**Instituto Politécnico Nacional.**

**Escuela Superior De Cómputo.**





**Materia:**

**Aplicaciones Para Comunicación En Red.**

**Tema:**

**Practica 01**

**(Reporte)**

**Profesor:**

**Axel Ernesto Moreno Cervantes.**

**Alumno:**

**Luis Enrique Rojas Alvarado.**

**Mario Alberto Miranda Sandoval.**

**Grupo:**

**3CM5**

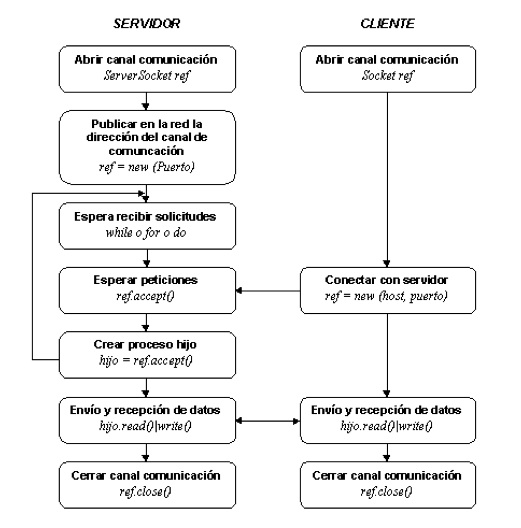
**Introducción.**

En la actualidad, muchos de los procesos que se ejecutan en una computadora requieren obtener o enviar información a otros procesos que se localizan en una computadora diferente.  Para lograr esta comunicación se utilizan los protocolos de comunicación TCP y UDP.

El protocolo TCP (Transmission Control Protocol) establece un conducto de comunicación punto a punto entre dos computadoras, es decir, cuando se requiere la transmisión de un flujo de datos entre dos equipos, el protocolo TCP establece un conducto exclusivo entre dichos equipos por el cual los datos serán transmitidos y este perdurará hasta que la transmisión haya finalizado, gracias a esto TCP garantiza que los datos enviados de un extremo de la conexión lleguen al otro extremo y en el mismo orden en que fueron enviados. Las características que posee TCP hacen que el protocolo sea conocido como un protocolo orientado a conexión.

Los sockets son una forma de comunicación entre procesos que se encuentran en diferentes máquinas de una red, los sockets proporcionan un punto de comunicación por el cual se puede enviar o recibir información entre procesos.

Los sockets tienen un ciclo de vida dependiendo si son sockets de servidor, que esperan a un cliente para establecer una comunicación, o socket cliente que busca a un socket de servidor para establecer la comunicación.

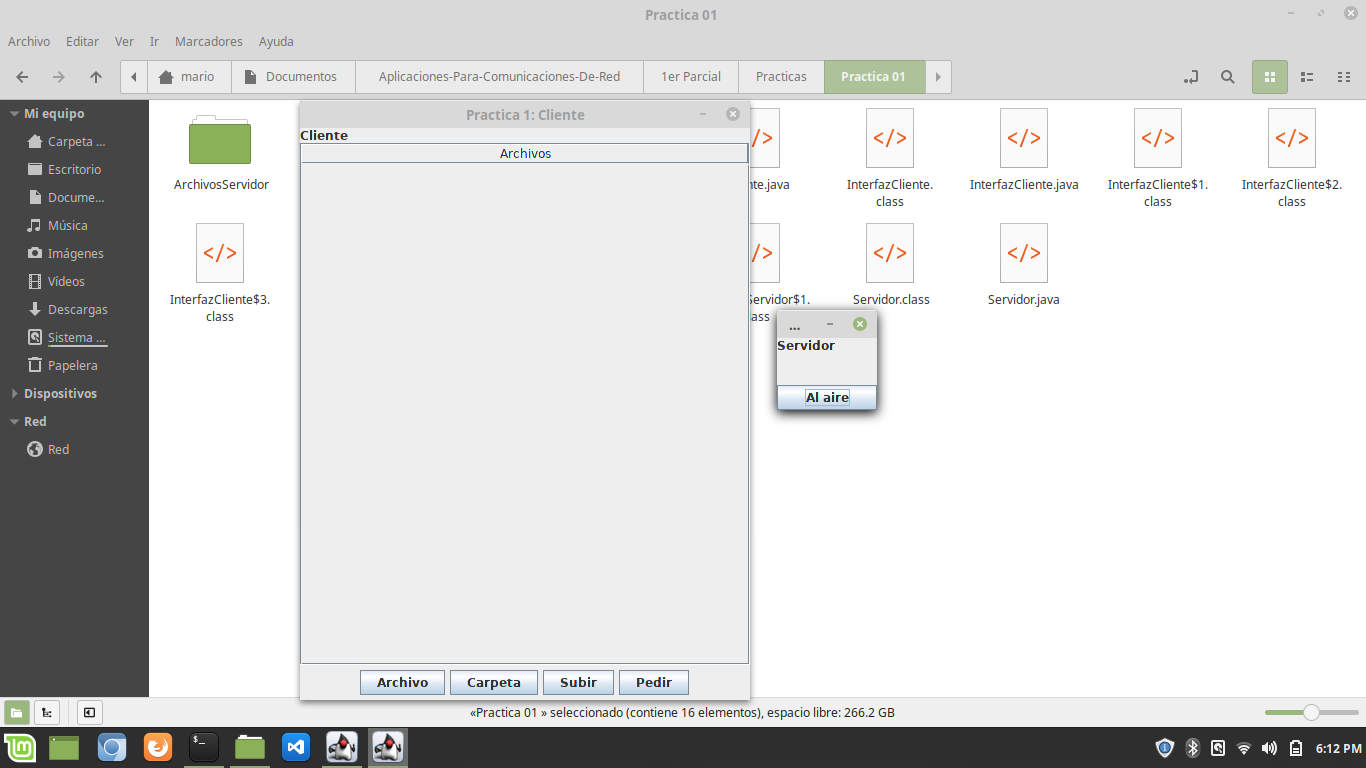


**Desarrollo.**

El desarrollo de esta práctica consta de, dos interfaces una de servidor y otra de cliente y de distintas clases que se encargan de gestionar los envíos y peticiones de archivos.

