se $a_{ij} + d_{ij} = 0 \leftrightarrow d_{ij} = -a_{ij} \leftrightarrow D = -A$.

2.1.3.2 Igualdade de Matrizes

Duas matrizes são consideradas iguais se as ordens das mesmas forem iguais e se todos os elementos correspondentes tiverem o mesmo valor.

2.1.3.3 Multiplicação por um escalar

Para realizar a multiplicação por um escalar é necessário multiplicar cada elemento da matriz por esse mesmo escalar. A nova matriz terá as mesmas dimensões da matriz anterior.

$$A = \begin{bmatrix} 3 & 25 \\ 10 & 4 \end{bmatrix} \quad A * 2 = \begin{bmatrix} 6 & 50 \\ 20 & 8 \end{bmatrix} \quad (14)$$

2.1.3.4 Multiplicação de Matrizes

Para ser possível realizar a multiplicação de duas matrizes, A e B , o número de colunas da matriz A deverá ser igual ao número de linhas da matriz B . A matriz gerada com o resultado da multiplicação terá tamanho igual ao de linhas da matriz A e de colunas da matriz B . Assim, o produto das matrizes $A = (a_{ij}) m \times p$ e $B = (b_{ij}) p \times n$ é a matriz $C = (c_{ij}) m \times n$, onde cada elemento c_{ij} é obtido por meio da soma dos produtos

dos elementos correspondentes da *i*-ésima linha de **A** pelos elementos da *j*-ésima coluna de **B**.

A matriz resultante será igual a:

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{bmatrix} * B = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} \rightarrow \begin{bmatrix} a_{11} * b_{11} + a_{12} * b_{21} & a_{11} * b_{12} + a_{12} * b_{22} \\ a_{21} * b_{11} + a_{22} * b_{21} & a_{21} * b_{12} + a_{22} * b_{22} \\ a_{31} * b_{11} + a_{32} * b_{21} & a_{31} * b_{12} + a_{32} * b_{22} \end{bmatrix} \quad (15)$$

Porém, para a multiplicação de matrizes não é válida a propriedade comutativa. Ou seja: $\mathbf{A} * \mathbf{B} \neq \mathbf{B} * \mathbf{A}$.

2.1.3.5 Potências de Matriz

Para encontrar o resultado de uma potenciação de matrizes segue-se as mesmas definições da multiplicação de matrizes. E para ser possível o cálculo, a matriz deve ser quadrada.

Pode-se definir uma matriz quadrada **A**, de ordem *m*: $A^0 = I_m$, sendo $A \neq 0$.

$$A^n = A.A.A \cdots A \text{ onde } n > 0. \quad (16)$$

2.1.4 Determinantes

Segundo (MONTEIRO, 2001), existe mais de uma maneira para se calcular o determinante de uma matriz.

2.1.4.1 Determinante de matriz quadrada de segunda ordem

Para chegar ao determinante de uma matriz quadrada de ordem igual a dois, é necessário realizar a subtração do produto dos elementos da diagonal principal, pelo produto dos elementos da diagonal secundária.

$$A_{mn} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}_{2 \times 2} \quad (17)$$

A fórmula utilizada para o cálculo de determinante de segunda ordem é:

$$Det = a_{11} \times a_{22} - a_{21} \times a_{12} \quad (18)$$

2.1.4.2 Determinante de matriz quadrada com ordem igual a três

2.1.4.2.1 Regra de Sarrus

Esta regra deve-se a Pierre Frédéric Sarrus (1798-1861), a qual se aplica para calcular o determinante de matrizes de ordem igual a três. Para esta regra se faz necessário copiar a primeira e a segunda coluna ao lado da matriz, e após soma-se os produtos dos elementos das diagonais principais subtraindo se a soma dos produtos das diagonais secundárias.

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \left| \begin{array}{cc} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{23} \end{array} \right|$$

$$\text{Det} = (a_{11} \times a_{22} \times a_{33} + a_{12} \times a_{23} \times a_{31} + a_{13} \times a_{21} \times a_{23}) - (a_{31} \times a_{22} \times a_{13} + a_{32} \times a_{23} \times a_{11} + a_{33} \times a_{21} \times a_{12}) \quad (19)$$

2.1.4.3 Determinante de matriz com ordem superior a três

Para o cálculo do determinante de matrizes com ordem igual ou superior a três, é utilizado o teorema de Laplace ou então o método de eliminação de Gauss.

2.1.4.3.1 Teorema de Laplace

Pierre-Simon Laplace, que viveu entre 1749-1827, foi quem desenvolveu o método que ficou conhecido como o teorema de Laplace.

O teorema consiste em selecionar uma linha ou coluna da matriz e então realizar a soma dos produtos dos elementos da coluna ou linha selecionada pelos seus respectivos cofatores.

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

$$\det A = a_{11} \cdot A_{11} + a_{21} \cdot A_{21} + a_{31} \cdot A_{31}, \text{ onde } A_{ij} \text{ é o cofator do elemento } a_{ij}.$$

$$A_{ij} = (-1)^{i+j} \cdot D_{ij} \quad (20)$$

2.1.5 Matriz Inversa

São chamadas matrizes inversas, aquelas que possuem o determinante diferente de zero, e que sejam quadradas. A representação de uma matriz invertível é dada por **A-1** como matriz inversa de **A**.

O cálculo da inversa de uma matriz, pode ser realizado por diferentes métodos, neste trabalho será abordado o método de escalonamento, conhecido também como Gauss-Jordan.

Para o cálculo da inversa, deve-se adicionar paralelamente a matriz principal à matriz identidade de mesma ordem.

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (21)$$

Em seguida, são aplicadas sucessivas operações elementares sobre as linhas da matriz que se deseja inverter, para transformá-la em uma matriz identidade, aplicando também, as mesmas operações à matriz identidade. Ao término do processo, a matriz identidade se transformou na matriz inversa procurada.

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} \quad (22)$$

Para que a realização de criptografia com matrizes seja mais rápida e pratica, ou até mesmo para que a criptografia seja possível, são necessários alguns conhecimentos básicos acerca do que são e para que são utilizadas as linguagens de programação.

2.2 CRIPTOGRAFIA

A criptografia existe a centenas de anos. Ela foi elaborada para se ter um meio mais seguro de transmitir informações sigilosas. Não se tem uma data correta da criação da criptografia, mas há indícios de sua utilização pelos egípcios aproximadamente 2 mil anos a.C., mas a ideia mais aceita é de sua utilização pelos povos mesopotâmios pelos anos de 500 a 600 a.C.

Existe dois tipos de criptografia mais utilizadas, a assimétrica e a simétrica.

(TANENBAUM, 2003), explica que a criptografia é usada para manter as informações importantes confidenciais, garantindo assim a integridade e autenticidade da informação, fazendo com que permaneça secreta para aqueles que não tem acesso as suas chaves.

2.2.1 Algoritmos Simétricos

Um algoritmo simétrico usa de apenas uma chave, tanto para encriptar quanto para descriptar uma mensagem. Então todos os algoritmos criados desde a cifra de Cesar até o ano de 1976, eram algoritmos simétricos.

Segundo (FIARRESGA, 2010), a proteção de uma cifra simétrica é determinada pelo tamanho da sua chave, sendo considerado uma chave de 40 Bits fraca, e uma maior do que 128 Bits as mais fortes e seguras.

O maior problema da chave simétrica é a necessidade de ser compartilhada de uma forma segura, que não comprometa toda a criptografia. A criptografia simétrica não permite confirmação de identidade, nem mesmo do escritor ou receptor.

Estas técnicas criptográficas foram muito utilizadas militarmente graças a possibilidade de manter alguma informação capturada segura, mesmo que fosse interceptada, ela passaria despercebida pelos inimigos já que, para alguém sem o conhecimento do assunto, não passaria de letras ou números embaralhados. O primeiro grande líder militar da história a utilizar da criptografia foi Júlio Cesar, um dos maiores imperadores romanos. Utilizando de conhecimentos lógicos, ele mesmo criou uma cifra criptográfica que é conhecida até os dias de hoje como cifra de Cesar, sendo utilizada mesmo após sua morte pelos seus predecessores.

Segundo (TANENBAUM, 2003), para os profissionais no assunto é diferenciado cifras de códigos. A cifra é caracterizada pela troca dos caracteres por outros caracteres, ou pela troca de bits por bits, considerando somente a codificação da mensagem, não seu significado. Já o código transcreve uma determinada palavra, por uma outra, criada a partir do código, podendo ser usado símbolos no lugar de uma palavra, mas os códigos não são mais utilizados nos dias atuais.

Como mencionado, a cifra de César foi a primeira cifra utilizada na história, e seu método de encriptação consistia em escolher um determinado número, usando este número trocava-se as letras subsequentemente. Assim, por exemplo, usando o número 4 para encriptar a mensagem "CRIPTOGRAFIA", a mensagem encriptada seria "FULSWRJUDILD". Por mais simples que pareça, demorou quase 800 anos para esta cifra ser quebrada.

A principal limitação da cifra de César era seu número limitado de 26 possíveis chaves, sendo que a frequência de suas letras poderia ser facilmente decifrada.

No intuito de diminuir a variação na frequência das letras, por volta do século XV, foram criadas as cifras poli alfabéticas, funcionando da seguinte maneira: era escolhido uma palavra-chave combinada antecipadamente entre o remetente e o destinatário, que era transformada em números entre 1 e 26, dependendo de sua ordem alfabética. Com os números da palavra-chave se convertia todas as letras da mensagem sequencialmente, usando os números transcritos da mesma maneira que a cifra de César. Desta maneira, cada número da palavra-chave se torna uma chave sequencial. Neste método, o que determina a força da criptografia é o tamanho da palavra-chave.

Mesmo a cifra poli alfabética diminuindo a frequência das letras, ela ainda não era perfeita, e durante mais quatro séculos não se encontrava uma forma mais eficiente, até a criação da cifra de chave única no século XIX. Ela se baseia na geração de números entre 1 e 26 aleatoriamente, sendo que a quantidade de números deverá ser maior que a quantidade de letras da mensagem, tornando assim a aleatoriedade impossível de descriptografar sem a sua chave única.

Para (TANENBAUM, 2003), apenas quatro grupos de pessoas tiveram contribuições para o avanço da criptografia. Os grupos são formados por militares, diplomatas, historiadores e amantes, dando maior desenvolvimento a criptografia os militares, já que estes conhecimentos foram amplamente utilizados durante toda a história conhecida.

Para (TANENBAUM, 2003):

As mensagens a serem criptografadas, conhecidas como texto simples, são transformadas por uma função que é parametrizada por uma chave. Em seguida, a saída do processo de criptografia, conhecida como texto cifrado, é transmitida, normalmente através de um mensageiro ou por rádio. Às vezes, o intruso pode não só escutar o que se passa no canal de comunicação (intruso passivo), como também pode gravar mensagens e reproduzi-las mais tarde, injetar suas próprias mensagens ou modificar mensagens legítimas antes que elas cheguem ao receptor (intruso ativo). (TANENBAUM, 2003, p. 546).

Neste contexto, a criptografia nada mais é do que transformar, por meio de algum algoritmo, um texto comum em um texto cifrado, de uma forma que os únicos que tenham a permissão ou a chave para decifrar e compreender a informação são aqueles que estão interligados com a mensagem, ou seja aquele que escreveu e aquele que irá receber a mensagem.

Então para (TANENBAUM, 2003), o método para solucionar uma mensagem criptografada se chama criptoanálise, já o de criar uma mensagem se chama criptografia. Os dois métodos juntos são chamados de criptologia.

A criptografia tem várias finalidades, entre elas está a de proteger as informações das pessoas, desde simples e-mails a transações bancárias. Qualquer atividade que se tem na internet envolve criptografia. Deste modo, ela está sempre presente, mesmo sem notarmos.

Assim como existem pessoas que buscam criar criptografias mais seguras (criptógrafos), também há aqueles que tentam derrubar os sistemas criptográficos (criptoanalistas).

Na segurança, as entidades americanas responsáveis pelo uso da criptografia impediram durante muitos anos que os algoritmos criptográficos fossem distribuídos mundialmente, já que tinham o conceito de que, monopolizando os algoritmos, dificultaria a vida dos criptoanalistas, o que ficou conhecido como criptografia privada. Esse método era chamado de segurança da obscuridade, mas pelo princípio de Kerckhoff os algoritmos devem permanecer públicos, e a única parte secreta seria a chave do algoritmo, porque seu princípio mostra que a segurança está unicamente na chave, que quanto maior, mais segura a criptografia será.

O que define o quão forte uma criptografia é, são o número de bits utilizados para gerar a chave criptográfica. Uma chave criptográfica de 8 bits tem a possibilidade de gerar 256 chaves diferentes, sendo muito fácil alguém especializado descriptografar esta chave. Já uma chave com 128 bits é considerada segura por poder gerar milhões de combinações de chaves diferentes, já existindo chaves com mais de 256 bits, o que acaba tornando uma tarefa quase impossível para alguém mal-intencionado descriptografar.

2.2.2 Algoritmos Assimétricos

Esta forma de criptografia foi desenvolvida na década de 70 por *Clifford Cocks*. Seu diferencial é o uso de duas chaves diferentes, uma pública e outra privada. A pública é usada pelo escritor para criptografar, e a privada utilizada pelo seu destinatário pra descriptografar a mensagem.

As técnicas mais usadas para fazer criptografia envolvem conceitos de Álgebra Linear, a partir do cálculo com matrizes. Um exemplo seriam duas matrizes, a original e a sua inversa, onde o escritor usa a matriz original como chave para criptografar a mensagem e o receptor da mensagem usa a inversa da matriz como chave para decodificar a mensagem. Esta forma impede a descoberta da frequência de escrita, o que dificulta a tarefa de um decodificador.

Para criar um exemplo de mensagem que será criptografada, é preciso transformar o alfabeto em números de 1 a 29. Para facilitar, a primeira etapa é a troca das letras originais pelos números. Usando então a seguinte tabela de referência:

Quadro 3 – Valores para conversão de letras

A	B	C	D	E	F	G	H	I	J
1	2	3	4	5	6	7	8	9	10
K	L	M	N	O	P	Q	R	S	T
11	12	13	14	15	16	17	18	19	20
U	V	W	X	Y	Z	.	,	#	
21	22	23	24	25	26	27	28	29	

Como exemplo será utilizada a seguinte frase “Meios de Criptografia.”.

Quadro 4 – Exemplo de mensagem convertida para números

M	E	I	O	S	#	D	E	#	C	R
13	5	9	15	19	29	4	5	29	3	18
I	P	T	O	G	R	A	F	I	A	.
9	16	20	15	7	18	1	6	9	1	27

Utilizando a tabela esta é a mensagem transformada em números “13, 5, 9, 15, 19, 29, 4, 5, 29, 3, 18, 9, 16, 20, 15, 7, 18, 1, 6, 9, 1, 27”. Como exemplo usaremos uma matriz 2x2 que é representada pela letra C, já que é a chave para codificação da mensagem, a matriz D representa a mensagem com seus caracteres transformados

em números, sendo que ela precisa ter o mesmo número de linhas que a matriz C, não importando o número de colunas, M é a matriz já criptografada. Para isso utiliza-se a fórmula C.D:

$$C = \begin{bmatrix} 3 & 1 \\ 2 & 1 \end{bmatrix} \text{ e } D = \begin{bmatrix} 13 & 5 & 9 & 15 & 19 & 29 & 4 & 5 & 29 & 3 & 18 \\ 9 & 16 & 20 & 15 & 7 & 18 & 1 & 6 & 9 & 1 & 29 \end{bmatrix}$$

$$C.D = \begin{bmatrix} 48 & 31 & 47 & 60 & 64 & 105 & 13 & 21 & 96 & 10 & 83 \\ 35 & 26 & 38 & 45 & 45 & 76 & 9 & 16 & 67 & 7 & 65 \end{bmatrix} \quad (21)$$

Desta maneira a mensagem se torna criptografada, sendo difícil de criar uma solução para descriptografar a mesma, sem ter exatamente a inversa da matriz C.

$$C^{-1} = \begin{bmatrix} 1 & 1 \\ 3 & 2 \end{bmatrix} \quad (22)$$

Para reverter o processo é utilizado a formula $D = C^{-1}.M$.

2.3 LINGUAGENS DE PROGRAMAÇÃO

Linguagens nada mais são do que uma forma de permitir a comunicação entre a pessoa e computador. Todas linguagens de programação criadas até hoje tem o mesmo objetivo: solucionar problemas. (PUGA e RISSETTI, 2008)

Na Babilônia algoritmos já eram escritos utilizando a linguagem natural. Os algoritmos eram utilizados para descrever ações do cotidiano das pessoas. A ação era feita manual, mas a descrição ajuda as pessoas a se orientarem quando faziam tal ação.

O primeiro programa foi criado por Augusta Ada. Augusta Ada e Charles Babbage trabalhavam juntos em várias ideias de Babbage. Charles criou a máquina diferencial, com o objetivo de realizar cálculos com polinômios. Ada por sua vez desenvolveu um programa com o objetivo de calcular os números de Bernoulli utilizando a máquina diferencial, porém este programa nunca chegou a ser testado por que a máquina também não chegou a ser concluída.

2.3.1 Primeiras linguagens de programação

2.3.1.1 Plankalkul

Konrad Zuse desenvolveu um ábaco mecânico, controlado por pinos de metal e por correias, isso em 1936.

A programação era feita através de fitas perfuradas. Zuse deu o nome de Z1. Em 1945 o projeto de criar uma linguagem de programação foi concluído e chamado de *Plankalkul*. O projeto continha ideias revolucionárias como por exemplo tipos de dados. (SEBESTA, 2010).

Zuse teve outras contribuições no desenvolvimento de programas, como ordenação de dados, busca em grafos, análise sintática, entre outros.

2.3.1.2 Fortran

Programar ainda era um grande desafio. Os programadores tinham que trocar vários fios de lugar, ou o trabalho era perfurar cartões que seriam executados pelas máquinas. Na definição de uma variável, por exemplo, os programadores tinham que saber exatamente a posição onde estava aquele valor.

Mas era preciso poupar tempo dos programadores, então surgiu o *Fortran*. O projeto foi liderado por John Backus e levou dois anos para ser concluído.

Com a chegada do *Fortran* era possível criar programas com mais agilidade, pois o compilador permitia que os códigos fossem gerados com mais qualidade.

Com a chegada do *Fortran II*, a compilação de programas se tornou mais prático. Pois o compilador permitia compilar um módulo inteiro, e não apenas um programa. (SEBESTA, 2010)

2.3.1.3 Lisp

John McCarthy trabalhava com Inteligência Artificial, e as operações que ele precisava o *Fortran* não suportava.

Então uma linguagem de programação que possibilitava o uso de expressões condicionais, recursão e coletor de lixo (no momento em que uma variável não era mais utilizada, o próprio compilador limpava a mesma, não necessitando mais o programador fazer esta ação).

Lisp é a linguagem mais popular quando o assunto é Inteligência Artificial. A partir do *Lisp* surgiram muitas outras linguagens funcionais, e muitas ideias implementadas no *Lisp* são usadas até hoje, como o coletor de lixo. (PUGA e RISSETTI, 2008)

2.3.1.4 *Algol*

Em 1957 várias linguagens de programação estavam surgindo, e não existia um padrão geral para elas, cada empresa universidade tinham seus padrões.

