

CLIENTE DE MENSAJERIA

CURSO:

INGENIERIA DE SOFTWARRE III

DOCENTE:

ING. IVAN SORIA SOLIS

ALUMNOS:

ANDREA MARGOT OCHOA TAPIA

NILO APARCO AREVALO

Tabla de contenido

**INTRODUCCION3**

1. **REQUERIMIENTO4**
2. **DEFINICIÓN DE NETBEANS4**
3. **IMPLEMENTACION4**
   1. servidor**4**
   2. cliente**12**

**INTRODUCCIÓN**

Si nos remontamos a la época prehistórica de la computación, en lugares selectos de los años ’70 y ‘80, distintas personas podían estar a varios kilómetros de distancia y sin embargo, charlar e intercambiar ideas en tiempo real por medio de una conexión a un servidor central o a distintos servidores. Ya sea mediante el programa *write* o el *talk*, esta forma de comunicación se basó en la necesidad de enviar y responder a mensajes, en distintas terminales Unix. Y así, como con tantas cosas que nacieron con este Sistema Operativo (Internet, el lenguaje C, Sistemas Multiprocesamiento/Multiusuario, etc.), la mensajería instantánea tal como la conocemos ahora se podría pensar que también.

La Mensajería Instantánea permite intercambiar mensajes de texto entre usuarios que previamente han aceptado comunicarse entre sí de esta manera. El procedimiento varía de un sistema a otro pero, en general, funciona así:

1. Un usuario se conecta a un [servidor](http://www.ecured.cu/index.php/Servidor), en el cual está almacenada su lista de [contactos](http://www.ecured.cu/index.php/Contactos), y se establece su estado: disponible, ocupado, ausente, etc.
2. El servidor le envía su lista de contactos y el estado de cada uno de ellos.
3. El servidor, automáticamente, informa de la presencia de este usuario a todos los usuarios de su lista de contactos que estén conectados en ese momento.
4. A partir de este momento, si un usuario quiere comunicarse con alguno de sus contactos, no tiene más que seleccionar el usuario deseado.
5. Para dar de alta un contacto en la lista de contactos, hace falta saber su dirección o alias y que el contacto autorice la inclusión.
6. Cuando el usuario cierra su programa de Mensajería Instantánea, el programa informa al servidor de la desconexión y éste, a su turno, notifica a todos los contactos. La mensajería instantánea requiere el uso de un cliente de mensajería instantánea que realiza el servicio y se diferencia del [correo electrónico](http://www.ecured.cu/index.php/Correo_electr%C3%B3nico) en que las conversaciones se realizan en tiempo real.

La mayoría de los servicios ofrecen el aviso de presencia, indicando cuando el cliente de una persona en la lista de contactos se conecta o en qué estado se encuentra, si está disponible para tener una conversación.

1. **REQUERIMIENTOS**

El propósito de este manual es facilitar al usuario la operación del programa que es un cliente de mensajería

* 1. Implementación del programa

1. Requerimientos de hardware

Contar con:

Computadora personal

1. Requerimiento de software

Contar con:

* Sistema operativo Windows.
* Netbeans IDE 6.9.1

1. **DEFINICIÓN DE NETBEANS**

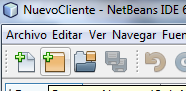
**NetBeans** es un [entorno de desarrollo integrado](http://es.wikipedia.org/wiki/Entorno_de_desarrollo_integrado) [libre](http://es.wikipedia.org/wiki/Software_libre), hecho principalmente para el [lenguaje de programación](http://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n) [Java](http://es.wikipedia.org/wiki/Java_(lenguaje_de_programaci%C3%B3n)). Existe además un número importante de módulos para extenderlo. NetBeans IDEes un producto libre y gratuito sin restricciones de uso.