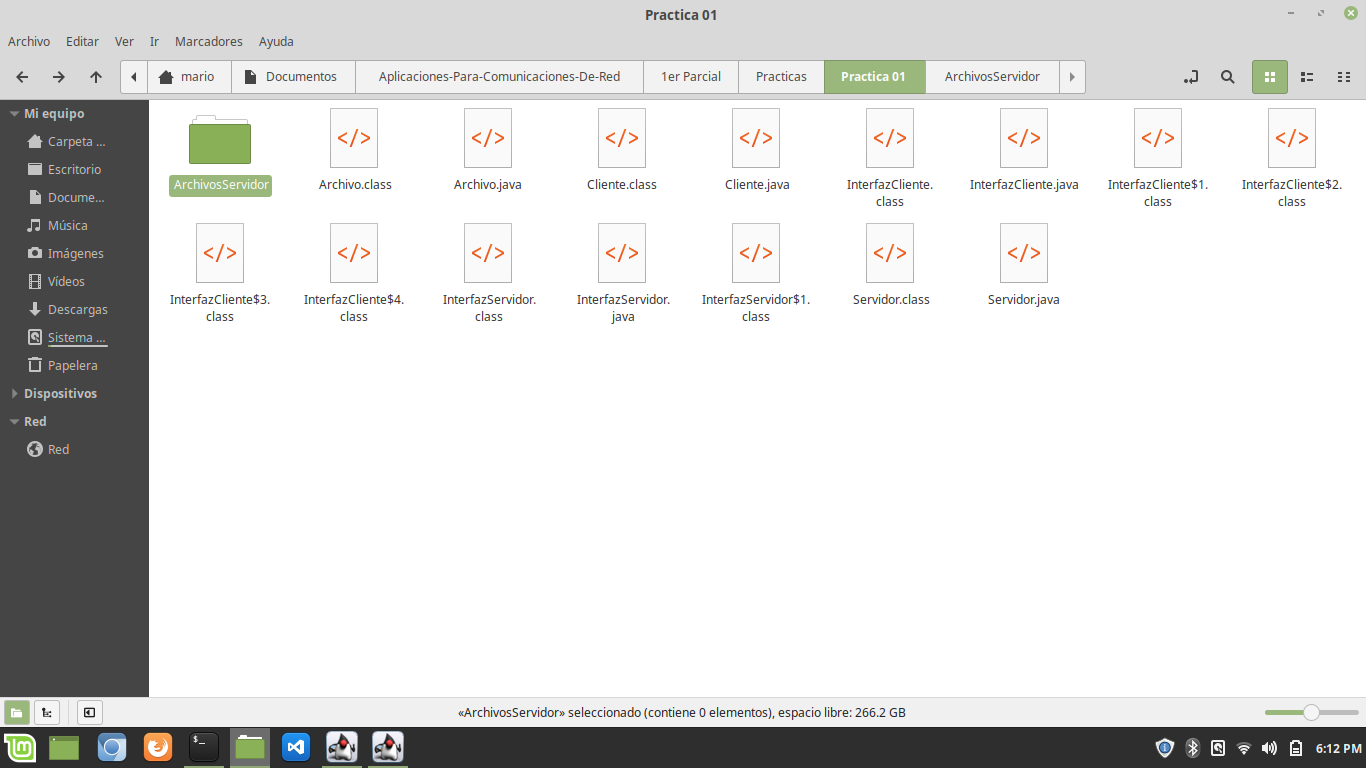
Además, que se creó una carpeta para el servidor a modo que esta sirva para guardar todos los archivos que se le son mandados a este servidor.

La separación de las clases genera que se llamen las funciones y la creación de los sockets y destrucción de los sockets en cada envió, donde destaca la clase archivo, para la gestión de estos.

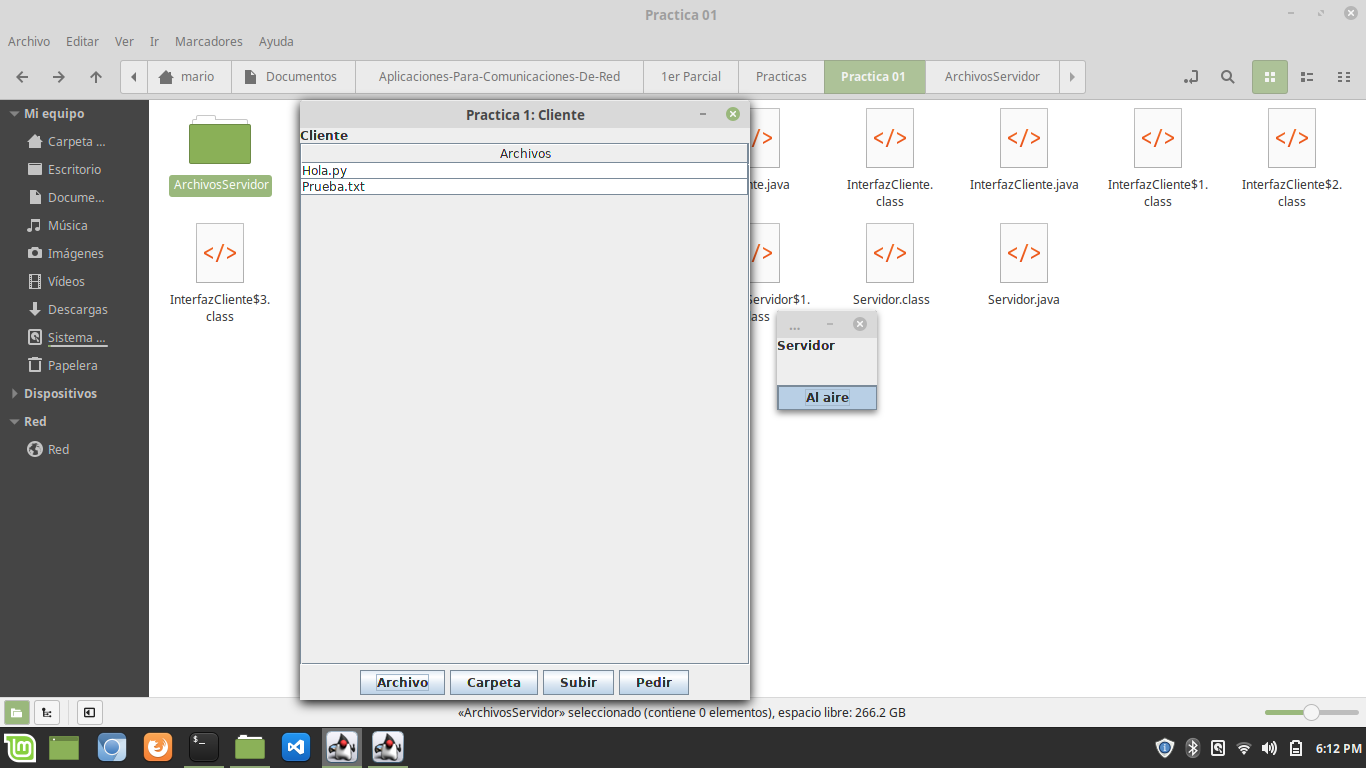


Interfaz de cliente y servidor.