Em 1958 foi criado um comitê internacional para o desenvolvimento do projeto. Depois do desenvolvimento do projeto, quase todas as linguagens que vieram depois do *Algol*, se utilizaram de seus padrões, como: blocos delimitadores, escopo de bloco para variáveis locais, tipos de variáveis definidos estaticamente, controles de fluxos (*if* e *else's*) alinhados e alocação da memória de forma dinâmica.

As linguagens anteriores ao *Algol* utilizavam, para controle de fluxo do algoritmo, o *goto* (vai-para). O *Algol* não se utilizava mais disso, o que acabou gerando muita polemica, pois, muitos programadores achavam difícil trabalhar sem o *goto*. Hoje a maioria das linguagens não utilizam este recurso.

O *Algol* não obteve um grande sucesso como o esperado, algumas das razões era que a linguagem foi considerada complicada e não se tinha suporte corporativo ou governamental. (SEBESTA, 2010)

2.3.1.5 *Java*

Criado em 1991 pela empresa *Sun*, a qual hoje pertence a *Oracle*. A ideia principal do projeto *Green*, era de permitir que os softwares pudessem ser executados em vários aparelhos eletrônicos.

Java foi a primeira linguagem na qual era possível interagir com dispositivos portáteis e com produtos eletrônicos de consumo.

O *Java* baseia-se no *C++*. As maiores vantagens da linguagem é a sua portabilidade, segurança, suporta a orientação a objeto, dinamismo, alta performance e roda independentemente da plataforma.

Java também se utiliza do coletor de lixo e se utiliza de pacotes. O principal objetivo do *Java* não é ser uma linguagem de pesquisa, mas sim uma linguagem voltada para a produção rápida e segura.

2.3.2 Níveis

As linguagens de programação podem ser apresentadas de variadas formas.¹

Os níveis de programação levam em consideração a proximidade com que o algoritmo escrito trabalha com o processamento do computador, ou seja, quanto mais perto do hardware o programador desenvolver seu algoritmo, mais baixo será o nível da linguagem de programação.

2.3.2.1 *Linguagem de Máquina*

O computador executa várias instruções que ficam armazenadas em sua memória na forma de código binário. Neste nível de programação o algoritmo é escrito utilizando código binário.

O código binário é formado somente pelos números 0 e 1. O número 0 representa a ausência de energia, enquanto o número 1 representa a presença de energia.

2.3.2.2 *Linguagens de baixo nível*

É também chamado de linguagem de montagem, ou linguagens *assembly*. Esta linguagem é mais utilizada quando determinados programas necessitam do contato direto com o *hardware* da máquina. Para estes casos a linguagem de baixo nível é extremamente útil.

As linguagens de baixo nível trabalham muito próximas da linguagem de máquina e um pouco mais distantes das linguagens de alto nível.

2.3.2.3 *Linguagem de Alto Nível*

Além da grande complexidade para desenvolver programas em linguagem de baixo nível, o programador deve ter profundos conhecimento acerca de como é o funcionamento da parte física e elétrica do computador.

Para facilitar o desenvolvimento de programas, não precisando utilizar linguagens de baixo nível, foi criado as linguagens de alto nível.

As linguagens de alto nível têm o objetivo de permitir com que os algoritmos criados estejam o mais perto possível de linguagem humana. Além disso, facilitam a

¹ BATISTA, Guilherme 2009. Nota de aula: Linguagens de Programação.

utilização da memória do computador, ou seja, o programador se preocupa apenas com o controle do algoritmo e com a entrada e saída dos dados (*input/output*), não precisando se preocupar, por exemplo, com a limpeza da memória.

O único ponto em que o programador deve realmente se preocupar é com a sintaxe do algoritmo. Uma linguagem de alto nível se utiliza de um alfabeto contendo letras, números e símbolos, permitindo assim a compreensão humana. As linguagens de alto nível se utilizam de algumas palavras predefinidas, mas permite que o programador construa suas próprias palavras.

Um exemplo disso pode ser a declaração de uma variável. O nome dado a esta variável fica a critério do programador. O compilador não irá guardar este nome de variável, este nome definido pelo programador, será apenas para que ele se oriente em seu algoritmo. Quando uma variável é declarada, esta variável irá armazenar um endereço de memória onde o valor será guardado até o fim da execução do algoritmo.

Sendo assim, o programador não se preocupa em que lugar da memória será armazenado este valor, ele apenas terá de saber qual o nome definido a esta variável em seu algoritmo, ou seja, ele apenas vai se preocupar com a sintaxe do algoritmo, deixando a parte de baixo nível com o compilador utilizado pela linguagem.

2.3.3 Tipos de Linguagens de Programação

As linguagens de programação também são divididas por tipos.

2.3.3.1 Linguagens de Script

As linguagens de script, são executadas dentro de outros programas, sendo assim, não ficam restritas a estes ambientes, podem ser executadas independentemente da linguagem utilizada.

Alguns exemplos desta linguagem são: *SQL*, *JavaScript* e *Lua*.

2.3.3.2 Linguagens Interpretadas

O código fonte é interpretado por uma máquina virtual para depois ser executado pela máquina, diferente da linguagem compilada, a linguagem interpretada, não gera outro programa em código de máquina, apenas interpreta o código fonte.

Um exemplo desta linguagem é o Java. O código fonte é primeiramente transformado para *bytecodes* e depois é interpretado pela máquina. Outros exemplos são: *C#*, *PHP* e *Python*.

2.3.3.3 Linguagens compiladas

Para o programa ser executado na máquina, primeiramente seu código fonte necessita de ser traduzido. Para efetuar a tradução dos códigos é utilizado o compilador.

2.3.3.3.1 Compilador

Para a execução de um programa no computador, é necessário que os códigos estejam na linguagem de máquina, isto por que esta é a única linguagem que o computador consegue interpretar.

Se um programa for escrito sem a utilização de linguagem de máquina, - algoritmos escritos baseando-se na linguagem humana, como o computador irá executar estas instruções? Pois bem, para isso cada linguagem possui o seu compilador.

A função do compilador é transformar o código escrito pelo programador, em código binário (linguagem de máquina), ou seja, o compilador cria um programa igual ao escrito pelo programador, porém em código de máquina. Após esta operação concluída, o computador consegue executar o programa, e assim executar as ações previstas no algoritmo.

Se a compilação apresentar algum erro, a compilação e parada, e o erro é mostrado, ajudando o programador a corrigi-lo. Um dos erros mais comuns é quando alguma das instruções do algoritmo, não está legível para o compilador. Após a correção, será possível a compilação novamente.

3 CAPÍTULO 3

No presente capítulo será detalhado o desenvolvimento de trabalho passo a passo.

3.1 INTERFACE DO SISTEMA

A interface do sistema é onde permite com que os usuários realizem suas ações dentro de sistema. O presente trabalho é dividido em 2 principais telas, a Tela do Servidor e a tela de Clientes.

3.1.1 Servidor

Para que haja a possibilidade de conversação entre usuários do chat, um usuário deve ser o responsável por criar o *chat*. Como demonstra a figura 1, a página para a inicialização de um novo grupo de conversa.

Figura 1 - Tela inicial para servidor

The screenshot shows a web application window titled "Inbox". In the top right corner, there is a "Sair" button. On the left side, there is a vertical menu with two buttons: "Conectar Chat" and "Novo Chat", with "Novo Chat" being the active selection. To the right of the menu, the text "Num. conexões permitidas" is displayed above a text input field containing the number "2". To the right of the input field is a small up/down arrow control. At the bottom right of the main content area, there is a "Criar" button.

A página inicial para a criação de um *chat* conta com três ações que o usuário pode estar executando.

A primeira ação é chamada no momento em que o usuário selecionar a opção “Sair”. Esta opção apenas finalizara a execução do *chat*.

A figura 2, demonstra a opção descrita acima em destaque.

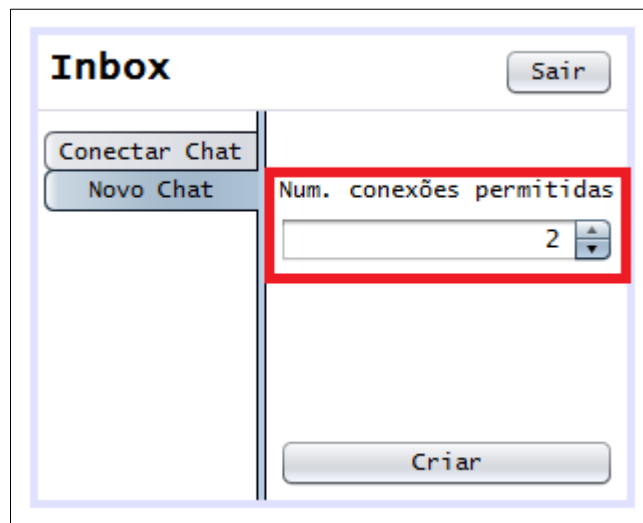
Figura 2 - Opção sair do servidor



A segunda opção disponível, como demonstra a figura 3, tem o objetivo de definir quantas conexões o *chat* possibilitará. Desta forma o usuário que estará responsável pela criação do chat, terá a opção de disponibilizar apenas a quantidade de conexões realmente necessárias.

Por exemplo, se o *chat* for privado, ele permitirá apenas uma conexão, assim ninguém mais terá acesso ao *chat*. A quantidade de conexões disponíveis deve estar entre 1 e 20, sendo que 20 é o máximo de pessoas que poderão estar se conectando ao *chat*. A quantidade de conexões disponíveis poderá ser alterada durante a execução do programa, conforme é mostrada na figura 9.

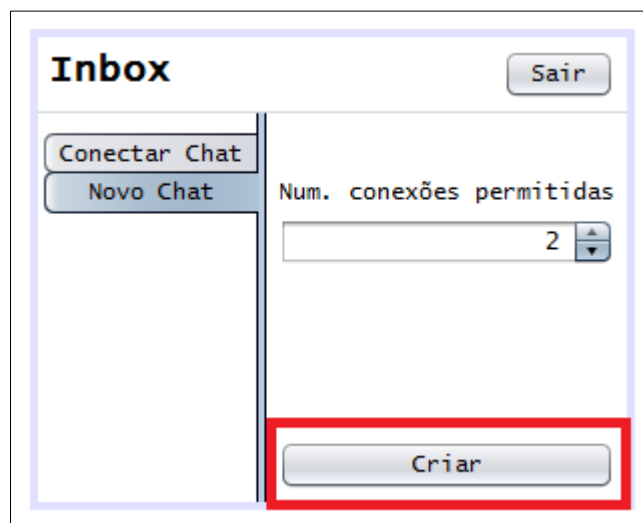
Figura 3 - Configurar número conexões permitidas no servidor



The screenshot shows a window titled "Inbox" with a "Sair" button in the top right. On the left, there are two buttons: "Conectar Chat" and "Novo Chat". The "Novo Chat" button is highlighted. To the right of these buttons, there is a label "Num. conexões permitidas" above a text input field containing the number "2". A red rectangle highlights the input field and the label. At the bottom right, there is a "Criar" button.

Com a quantidade de conexões disponíveis já definida, basta o usuário deve selecionar a opção “Criar”, destacada na figura 4.

Figura 4 - Opção criar do servidor

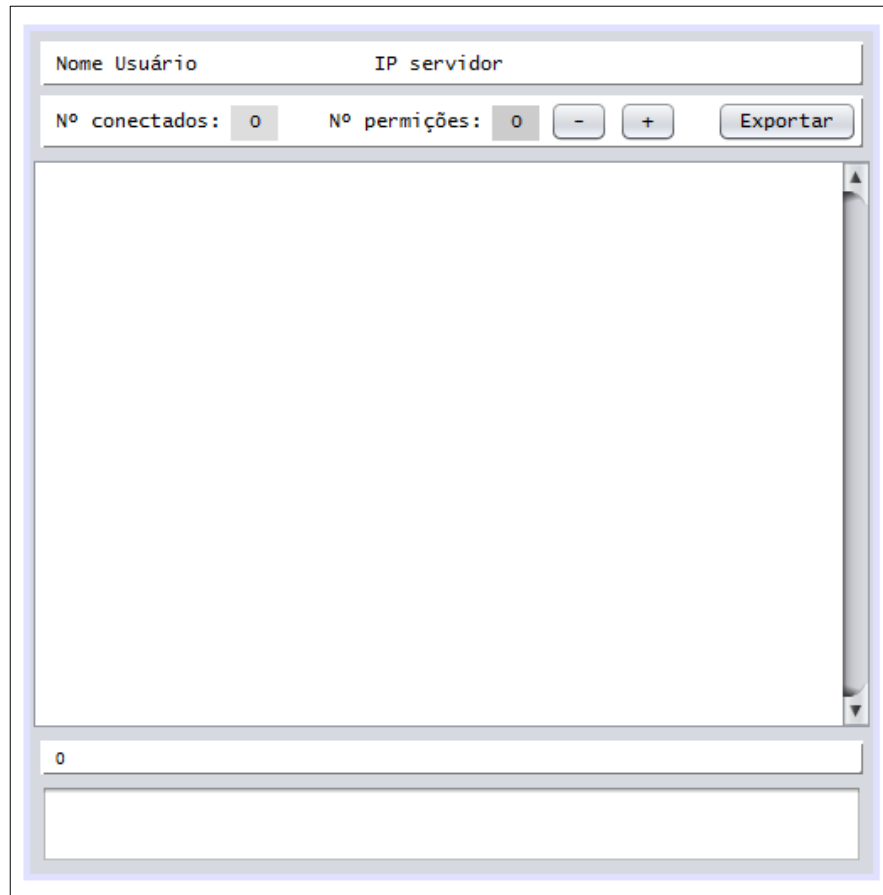


This screenshot is identical to the previous one, but with a red rectangle highlighting the "Criar" button at the bottom right instead of the input field.

Após selecionada a opção de criar o chat, é iniciado uma serie de validações internas do sistema. Por exemplo, não é permitido criar dois *chats* na mesma rede, usando a mesma porta para se comunicarem. Caso isso ocorra, ou caso haja quaisquer outros erros durante a criação do *chat*, o usuário é notificado e a tela demonstrada na figura 5, não é iniciada.

Se nenhum erro for encontrado, a tela da figura 5 é iniciada, e a tela da figura 1, é fechada.

Figura 5 - Tela do servidor criado

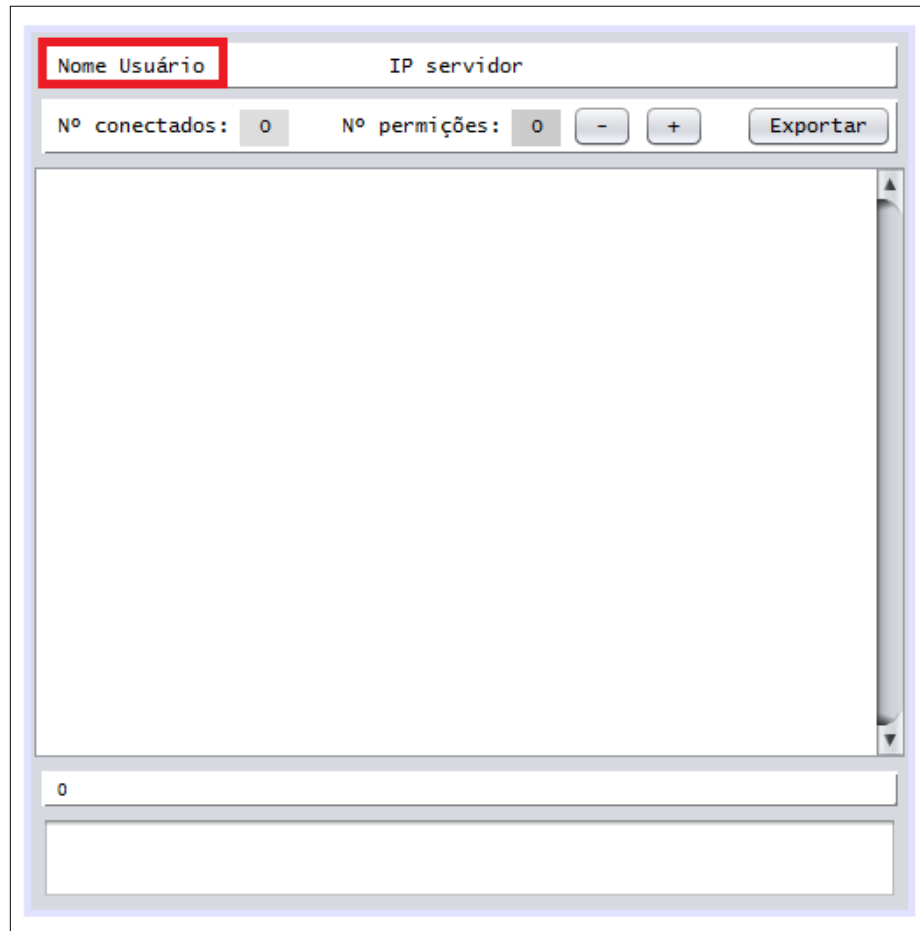


Juntamente com a inicialização da tela, se inicia o processo em que é criado o *ServerSocket*. A classe *ServerSocket* é disponibilizada pela API do *Java*, e permite a troca de informações entre usuários conectados a ele.

Há algumas informações que são importantes para o usuário que criou o *chat*.

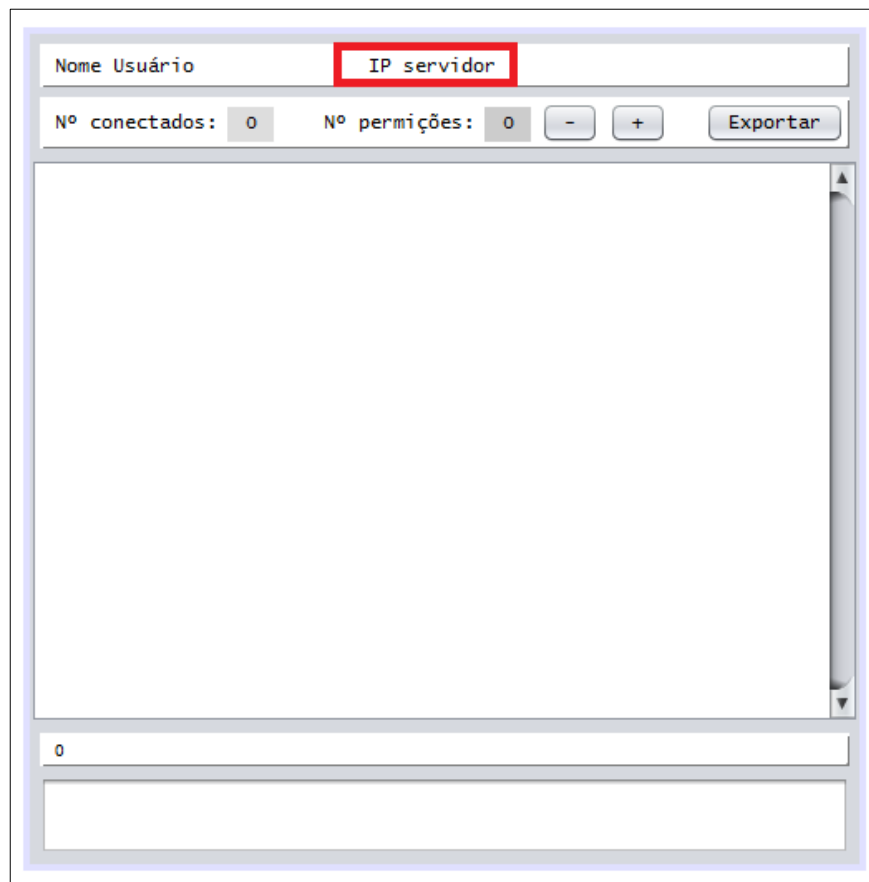
A área destacada na figura 6, mostra o local onde estará o nome do usuário. O nome do usuário é o mesmo nome do usuário que está logado na máquina, dando assim mais credibilidade ao *chat*, evitando assim que haja falsificações de nomes.

