Instituto Politécnico Nacional

Escuela Superior de Cómputo

Desarrollo de Sistemas Distribuidos.

Tarea 5, chat multicast.

.

Alumno:

Hernández Escudero Luis Hugo

Profesor:

Pineda Guerrero Carlos

16-ABR-2021

Creación de la máquina virtual.

La máquina virtual lleva por nombre mi número de boleta, 2016310335B, la letra B al final solo era para cumplir con un nombre adecuado y la conexión es a través de un escritorio remoto.

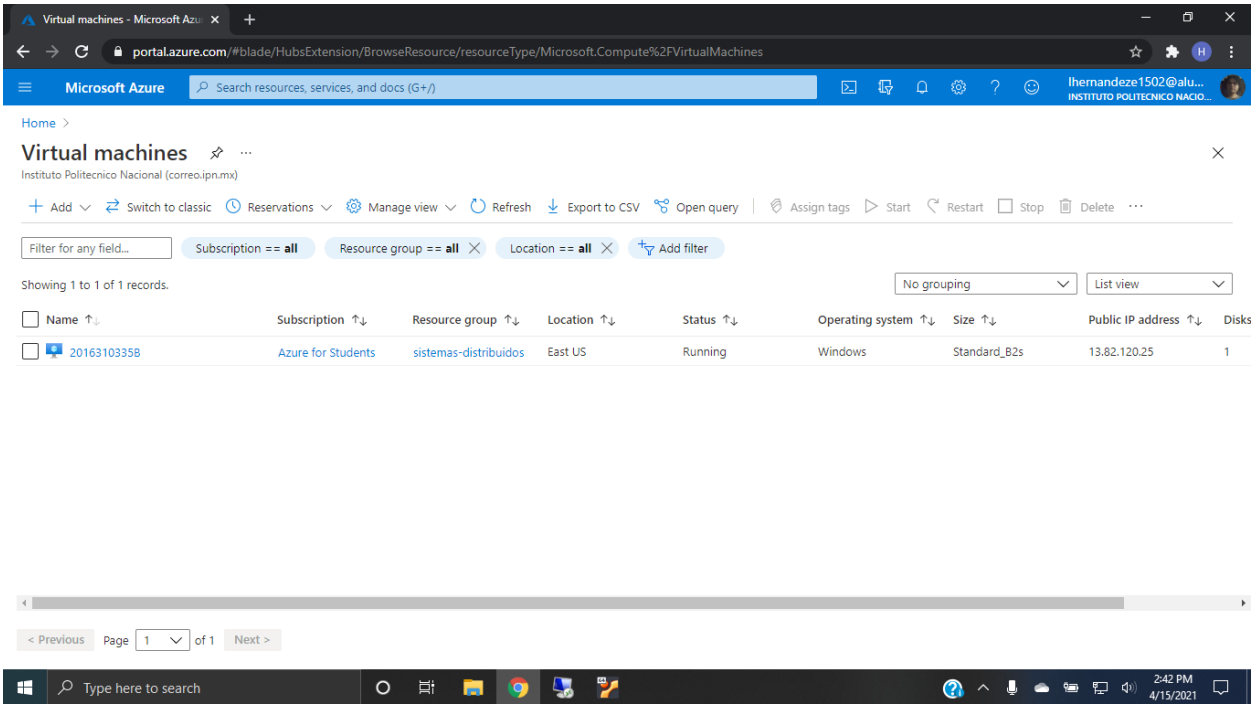

Imagen 1. creación de la máquina virtual en Azure.

Compilación del programa.

Para compilar el programa basta con tener un compilador de java. Introduciendo el siguiente comando se puede compilar el código.

```
> javac Chat.java
```

Al compilar el archivo, se genera el archivo Chat.class y ya es posible ejecutarlo.