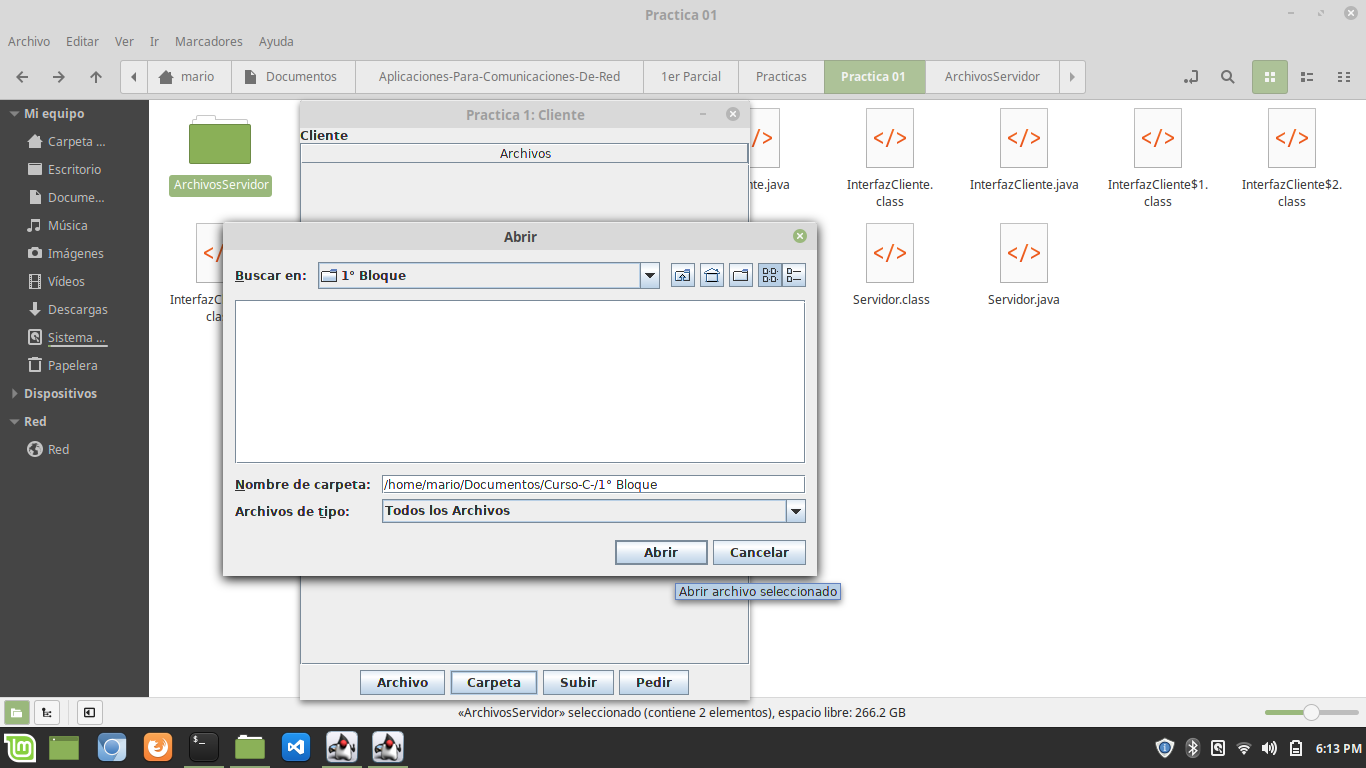
**Pruebas.**



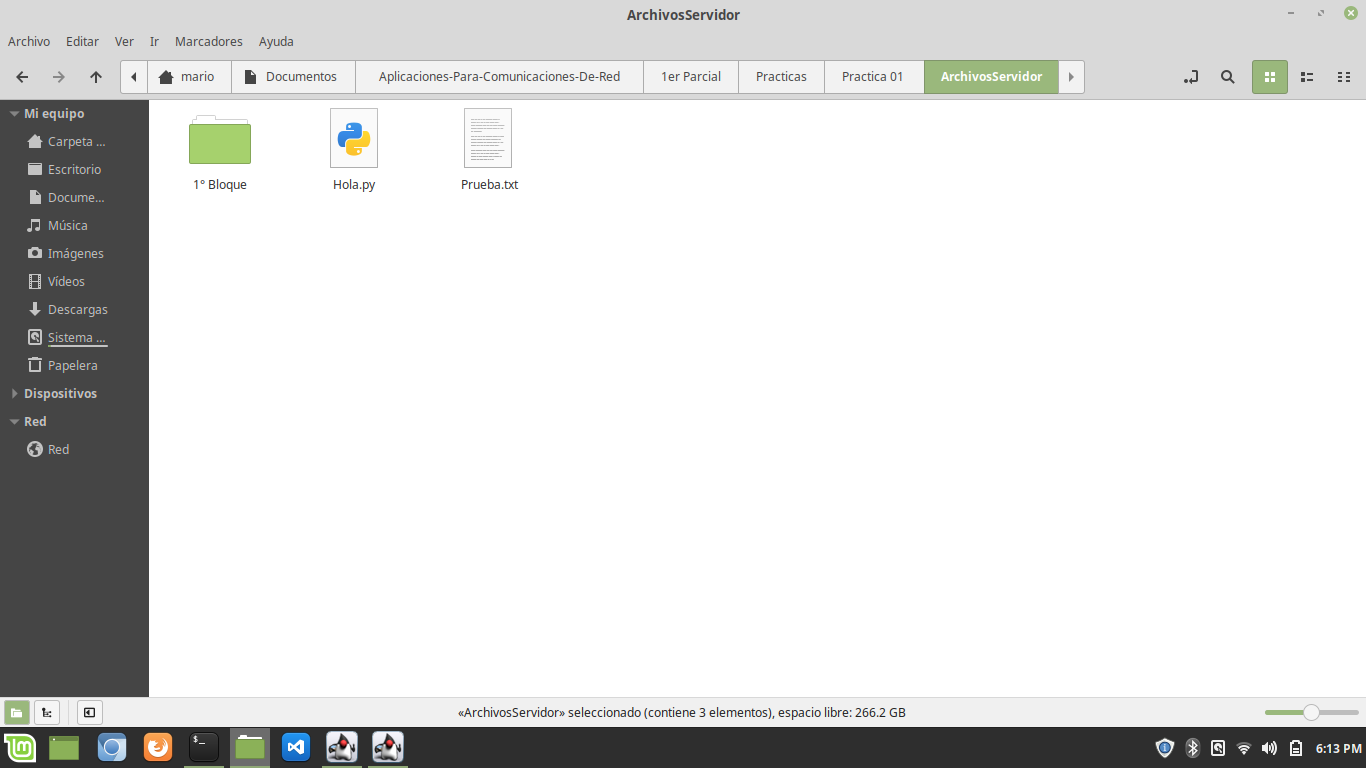
Carpeta del servidor para guardar los archivos.