Figura 6 - Local onde é mostrado o nome do usuário servidor



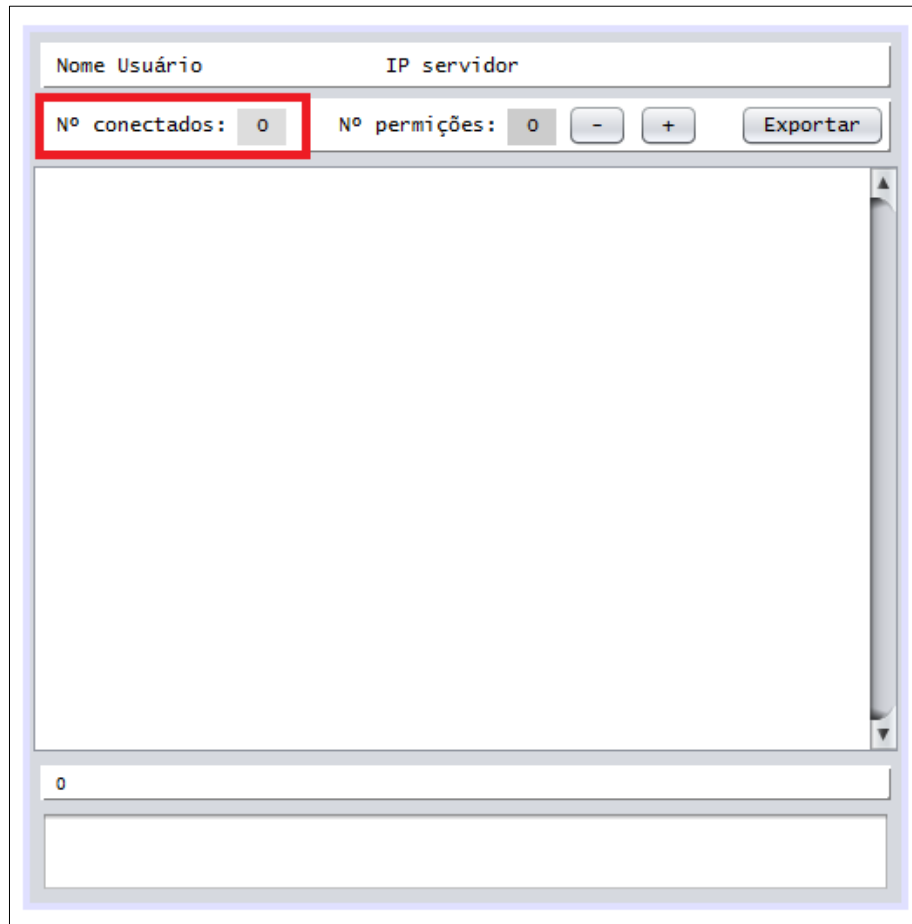
Para que um usuário se conecte ao *chat*, o mesmo deverá saber o endereço **IP** da máquina do usuário que criou o *chat*, ou seja, sem esta informação não é possível realizar a conexão. A figura 7, traz em destaque a área onde está informado o endereço para usuários realizarem a conexão.

Figura 7 - Local onde é mostrado o IP do servidor



Para se saber a quantidades de conexões que estão sendo utilizadas, a cada nova conexão ou cada vez que um usuário sair do *chat*, o número de conectados deve atualizado, o número de conectados é mostrado em destaque na figura 8. Com número de conexões atualizados, sempre se sabe com quantos usuários ativos o chat está.

Figura 8 - Número de usuários conectados ao servidor



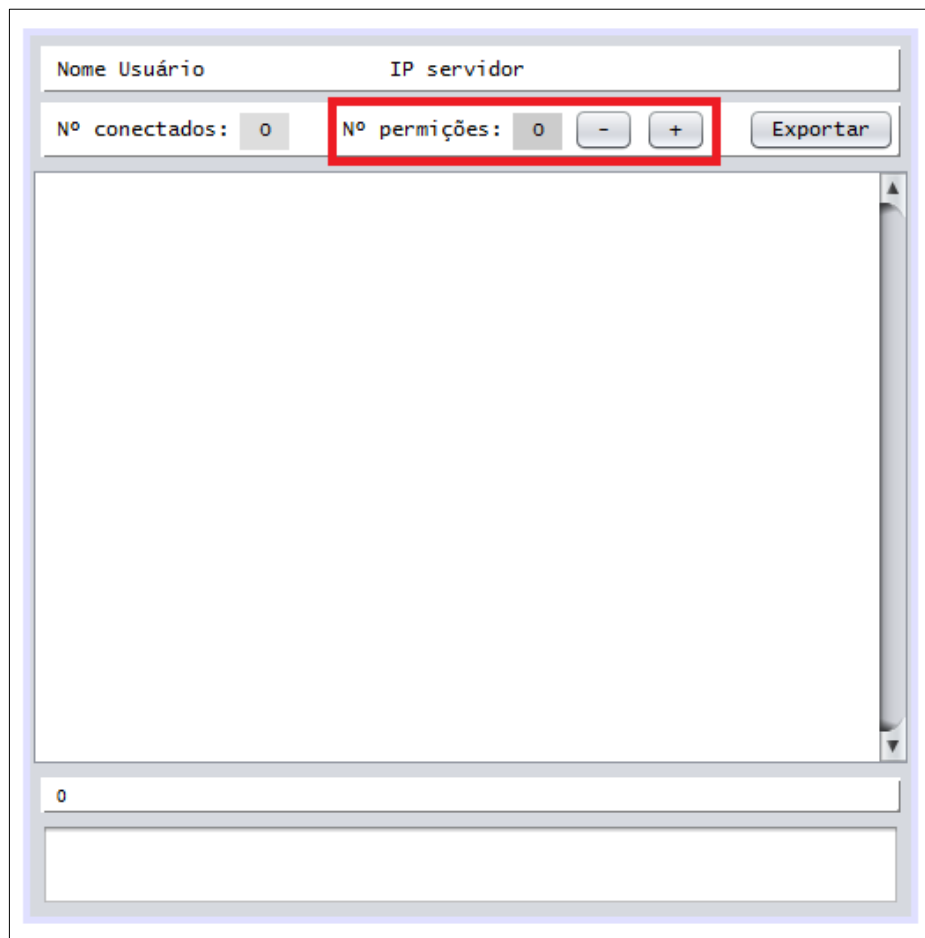
Havendo a necessidade de possibilitar novas conexões, sempre levando em consideração que o limite de conexões é 20, o usuário poderá estar disponibilizando-as.

Como é mostrado em destaque na figura 9, há duas opções que o usuário tem, uma que possibilita decrementar e outra que possibilita incrementar o número de conexões.

No momento em que é decrementado a quantidade não pode ser menor do que a quantidade de usuários conectados. Por exemplo, há três usuários conectados, então o número de conexões permitidas deve ser maior ou igual à três. Caso não haja nenhuma conexão ativa, o limite será um, como já validado na criação do *chat*.

Quando é incrementada uma conexão o limite será de vinte conexões.

Figura 9 - Opções número de conexões do servidor



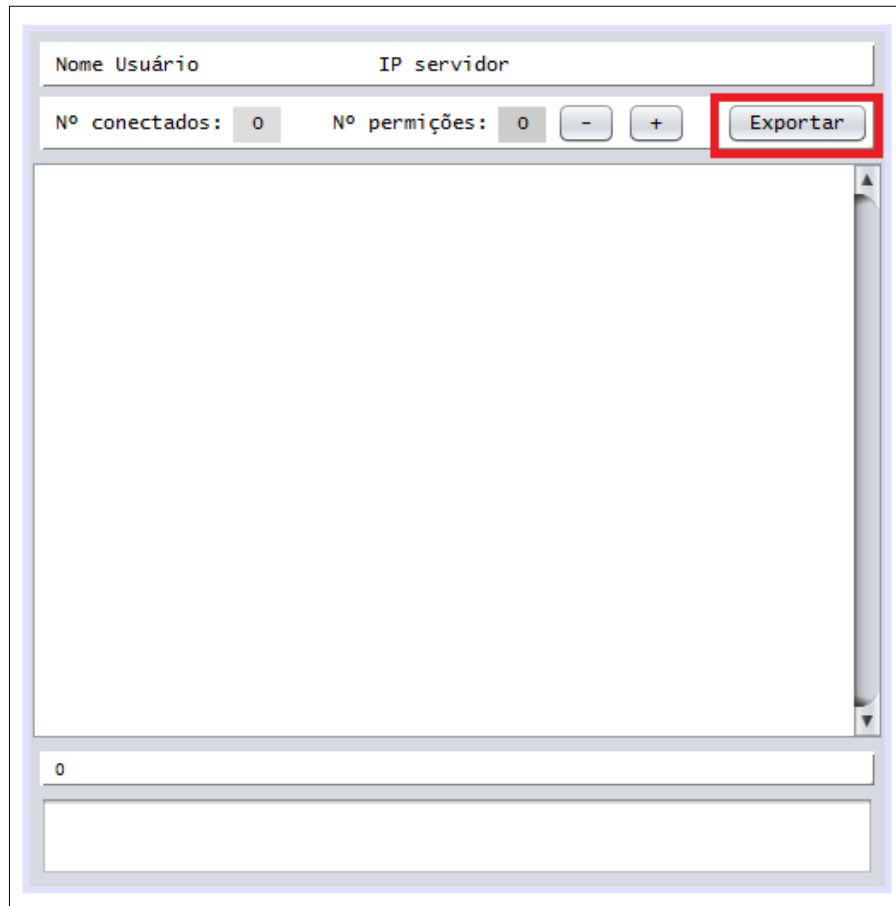
A interface de configuração de conexões do servidor apresenta um formulário com os seguintes elementos:

- Dois campos de texto no topo rotulados "Nome Usuário" e "IP servidor".
- Uma barra de controle contendo:
 - O rótulo "Nº conectados:" seguido de um campo de texto com o valor "0".
 - O rótulo "Nº permissões:" seguido de um campo de texto com o valor "0".
 - Dois botões de controle, "-" e "+", localizados imediatamente à direita do campo de "Nº permissões".
 - Um botão "Exportar" à extrema direita da barra.
- Uma grande área de visualização central, atualmente vazia, com uma barra de rolagem vertical à direita.
- Dois campos de texto adicionais na base da interface, o primeiro contendo o valor "0".

Um retângulo vermelho na imagem destaca a seção contendo "Nº permissões:", o campo de texto "0", e os botões "-" e "+".

A função “Exportar”, como destacada na figura 10, permite a criação de um arquivo TXT com todas as mensagens trocadas até o momento no chat. Após o arquivo ser gerado, o sistema dará um alerta indicando o local onde foi salvo o arquivo.

Figura 10 - Função exportar mensagens servidor

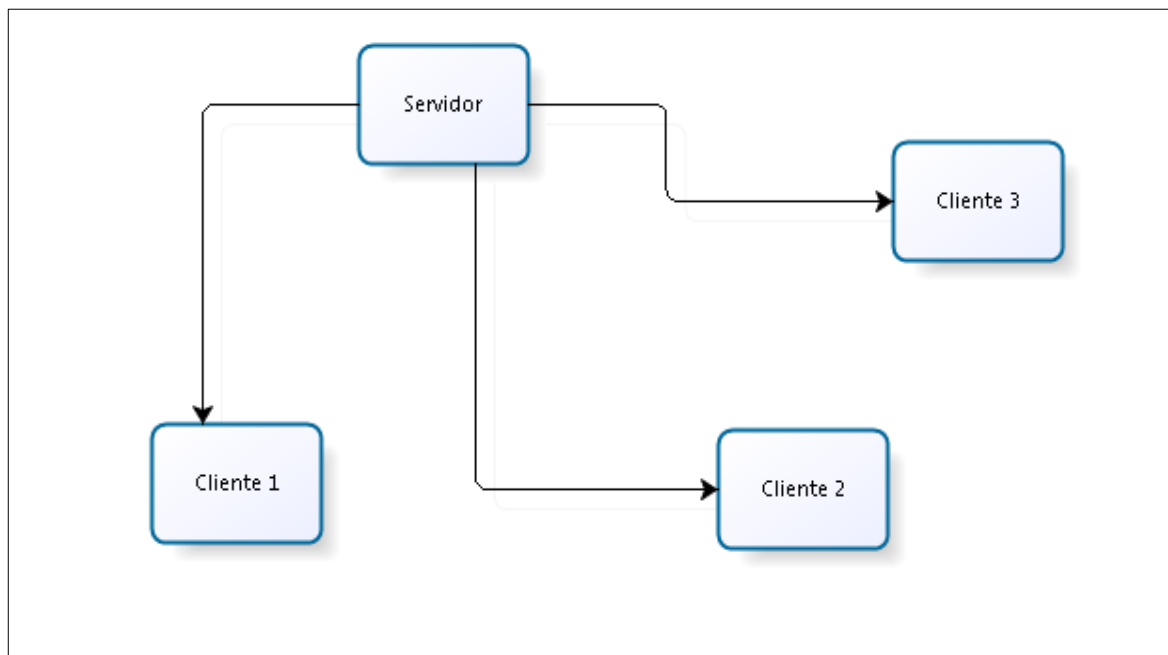


Além do servidor ter a função liberar ou bloquear algumas conexões, o servidor também é responsável por receber todas as mensagens que são enviadas no chat, e distribuí-las entre os conectados.

O exemplo da figura 11 demonstra um chat onde existe a conexão de três clientes e o servidor. Os clientes conectados ao servidor não têm contato entre eles, apenas o servidor tem a possibilidade de enviar mensagens a todos os usuários conectados.

Sendo assim, a cada vez que um cliente conectado ao chat enviar uma mensagem, a mensagem será recebida pelo servidor, e então o servidor tem a responsabilidade de repassar a mensagem para os outros usuários conectados. Assim, não há a possibilidade de um cliente conectado enviar uma mensagem a outro cliente, a mensagem apenas poderá ser enviada ao servidor.

Figura 11 - Esquema de conexões ao servidor



3.1.2 Cliente

No momento em que algum usuário realizar a conexão com algum chat, o mesmo já deverá ter sido criado, ou seja, o cliente tem que saber o endereço IP onde o chat foi criado.

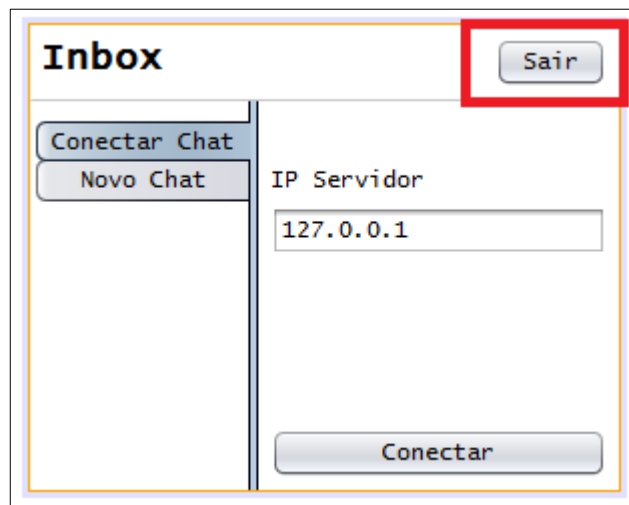
A figura 12, demonstra a tela inicial do cliente, a partir desta tela é possível realizar a conexão com o servidor.

Figura 12 - Tela configurações para cliente conectar

A imagem mostra uma interface gráfica de usuário (GUI) intitulada "Inbox" no canto superior esquerdo. No canto superior direito, há um botão "Sair". À esquerda, há uma barra de menu com duas opções: "Conectar Chat" (destacada com um fundo azul) e "Novo Chat". À direita da barra de menu, há um campo de texto rotulado "IP Servidor" com o valor "127.0.0.1" inserido. No rodapé da interface, há um botão "Conectar".

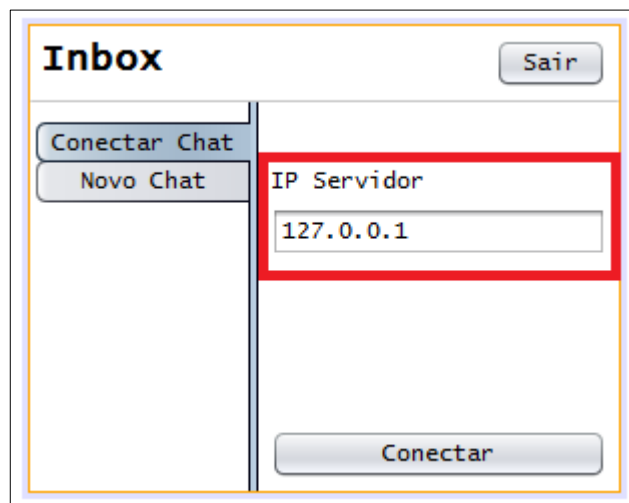
Ao selecionar a opção "Sair", destacada na figura 13, o usuário estará encerrando a execução do programa.

Figura 13 - Opção sair do cliente



A área com destaque na figura 14, mostra o local onde o IP do servidor deve ser digitado para possibilitar a conexão. Por padrão, o campo está preenchido com IP do endereçamento local, 127.0.0.1.

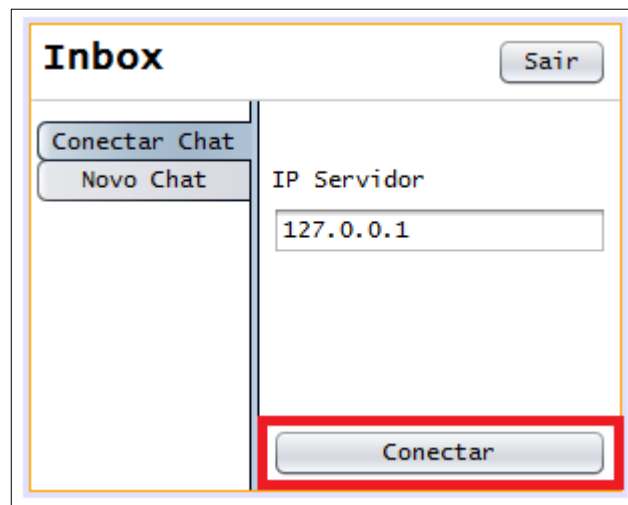
Figura 14 - Configurando o IP do servidor



Após definido o IP do servidor, e o usuário ter selecionado a opção “Conectar”, destacada na figura 15, o sistema fará a validação do IP antes de tentar realizar a conexão com o servidor. A validação do IP se dá a partir da verificação de cada conjunto de números digitados. O IP é composto de 4 partes, todas separadas por um ponto, em cada uma destas partes, o número ali inserido, deve estar entre 0 e 255.

Se o IP não for validado, o sistema alerta o usuário, e pede que o IP seja informado novamente.