La plataforma NetBeans permite que las aplicaciones sean desarrolladas a partir de un conjunto de [componentes de software](http://es.wikipedia.org/wiki/Componente_de_software) llamados *módulos*. Un módulo es un archivo Java que contiene clases de java escritas para interactuar con las APIs de NetBeans y un archivo especial (manifest file) que lo identifica como módulo. Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándole nuevos módulos. Debido a que los módulos pueden ser desarrollados independientemente, las aplicaciones basadas en la plataforma NetBeans pueden ser extendidas fácilmente por otros desarrolladores de software.

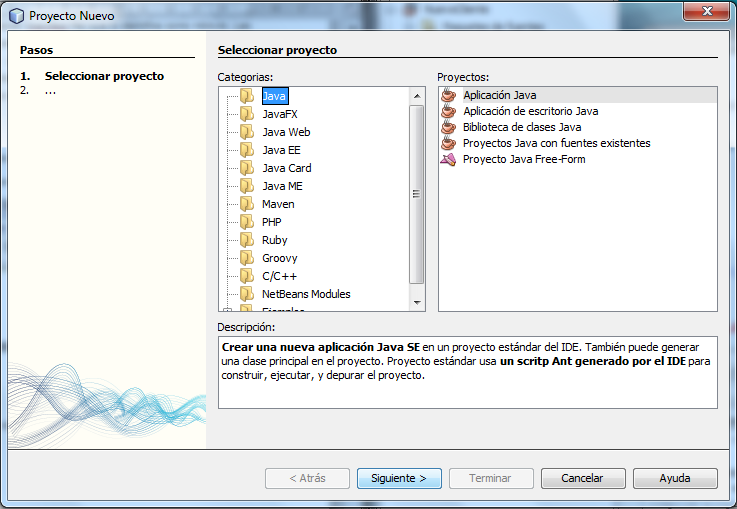
1. **IMPLEMENTACIÓN** 
   1. **Servidor**

Para el servidor debemos crear un proyecto al cual lo llamaremos NuevoServidor

Para ello hacemos clic en el botón nuevo proyecto como se muestra

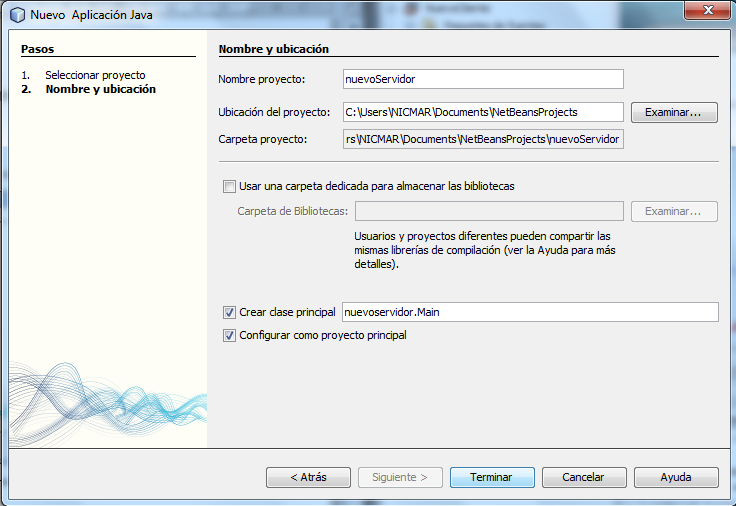


Luego nos aparecerá la siguiente pantalla



Le damos clic en siguiente

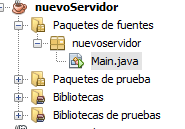
Luego nos aparecerá la siguiente ventana



Escribir en nombre del proyecto “nuevoservidor”

Luego clic en terminar

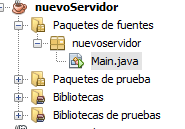
Luego nos aparece en el lado izquierdo de la ventana el proyecto que hemos creado como se puede apreciar



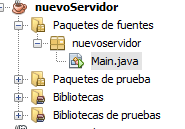
Como ya tenemos el main solo creriamos un formulario y una clase para manejar el servidor