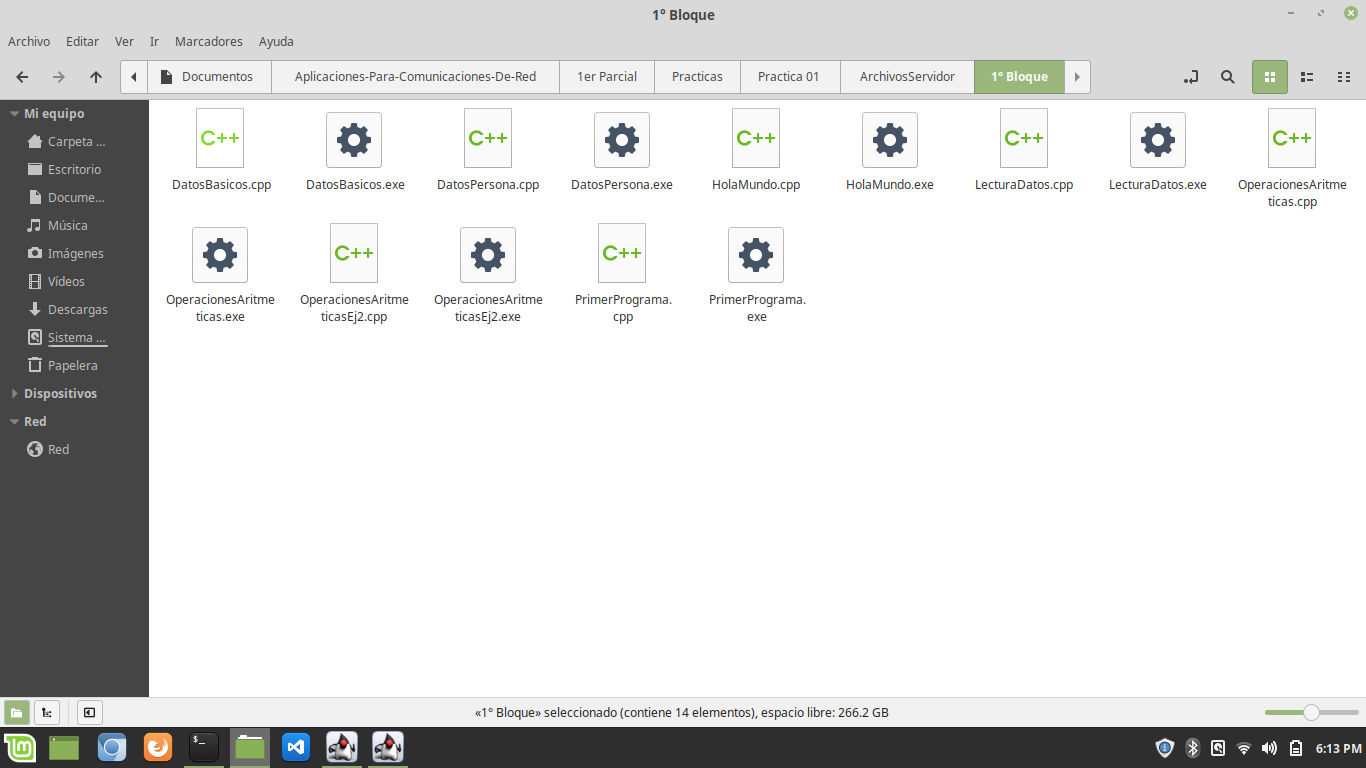


Enviando múltiples archivos por separado.