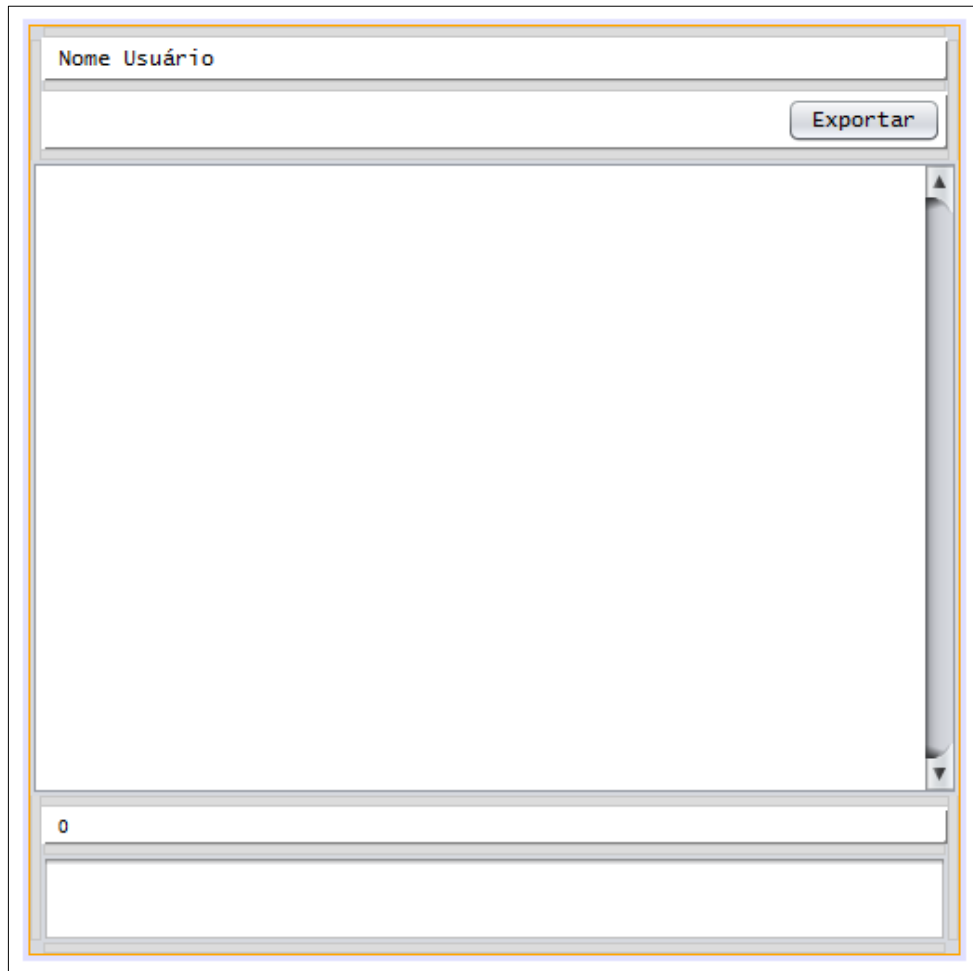
Figura 15 - Opção conectar do cliente



The image shows a software window titled "Inbox". In the top right corner, there is a button labeled "Sair". On the left side, there is a vertical menu with two buttons: "Conectar Chat" and "Novo Chat". The "Conectar Chat" button is highlighted with a blue background. To the right of the menu, there is a label "IP Servidor" above a text input field containing the value "127.0.0.1". At the bottom right of the window, there is a button labeled "Conectar", which is highlighted with a red rectangular border.

Com o IP já validado e a opção “Conectar” selecionada, o sistema irá iniciar a tela representada na figura 16. No momento em que a tela é iniciada, é enviada uma solicitação de conexão ao servidor, ou seja, o usuário apenas se conectará se o servidor autorizar. Caso a solicitação seja negada, o usuário não está autorizado a se conectar, então o usuário é informado e o chat é fechado.

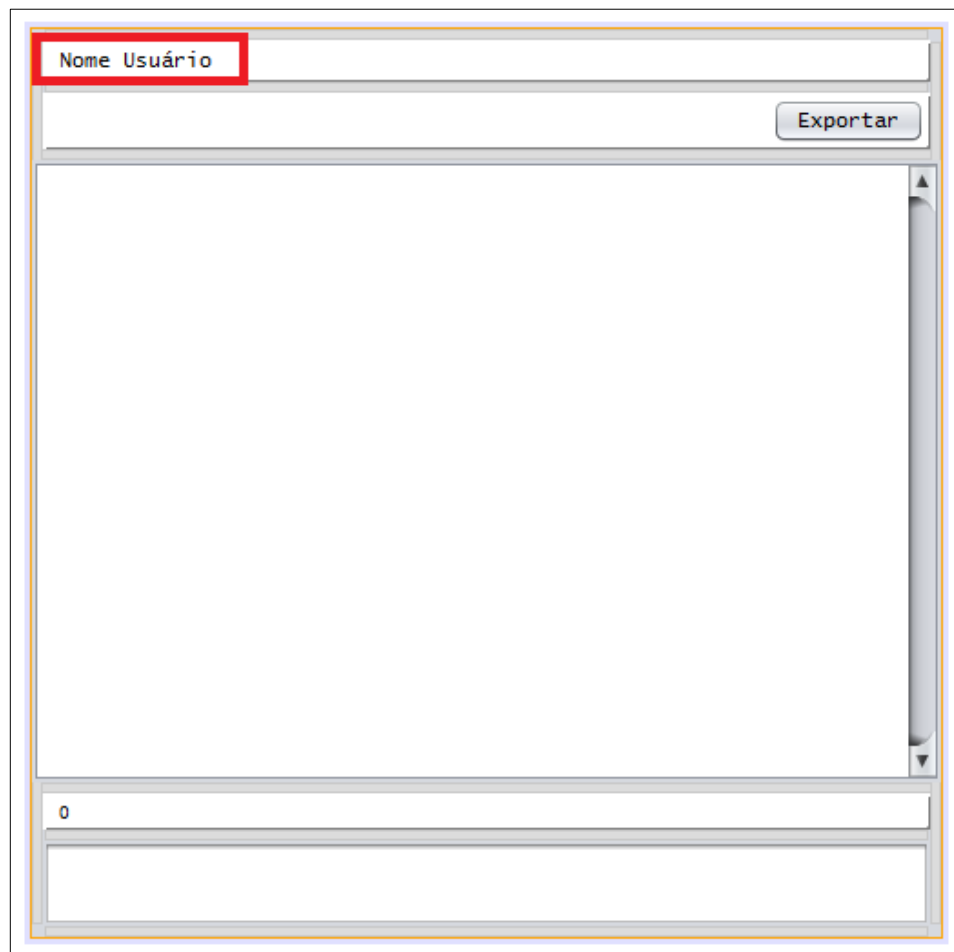
Figura 16 - Tela para cliente conectado



Juntamente com a inicialização da tela, se inicia o processo em que é criado o *Socket*. Esta classe é disponibilizada pela API do *Java*, e permite que seja realizada a conexão com o *ServerSocket* – a classe permite a criação de um servidor por onde as informações passam ao ser enviadas aos usuários.

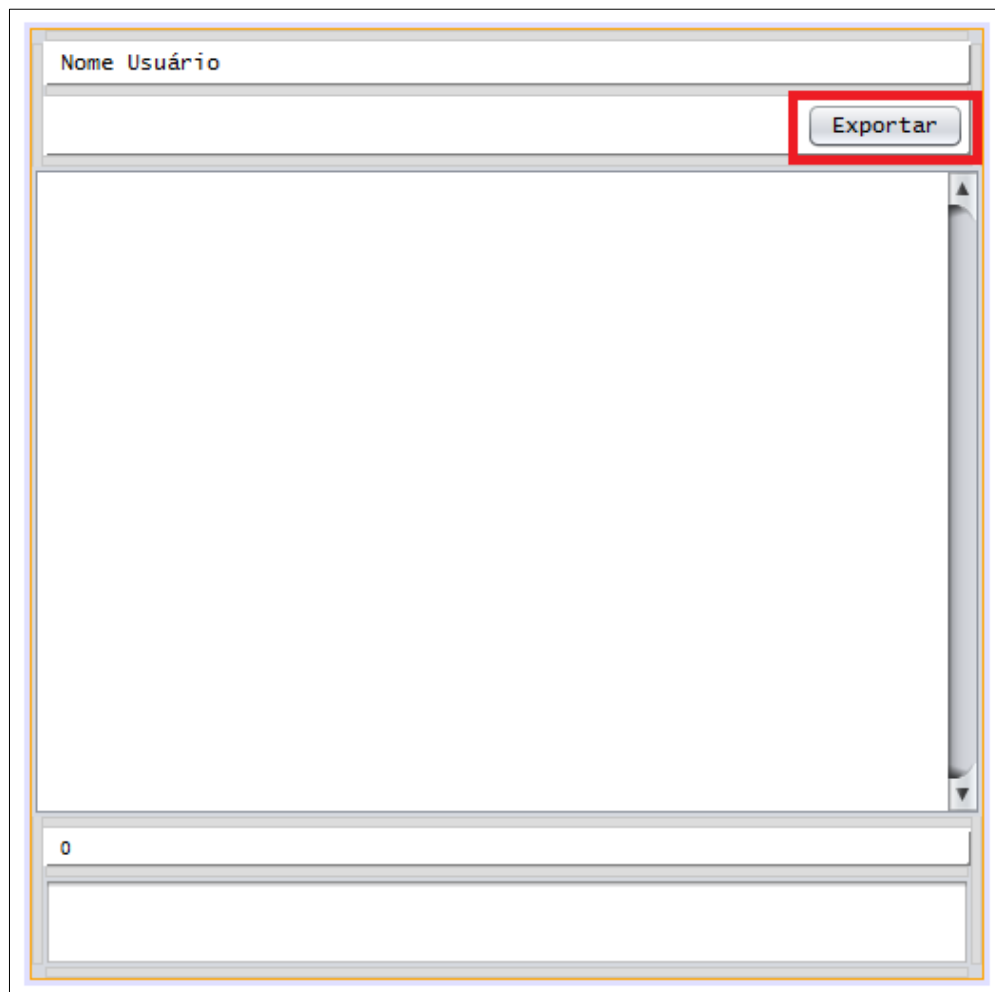
A área destacada na figura 17, mostra o local onde estará o nome do usuário. O nome é o mesmo nome do usuário que está logado na máquina, dando assim mais credibilidade ao *chat*, evitando assim que haja falsificações de nomes.

Figura 17 - Local onde o mostra nome do cliente



A função “Exportar”, como destacada na figura 18, permite a criação de um arquivo TXT com todas as mensagens trocadas até o momento no chat. Após o arquivo ser gerado, o sistema dará um alerta indicando o local onde foi salvo o arquivo.

Figura 18 - Função exportar do cliente



3.2 CRIPTOGRAFIA DE MENSAGENS

As mensagens enviadas no chat, são todas criptografadas no momento em que são enviadas, e descriptografadas no momento em que é recebida pelo destinatário. Sendo assim, as mensagens trafegam na rede com total segurança, não correndo o risco de serem facilmente acessadas por qualquer interceptação da rede.

Quaisquer caracteres que estejam inseridos na tabela ASCII, são aceitos nas mensagens. O limite de caracteres de cada mensagem é de 140. A figura 19 destaca o local onde é mostrada a contagem de caracteres da mensagem. Desta forma o usuário que está digitando, tem a informação instantânea deste dado. Caso o usuário ainda insista no envio da mensagem com quantidade de caracteres maior que 140, ou até mesmo uma mensagem vazia, o sistema alertará o usuário sobre o problema ocorrido.

Para o exemplo de criptografia e descryptografia de uma mensagem, será usada seguinte frase: “Oi! Qual é o seu nome? ”.

Abaixo está descrito como todo o processo é realizado.

3.2.1 Criptografia de mensagem

No momento em que é iniciado o processo de envio da mensagem, tanto partindo do cliente e indo ao servidor, ou vice-versa, é realizado a criptografia da mesma.

O primeiro passo é transformar a mensagem em um *array* de números. Este processo é realizado utilizando-se da tabela ASCII, ou seja, cada caractere da mensagem equivale a um valor numérico. Abaixo é mostrado os valores correspondentes a cada caractere da mensagem de exemplo.

79, 105, 33, 32, 81, 117, 97, 108, 32, 233, 32, 111, 32, 115, 101, 117, 32, 110, 111, 109, 101, 63

No próximo passo o *array* é transformado em uma matriz de 3 linhas por n colunas. O processo consiste em dividir a quantidade de caracteres encontrados na mensagem, pelo número de linhas, no caso, 3. Caso a divisão não de exata, o algoritmo fica responsável por adicionar 0s no final do *array* até a quantidade de caracteres ser divisível por 3. Assim a matriz A sempre terá 3 linhas, e o número de colunas será variável de acordo com a quantidade de caracteres que a mensagem conter, como no exemplo abaixo.

$$A = \begin{bmatrix} 79 & 105 & 33 & 32 & 81 & 117 & 97 & 108 \\ 32 & 233 & 32 & 111 & 32 & 115 & 101 & 117 \\ 32 & 110 & 111 & 109 & 101 & 63 & 0 & 0 \end{bmatrix}$$

Após ter a mensagem transformada em uma matriz A, o próximo passo é se utilizar a matriz chave, e multiplicar pela matriz A. A matriz CHAVE, é definida estaticamente no algoritmo, ou seja, não há possibilidade de ela ser capturada ou até mesmo ser mudada no momento em que a mensagem for enviada. A matriz CHAVE utilizada no trabalho é descrita abaixo.

$$\text{CHAVE} = \begin{bmatrix} 2 & 1 & -2 \\ 3 & 2 & 1 \\ -4 & -2 & 2 \end{bmatrix}$$

Após a multiplicação das matrizes obtêm-se a matriz resultado R, como demonstrado abaixo:

$$R = \begin{bmatrix} 126 & 223 & -124 & -43 & -8 & 223 & 295 & 333 \\ 333 & 891 & 274 & 427 & 408 & 644 & 493 & 558 \\ -316 & -666 & 26 & -132 & -186 & -572 & -590 & -666 \end{bmatrix}$$

A partir da matriz resultante, é realizada a transposição da mesma, tendo como resultado a seguinte matriz:

$$R^T = \begin{bmatrix} 126 & 333 & -316 \\ 223 & 891 & -666 \\ -124 & 274 & 26 \\ -43 & 427 & -132 \\ -8 & 408 & -186 \\ 223 & 644 & -572 \\ 295 & 493 & -590 \\ 333 & 558 & -666 \end{bmatrix}$$

Finalizando o processo de criptografia, é percorrido todos os valores da matriz R^T e os mesmos concatenados em variável do tipo *String*, separados por / (barra), como mostrado abaixo.

126/333/-316/223/891/-666/-124/274/26/-43/427/-132/-8/408/-186/223
/644/-572/295/493/-590/333/558/-666/

Com o objetivo de não deixar os valores explícitos na rede, a *String* com os valores, é transformada em uma *String* de *bytes*. *Byte*, é uma unidade de medida utilizada pelo computador. Transforma em *bytes*, é o mesmo que transformar cada caractere em um código da tabela ASCII.

Abaixo está demonstrado o resultado final da criptografia:

49505446484751515146484745514954464847505051464847565749464847455454
54464847454950524648475055524648475054464847455251464847525055464847
45495150464847455646484752485646484745495654464847505051464847545252
46484745535550464847505753464847525751464847455357484648475151514648
4753535646484745545454464847

Enquanto a mensagem estiver trafegando pela rede, caso algum usuário mal-intencionado realizar a captura dos dados, o que estará visível para ele será apenas os *bytes* da mensagem criptografada. Para realizar a descriptografia da mensagem, terá que ser realizado o processo inverso da criptografia, como a pessoa que fez a captura dos dados não tem acesso a chave usada para a criptografia, impede a ação do mesmo.

3.2.2 Descriptografia de mensagem

Realizando o processo inverso da criptografia, chega-se a mensagem inicial.

O primeiro passo é transformar os *bytes* da mensagem para os valores da matriz, segundo demonstra o exemplo mais abaixo.

Dois caracteres da mensagem criptografada equivalem a um caractere da mensagem contendo os valores da matriz separados por / (barra), ou seja, se parando a mensagem criptografada de dois em dois caracteres, e transformando o número encontrado em um caractere através da tabela ASCII, obtêm-se os números da matriz separados por / (barra).

49505446484751515146484745514954464847505051464847565749464847455454
54464847454950524648475055524648475054464847455251464847525055464847
45495150464847455646484752485646484745495654464847505051464847545252
46484745535550464847505753464847525751464847455357484648475151514648
4753535646484745545454464847

=

126/333/-316/223/891/-666/-124/274/26/-43/427/-132/-8/408/-186/223
/644/-572/295/493/-590/333/558/-666/

Os números encontrados são separados um por um e após transformados em uma matriz de acordo como a mostrada abaixo.

$$A = \begin{bmatrix} 126 & 333 & -316 \\ 223 & 891 & -666 \\ -124 & 274 & 26 \\ -43 & 427 & -132 \\ -8 & 408 & -186 \\ 223 & 644 & -572 \\ 295 & 493 & -590 \\ 333 & 558 & -666 \end{bmatrix}$$

Com a transposição da matriz A, obtêm-se a seguinte matriz:

$$A^T = \begin{bmatrix} 126 & 223 & -124 & -43 & -8 & 223 & 295 & 333 \\ 333 & 891 & 274 & 427 & 408 & 644 & 493 & 558 \\ -316 & -666 & 26 & -132 & -186 & -572 & -590 & -666 \end{bmatrix}$$

Como antes na criptografia, esta matriz foi encontrada a partir da multiplicação da matriz CHAVE com a matriz de caracteres no padrão ASCII, agora para se encontrar a matriz contendo os caracteres no padrão ASCII, é necessário multiplicar a matriz inversa da matriz CHAVE pela matriz A^T .

A matriz resultante desta multiplicação, será a matriz contendo os valores ASCII de cada caractere da mensagem enviada. Esta matriz resultante do exemplo citado acima, é a seguinte:

$$CHAVE^{-1} * A^T = \begin{bmatrix} 79 & 105 & 33 & 32 & 81 & 117 & 97 & 108 \\ 32 & 233 & 32 & 111 & 32 & 115 & 101 & 117 \\ 32 & 110 & 111 & 109 & 101 & 63 & 0 & 0 \end{bmatrix}$$

Para se encontrar os caracteres da mensagem, basta percorrer as linhas da matriz e transformar o valor encontrado em um caractere. Caso o valor encontrado seja 0, o valor é ignorado, pois esse valor foi adicionado no processo de criptografia para se completar a matriz $3 \times n$.

Após todo este processo chega-se a mensagem inicial, “Oi! Qual é o seu nome?”.

3.3 TESTES E VALIDAÇÕES

Para ter a certeza de que os cálculos utilizados e de que o processo de criptografia obteve sucesso, foi realizado uma série de testes.

3.3.1 Operações com matrizes

Primeiramente para se validar o sucesso no cálculo de inversa de matrizes, foi realizado na ferramenta Excel, o cálculo da inversa da matriz CHAVE utilizada na criptografia. Para os exemplos de cálculos abaixo irá se utilizar o mesmo exemplo utilizado no processo de criptografia.

A figura 19 mostra o exemplo do cálculo realizado no Excel.

Figura 19 - Inversão de matriz no Excel

C9							
	A	B	C	D	E	F	G
1							
2							
3			Matriz CHAVE				
4			2	1	-2		
5			3	2	1		
6			-4	-2	2		
7							
8			Matriz INVERSA				
9			-3	-1	-2,5		
10			5	2	4		
11			-1	0	-0,5		
12							
13							
14							
15							
16							
17							

Já a figura 20 mostra o resultado do cálculo da inversa realizada no algoritmo desenvolvido no sistema.