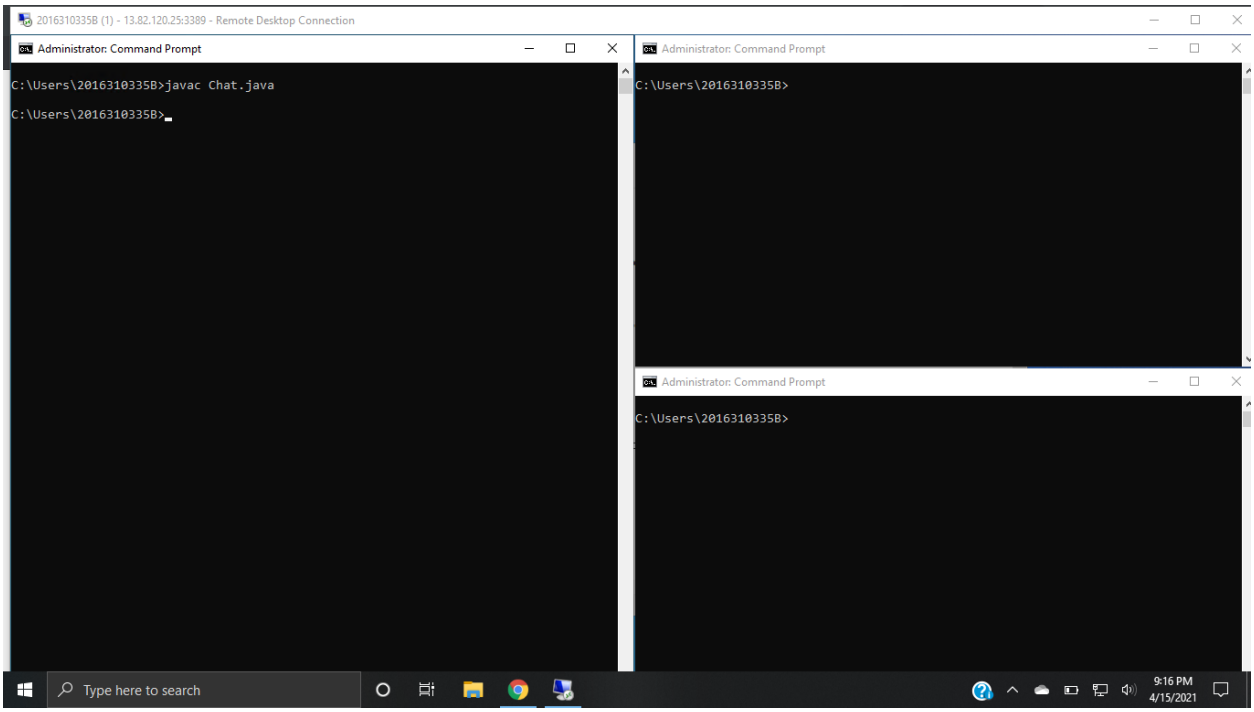

Imagen 2. Compilación del programa en consola del lado izquierdo.

Ejecución

Para ejecutar el programa es necesario pasar como parámetro el nombre de usuario que será utilizado para identificarse con los demás usuarios. De la siguiente forma se puede ejecutar el programa.

```
> java Chat <nombre-de-usuario>
```