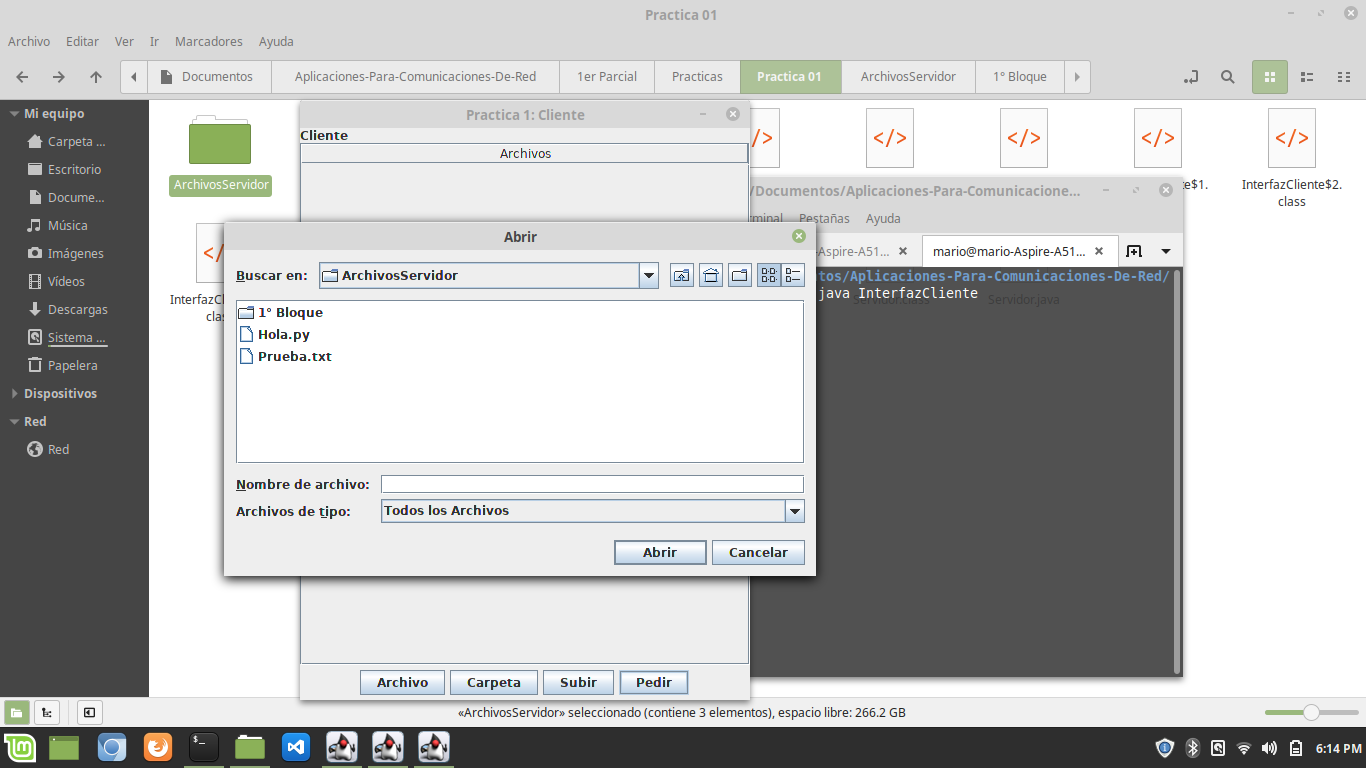


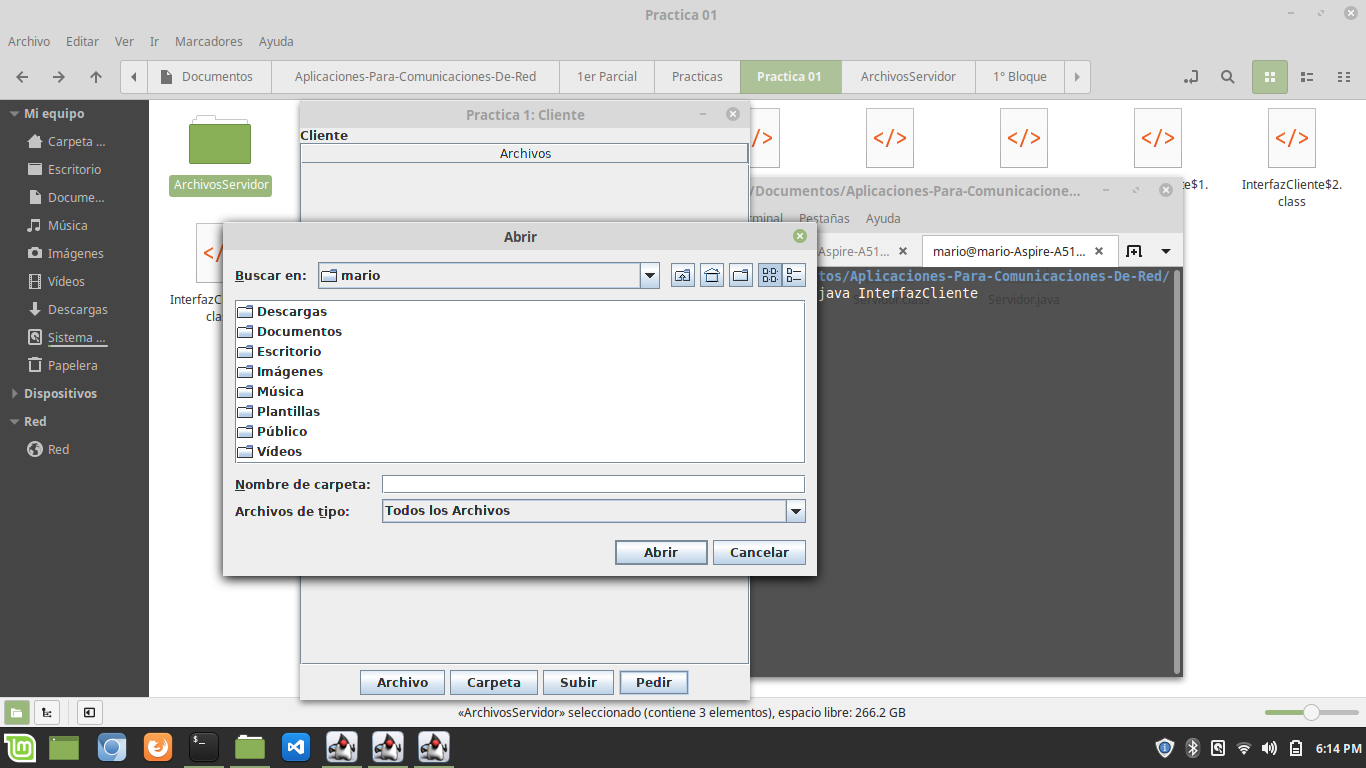
Mandando una carpeta con su contenido.