Figura 20 - Inversão de matriz no algoritmo desenvolvido

3	3	Refr.	Alea.	
2		1	-2	
3		2	1	
-4		-2	2	

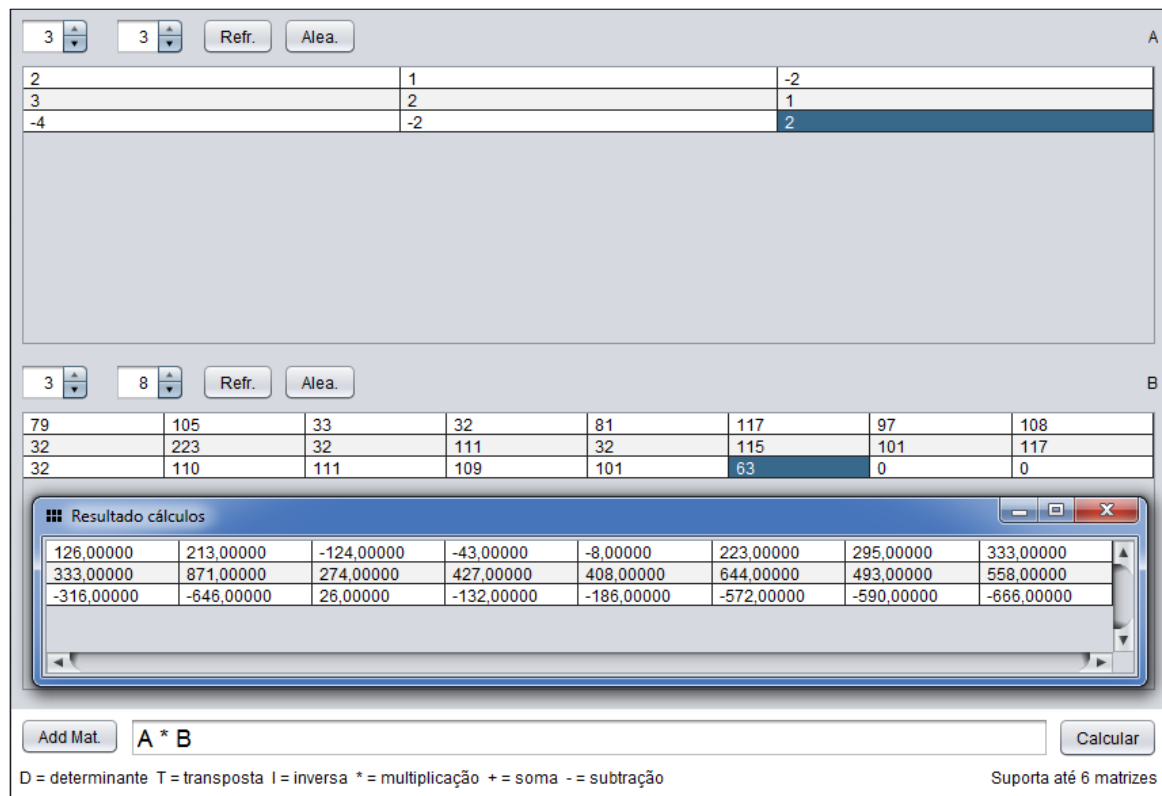
Resultado cálculos		
-3,00000	-1,00000	-2,50000
5,00000	2,00000	4,00000
-1,00000	-0,00000	-0,50000

Add Mat.	IA	Calcular
----------	----	----------

D = determinante T = transposta I = inversa * = multiplicação += soma -= subtração

Suporta até 6 matrizes

Figura 22 - Multiplicação de matriz no algoritmo desenvolvido



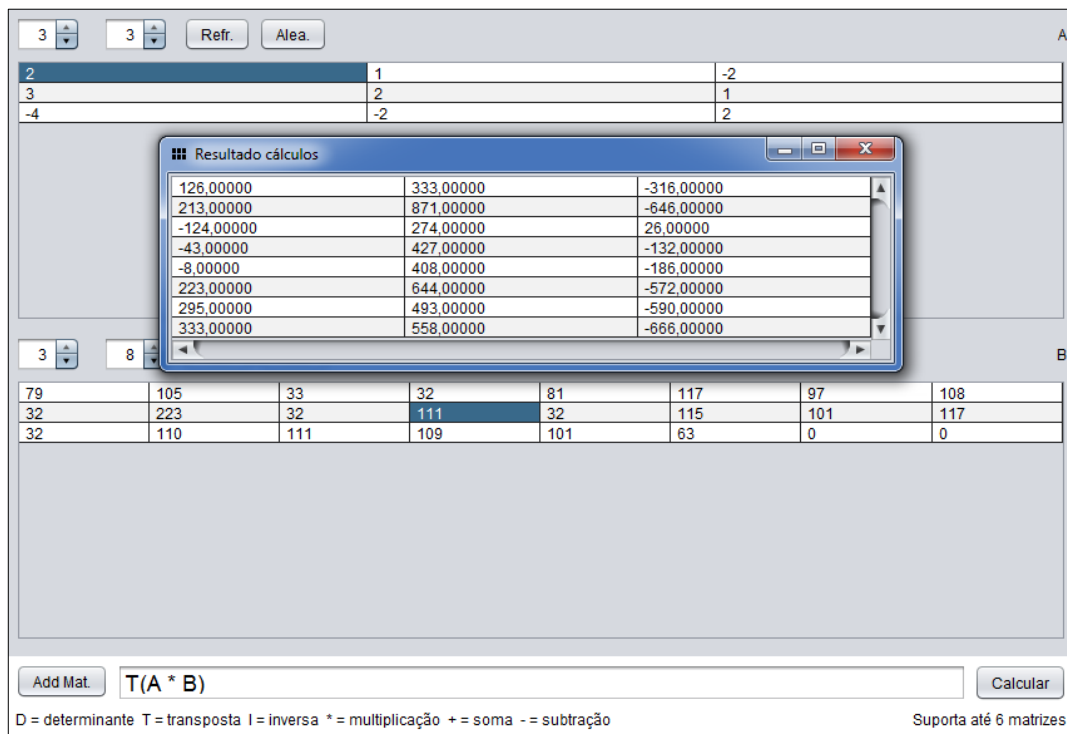
O cálculo de transposição de matrizes também foi testado. A figura 23 demonstra o resultado do cálculo realizado na ferramenta Excel.

Figura 23 - Transposição de matriz no Excel

13		Matriz RESULTANTE							
14		126	213	-124	-43	-8	223	295	333
15		333	871	274	427	408	644	493	558
16		-316	-646	26	-132	-186	-572	-590	-666
17									
18		Matriz TRANSPOSTA							
19		126	333	-316					
20		213	871	-646					
21		-124	274	26					
22		-43	427	-132					
23		-8	408	-186					
24		223	644	-572					
25		295	493	-590					
26		333	558	-666					
27									

E o cálculo de transposição de matrizes no algoritmo desenvolvido é mostrado abaixo na figura 24.

Figura 24 - Transposição de matriz no algoritmo desenvolvido

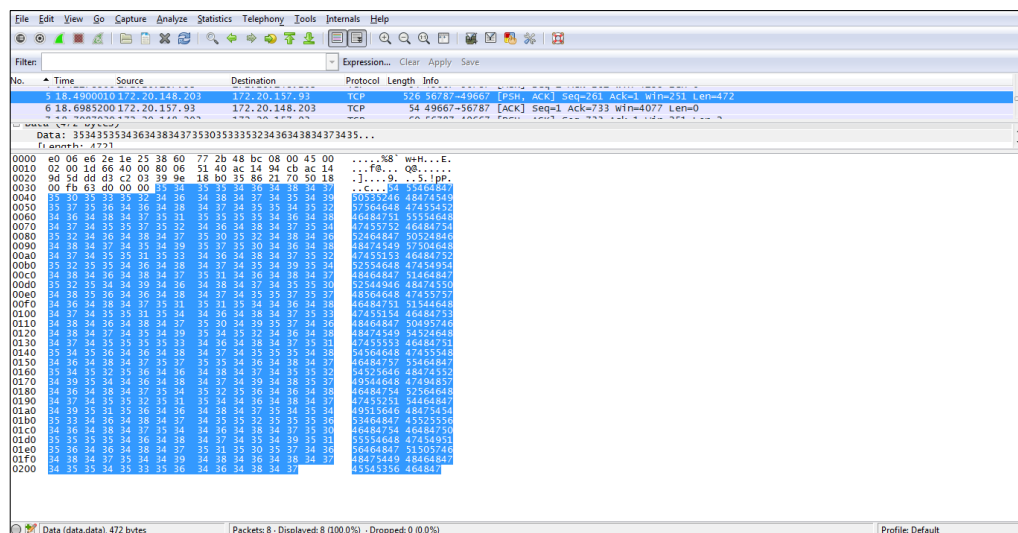


3.3.2 Criptografia das mensagens

Com objetivo de verificar se realmente as mensagens que estão sendo enviadas na rede estão criptografadas, foi-se utilizado a ferramenta Wireshark para realizar a captura dos dados.

Durante o envio da mensagem utilizada como exemplo foi feita a captura da mensagem, como demonstra a figura 25.

Figura 25 - Software sniffer com a mensagem criptografada capturada



Desta forma comprovou-se que as mensagens que são enviadas estão criptografadas, garantindo assim a segurança dos usuários.

CONCLUSÃO

Para que seja possível compartilhar informações de forma rápida e segura, são necessários algoritmos complexos. Que sejam capazes de proteger as informações, mantendo a agilidade durante o processo de envio de informações.

O objetivo principal deste trabalho foi desenvolver um algoritmo utilizando a linguagem Java, aplicando os conceitos de matrizes para a criptografia de mensagens compartilhadas entre usuários de um chat privado online. O algoritmo, denominado Secretum, é capaz de promover a troca de mensagens criptografadas, com até 20 usuários simultâneos. Diante disso, após o desenvolvimento do trabalho e testes realizados os objetivos do trabalho foram atingidos.

Para o desenvolvimento do trabalho foram levantadas hipóteses, a primeira hipótese do trabalho afirma que o algoritmo desenvolvido impossibilita o uso indevido dos dados, compartilhados entre usuários. Ao realizar testes consecutivos, com o sniffer WireShark, foi constatado que todas as informações do chat que são enviadas para a rede, estão criptografadas, conforme pode ser observado na figura 25. Assim foi possível corroborar esta hipótese.

Já a segunda hipótese afirma que o algoritmo permite que os usuários tenham segurança nas suas mensagens. A mesma foi corroborada, como visto anteriormente, foram realizados testes com a ferramenta Sniffer, todas as mensagens enviadas a rede, estão sendo criptografadas.

A terceira hipótese garante que é possível criptografar mensagens com até 140 caracteres ou mais. Durante testes foi comprovado que é possível utilizar mais de 140, porém, para manter a agilidade durante a troca de mensagens, ficou delimitado este valor. Portanto, esta hipótese foi corroborada.

Neste trabalho foi possível comprovar que a transmissão de mensagens pode ser feita de modo seguro, através da criptografia. Como sugestão para trabalhos futuros, criar um servidor utilizando banco de dados, para centralizar as informações.

REFERÊNCIAS

- CAYLEY, A. **A Memoir on the Theory of Matrices**. [S.l.]: Philosophical Transactions of the Royal Society of London, v. 148, 1858. p 17-37.
- FIARRESGA, V. C. M. **Criptografia e Matemática**. [S.l.]: Universidade de Lisboa, 2010.
- KUERTEN, C. **Algumas aplicações de matrizes**. Universidade Federal de Santa Catarina. Florianópolis, p. 60. 2002.
- LOVATO, A. **Metodologia da pesquisa**. Três de Maio: SETREM, 2013.
- MONTEIRO, A. **Álgebra Linear e Geometria Analítica**. Lisboa: McGraw-Hill, 2001.
- NAKAMURA, E. T.; GEUS, P. L. D. **Segurança de Redes em Ambientes Cooperativos**. São Paulo: Novatec, 2007.
- NOTHCUTT, S. et al. **Desvendando Segurança em Redes**. Tradução de Daniel Vieira. Rio de Janeiro: Campus, 2002.
- PUGA, S.; RISSETTI, G. **Lógica de programação e estruturas de dados**. 2ª. ed. São Paulo: Pearson Prentice Hall, 2008.
- SEBESTA, R. W. **Conceitos de Linguagens de Programação**. [S.l.]: Artmed Editora SA, 2010. Disponível em: <https://books.google.com.br/books?id=vPldwBmt-9wC&printsec=frontcover&dq=linguagens+de+programacao&hl=pt-BR&sa=X&redir_esc=y#v=onepage&q&f=false>.
- TANENBAUM, A. S. **Redes de Computadores**. Tradução de Vanderberg D. de SOUZA. 4. ed. Amsterdam: Editora Campus, 2003.

APÊNDICE A – CLASSES PROJETO

1. CRIPTOGRAFIA

```
public class Criptografia {

    private static final int numero_linhas = 3;

    //pega o texto e transforma ele em um vetor
    private static double[] textoParaVetor(String texto, boolean
remover_separador) {
        double[] result;
        if (!remover_separador) {
            result = new double[texto.length()];
            for (int i = 0; i < result.length; i++) {
                result[i] = (double) texto.charAt(i);
            }
        } else {
            String[] chars = texto.split(Utils.caracter_padrao);
            result = new double[chars.length];
            for (int i = 0; i < result.length; i++) {
                result[i] = Double.valueOf(chars[i]);
            }
        }
        return result;
    }

    private static double[][] textoParaMatriz(String texto, boolean
criptografa) {
        double[] chars = textoParaVetor(texto, !criptografa);
        int lenght_matriz = chars.length;

        while (lenght_matriz % numero_linhas != 0) {
            lenght_matriz++;
        }

        double[][] letras;
        if (!criptografa) {
            letras = new double[lenght_matriz /
numero_linhas][numero_linhas];
        } else {
            letras = new double[numero_linhas][lenght_matriz /
numero_linhas];
        }
        int posicao_letra = 0;
        for (double[] linha : letras) {
            for (int j = 0; j < letras[0].length; j++) {
```

```

        if (posicao_letra < chars.length) {
            linha[j] = (double) chars[posicao_letra];
            posicao_letra++;
        } else {
            linha[j] = 0;
        }
    }
}
return letras;
}

/**
 * Criptografia o texto passado como parametro.
 * @param texto O texto pode conter qualquer caractere e não têm limite
de caracteres
 * @return String de texto criptografado
 */
public static String criptografa(String texto) {
    double[][] mat = textoParaMatriz(texto, true);
    double[][] multi =
OperacaoMatrizes.multiplicaMatrizes(OperacaoMatrizes.matrizBase(), mat);
    double[][] result = OperacaoMatrizes.transpoemMatriz(multi);
    return bytesParaTexto(matrizParaTexto(result).getBytes());
}

private static byte[] stringParaByte(String bytes) {
    int length = bytes.length();
    byte[] result = new byte[length / 2];
    String s;
    int posicao_byte = 0;
    for (int i = 0; i < length; i += 2) {
        s = bytes.substring(i, i + 2);
        result[posicao_byte] = Byte.valueOf(s);
        posicao_byte++;
    }
    return result;
}

private static String bytesParaTexto(byte[] b) {
    StringBuilder sb = new StringBuilder();
    for (int i = 0; i < b.length; i++) {
        sb.append(b[i]);
    }
    return sb.toString();
}

private static String matrizParaTexto(double[][] chars) {
    StringBuilder sb = new StringBuilder();
    for (double[] linha : chars) {
        for (int j = 0; j < linha.length; j++) {
            sb.append(linha[j]).append(Utils.caracter_padrao);
        }
    }
    return sb.toString();
}

/**
 * Dexcriptografa o texto passado, transformando-o em um texto legível.
 * @param texto Passar o texto criptografado para descriptografar
 * @return Retorna o texto descriptografado
 */

```

```

        public static String descriptografa(String texto) {
            double[][] chars =
OperacaoMatrizes.transpoemMatriz(textoParaMatriz(new
String(stringParaByte(texto)), false));
            double[][] result =
OperacaoMatrizes.multiplicaMatrizes(OperacaoMatrizes.matrizBaseInversa(),
chars);
            StringBuilder sb = new StringBuilder();
            for (double[] linha : result) {
                for (int j = 0; j < linha.length; j++) {
                    if (linha[j] != 0) {
                        sb.append((char) linha[j]);
                    }
                }
            }
            return sb.toString();
        }
    }
}

```

2. UTILITÁRIOS

```

public class Utils {

    public static final int porta_conectar = 56787;
    public static final String arquivo_audio = "resources/notificacao.wav";
    public static final String padrao_desligar = "D";
    public static final int tempo_delay_animacao = 3000;

    public static final String padrao_notificacao = "N";
    public static final String padrao_envio_nome = "EN";

    public static final String caracter_padrao = "/";
    public static final String mensagem_padrao_erro = "O servidor não está
operando";

    public static String horaMsg() {
        return new SimpleDateFormat("HH:mm").format(new Date());
    }

    public static void audioRecebeMensagem(Class c) {
        try {
            URL sound = c.getClassLoader().getResource(arquivo_audio);
            AudioInputStream audio =
AudioSystem.getAudioInputStream(sound);
            Clip clip = AudioSystem.getClip();
            clip.open(audio);
            clip.start();
        } catch (Exception e) {
        }
    }

    public static void notificacao(String mensagem) {
        showMessageDialog(null, mensagem);
    }

    public static String nomeUsuario() {
        return System.getProperty("user.name");
    }

    public static String formataMensagem(String msg, String usuario) {
        if (!usuario.isEmpty()) {

```

```

        return "(" + Utils.horaMsg() + ") " + usuario + ": " + msg;
    }
    return "(" + Utils.horaMsg() + ") " + msg;
}

public static boolean validaMensagem(String mensagem) {
    return mensagem.length() <= 140 && mensagem.length() > 0;
}

public static String[] quebraString(String texto) {
    return texto.split("(?<=\\G.{58})");
}

public static void criarArquivoText(String texto) {
    try {
        File f = new File("arquivo_mensagens.txt");
        f.createNewFile();
        PrintWriter escrever = new PrintWriter(f);
        escrever.print(texto);
        escrever.flush();
        escrever.close();
        Utils.notificacao("O arquivo foi salvo em:\n " +
f.getAbsolutePath());
    } catch (Exception ex) {
    }
}

public static boolean validarIP(String ip) {
    Pattern pat = Pattern.compile("^(([1]?[0-9]{1,2}|2([0-4][0-9]|5[0-5]))\\.){3}([1]?[0-9]{1,2}|2([0-4][0-9]|5[0-5]))$");
    Matcher mat = pat.matcher(ip);
    return mat.matches();
}
}

```

3. SERVIDOR EM SEGUNDO PLANO

```

public class TrayItemChat {

    private JFrame janela;
    private TrayIcon trayIcon;
    private SystemTray tray;

    public TrayItemChat(JFrame janela) {
        this.janela = janela;
    }

    private void abreJanela() {
        janela.setVisible(true);
        tray.remove(trayIcon);
    }

    public void iniciar() {

        if (!SystemTray.isSupported()) {
            return;
        }

        PopupMenu popup = new PopupMenu();
    }
}

```

```

        trayIcon = new TrayIcon(new
ImageIcon(janela.getClass().getClassLoader().getResource("resources/icone_t
ray.png")).getImage(), "Secretum");
        tray = SystemTray.getSystemTray();

        MenuItem abrir = new MenuItem("Abrir");
        abrir.addActionListener((ActionListener) -> {
            abreJanela();
        });

        MenuItem fechar = new MenuItem("Fechar");
        fechar.addActionListener((ActionListener) -> {
            System.exit(0);
        });

        popup.add(abrir);
        popup.addSeparator();
        popup.add(fechar);

        trayIcon.addActionListener((ActionListener) -> {
            abreJanela();
        });

        trayIcon.setPopupMenu(popup);

        try {
            tray.add(trayIcon);
        } catch (Exception e) {
        }
    }
}