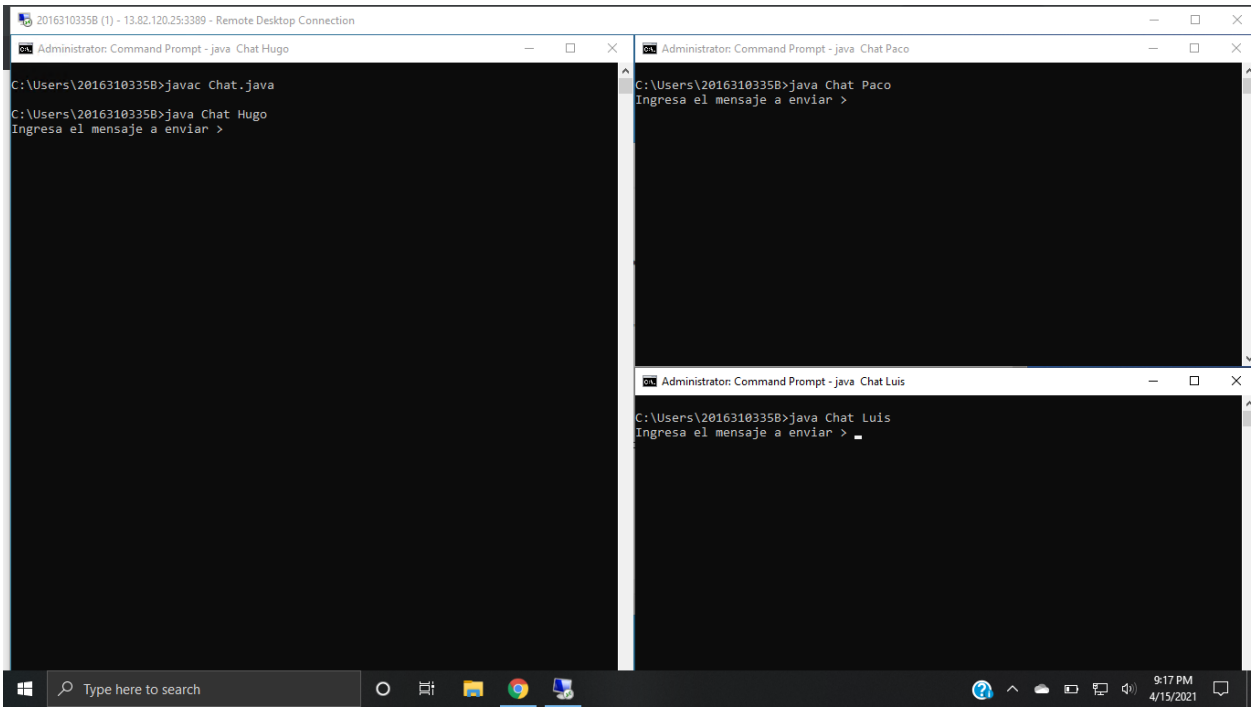

Imagen 3. Ejecución del programa.

Para mostrar el correcto funcionamiento del programa se va a ejecutar el programa en 3 consolas. En la imagen 3, en la consola de la izquierda escribirá el usuario Hugo, en la consola superior derecha escribirá Paco y en la consola inferior derecha el usuario Luis.

El primer usuario en escribir es el usuario Hugo, posteriormente los usuarios Paco y Luis le responden.