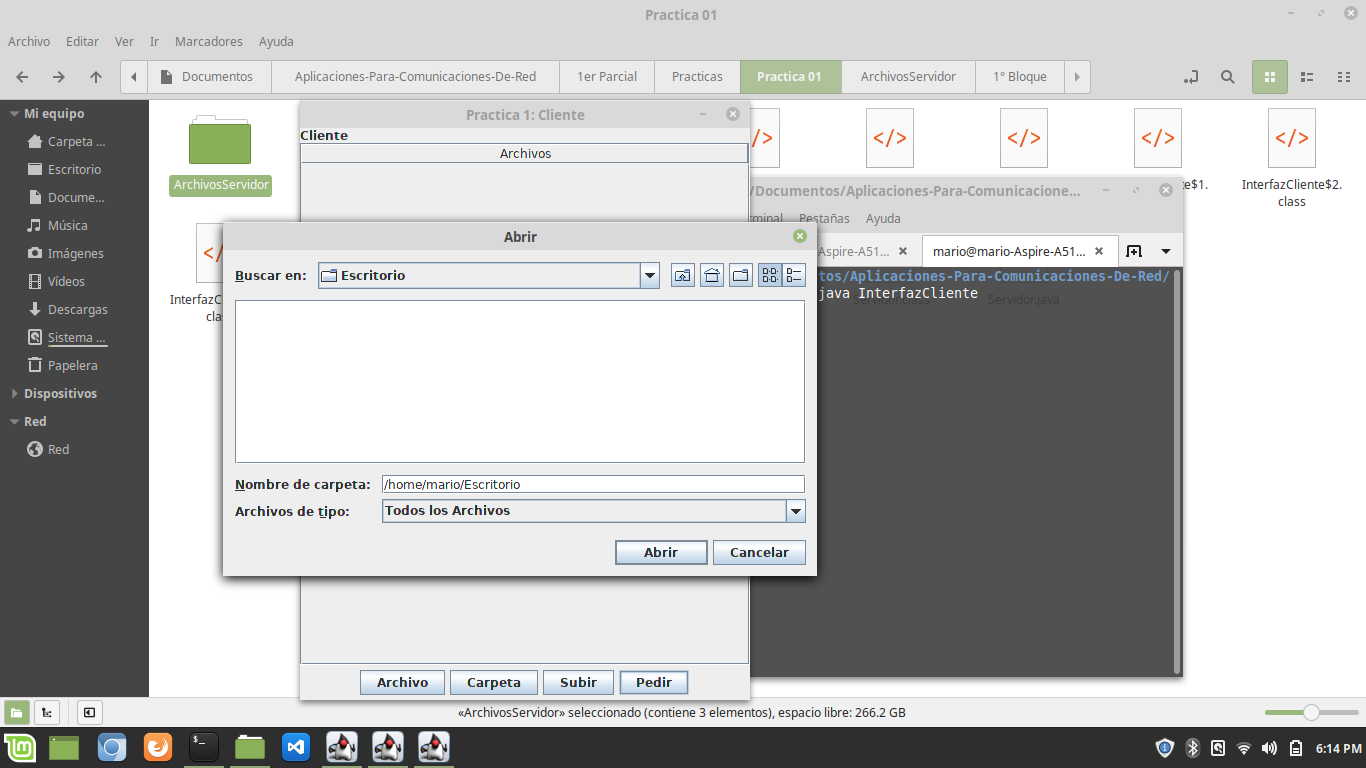


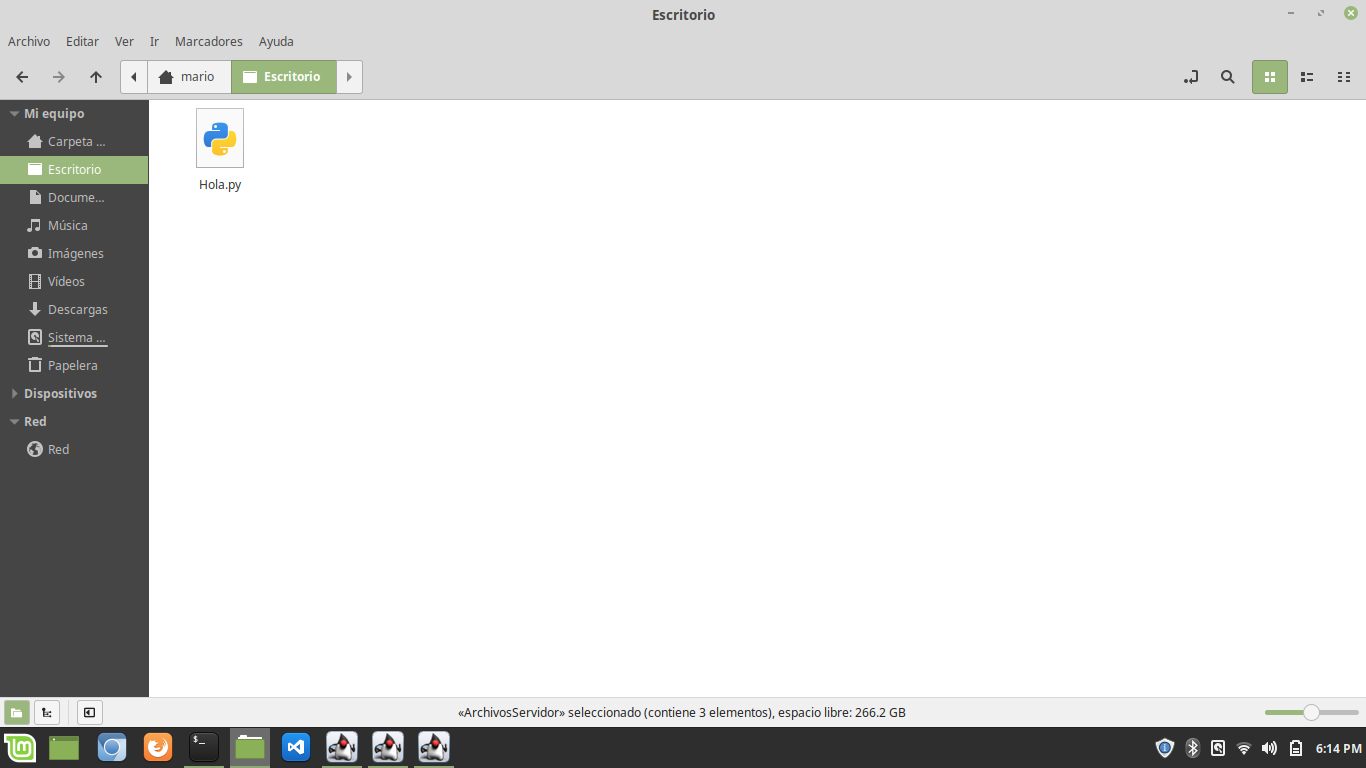
Archivos recibidos en el servidor.





Pidiendo un archivo, el primer cuadro muestra el archivo a ser pedido que se encuentra el servidor, mientras que el segundo muestra al usuario donde escoger la ubicación para guardar su archivo.





Como se puede ver se seleccionó el destino en el escritorio, y el archivo ha llegado a su destino.

**Código.**

**Archivo.java**

1. **import** java.io.File;
3. **public** **class** Archivo {
4. **public** Archivo(String nombre, **long** tamanio, String path, File file) {
5. **this**.nombre = nombre;
6. **this**.tamanio = tamanio;
7. **this**.path = path;
8. **this**.file = file;
9. }
11. **public** String getNombre() { **return** nombre; }
12. **public** **long** getTamanio() { **return** tamanio; }
13. **public** String getPath() { **return** path; }
14. **public** File getFile() { **return** file; }
16. **public** **void** setNombre(String nombre) { **this**.nombre = nombre; }
17. **public** **void** setTamanio(**long** tamanio) { **this**.tamanio = tamanio; }
18. **public** **void** setPath(String path) { **this**.path = path; }
19. **public** **void** setFile(File file) { **this**.file = file; }
21. **private** String nombre;
22. **private** **long** tamanio;
23. **private** String path;
24. **private** File file;
25. }