```

4. OPERAÇÕES COM MATRIZES

```

public class OperacaoMatrizes {

    public static final DecimalFormat formata_double = new
DecimalFormat("0.00000");

    public static double[][] matrizBase() {
        return new double[][]{{2, 1, -2}, {3, 2, 1}, {-4, -2, 2}};
    }

    public static double numerosPrimosPadrao() {
        return 3d * 2d;
    }

    public static double[][] matrizBaseInversa() {
        return inversaMatriz(matrizBase(), false);
    }

    public static double[] somaVetores(double[] a, double[] b) {
        if (a.length != b.length) {
            throw new Error("colunas vetor a != colunas vetor b");
        }

        double[] result = new double[a.length];
        for (int i = 0; i < a.length; i++) {
            result[i] = a[i] + b[i];
        }
        return result;
    }
}

```

```

    }

    public static double[] diminuirVetores(double[] a, double[] b) {
        if (a.length != b.length) {
            throw new Error("colunas vetor a != colunas vetor b");
        }

        double[] result = new double[a.length];
        for (int i = 0; i < a.length; i++) {
            result[i] = a[i] - b[i];
        }
        return result;
    }

    public static void imprimeVetor(double[] a) {
        for (double b : a) {
            System.out.print(formata_double.format(b) + "\t");
        }
    }

    public static void imprimeMatriz(double[][] a) {
        for (double[] linha : a) {
            imprimeVetor(linha);
            System.out.println();
        }
    }

    public static double[][] transpoemMatriz(double[][] a) {
        double[][] result = new double[a[0].length][a.length];
        for (int i = 0; i < a.length; i++) {
            for (int j = 0; j < a[0].length; j++) {
                result[j][i] = a[i][j];
            }
        }
        return result;
    }

    private static double linhaxColuna(double[] a, double[] b) {
        double soma = 0;
        for (int i = 0; i < a.length; i++) {
            soma += a[i] * b[i];
        }
        return soma;
    }

    public static double[] vetorPorEscalar(double[] a, double num) {
        double[] result = new double[a.length];
        for (int i = 0; i < a.length; i++) {
            result[i] = a[i] * num;
        }
        return result;
    }

    public static double[][] matrizPorEscalar(double[][] a, double num) {
        double[][] result = new double[a.length][a[0].length];
        for (int i = 0; i < a.length; i++) {
            result[i] = vetorPorEscalar(a[i], num);
        }
        return result;
    }
}

```

```

    public static double[][] multiplicaMatrizes(double[][] a, double[][] b)
    {
        if (a[0].length != b.length) {
            throw new Error("colunas matriz a != linhas matriz b");
        }
        double[][] transpostaB = transpoemMatriz(b);
        double[][] mat_produto = new double[a.length][b[0].length];
        for (int linhaA = 0; linhaA < a.length; linhaA++) {
            for (int linhaB = 0; linhaB < transpostaB.length; linhaB++) {
                mat_produto[linhaA][linhaB] = linhaxColuna(a[linhaA],
transpostaB[linhaB]);
            }
        }
        return mat_produto;
    }

    public static double[][] matrizIdentidade(int ordem) {
        double[][] result = new double[ordem][ordem];
        for (int i = 0; i < ordem; i++) {
            for (int j = 0; j < ordem; j++) {
                result[i][j] = (i == j) ? 1 : 0;
            }
        }
        return result;
    }

    public static double[][] somaMatrizes(double[][] a, double[][] b) {
        if (a.length != b.length || a[0].length != b[0].length) {
            throw new Error("as matrixes devem possuir a mesma ordem");
        }

        double[][] result = new double[a.length][a[0].length];

        for (int i = 0; i < a.length; i++) {
            result[i] = somaVetores(a[i], b[i]);
        }

        return result;
    }

    public static double[][] diminuiMatrizes(double[][] a, double[][] b) {
        if (a.length != b.length || a[0].length != b[0].length) {
            throw new Error("as matrixes devem possuir a mesma ordem");
        }

        double[][] result = new double[a.length][a[0].length];

        for (int i = 0; i < a.length; i++) {
            result[i] = diminuirVetores(a[i], b[i]);
        }

        return result;
    }

    public static double inversoNumero(double num) {
        return (1 / num);
    }

    public static double[] linhaxLinhaMultiplicaLinha(double[] a, double[]
b, double num) {
        a = vetorPorEscalar(a, num);
    }

```

```

        return somaVetores(a, b);
    }

    public static double determinanteMatriz(double[][] a) {
        if (a.length != a[0].length) {
            throw new Error("a coluna deve ser quadrada");
        }

        if (todosValoresColunaSaoZero(0, a)) {
            return 0;
        }

        double[][] calcs = a.clone();
        int l = calcs.length;

        //deixa o pivo inicial diferente de 0 se o mesmo for 0
        int linha_somar = 1;
        while (calcs[0][0] == 0d) {
            calcs[0] = somaVetores(calcs[0], calcs[linha_somar]);
            linha_somar++;
        }

        ArrayList<Double> multiplicacoes = new ArrayList<>();

        for (int i = 0; i < l; i++) {
            for (int j = 0; j < l; j++) {
                if (i == j) {
                    double valor_normal = calcs[i][j];
                    multiplicacoes.add(valor_normal);
                    double valor_inverso = inversoNumero(valor_normal);
                    calcs[i] = vetorPorEscalar(calcs[i], valor_inverso);
                    break;
                }
            }
            for (int k = 0; k < l; k++) {
                if (k != i && calcs[k][i] != 0) {
                    double v = -calcs[k][i];
                    calcs[k] = linhaXLinhaMultiplicaLinha(calcs[i],
calcs[k], v);
                }
            }
        }

        double valor = 1;
        for (Double v : multiplicacoes) {
            valor *= v;
        }

        return valor;
    }

    private static boolean todosValoresColunaSaoZero(int col, double[][] a)
    {
        for (double[] linha : a) {
            if (linha[col] != 0d) {
                return false;
            }
        }
        return true;
    }
}

```



```

    public static double[][] inversaMatriz(double[][] a, boolean verif_det)
    {
        if (a.length != a[0].length) {
            throw new Error("a coluna deve ser quadrada");
        }

        if (verif_det) {
            if (determinanteMatriz(a) == 0d) {
                throw new Error("o determinante é 0, nao existe inversa");
            }
        }

        double[][] calcs = a.clone();
        int l = calcs.length;
        double[][] iden = matrizIdentidade(l);

        int linha_somar = 1;
        while (calcs[0][0] == 0d) {
            calcs[0] = somaVetores(calcs[0], calcs[linha_somar]);
            iden[0] = somaVetores(iden[0], iden[linha_somar]);
            linha_somar++;
        }

        for (int i = 0; i < l; i++) {
            for (int j = 0; j < l; j++) {
                if (i == j) {
                    double valor = inversoNumero(calcs[i][j]);
                    calcs[i] = vetorPorEscalar(calcs[i], valor);
                    iden[i] = vetorPorEscalar(iden[i], valor);
                    break;
                }
            }
            for (int k = 0; k < l; k++) {
                if (k != i && calcs[k][i] != 0) {
                    double v = -calcs[k][i];
                    calcs[k] = linhaxLinhaMultiplicaLinha(calcs[i],
calcs[k], v);
                    iden[k] = linhaxLinhaMultiplicaLinha(iden[i], iden[k],
v);
                }
            }
        }

        return iden;
    }
}

```

5. ALERTA DE NOVA MENSAGEM

```

public class IconeAlerta {

    Dialog d;
    Window w;

    public IconeAlerta(Window w) {
        this.w = w;
    }

    public void alertar() {
        d = new Dialog(w);
    }
}

```

```

d.setUndecorated(true);
d.setSize(0, 0);
d.setModal(false);

d.addWindowFocusListener(new WindowAdapter() {

    @Override
    public void windowGainedFocus(WindowEvent e) {
        w.requestFocus();
        d.setVisible(false);
        super.windowGainedFocus(e);
    }
});

w.addWindowFocusListener(new WindowAdapter() {

    @Override
    public void windowGainedFocus(WindowEvent e) {
        d.setVisible(false);
        super.windowGainedFocus(e);
    }
});

if (!w.isFocused()) {
    d.setVisible(false);
}
d.setLocation(0, 0);
d.setLocationRelativeTo(w);
d.setVisible(true);
}
}

```

APÊNDICE B – INTERFACE USUÁRIO

1. TELA INICIAL

```
package chatcomservidor;

import classes.Utils;
import javax.swing.ImageIcon;

public class Inicial extends javax.swing.JFrame {

    public Inicial() {
        initComponents();
    }

    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jTabbedPane1 = new javax.swing.JTabbedPane();
        jPanel2 = new javax.swing.JPanel();
        jLabel2 = new javax.swing.JLabel();
        txtIPConectarChat = new javax.swing.JTextField();
        btnConectarCliente = new javax.swing.JButton();
        jPanel3 = new javax.swing.JPanel();
        jLabel3 = new javax.swing.JLabel();
        btnCriarChat = new javax.swing.JButton();
        txtMaximoConectados = new javax.swing.JSpinner();
        jPanel1 = new javax.swing.JPanel();
        jLabel1 = new javax.swing.JLabel();
        jButton1 = new javax.swing.JButton();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        setBackground(new java.awt.Color(255, 255, 255));
        setIconImage(new
ImageIcon(getClass().getClassLoader().getResource("resources/icone.png")).get
Image());
        setResizable(false);

        jTabbedPane1.setBackground(new java.awt.Color(255, 255, 255));
        jTabbedPane1.setTabPlacement(javax.swing.JTabbedPane.LEFT);
        jTabbedPane1.setFont(new java.awt.Font("Lucida Console", 0, 12));
        // NOI18N
        jTabbedPane1.setOpaque(true);

        jPanel2.setBackground(new java.awt.Color(255, 255, 255));
```



```

jPanel3.setBackground(new java.awt.Color(255, 255, 255));

jLabel3.setFont(new java.awt.Font("Lucida Console", 0, 12)); //
NOI18N
jLabel3.setText("Num. conexões permitidas");

btnCriarChat.setFont(new java.awt.Font("Lucida Console", 0, 12));
// NOI18N
btnCriarChat.setText("Criar");
btnCriarChat.addActionListener(new java.awt.event.ActionListener()
{
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnCriarChatActionPerformed(evt);
    }
});

txtMaximoConectados.setFont(new java.awt.Font("Lucida Console", 0,
12)); // NOI18N
txtMaximoConectados.setValue(2);

javax.swing.GroupLayout jPanel3Layout = new
javax.swing.GroupLayout(jPanel3);
jPanel3.setLayout(jPanel3Layout);
jPanel3Layout.setHorizontalGroup(

jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(jPanel3Layout.createSequentialGroup()
        .addGap(10, 10, 10)
        .addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(btnCriarChat,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
            .addGroup(jPanel3Layout.createSequentialGroup()
                .addComponent(txtMaximoConectados)
                .addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(jLabel3)
                    .addGap(0, 0, Short.MAX_VALUE))
                .addGap(10, 10, 10))
            .addGap(10, 10, 10))
        .addContainerGap(10, true))
);
jPanel3Layout.setVerticalGroup(

jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(jPanel3Layout.createSequentialGroup()
        .addGap(10, 10, 10)
        .addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel3Layout.createSequentialGroup()
                .addGap(10, 10, 10)
                .addComponent(jLabel3)
                .addGap(10, 10, 10))
            .addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(txtMaximoConectados,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(10, 10, 10))
            .addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(btnCriarChat)
                .addGap(10, 10, 10))
            .addGap(10, 10, 10))
        .addContainerGap(10, true))
);

```

```

jTabbedPane1.addTab("Novo Chat", jPanel3);

jPanel1.setBackground(new java.awt.Color(255, 255, 255));

jLabel1.setFont(new java.awt.Font("Lucida Console", 1, 18)); //
NOI18N
jLabel1.setText("Secretum");

jButton1.setFont(new java.awt.Font("Lucida Console", 0, 12)); //
NOI18N
jButton1.setText("Sair");
jButton1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton1ActionPerformed(evt);
    }
});

javax.swing.GroupLayout jPanel1Layout = new
javax.swing.GroupLayout(jPanel1);
jPanel1.setLayout(jPanel1Layout);
jPanel1Layout.setHorizontalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(jPanel1Layout.createSequentialGroup()
        .addGap(10, 10, 10)
        .addComponent(jLabel1)
        .addContainerGap(10, Short.MAX_VALUE))
    .addGroup(jPanel1Layout.createSequentialGroup()
        .addGap(10, 10, 10)
        .addComponent(jButton1)
        .addContainerGap(10, Short.MAX_VALUE))
    );
jPanel1Layout.setVerticalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(jPanel1Layout.createSequentialGroup()
        .addGap(10, 10, 10)
        .addComponent(jLabel1)
        .addGap(10, 10, 10)
        .addComponent(jButton1)
        .addContainerGap(10, Short.MAX_VALUE))
    );

javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
getContentPane().setLayout(layout);
layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addComponent(jTabbedPane1,
        javax.swing.GroupLayout.Alignment.TRAILING)
    .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
        javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
    );
layout.setVerticalGroup(

```

```

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
    .addComponent(jPanell,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
    .addGap(1, 1, 1)
    .addComponent(jTabbedPanel))
);

pack();
setLocationRelativeTo(null);
} // </editor-fold>

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    System.exit(0);
}

private void btnCriarChatActionPerformed(java.awt.event.ActionEvent
evt) {
    int conectados =
Integer.valueOf(txtMaximoConectados.getValue().toString());
    if (conectados >= 1 && conectados <= 20) {
        Servidor servidor = new Servidor(conectados, this);
        servidor.setVisible(true);
        this.dispose();
    } else {
        Utils.notificacao("O número de conexões permitidas devem estar
entre 1 e 20");
    }
}

private void
btnConectarClienteActionPerformed(java.awt.event.ActionEvent evt) {
    String ip = txtIPConectarChat.getText();
    if (Utils.validarIP(ip)) {
        Cliente cliente = new Cliente(ip, this);
        cliente.setVisible(true);
        this.dispose();
    } else {
        Utils.notificacao("IP Inválido");
    }
}

public static void main(String args[]) {
    //<editor-fold defaultstate="collapsed" desc=" Look and feel
setting code (optional) ">
    try {
        for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
            if ("Nimbus".equals(info.getName())) {
                javax.swing.UIManager.setLookAndFeel(info.getClassName());
                break;
            }
        }
    } catch (Exception ex) {
    }
} //</editor-fold>

```

```

        java.awt.EventQueue.invokeLater(() -> {
            new Inicial().setVisible(true);
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JButton btnConectarCliente;
    private javax.swing.JButton btnCriarChat;
    private javax.swing.JButton jButton1;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JPanel jPanel3;
    private javax.swing.JTabbedPane jTabbedPane1;
    private javax.swing.JTextField txtIPConectarChat;
    private javax.swing.JSpinner txtMaximoConectados;
    // End of variables declaration
}

```

2. TELA SERVIDOR

```

public class Servidor extends javax.swing.JFrame {

    //timer para limpar o campo de avisos depois de um tempo
    private javax.swing.Timer timer;

    //guarda os dados dos clientes conectados
    private ArrayList<UsuariosConectados> conexoes_usuarios;

    //número máximo de clientes que o chat aceita
    private int maximo_conectados;

    //número de clientes que estão conectados ao chat no momento
    private int numero_conectados;

    //sequencial de códigos dos clientes já conectados
    private int codigo_usuario;

    //nome do usuário servidor
    private String nome_usuario;

    //form inicial, usado apenas se acontecer algum erro aqui
    private final Inicial inicial;

    //socket de conexão do servidor, a hora que este fechar, fecha o chat
    private ServerSocket servidor;

    //classe para mostrar ícone piscando quando recebe mensagem
    private IconeAlerta alertas;

    //construtor da classe servidor, onde são passado o número máximo de
    //conexões permitidas,
    public Servidor(int maximo_usuarios, Inicial inicial) {
        initComponents();
        this.inicial = inicial;
        //seta as configurações iniciais do chat
        configuracoesIniciais(maximo_usuarios);
        //inicia em um processo paralelo, a criação do servidor
    }
}

```



```

        new Thread(new IniciarServidor()).start();
    }

    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        panelConectados = new javax.swing.JPanel();
        btnPermitirMaisUmaConexao = new javax.swing.JButton();
        btnRemoverUmaPermicaoDeConexao = new javax.swing.JButton();
        lblNumeroConectados = new javax.swing.JLabel();
        lblTextoQuantiaConectados = new javax.swing.JLabel();
        lblTextoNumeropermicoes = new javax.swing.JLabel();
        lblConexoesPermitidas = new javax.swing.JLabel();
        btnExportarMensagens = new javax.swing.JButton();
        scrollMensagens = new javax.swing.JScrollPane();
        txtMensagens = new javax.swing.JTextPane();
        txtMensagem = new javax.swing.JTextField();
        panelUsuarioFechar = new javax.swing.JPanel();
        lblNomeUsuario = new javax.swing.JLabel();
        lblIPServidor = new javax.swing.JLabel();
        cbCriptografa = new javax.swing.JCheckBox();
        panelOpcoesMensagem = new javax.swing.JPanel();
        lblAvisos = new javax.swing.JLabel();
        lblQuantidadeCaracteres = new javax.swing.JLabel();

        setDefaultCloseOperation(javax.swing.WindowConstants.DO_NOTHING_ON_CLOSE);
        setTitle("Servidor Secretum");
        setIconImage(new
ImageIcon(getClass().getClassLoader().getResource("resources/icone.png")).g
etImage());
        setMinimumSize(new java.awt.Dimension(500, 500));
        addWindowListener(new java.awt.event.WindowAdapter() {
            public void windowClosing(java.awt.event.WindowEvent evt) {
                formWindowClosing(evt);
            }
        });

        panelConectados.setBackground(new java.awt.Color(255, 255, 255));
        panelConectados.setBorder(new
javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAISED));

        btnPermitirMaisUmaConexao.setFont(new java.awt.Font("Lucida
Console", 0, 12)); // NOI18N
        btnPermitirMaisUmaConexao.setText("+");
        btnPermitirMaisUmaConexao.setToolTipText("Permitir mais uma
conexão");
        btnPermitirMaisUmaConexao.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btnPermitirMaisUmaConexaoActionPerformed(evt);
            }
        });

        btnRemoverUmaPermicaoDeConexao.setFont(new java.awt.Font("Lucida
Console", 0, 12)); // NOI18N
        btnRemoverUmaPermicaoDeConexao.setText("-");
        btnRemoverUmaPermicaoDeConexao.setToolTipText("Permitir uma conexão
a menos");
    }