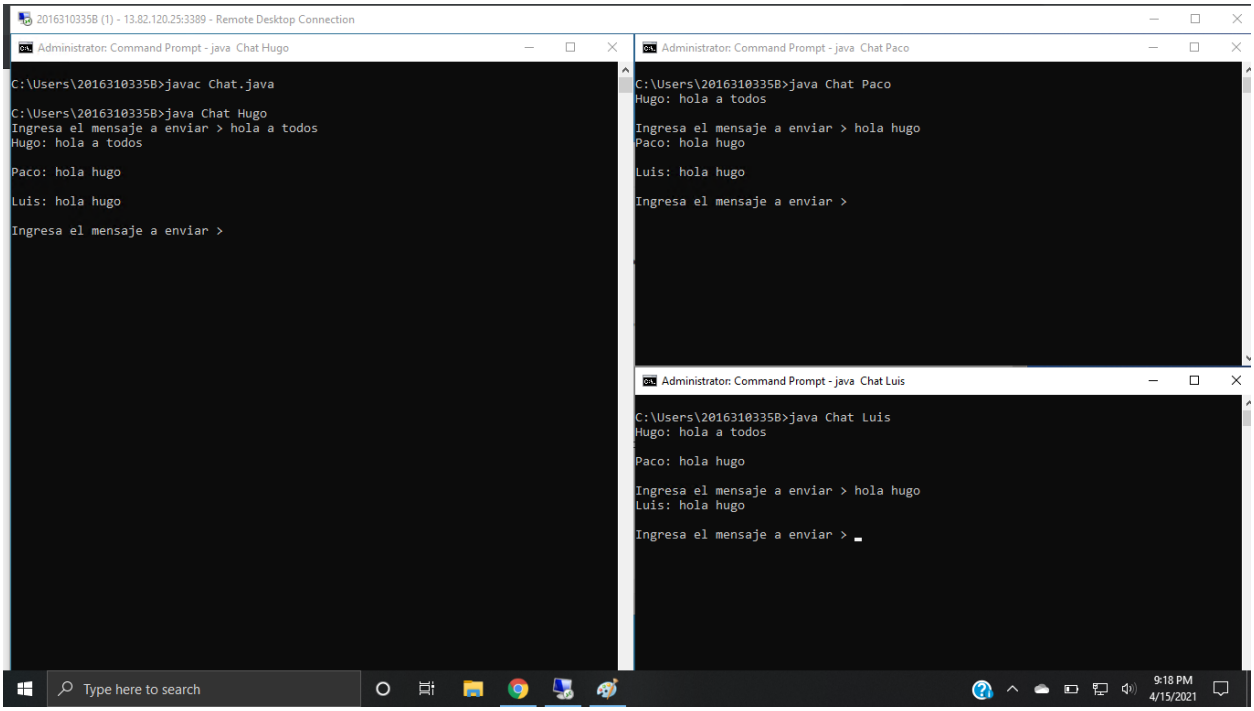

Imagen 4. Inicio de la conversación.

Posteriormente el usuario Hugo envía un mensaje al grupo que contiene caracteres especiales, como signos de interrogación y vocales acentuadas.

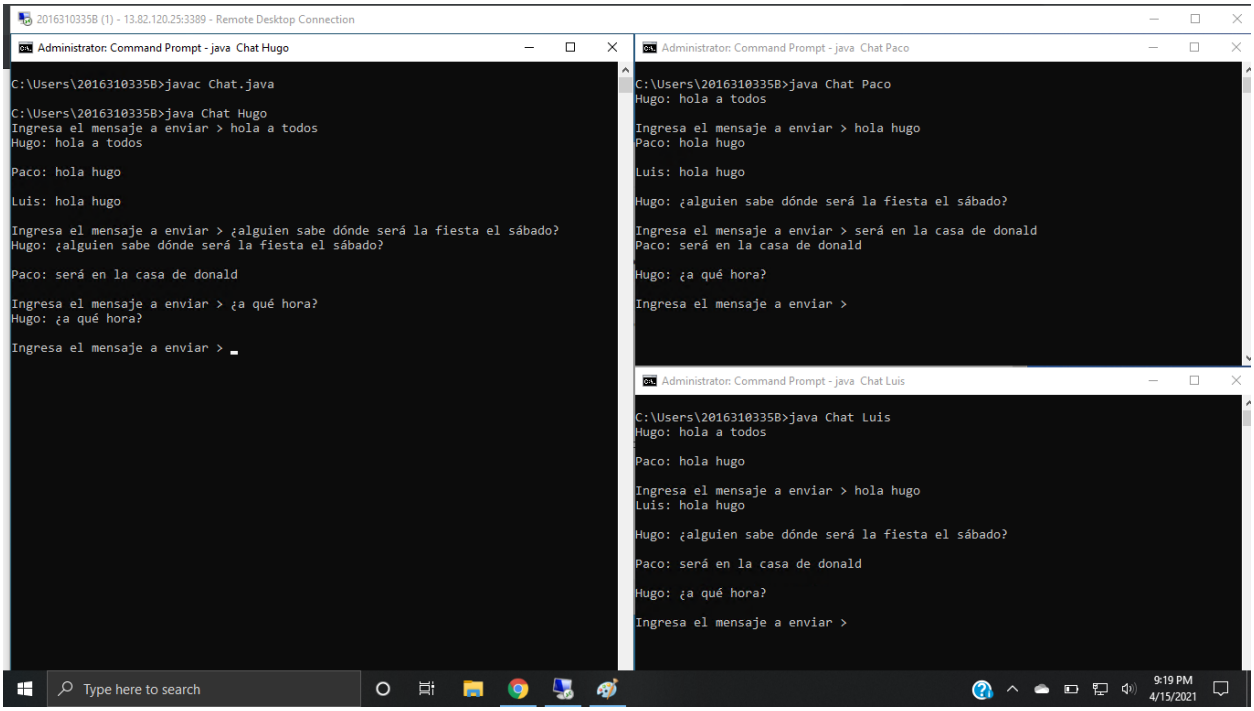

Imagen 5. Envío de mensajes con caracteres especiales.

Posteriormente los usuarios se despiden y salen del grupo al detener el programa.