**InterfazCliente.java**

1. **import** java.awt.\*;
2. **import** javax.swing.\*;
3. **import** java.awt.event.\*;
4. **import** javax.swing.table.\*;
5. **import** java.io.\*;
6. **import** java.util.\*;
8. **public** **class** InterfazCliente **extends** JFrame {
9. **private** **static** **final** **long** serialVersionUID = 1L;
10. **public** InterfazCliente() {
11. setTitle("Practica 1: Cliente");
12. setBounds(300, 100, 450, 600);
13. setResizable(**false**);
15. panelPrincipal = **new** JPanel();
16. panelInferior = **new** JPanel();
17. cliente = **new** JLabel("Cliente");
18. elegirArchivo = **new** JButton("Archivo");
19. elegirCarpeta = **new** JButton("Carpeta");
20. subirArchivo = **new** JButton("Subir");
21. pedirArchivo = **new** JButton("Pedir");
22. tablaCliente = **new** JTable();
23. misArchivos = **new** ArrayList<>();
24. modelo = (DefaultTableModel) tablaCliente.getModel();
25. modelo.addColumn("Archivos");
27. elegirArchivo.addActionListener(**new** ActionListener() {
28. **public** **void** actionPerformed(ActionEvent e) {
29. file = **new** JFileChooser();
30. file.requestFocus();
31. **int** r = file.showOpenDialog(InterfazCliente.**this**);
32. **if**(r == JFileChooser.APPROVE\_OPTION) {
33. File f = file.getSelectedFile();
34. misArchivos.add(**new** Archivo(f.getName(), f.length(), f.getAbsolutePath(), f));
35. modelo.addRow(**new** Object[] { f.getName() });
36. }
37. }
38. });
40. subirArchivo.addActionListener(**new** ActionListener() {
41. **public** **void** actionPerformed(ActionEvent e) {
42. miCliente = **new** Cliente(HOST, PUERTO);
43. miCliente.enviarArchivo(misArchivos, "");
44. modelo.setRowCount(0);
45. }
46. });
48. elegirCarpeta.addActionListener(**new** ActionListener() {
49. **public** **void** actionPerformed(ActionEvent e) {
50. JFileChooser jf = **new** JFileChooser();
51. jf.setFileSelectionMode(JFileChooser.DIRECTORIES\_ONLY);
52. jf.requestFocus();
53. **int** r = jf.showOpenDialog(InterfazCliente.**this**);
54. **if** (r == JFileChooser.APPROVE\_OPTION) {
55. File archivoSeleccionado = jf.getSelectedFile();
56. modelo.addRow(**new** Object[]{"Carpeta: " + archivoSeleccionado.getName()});
57. modelo.setRowCount(0);
58. miCliente = **new** Cliente(HOST, PUERTO);
59. miCliente.carpetas(archivoSeleccionado, "", misArchivos);
60. }
61. }
62. });
64. pedirArchivo.addActionListener(**new** ActionListener() {
65. **public** **void** actionPerformed(ActionEvent e) {
66. JFileChooser jf = **new** JFileChooser("./ArchivosServidor/");
67. jf.requestFocus();
68. **int** r = jf.showOpenDialog(InterfazCliente.**this**);
69. **if** (r == JFileChooser.APPROVE\_OPTION) {
70. JFileChooser jf2 = **new** JFileChooser();
71. jf2.requestFocus();
72. jf2.setFileSelectionMode(JFileChooser.DIRECTORIES\_ONLY);
73. **int** r2 = jf2.showOpenDialog(InterfazCliente.**this**);
74. **if**(r2 == JFileChooser.APPROVE\_OPTION) {
75. File f = jf.getSelectedFile();
76. File f2 = jf2.getSelectedFile();
77. Archivo a = **new** Archivo(f.getName(), f.length(), f.getAbsolutePath(), f);
78. miCliente = **new** Cliente(HOST, PUERTO);
79. miCliente.peticionArchivo(a, f2.getAbsolutePath());
80. }
81. }
82. }
83. });
85. panelInferior.add(elegirArchivo);
86. panelInferior.add(elegirCarpeta);
87. panelInferior.add(subirArchivo);
88. panelInferior.add(pedirArchivo);
89. panelPrincipal.setLayout(**new** BorderLayout());
90. panelPrincipal.add(cliente, BorderLayout.NORTH);
91. panelPrincipal.add(**new** JScrollPane(tablaCliente), BorderLayout.CENTER);
92. panelPrincipal.add(panelInferior, BorderLayout.SOUTH);
93. add(panelPrincipal);
95. setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);
96. setVisible(**true**);
97. }
99. **public** **static** **void** main(String[] args) {
100. **new** InterfazCliente();
101. }
103. **private** JPanel panelPrincipal;
104. **private** JLabel cliente;
105. **private** JTable tablaCliente;
106. **private** DefaultTableModel modelo;
107. **private** JPanel panelInferior;
108. **private** JButton elegirArchivo;
109. **private** JButton subirArchivo;
110. **private** JButton elegirCarpeta;
111. **private** JButton pedirArchivo;
112. **private** JFileChooser file;
113. **private** ArrayList <Archivo> misArchivos;
114. **private** Cliente miCliente;
115. **private** **final** **int** PUERTO = 9000;
116. **private** **final** String HOST = "127.0.0.1";
117. }

**InterfazServidor.java**

1. **import** java.awt.\*;
2. **import** java.awt.event.ActionEvent;
3. **import** java.awt.event.ActionListener;
5. **import** javax.swing.\*;
7. **public** **class** InterfazServidor **extends** JFrame{
8. **private** **static** **final** **long** serialVersionUID = 2L;
9. **public** InterfazServidor() {
10. setBounds(600, 100, 100, 100);
11. setTitle("Practica 1: Servidor");
12. setResizable(**false**);
14. panelPrincipal = **new** JPanel();
15. servidor = **new** JLabel("Servidor");
16. conectar = **new** JButton("Al aire");
18. conectar.addActionListener(**new** ActionListener() {
19. **public** **void** actionPerformed(ActionEvent arg0) {
20. miServidor = **new** Servidor(PUERTO);
21. miServidor.conectar();
22. }
23. });
25. panelPrincipal.setLayout(**new** BorderLayout());
26. panelPrincipal.add(servidor, BorderLayout.NORTH);
27. panelPrincipal.add(conectar, BorderLayout.SOUTH);
28. add(panelPrincipal);
30. setDefaultCloseOperation(3);
31. setVisible(**true**);
32. }
34. **public** **static** **void** main(String[] args) {
35. **new** InterfazServidor();
36. }
38. **private** JPanel panelPrincipal;
39. **private** JLabel servidor;
40. **private** JButton conectar;
41. **private** Servidor miServidor;
42. **private** **final** **int** PUERTO = 9000;
43. }

**Cliente.java**

1. **import** java.io.\*;
2. **import** java.net.\*;
3. **import** java.util.\*;
5. **public** **class** Cliente {
6. **public** Cliente(String host, **int** puerto){
7. **this**.host = host;
8. **this**.puerto = puerto;
9. }
11. **public** **void** enviarArchivo(ArrayList <Archivo> misArchivos, String destino) {
12. **try** {
13. **for**(Archivo a : misArchivos) {
14. File f = a.getFile();
15. cliente = **new** Socket(**this**.host, **this**.puerto);
16. dis = **new** DataInputStream(**new** FileInputStream(f.getAbsoluteFile()));
17. dos = **new** DataOutputStream(cliente.getOutputStream());
19. **long** e = 0;
20. **int** n = 0;
21. **int** porcentaje = 0;
23. dos.writeInt(0);
24. dos.flush();
25. dos.writeUTF(a.getNombre());
26. dos.flush();
27. dos.writeLong(a.getTamanio());
28. dos.flush();
29. dos.writeUTF(destino);
30. dos.flush();
32. **while** (e < a.getTamanio()) {
33. **byte**[] b = **new** **byte**[2000];
34. n = dis.read(b);
35. e += n;
36. dos.write(b, 0, n);
37. dos.flush();
38. porcentaje = (**int**) ((e \* 100) / a.getTamanio());
39. System.out.print("\rPorcentaje:" + porcentaje + "%");
40. }
42. System.out.println("Archivo Enviado");
43. dos.close();
44. dis.close();
45. cliente.close();
46. }
47. misArchivos.clear();
48. } **catch**(Exception e) { e.printStackTrace(); };
49. }
51. **public** **void** carpetas(File carpeta, String destino, ArrayList<Archivo> misArchivos) {
52. **if**(destino.equals(""))
53. destino = carpeta.getName();
54. **else**
55. destino = destino + "\\" + carpeta.getName();
57. **for**(File archivo : carpeta.listFiles()) {
58. **if**(archivo.isDirectory())
59. carpetas(carpeta, destino, misArchivos);
60. **else**{
61. misArchivos.add(**new** Archivo(archivo.getName(), archivo.length(), destino, archivo));
62. }
63. }
64. enviarArchivo(misArchivos ,destino);
65. }
67. **public** **void** peticionArchivo(Archivo archivoPedido, String destino) {
68. **try** {
69. cliente = **new** Socket(**this**.host, **this**.puerto);
70. dos = **new** DataOutputStream(cliente.getOutputStream());
71. dos.writeInt(1);
72. dos.flush();
73. dos.writeUTF(archivoPedido.getNombre());
74. dos.flush();
75. dos.writeLong(archivoPedido.getTamanio());
76. dos.flush();
77. dos.writeUTF(archivoPedido.getPath());
78. dos.flush();
80. System.out.println(destino);
81. dis = **new** DataInputStream(cliente.getInputStream());
82. String nombre = dis.readUTF();
83. **long** tam = dis.readLong();
84. String ruta = dis.readUTF();
86. **long** r = 0;
87. **int** n = 0;
88. **int** porcentaje = 0;
89. DataOutputStream archivo = **new** DataOutputStream(**new** FileOutputStream(destino + "/" + nombre));
91. **while**(r < tam) {
92. **byte**[] b = **new** **byte**[2000];
93. n = dis.read(b);
94. r += n;
95. archivo.write(b, 0, n);
96. archivo.flush();
97. porcentaje = (**int**) ((r\*100) / tam);
98. System.out.println("Se ha recibido " + porcentaje + "%");
99. }
101. System.out.println("Se ha recicibido el archivo");
103. archivo.close();
104. dos.close();
105. dis.close();
106. } **catch** (Exception e) { e.printStackTrace(); }
107. }
109. **private** String host;
110. **private** **int** puerto;
111. **private** Socket cliente;
112. **private** DataInputStream dis;
113. **private** DataOutputStream dos;
114. }