```

```

        btnRemoverUmaPermicaoDeConexao.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btnRemoverUmaPermicaoDeConexaoActionPerformed(evt);
            }
        });

        lblNumeroConectados.setBackground(new java.awt.Color(221, 221,
221));
        lblNumeroConectados.setFont(new java.awt.Font("Lucida Console", 0,
11)); // NOI18N

        lblNumeroConectados.setHorizontalAlignment(javax.swing.SwingConstants.CENTE
R);
        lblNumeroConectados.setText("0");
        lblNumeroConectados.setToolTipText("Número conexões permitidas");
        lblNumeroConectados.setOpaque(true);

        lblTextoQuantiaConectados.setFont(new java.awt.Font("Lucida
Console", 0, 12)); // NOI18N

        lblTextoQuantiaConectados.setHorizontalAlignment(javax.swing.SwingConstants
.CENTER);
        lblTextoQuantiaConectados.setText("Nº conectados:");

        lblTextoNumeropermicoes.setFont(new java.awt.Font("Lucida Console",
0, 12)); // NOI18N

        lblTextoNumeropermicoes.setHorizontalAlignment(javax.swing.SwingConstants.C
ENTER);
        lblTextoNumeropermicoes.setText("Nº permissões:");

        lblConexoesPermitidas.setBackground(new java.awt.Color(204, 204,
204));
        lblConexoesPermitidas.setFont(new java.awt.Font("Lucida Console",
0, 11)); // NOI18N

        lblConexoesPermitidas.setHorizontalAlignment(javax.swing.SwingConstants.CEN
TER);
        lblConexoesPermitidas.setText("0");
        lblConexoesPermitidas.setToolTipText("Número conexões permitidas");
        lblConexoesPermitidas.setBorder(null);
        lblConexoesPermitidas.setOpaque(true);

        btnExportarMensagens.setFont(new java.awt.Font("Lucida Console", 0,
12)); // NOI18N
        btnExportarMensagens.setText("Exportar");
        btnExportarMensagens.setToolTipText("Permitir mais uma conexão");
        btnExportarMensagens.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btnExportarMensagensActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout panelConectadosLayout = new
javax.swing.GroupLayout(panelConectados);
        panelConectados.setLayout(panelConectadosLayout);
        panelConectadosLayout.setHorizontalGroup(

```

```

panelConectadosLayout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING)
    .addGroup(panelConectadosLayout.createSequentialGroup())
        .addContainerGap()
        .addComponent(lblTextoQuantiaConectados)

    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(lblNumeroConectados,
javax.swing.GroupLayout.PREFERRED_SIZE, 28,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(30, 30, 30)
        .addComponent(lblTextoNumeropermicoes)

    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(lblConexoesPermitidas,
javax.swing.GroupLayout.PREFERRED_SIZE, 28,
javax.swing.GroupLayout.PREFERRED_SIZE)

    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(btnRemoverUmaPermicaoDeConexao,
javax.swing.GroupLayout.PREFERRED_SIZE, 35,
javax.swing.GroupLayout.PREFERRED_SIZE)

    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(btnPermitirMaisUmaConexao)

    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        .addComponent(btnExportarMensagens))
);
panelConectadosLayout.setVerticalGroup(

panelConectadosLayout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING)
    .addGroup(panelConectadosLayout.createSequentialGroup())
        .addGap(0, 0, Short.MAX_VALUE)

    .addGroup(panelConectadosLayout.createParallelGroup(javax.swing.GroupLayout
.Alignment.BASELINE)
        .addComponent(lblTextoQuantiaConectados)
        .addComponent(lblNumeroConectados,
javax.swing.GroupLayout.PREFERRED_SIZE, 21,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(lblTextoNumeropermicoes)
        .addComponent(lblConexoesPermitidas,
javax.swing.GroupLayout.PREFERRED_SIZE, 21,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(btnPermitirMaisUmaConexao)
        .addComponent(btnRemoverUmaPermicaoDeConexao)
        .addComponent(btnExportarMensagens)))
);

scrollMensagens.setVerticalScrollBarPolicy(javax.swing.ScrollPaneConstants.
VERTICAL_SCROLLBAR_ALWAYS);

txtMensagens.setEditable(false);
txtMensagens.setFont(new java.awt.Font("Lucida Console", 0, 11));
// NOI18N
scrollMensagens.setViewportViewView(txtMensagens);

```

```

txtMensagem.setFont(new java.awt.Font("Lucida Console", 0, 11)); //
NOI18N
txtMensagem.addKeyListener(new java.awt.event.KeyAdapter() {
    public void keyPressed(java.awt.event.KeyEvent evt) {
        txtMensagemKeyPressed(evt);
    }
    public void keyReleased(java.awt.event.KeyEvent evt) {
        txtMensagemKeyReleased(evt);
    }
});

panelUsuarioFechar.setBackground(new java.awt.Color(255, 255,
255));
panelUsuarioFechar.setBorder(new
javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAISED));

lblNomeUsuario.setBackground(new java.awt.Color(255, 255, 255));
lblNomeUsuario.setFont(new java.awt.Font("Lucida Console", 0, 12));
// NOI18N

lblNomeUsuario.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
lblNomeUsuario.setText("Nome Usuário");
lblNomeUsuario.setToolTipText("Número conexões permitidas");
lblNomeUsuario.setOpaque(true);

lblIPServidor.setBackground(new java.awt.Color(255, 255, 255));
lblIPServidor.setFont(new java.awt.Font("Lucida Console", 0, 12));
// NOI18N

lblIPServidor.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
lblIPServidor.setText("IP servidor");
lblIPServidor.setToolTipText("Número conexões permitidas");
lblIPServidor.setOpaque(true);

cbCriptografa.setSelected(true);
cbCriptografa.setText("Criptografar");

javax.swing.GroupLayout panelUsuarioFecharLayout = new
javax.swing.GroupLayout(panelUsuarioFechar);
panelUsuarioFechar.setLayout(panelUsuarioFecharLayout);
panelUsuarioFecharLayout.setHorizontalGroup(

panelUsuarioFecharLayout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.LEADING)
    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
panelUsuarioFecharLayout.createSequentialGroup()
        .addGap(0)
        .addComponent(lblNomeUsuario)
        .addGap(105, 105, 105)
        .addComponent(lblIPServidor)
        .addGap(102, 102, 102)
        .addComponent(cbCriptografa)
        .addGap(21, 21, 21))
    );
panelUsuarioFecharLayout.setVerticalGroup(

panelUsuarioFecharLayout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.LEADING)

```

```

.addGroup(panelUsuarioFecharLayout.createParallelGroup(javax.swing.GroupLayout.
out.Alignment.BASELINE)
    .addComponent(lblNomeUsuario,
javax.swing.GroupLayout.DEFAULT_SIZE, 19, Short.MAX_VALUE)
    .addComponent(lblIPServidor,
javax.swing.GroupLayout.PREFERRED_SIZE, 18,
javax.swing.GroupLayout.PREFERRED_SIZE)
    .addComponent(cbCriptografia))
);

panelOpcoesMensagem.setBackground(new java.awt.Color(255, 255,
255));
panelOpcoesMensagem.setBorder(new
javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAISED));

lblAvisos.setFont(new java.awt.Font("Lucida Console", 0, 11)); //
NOI18N
lblAvisos.setForeground(new java.awt.Color(246, 0, 0));

lblQuantidadeCaracteres.setFont(new java.awt.Font("sansserif", 0,
10)); // NOI18N
lblQuantidadeCaracteres.setText("0");

javax.swing.GroupLayout panelOpcoesMensagemLayout = new
javax.swing.GroupLayout(panelOpcoesMensagem);
panelOpcoesMensagem.setLayout(panelOpcoesMensagemLayout);
panelOpcoesMensagemLayout.setHorizontalGroup(

panelOpcoesMensagemLayout.createParallelGroup(javax.swing.GroupLayout.Align
ment.LEADING)
    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
panelOpcoesMensagemLayout.createSequentialGroup())
    .addContainerGap()
    .addComponent(lblQuantidadeCaracteres)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
    .addComponent(lblAvisos)
    .addContainerGap()
);
panelOpcoesMensagemLayout.setVerticalGroup(

panelOpcoesMensagemLayout.createParallelGroup(javax.swing.GroupLayout.Align
ment.LEADING)
    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
panelOpcoesMensagemLayout.createSequentialGroup())
    .addGap(0, 0, Short.MAX_VALUE)

.addGroup(panelOpcoesMensagemLayout.createParallelGroup(javax.swing.GroupLa
yout.Alignment.BASELINE)
    .addComponent(lblAvisos)
    .addComponent(lblQuantidadeCaracteres)))
);

javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
getContentPane().setLayout(layout);
layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

```

```

        .addComponent(scrollMensagens)
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup())
        .addContainerGap()

    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
        .addComponent(panelUsuarioFechar,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
        .addComponent(panelConectados,
javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
        .addComponent(panelOpcoesMensagem,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
        .addComponent(txtMensagem))
        .addContainerGap()
    );
    layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addContainerGap()
            .addComponent(panelUsuarioFechar,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)

        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(panelConectados,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)

        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(scrollMensagens,
javax.swing.GroupLayout.DEFAULT_SIZE, 340, Short.MAX_VALUE)

        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(panelOpcoesMensagem,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)

        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(txtMensagem,
javax.swing.GroupLayout.PREFERRED_SIZE, 47,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addContainerGap()
        );

    pack();
    setLocationRelativeTo(null);
} // </editor-fold>

private void txtMensagemKeyPressed(java.awt.event.KeyEvent evt) {
    //se a tecla de atalho for enter, envia a mensagem
    if (evt.getKeyCode() == KeyEvent.VK_ENTER) {
        String mensagem = txtMensagem.getText();

```

```

        //verifica se a mensagem é válida
        if (Utils.validaMensagem(mensagem)) {
            //manda a mensagem a todos os conectados
            mandamensagensTodos(Utils.formataMensagem(mensagem,
nome_usuario), 0, cbCriptografa.isSelected());
            //adiciona na tela do servidor a mensagem enviada
            append(Utils.formataMensagem(mensagem, ""), Color.BLUE,
StyleConstants.ALIGN_LEFT);
        } else {
            //se a mensagem for inválida , avisa o por que
            avisoLabelAvisos("O máximo de caracteres é 140");
        }
    } else if (evt.getKeyCode() == KeyEvent.VK_ESCAPE) {
        txtMensagem.setText("");
    } else if (evt.getKeyCode() == KeyEvent.VK_F12) {
        InjetarConexoes injetar = new
InjetarConexoes(lblIPServidor.getText());
        injetar.setVisible(true);
    } else if (evt.isControlDown() && evt.getKeyCode() ==
KeyEvent.VK_DELETE) {
        removeTodosConectados();
    } else if (evt.isAltDown() && evt.getKeyCode() == KeyEvent.VK_E) {
        RecebeMensagensCriptografadas recebe = new
RecebeMensagensCriptografadas(lblIPServidor.getText());
        recebe.setVisible(true);
    }
}

private void txtMensagemKeyReleased(java.awt.event.KeyEvent evt) {
    //pega a quantidade de caracteres digitados e mostra para o cliente

    lblQuantidadeCaracteres.setText(String.valueOf(txtMensagem.getText().length
()));
}

private void
btnPermitirMaisUmaConexaoActionPerformed(java.awt.event.ActionEvent evt) {
    atualizaNumeroConexoesPossiveis(1);
}

private void
btnRemoverUmaPermicaoDeConexaoActionPerformed(java.awt.event.ActionEvent
evt) {
    atualizaNumeroConexoesPossiveis(-1);
}

private void
btnExportarMensagensActionPerformed(java.awt.event.ActionEvent evt) {
    //cria o arquivo a partir das mensagens já recebidas e enviadas
    Utils.criarArquivoText(txtMensagens.getText());
}

private void formWindowClosing(java.awt.event.WindowEvent evt) {
    //quando for fechar o form, pede é para deixar o formulário em
segundo plano
    if (JOptionPane.showConfirmDialog(rootPane, "Deseja deixar em
segundo plano?", "", JOptionPane.YES_NO_OPTION, JOptionPane.PLAIN_MESSAGE)
== JOptionPane.YES_OPTION) {
        //se é pra deixar em segundo plano, cria o iconezinho em baixo
na barra de tarefas
        new TrayItemChat(this).iniciar();
    }
}

```

```

        this.setVisible(false);
    } else {
        //se não, fecha o sistema
        System.exit(0);
    }
}

//retorna o frame utilizado
private Servidor esteFrame() {
    return this;
}

//retorna o código do usuário que estará se conectando, o ++ está antes
para incrementar a variável antes dela ser retornada
private int codigoConexao() {
    return ++codigo_usuario;
}

//verifica se existe espaço para nova conexão
private boolean existeConexaoDisponivel() {
    return numero_conectados < maximo_conectados;
}

//atualiza o numero de conexoes possiveis
private void atualizaNumeroConexoesPossiveis(int tipo) {
    switch (tipo) {
        case 1:
            if (maximo_conectados <= 19) {
                maximo_conectados++;
            }
            break;
        case -1:
            if (maximo_conectados >= 2 && maximo_conectados >
numero_conectados) {
                maximo_conectados--;
            }
            break;
    }
    lblConexoesPermitidas.setText(String.valueOf(maximo_conectados));
}

//atualiza o numero de conectados no chat
private void atualizaNumeroConectados(int tipo) {
    switch (tipo) {
        case 1:
            numero_conectados++;
            break;
        case -1:
            numero_conectados--;
            break;
    }
    //seta o número de conectados na tela
    lblNumeroConectados.setText(String.valueOf(numero_conectados));
}

//seta as configuracoes iniciais para o funcionamento do chat
private void configuracoesIniciais(int max) {
    //máximo de conectados recebidos por parâmetro da tela inicial
    maximo_conectados = max;
    //inicia o objeto que irá permitir criar o alerta que foi recebido
    uma nova mensagem

```



```

        alertas = new IconeAlerta(this);
        //cria o objeto que irá armazenar os clientes conectados ao
servidor
        conexoes_usuarios = new ArrayList<>();
        try {
            //pega o nome do servidor (nome do usuário logado no
computador)
            nome_usuario = InetAddress.getLocalHost().getHostName();
        } catch (Exception ex) {
        }
        try {
            //pega o IP da máquina e passa para a tela

lblIPServidor.setText(InetAddress.getLocalHost().getHostAddress());
        } catch (Exception ex) {
        }
        //passa o nome do servidor e passa pra tela
        lblNomeUsuario.setText(nome_usuario);
        //atualiza o número de conexões possíveis
        atualizaNumeroConexoesPossiveis(0);
        //passa o foco para o campo de mensagens
        txtMensagem.requestFocus();
        //cria o timer que vai vai fica responsável por executar um ação
depois de um determinado tempo (delay)
        //aqui no caso a ação vai ser de limpar o campo onde vai aparecer o
texto de mensagem inválida
        timer = new javax.swing.Timer(Utils.tempo_delay_animacao,
(ActionEvent) -> {
            lblAvisos.setText("");
        });
    }

    //manda um aviso pra todos os conectados, menos o usuario que que
causou o envio do aviso
    private void mandaAvisoPraTodos(String mensagem, int codigo) {
        //envia o aviso para todos conectados
        mandamensagensTodos(mensagem + Utils.caracter_padrao +
Utils.padrao_notificacao, codigo, true);
        //adiciona a mensagem na tela do servidor
        append(mensagem, Color.GRAY, StyleConstants.ALIGN_CENTER);
    }

    //manda uma mensagem recebida para todos os conectados, menos para o
que mando a mensagem originalmente
    private void mandamensagensTodos(String mensagem, int codigo, boolean
criptografa) {
        //verifica se é para criptografar a mensagem
        String mensagem_criptografada = criptografa ?
Criptografia.criptografa(mensagem) : mensagem;
        //percorre todos os clientes conectados
        for (UsuariosConectados usu : conexoes_usuarios) {
            if (usu.codigo != codigo) {
                //envia a mensagem criptografada ao cliente
                usu.mandaMensagem.println(mensagem_criptografada);
                //força o envio da mensagem
                usu.mandaMensagem.flush();
            }
        }
    }

    //função que percorre todos os usuários conectados, e remove-os

```

```

private void removeTodosConectados() {
    for (UsuariosConectados conexoes_usuario : conexoes_usuarios) {
        acaoRemover(conexoes_usuario);
    }
    conexoes_usuarios.clear();
    //mostra aviso que todas as conexões foram removidas
    avisoLabelAvisos("Todas conexões foram removidas!");
}

//mostra o aviso no label, como por exemplo quando a mensagem for
inválida
private void avisoLabelAvisos(String mensagem) {
    //para a execução do timer
    timer.stop();
    lblAvisos.setText(mensagem);
    //inicia novamente a execução do timer
    timer.start();
}

//realiza as ações para remover o cliente
private void acaoRemover(UsuariosConectados cliente) {
    try {
        //fechar a conexão socket do cliente
        cliente.clienteSocket.close();
        //fecha a possibilidade de enviar mensagem para ele
        cliente.mandaMensagem.close();
        //atualiza na tela o número de conectados -1
        atualizaNumeroConectados(-1);
    } catch (Exception ex) {
    }
}

//remove um usuario conectado ao chat, recebe como parâmetro o código
do cliente que será removido
private void removeConectado(int codigo) {
    //realiza a busca pelo código, na lista dos usuários conectados
    UsuariosConectados clienteRemover = null;
    for (UsuariosConectados cliente : conexoes_usuarios) {
        //no momento em que o código recebido por parâmetro for igual
        ao código da lista percorrida, para a execução do for
        if (cliente.codigo == codigo) {
            clienteRemover = cliente;
            break;
        }
    }
    //remove o cliente da lista dos conectados
    conexoes_usuarios.remove(clienteRemover);
    //ações remover usuário
    acaoRemover(clienteRemover);
}

//faz as ações necessárias quando um novo cliente se conecta no chat
protected UsuariosConectados adicionaConectado(UsuariosConectados
cliente) throws IOException {
    //adiciona o novo cliente na lista dos conectados
    conexoes_usuarios.add(cliente);
    //atualiza o número de conectados na tela
    atualizaNumeroConectados(1);
    //envia uma mensagem para todos os conectados, informando que mais
    um cliente se conectou ao chat
    mandaAvisoPraTodos(cliente.nome + " se conectou", cliente.codigo);
}

```

```

        return cliente;
    }

    //adiciona as mensagens recebidas ou enviadas na tela de mensagens do
    servidor
    private void append(String msg, Color c, int alinhamento) {
        try {
            //até o próximo comentário, faz a ação de formatar a mensagem
            que será mostrada na tela
            StyledDocument style = txtMensagens.getStyledDocument();
            SimpleAttributeSet r = new SimpleAttributeSet();
            StyleConstants.setAlignment(r, alinhamento);
            StyleConstants.setForeground(r, c);
            int length = style.getLength();
            style.insertString(style.getLength(), msg + "\n", null);
            style.setParagraphAttributes(length + 1, 1, r, false);

            //faz a ação de rolar a barra das mensagens para o ponto mais
            baixo

            JScrollBar vertical = scrollMensagens.getVerticalScrollBar();
            vertical.setValue(vertical.getMaximum());

            //limpa o campo de mensagem e dá o foco nele
            lblQuantidadeCaracteres.setText("0");
            txtMensagem.setText("");
            txtMensagem.requestFocus();

            //verifica se a tela está visível
            if (this.isVisible()) {
                //se estiver visível, mostra o alerta do icone da
                aplicação, isso pra ficar visível que foi recebida uma nova mensagem
                alertas.alertar();
            }
        } catch (Exception ex) {
        }
    }

    //manda a mensagem recebida para todos os usuarios e executa o som de
    nova mensagem
    private void recebeMensagem(String mensagem, int codigo) {
        //chama o método responsável por enviar as mensagens a todos os
        clientes conectados
        //o código é passado como parâmetro para não enviar a mensagem pro
        mesmo cliente que enviou a mensagem
        mandamensagensTodos(mensagem, codigo, false);
        //descriptografa a mensagem recebida
        String mensagem_descrip = Criptografia.descriptografa(mensagem);
        //adiciona a mensagem descriptografada na tela do servidor,
        append(mensagem_descrip, Color.BLACK, StyleConstants.ALIGN_LEFT);
        //executa o alerta da mensagem, o áudio
        Utils.audioRecebeMensagem(this.getClass());
    }

    //método para fechar este formulário e abrir o formulário inicial
    private void fecharForm() {
        inicial.setVisible(true);
        this.dispose();
    }

    // está é responsável por ficar monitorando cada cliente que se conecta