1. **Creando el formulario**

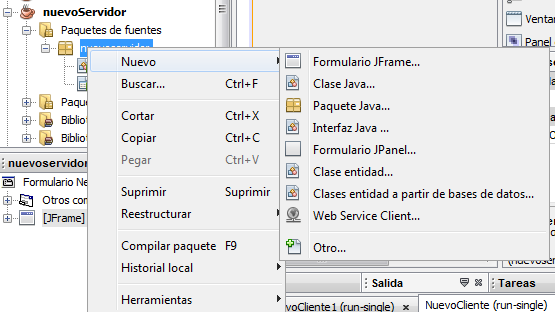
Para crear el formulario deglosamos el paquetes de fuentes



En nuevo servidor le damos clic derecho



Hacemos clic derecho en nuevo y por ultimo clic en formulario Jframe como se aprecia en la imagen



Al cual lo llamaremos ventana

Cuando abrimos el formulario hacemos clic en diseño



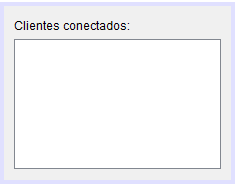
Donde nos aprecera la siguiente pantalla



Donde diseñaremos nuestra ventana de servidor para lo cual insertaremos de la paleta “lista” y “label”

Y lo diseñaremos de la siguiente manera



En la parte del código “fuente” insertaremos en siguiente código

package nuevoservidor;

import javax.swing.\*;

public class Ventana extends javax.swing.JFrame {

DefaultListModel listaModelo;

public Ventana() {

listaModelo = new DefaultListModel();

initComponents();

agregarElementoLista("\*\*clientes conectados\*\*");

// quitarElementoLista(1); }

public void agregarElementoLista(String elemento){

int index = jList1.getSelectedIndex(); //get selected index

if (index == -1) { //no selection, so insert at beginning

index = 0;

} else { //add after the selected item

index++; }

listaModelo.insertElementAt(elemento, index); }

public void quitarElementoLista(int posicion){

listaModelo.removeElementAt(posicion);

public static void main(String args[]) {

java.awt.EventQueue.invokeLater(new Runnable() {

public void run() {

new Ventana().setVisible(true); } }); }

// Variables declaration - do not modify

private javax.swing.JLabel jLabel1;

public javax.swing.JList jList1;

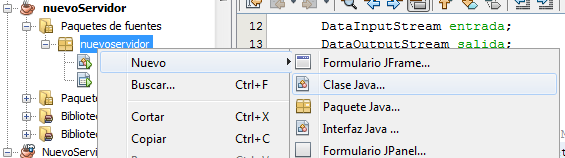
private javax.swing.JScrollPane jScrollPane1;

// End of variables declaration

}

1. **Creando clase manejador del servidor**

Al igual que el formulario creamos la clase manejadordel servidor



Donde escribiremos el siguiente código

package nuevoservidor;

import java.net.\*;

import java.io.\*;

import java.util.\*;

public class ManejadorServidor extends Thread {

Socket conexion;

DataInputStream entrada;

DataOutputStream salida;

static Vector manejador = new Vector();

Ventana venta = new Ventana();

int contador=-1;

/\*\* Creates a new instance of ManejadorServidor \*/

public ManejadorServidor(Socket conexion) throws IOException {

this.conexion = conexion;

entrada = new DataInputStream(new BufferedInputStream(conexion.getInputStream()));

salida = new DataOutputStream(new BufferedOutputStream(conexion.getOutputStream()));

}

public void run(){

String nombre = conexion.getInetAddress().toString();

try{

emitirMensaje(nombre +" se ha unido.");

manejador.addElement(this);

while(true){

String msg = entrada.readUTF();

emitirMensaje(nombre +": "+msg);

}

}

catch(IOException e){

//emitirMensaje(nombre +" Ha lanzado un Error.");

}

finally{

System.out.println("numEEro: "+manejador.capacity()+" "+manejador.lastIndexOf(this));

venta.quitarElementoLista(0);//manejador.lastIndexOf(this));

manejador.removeElement(this);

emitirMensaje(nombre + " ha abandonado la conexión.");

try{

salida.close();