**Servidor.java**

1. **import** java.io.\*;
2. **import** java.net.\*;
4. **public** **class** Servidor {
5. **public** Servidor(**int** puerto) {
6. **this**.puerto = puerto;
7. }
9. **public** **void** conectar() {
10. **try** {
11. servidor = **new** ServerSocket(puerto);
12. servidor.setReuseAddress(**true**);
13. System.out.println("Servidor establecido");
14. **for**(;;) {
15. cliente = servidor.accept();
16. DataInputStream dis = **new** DataInputStream(cliente.getInputStream());
17. **int** accion = dis.readInt();
18. String nombre = dis.readUTF();
19. **long** tam = dis.readLong();
20. String ruta = dis.readUTF();
22. **if** (accion == 0) {
23. nombre = directorio(nombre, ruta);
24. nombre = "./ArchivosServidor/" + nombre;
25. recibirArchivo(nombre, tam, dis);
26. } **else** **if**(accion == 1) {
27. enviaArchivoPedido(nombre, tam, ruta);
28. }
30. cliente.close();
31. }
32. } **catch** (Exception e) { e.printStackTrace(); }
33. }
35. **public** **void** recibirArchivo(String nombre, **long** tam, DataInputStream dis) {
36. **try** {
37. **int** porcentaje = 0;
38. **long** r = 0;
39. **int** n = 0;
40. DataOutputStream dos = **new** DataOutputStream(**new** FileOutputStream(nombre));
41. **while**(r < tam) {
42. **byte**[] b = **new** **byte**[2000];
43. n = dis.read(b);
44. r += n;
45. dos.write(b, 0, n);
46. dos.flush();
47. porcentaje = (**int**) ((r\*100) / tam);
48. System.out.print("\rSe ha recibido " + porcentaje + "%");
49. }
51. dos.close();
52. } **catch**(Exception e) { e.printStackTrace(); }
53. }
55. **public** String directorio(String nombre, String ruta) {
56. **if**(!ruta.equals("")) {
57. File carpeta = **new** File("./ArchivosServidor/" + ruta);
58. **if**(!carpeta.exists()) {
59. **try** {
60. **if** (carpeta.mkdir())
61. System.out.println("Carpeta creada");
62. **else**
63. System.out.println("No se creo la carpeta");
64. }**catch**(SecurityException se) { se.printStackTrace(); }
65. }
66. nombre = ruta + "//" + nombre;
67. }
68. **return** nombre;
69. }
71. **public** **void** enviaArchivoPedido(String nombre, **long** tam, String ruta) {
72. //System.out.println("Todo bien");
73. **try** {
74. **int** porcentaje = 0;
75. **long** r = 0;
76. **int** n = 0;
77. DataInputStream dis = **new** DataInputStream(**new** FileInputStream(ruta));
78. DataOutputStream dos = **new** DataOutputStream(cliente.getOutputStream());
80. dos.writeUTF(nombre);
81. dos.flush();
82. dos.writeLong(tam);
83. dos.flush();
84. dos.writeUTF(ruta);
85. dos.flush();
87. **while**(r < tam) {
88. **byte**[] b = **new** **byte**[2000];
89. n = dis.read(b);
90. r += n;
91. dos.write(b, 0, n);
92. dos.flush();
93. porcentaje = (**int**) ((r\*100) / tam);
94. System.out.print("\rPorcentaje" + porcentaje + "%");
95. }
96. System.out.println("");
97. dos.close();
98. dis.close();
99. } **catch**(Exception e) { e.printStackTrace(); }
100. }
102. **private** **int** puerto;
103. **private** ServerSocket servidor;
104. **private** Socket cliente;
105. }

**Dificultades Encontradas.**

La mayor dificultad fue pedir los archivos y su ruta donde se almacenarían, ya que no encontrábamos los métodos necesarios para poder manejar esas ultimas partes, de ahí en fuera todo normal.

**Conclusiones.**

**Luis Enrique Rojas Alvarado.**

Este programa es muy interesante debido a que es algo con lo que tenemos contacto todos los días, por ejemplo, para hacer algún programa o reporte de práctica entre compañeros porque nos compartimos los archivos por medio de Facebook o alguna otra aplicación.

La parte de la comunicación fue bastante simple debido a que se explicó muy bien dentro del curso en el salón y se resolvieron todas las dudas por parte del profesor, sin duda, una práctica con un nivel de complejidad bastante aceptable y acorde a lo visto hasta el momento.

**Mario Alberto Miranda Sandoval.**

Acerca de esta aplicación, fue algo fácil e interesante hacerla, ya que ahora comprendo mejor el envío de archivos mediante sockets de flujo, además de como el modo de hacerlo enviando por partes dentro del DataOutputStream y el buffer mandado.

Sobre las peticiones al servidor, fue relativamente fácil ya que, con el mandado de la información del archivo o carpeta, se mandaba una pequeña bandera a modo de descriptor, el único inconveniente fue el manejar la ruta elegida del archivo, ya que desconocíamos que el método getSelectedFile(), funcionaba para carpetas y si, ya después de recibir el archivo seleccionado con el método getAbsolutePath() fue suficiente para poder terminar la práctica.