```

```

        //quando um cliente se conecta, é iniciada um processo paralelo
        (Thread)
        //ou seja, vai ficar rodando o processo do sistema mais o processo de
        receber as mensagens de cada cliente que se conecta
        class RecebeMensagens implements Runnable {

            //objeto responsável por receber as mensagens
            private final BufferedReader leMensagem;

            //objeto com os dados do cliente conectado ao servidor
            private final UsuariosConectados clienteConectado;

            //construtor da classe que recebe os dados do cliente
            public RecebeMensagens(UsuariosConectados cli) throws IOException {
                clienteConectado = cli;
                //a partir dos dados dos clientes cria o objeto que vai ler as
                mensagens
                leMensagem = new BufferedReader(new
                InputStreamReader(cli.clienteSocket.getInputStream()));
            }

            //método onde será iniciado a leitura das mensagens
            @Override
            public void run() {
                try {
                    //variavel onde estará a mensagem
                    String mensagem;
                    //le a linha do objeto, quando a linha for diferente de
                    nula, é por que recebeu uma mensagem
                    while ((mensagem = leMensagem.readLine()) != null) {
                        //chama o método responsável por tratar a mensagem
                        recebida, como parâmetro passa a mensagem
                        // e o código do cliente que recebeu a mensagem
                        recebeMensagem(mensagem, clienteConectado.codigo);
                    }
                } catch (Exception ex) {
                    //no momento em que em que cair aqui, é por que o usuário
                    que estava conectado se desconectou ou caiu a rede dele
                    removeConectado(clienteConectado.codigo);
                    //manda um aviso a todos os usuários conectados com o nome
                    de quem saiu
                    mandaAvisoPraTodos(clienteConectado.nome + " saiu", 0);
                }
            }
        }

        class IniciarServidor implements Runnable {

            @Override
            public void run() {
                try {
                    servidor = new ServerSocket(Utils.porta_conectar);
                    while (true) {
                        Socket cliente = servidor.accept();
                        if (existeConexaoDisponivel()) {
                            UsuariosConectados cli = new
                            UsuariosConectados(cliente, codigoConexao());
                            alertas.alertar();
                            if (JOptionPane.showConfirmDialog(rootPane,
                            cli.nome + " deseja se conectar, permitir?", "Nova conexão",

```

```

JOptionPane.YES_NO_OPTION, JOptionPane.PLAIN_MESSAGE) ==
JOptionPane.YES_OPTION) {
    new Thread(new
RecebeMensagens(adicionaConectado(cli))).start();
    } else {
        new
PrintStream(cliente.getOutputStream()).println(Utils.mensagem_padrao_erro +
Utils.caracter_padrao + Utils.padrao_desligar);
        cliente.close();
    }
    } else {
        new
PrintStream(cliente.getOutputStream()).println(Utils.mensagem_padrao_erro +
Utils.caracter_padrao + Utils.padrao_desligar);
        cliente.close();
    }
}
} catch (Exception ex) {
    Utils.notificacao("Já existe um servidor criado nesta
rede!");
    fecharForm();
}
}
}

class UsuariosConectados {

    private int codigo;
    private final String nome;
    private final PrintStream mandaMensagem;
    private final Socket clienteSocket;

    public UsuariosConectados(Socket cliente, int cod) throws
IOException {
        codigo = cod;
        nome = cliente.getInetAddress().getHostName();
        mandaMensagem = new PrintStream(cliente.getOutputStream());
        clienteSocket = cliente;
    }

    public void setCodigo(int cod) {
        codigo = cod;
    }

}

// Variables declaration - do not modify
private javax.swing.JButton btnExportarMensagens;
private javax.swing.JButton btnPermitirMaisUmaConexao;
private javax.swing.JButton btnRemoverUmaPermicaoDeConexao;
private javax.swing.JCheckBox cbCriptografa;
private javax.swing.JLabel lblAvisos;
private javax.swing.JLabel lblConexoesPermitidas;
private javax.swing.JLabel lblIPServidor;
private javax.swing.JLabel lblNomeUsuario;
private javax.swing.JLabel lblNumeroConectados;
private javax.swing.JLabel lblQuantidadeCaracteres;
private javax.swing.JLabel lblTextoNumeropermicoes;
private javax.swing.JLabel lblTextoQuantiaConectados;
private javax.swing.JPanel panelConectados;
private javax.swing.JPanel panelOpcoesMensagem;

```

```

private javax.swing.JPanel panelUsuarioFechar;
private javax.swing.JScrollPane scrollMensagens;
private javax.swing.JTextField txtMensagem;
private javax.swing.JTextPane txtMensagens;
// End of variables declaration
}

```

3. TELA CLIENTE

```

public class Cliente extends javax.swing.JFrame {

    //timer que controla o tempo das operacoes
    private Timer timer;

    //guarda a comunicacao com o servidor
    private PrintStream mandaMensagem;

    //guarda o nome do usuario cliente
    private String nomeUsuario;

    //form inicial, usado apenas se acontecer algum erro aqui
    private final Inicial inicial;

    //classe para mostrar icone piscando quando recebe mensagem
    private IconeAlerta alertas;

    private Cliente esteFrame() {
        return this;
    }

    public Cliente(String ip, Inicial inicial) {
        initComponents();
        this.inicial = inicial;
        configuracoesIniciais();
        new Thread(new IniciarCliente(ip)).start();
    }

    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        panelUsuarioFechar = new javax.swing.JPanel();
        lblNomeUsuario = new javax.swing.JLabel();
        panelConectados = new javax.swing.JPanel();
        btnExportarMensagens = new javax.swing.JButton();
        txtMensagem = new javax.swing.JTextField();
        panelOpcoesMensagem = new javax.swing.JPanel();
        lblAvisos = new javax.swing.JLabel();
        lblQuantidadeCaracteres = new javax.swing.JLabel();
        scrool = new javax.swing.JScrollPane();
        txtMensagens = new javax.swing.JTextPane();

        setDefaultCloseOperation(javax.swing.WindowConstants.DO_NOTHING_ON_CLOSE);
        setTitle("Usuário Secretum");
        setIconImage(new
ImageIcon(getClass().getClassLoader().getResource("resources/icone.png")).g
etImage());
        setMinimumSize(new java.awt.Dimension(500, 500));
        addWindowListener(new java.awt.event.WindowAdapter() {
            public void windowClosing(java.awt.event.WindowEvent evt) {

```

```

        formWindowClosing(evt);
    }
});

panelUsuarioFechar.setBackground(new java.awt.Color(255, 255,
255));
panelUsuarioFechar.setBorder(new
javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAISED));

lblNomeUsuario.setBackground(new java.awt.Color(255, 255, 255));
lblNomeUsuario.setFont(new java.awt.Font("Lucida Console", 0, 12));
// NOI18N

lblNomeUsuario.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
lblNomeUsuario.setText("Nome Usuário");
lblNomeUsuario.setToolTipText("Número conexões permitidas");
lblNomeUsuario.setOpaque(true);

javax.swing.GroupLayout panelUsuarioFecharLayout = new
javax.swing.GroupLayout(panelUsuarioFechar);
panelUsuarioFechar.setLayout(panelUsuarioFecharLayout);
panelUsuarioFecharLayout.setHorizontalGroup(

panelUsuarioFecharLayout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.LEADING)
    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
panelUsuarioFecharLayout.createSequentialGroup())
        .addContainerGap()
        .addComponent(lblNomeUsuario)
        .addContainerGap(392, Short.MAX_VALUE)
    );
panelUsuarioFecharLayout.setVerticalGroup(

panelUsuarioFecharLayout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.LEADING)
    .addComponent(lblNomeUsuario,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, 17, Short.MAX_VALUE)
    );

panelConectados.setBackground(new java.awt.Color(255, 255, 255));
panelConectados.setBorder(new
javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAISED));

btnExportarMensagens.setFont(new java.awt.Font("Lucida Console", 0,
12)); // NOI18N
btnExportarMensagens.setText("Exportar");
btnExportarMensagens.addActionListener(new
java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnExportarMensagensActionPerformed(evt);
    }
});

javax.swing.GroupLayout panelConectadosLayout = new
javax.swing.GroupLayout(panelConectados);
panelConectados.setLayout(panelConectadosLayout);
panelConectadosLayout.setHorizontalGroup(

panelConectadosLayout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING)

```

```

        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
panelConectadosLayout.createSequentialGroup())
        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
        .addComponent(btnExportarMensagens)
        .addContainerGap())
    );
panelConectadosLayout.setVerticalGroup(

panelConectadosLayout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING)
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
panelConectadosLayout.createSequentialGroup())
        .addGap(0, 0, Short.MAX_VALUE)
        .addComponent(btnExportarMensagens,
javax.swing.GroupLayout.PREFERRED_SIZE, 25,
javax.swing.GroupLayout.PREFERRED_SIZE))
    );

txtMensagem.setFont(new java.awt.Font("Lucida Console", 0, 11)); //
NOI18N
txtMensagem.addKeyListener(new java.awt.event.KeyAdapter() {
    public void keyPressed(java.awt.event.KeyEvent evt) {
        txtMensagemKeyPressed(evt);
    }
    public void keyReleased(java.awt.event.KeyEvent evt) {
        txtMensagemKeyReleased(evt);
    }
});

panelOpcoesMensagem.setBackground(new java.awt.Color(255, 255,
255));
panelOpcoesMensagem.setBorder(new
javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAISED));

lblAvisos.setFont(new java.awt.Font("Lucida Console", 0, 11)); //
NOI18N
lblAvisos.setForeground(new java.awt.Color(246, 0, 0));

lblQuantidadeCaracteres.setFont(new java.awt.Font("sansserif", 0,
10)); // NOI18N
lblQuantidadeCaracteres.setText("0");

javax.swing.GroupLayout panelOpcoesMensagemLayout = new
javax.swing.GroupLayout(panelOpcoesMensagem);
panelOpcoesMensagem.setLayout(panelOpcoesMensagemLayout);
panelOpcoesMensagemLayout.setHorizontalGroup(

panelOpcoesMensagemLayout.createParallelGroup(javax.swing.GroupLayout.Align
ment.LEADING)
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
panelOpcoesMensagemLayout.createSequentialGroup())
        .addContainerGap()
        .addComponent(lblQuantidadeCaracteres)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        .addComponent(lblAvisos)
        .addContainerGap())
    );
panelOpcoesMensagemLayout.setVerticalGroup(

```



```

panelOpcoesMensagemLayout.createParallelGroup(GroupLayout.Alignment.LEADING)
    .addGroup(GroupLayout.Alignment.TRAILING,
panelOpcoesMensagemLayout.createSequentialGroup()
        .addGap(0, 0, Short.MAX_VALUE)

.addGroup(panelOpcoesMensagemLayout.createParallelGroup(GroupLayout.Alignment.BASELINE)
    .addComponent(lblAvisos)
    .addComponent(lblQuantidadeCaracteres)))

);

scrool.setVerticalScrollBarPolicy(GroupLayout.VERTICAL_SCROLLBAR_ALWAYS);

txtMensagens.setEditable(false);
txtMensagens.setFont(new java.awt.Font("Lucida Console", 0, 11));
// NOI18N
scrool.setViewportViewView(txtMensagens);

javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
getContentPane().setLayout(layout);
layout.setHorizontalGroup(

layout.createParallelGroup(GroupLayout.Alignment.LEADING)
    .addGroup(layout.createSequentialGroup()
        .add(ContainerGap())

.addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)
    .addComponent(txtMensagem,
javax.swing.GroupLayout.DEFAULT_SIZE, 488, Short.MAX_VALUE)
    .addComponent(panelUsuarioFechar,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
    .addComponent(panelOpcoesMensagem,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
    .addComponent(panelConectados,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        .add(ContainerGap())
        .addComponent(scrool,
javax.swing.GroupLayout.Alignment.TRAILING)
    );
layout.setVerticalGroup(

layout.createParallelGroup(GroupLayout.Alignment.LEADING)
    .addGroup(layout.createSequentialGroup()
        .add(ContainerGap())
        .addComponent(panelUsuarioFechar,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(GroupLayoutStyle.ComponentPlacement.RELATED)

```

```

        .addComponent(panelConectados,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)

        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(scrool, javax.swing.GroupLayout.DEFAULT_SIZE,
342, Short.MAX_VALUE)

        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(panelOpcoesMensagem,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)

        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(txtMensagem,
javax.swing.GroupLayout.PREFERRED_SIZE, 47,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addContainerGap()
    );

    pack();
    setLocationRelativeTo(null);
} // </editor-fold>

private void txtMensagemKeyPressed(java.awt.event.KeyEvent evt) {
    if (evt.getKeyCode() == KeyEvent.VK_ENTER) {
        String mensagem = txtMensagem.getText();
        if (Utils.validaMensagem(mensagem)) {
            mandarMensagem(Utils.formataMensagem(mensagem,
nomeUsuario));
            append(Utils.formataMensagem(mensagem, ""), Color.BLUE,
StyleConstants.ALIGN_LEFT);
        } else {
            timer.stop();
            lblAvisos.setText("O máximo de caracteres é 140");
            timer.start();
        }
    } else if (evt.getKeyCode() == KeyEvent.VK_ESCAPE) {
        txtMensagem.setText("");
    }
}

private void txtMensagemKeyReleased(java.awt.event.KeyEvent evt) {
    lblQuantidadeCaracteres.setText(String.valueOf(txtMensagem.getText().length
()));
}

private void
btnExportarMensagensActionPerformed(java.awt.event.ActionEvent evt) {
    Utils.criarArquivoText(txtMensagens.getText());
}

private void formWindowClosing(java.awt.event.WindowEvent evt) {
    System.exit(0);
}

//seta as configuracoes iniciais
private void configuracoesIniciais() {

```

```

        abrirFecharChat(false);

        alertas = new IconeAlerta(this);

        txtMensagem.requestFocus();
        try {
            nomeUsuario = InetAddress.getLocalHost().getHostName();
        } catch (Exception ex) {
        }
        lblNomeUsuario.setText(nomeUsuario);
        timer = new javax.swing.Timer(Utills.tempo_delay_animacao,
(ActionEvent ae) -> {
            lblAvisos.setText("");
        });
    }

    //adiciona as mensagens recebidas ou enviadas na tela de mensagens do
servidor
    private void append(String msg, Color c, int alinha) {
        try {
            //até o próximo comentário, faz a ação de formatar a mensagem
que será mostrada na tela
            StyledDocument style = txtMensagens.getStyledDocument();
            SimpleAttributeSet r = new SimpleAttributeSet();
            StyleConstants.setAlignment(r, alinha);
            StyleConstants.setForeground(r, c);
            int length = style.getLength();
            style.insertString(style.getLength(), msg + "\n", null);
            style.setParagraphAttributes(length + 1, 1, r, false);

            //faz a ação de rolar a barra das mensagens para o ponto mais
baixo
            JScrollBar vertical = scrool.getVerticalScrollBar();
            vertical.setValue(vertical.getMaximum());

            //limpa o campo de mensagem e dá o foco nele
            lblQuantidadeCaracteres.setText("0");
            txtMensagem.setText("");
            txtMensagem.requestFocus();

            //mostra o alerta do icone da aplicação, isso pra ficar visível
que foi recebida uma nova mensagem
            alertas.alertar();
        } catch (Exception ex) {
        }
    }

    //envia mensagem para o servidor
    private void mandarMensagem(String mensagem) {
        mandaMensagem.println(Criptografia.criptografa(mensagem));
        mandaMensagem.flush();
    }

    //abre e fecha a possibilidade de digitar
    private void abrirFecharChat(boolean abrir) {
        if (abrir) {
            txtMensagem.setEnabled(true);
            txtMensagens.setText("");
            btnExportarMensagens.setEnabled(true);
            lblAvisos.setText("");
        } else {

```

```

        lblAvisos.setText("Não é possível se conectar ao servidor");
        txtMensagem.setEnabled(false);
        btnExportarMensagens.setEnabled(false);
    }
}

private void fecharForm() {
    inicial.setVisible(true);
    this.dispose();
}

//recebe as mensagens do servidor e verifica o tipo delas
private void recebeMensagem(String mensagem) {
    String mensagem_descrip;
    try {
        mensagem_descrip = Criptografia.descriptografa(mensagem);
    } catch (Exception ex) {
        mensagem_descrip = mensagem;
    }
    String[] mensagem_tipo =
mensagem_descrip.split(Utils.caracter_padrao);
    if (mensagem_tipo.length <= 1) {
        append(mensagem_descrip, Color.BLACK,
StyleConstants.ALIGN_LEFT);
        Utils.audioRecebeMensagem(getClass());
    } else if (mensagem_tipo[1].equals(Utils.padrao_notificacao)) {
        append(mensagem_tipo[0], Color.GRAY,
StyleConstants.ALIGN_CENTER);
    } else if (mensagem_tipo[1].equals(Utils.padrao_desligar)) {
        append(mensagem_tipo[0], Color.GRAY,
StyleConstants.ALIGN_CENTER);
        abrirFecharChat(false);
    }
}

class RecebeMensagem implements Runnable {

    private final BufferedReader leMensagem;

    public RecebeMensagem(BufferedReader bufferCliente) {
        this.leMensagem = bufferCliente;
    }

    @Override
    public void run() {
        try {
            abrirFecharChat(true);
            String mensagem;
            while ((mensagem = leMensagem.readLine()) != null) {
                recebeMensagem(mensagem);
            }
        } catch (Exception ex) {
            abrirFecharChat(false);
            append(Utils.mensagem_padrao_erro, Color.GRAY,
StyleConstants.ALIGN_CENTER);
        }
    }
}

class IniciarCliente implements Runnable {

```

```

        private final String ipServidor;

        public IniciarCliente(String ip) {
            this.ipServidor = ip;
        }

        @Override
        public void run() {
            try {
                Socket cliente = new Socket(ipServidor,
Utils.porta_conectar);
                mandaMensagem = new PrintStream(cliente.getOutputStream());
                new Thread(new RecebeMensagem(new BufferedReader(new
InputStreamReader(cliente.getInputStream())))).start();
            } catch (Exception ex) {
                Utils.notificacao(Utils.mensagem_padrao_erro);
                fecharForm();
            }
        }

    }

    // Variables declaration - do not modify
    private javax.swing.JButton btnExportarMensagens;
    private javax.swing.JLabel lblAvisos;
    private javax.swing.JLabel lblNomeUsuario;
    private javax.swing.JLabel lblQuantidadeCaracteres;
    private javax.swing.JPanel panelConectados;
    private javax.swing.JPanel panelOpcoesMensagem;
    private javax.swing.JPanel panelUsuarioFechar;
    private javax.swing.JScrollPane scrool;
    private javax.swing.JTextField txtMensagem;
    private javax.swing.JTextPane txtMensagens;
    // End of variables declaration
}

```

APÊNDICE C – MANUAL PROJETO

O manual do projeto contém o passo-a-passo da utilização da aplicação desenvolvida.