}

catch(IOException e){}

}

}

public static void emitirMensaje(String mensaje){

synchronized(manejador){

Enumeration e = manejador.elements();

while(e.hasMoreElements()){

ManejadorServidor ms = (ManejadorServidor)e.nextElement();

try{

synchronized(ms.salida){

ms.salida.writeUTF(mensaje);

}

ms.salida.flush();

}

catch(IOException er){

ms.stop();

}

}

}

}

}

1. **Main**

En el main escribiremos el siguiente código

package nuevoservidor;

import java.net.\*;

import java.io.\*;

import java.util.\*;

import javax.swing.\*;

public class Main {

Ventana ventana;

public Main(int puerto) throws IOException {

ServerSocket servidor = new ServerSocket(puerto);

ventana = new Ventana();

ventana.setVisible(true);

while(true){

Socket cliente = servidor.accept();

System.out.println("Recibido de: "+cliente.getInetAddress());

ventana.agregarElementoLista(cliente.getInetAddress().toString());

ManejadorServidor manser = new ManejadorServidor(cliente);

manser.start();

}

}

public static void main(String[] args) throws IOException {

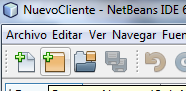
new Main(12345);

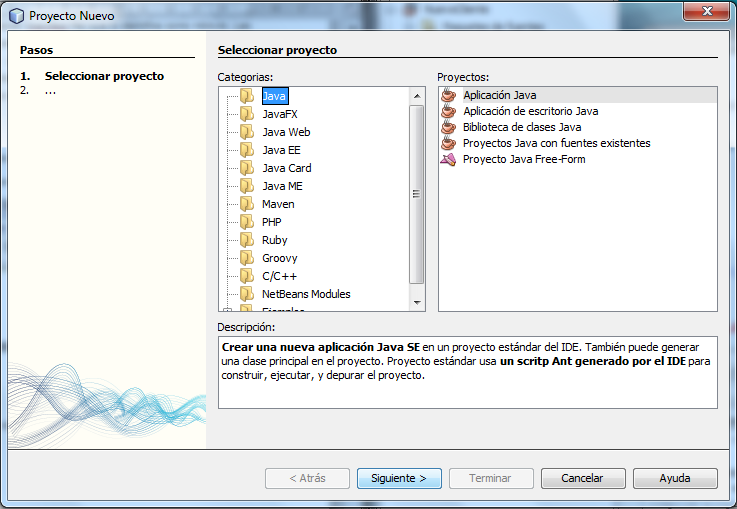
}

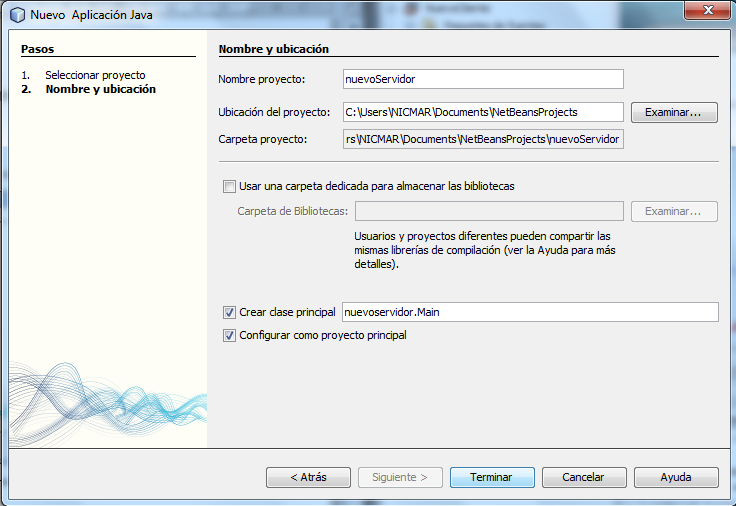
}

* 1. **CLIENTE**

Para el cliente crearemos un proyecto igual que el servidor y solo crearemos una clase ala que llamaremos ventana

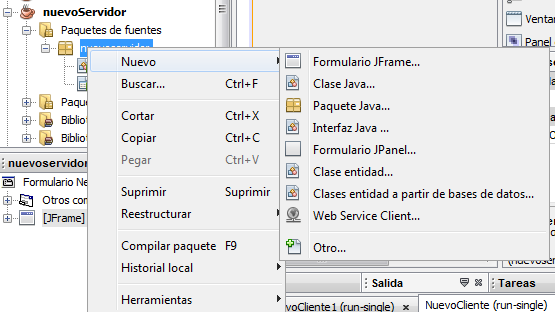




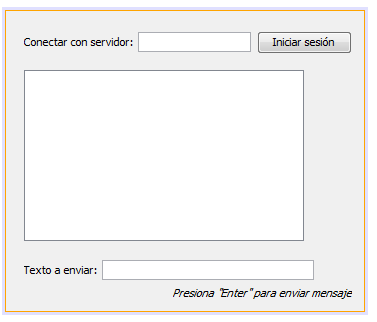


Nuevocliente

1. Creando la clases ventana



Donde realizaremos el siguiente diseño



Luego en la parte del código tendremos que escribir

package nuevocliente1;

import java.io.\*;

import java.net.\*;

public class Ventana extends javax.swing.JFrame implements Runnable {

DataInputStream entrada;

DataOutputStream salida;

Thread hilo;

String host="";

/\*\* Creates new form Ventana \*/

/\* public Ventana() {

initComponents();

}\*/

public Ventana() {

initComponents();

setVisible(true);

jTextArea1.setVisible(false);

jTextField1.setEditable(false);

}

public void iniciar(InputStream e, OutputStream s){

jTextArea1.setVisible(true);

jTextField1.setEditable(true);

jTextField2.setEditable(false);

jButton1.setEnabled(false);

entrada = new DataInputStream(new BufferedInputStream(e));

salida = new DataOutputStream(new BufferedOutputStream(s));

hilo = new Thread(this);

hilo.start();

}

public void run(){

try{

while(true){

String lineaEntrada = entrada.readUTF();

jTextArea1.append(lineaEntrada+"\n");

}

}

catch(IOException e){

jTextArea1.append("\*Error lectura.\n");

}

finally{

hilo = null;

try{

salida.close();

}

catch(IOException e){

jTextArea1.append("\*Error al cerrar.\n");

}

}

}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

// TODO: Agrege su codigo aqui:

String host = jTextField2.getText().toString();

try{

Socket s = new Socket (host,12345);

iniciar(s.getInputStream (), s.getOutputStream ());

}

catch(IOException e){}

}

private void formWindowClosing(java.awt.event.WindowEvent evt) {

// TODO: Agrege su codigo aqui:

hilo.stop();

System.out.println("hilo trminado.");

}

private void formWindowClosed(java.awt.event.WindowEvent evt) {

// TODO: Agrege su codigo aqui:

hilo.stop();

System.out.println("salimos y detenemos hilo.");

}

private void jTextField1ActionPerformed(java.awt.event.ActionEvent evt) {

// TODO: Agrege su codigo aqui:

try{

salida.writeUTF(jTextField1.getText().toString());

salida.flush();

jTextField1.setText("");

}

catch(IOException e){

hilo.stop();

}

}

private void jTextField2ActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

}

/\*\*

\* @param args the command line arguments

\*/

public static void main(String args[]) throws IOException{

}

// Variables declaration - do not modify

private javax.swing.JButton jButton1;

private javax.swing.JLabel jLabel1;

private javax.swing.JLabel jLabel2;

private javax.swing.JLabel jLabel3;

private javax.swing.JScrollPane jScrollPane1;

public javax.swing.JTextArea jTextArea1;

public javax.swing.JTextField jTextField1;

private javax.swing.JTextField jTextField2;

// End of variables declaration

}

1. **MAIN**

En el main codificaremos en siguiente código

package nuevocliente1;

import java.net.\*;

import java.io.\*;

import java.awt.event.\*;

import java.awt.\*;

public class Main{

Ventana ventana;

public Main() throws IOException{

Ventana ventana = new Ventana();

}

public static void main(String[] args) throws IOException {

new Main();

}

}