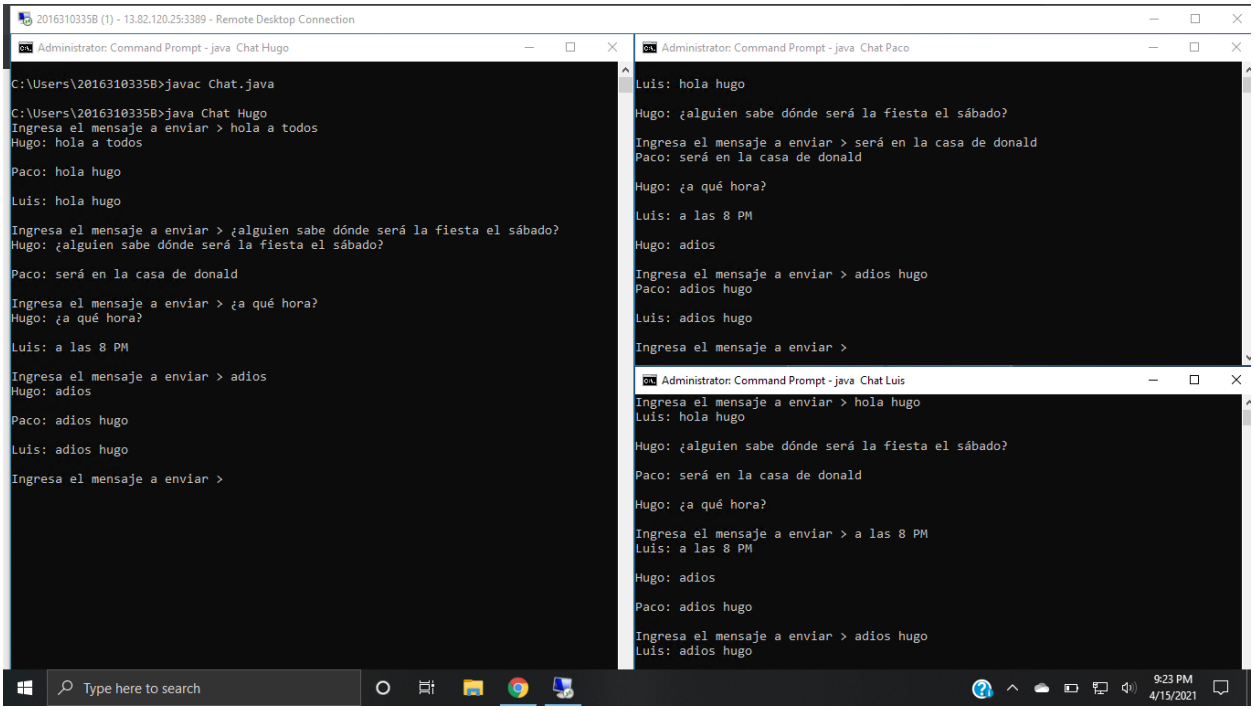

Imagen 6. Despedida de los usuarios

Código del programa.

El programa tiene dos secciones principales, la sección de lectura donde se está al pendiente de cualquier mensaje que escriban otros usuarios en el grupo y la sección de escritura donde se lee el mensaje que el usuario escribe y lo envía al grupo para que los otros usuarios lo lean.

La sección de lectura se realiza en un hilo diferente al hilo principal del programa, dentro de este hilo en un ciclo infinito se ejecuta el método recibe_mensaje_multicast() que recibe como parámetro un socket,

En la sección de escritura que se ejecuta sobre el hilo principal, dentro de un ciclo while infinito se espera a que el usuario escriba un mensaje y posteriormente lo envíe al grupo. Cabe mencionar que la longitud máxima del mensaje es de 1024 bytes, o bien 1024 caracteres, considerando al nombre de usuario que va concatenado al inicio de cada mensaje, para que los demás usuarios en el grupo puedan identificar la fuente del mensaje y un marcador de longitud fija que es concatenado al final del mensaje escrito por el usuario, para que en caso de no completar los 1024 bytes con el nombre y mensaje, se pueda identificar su fin y no se cree un String con un buffer que puede contener basura.

```
1. import java.net.*;
2. import java.io.*;
3. import java.util.*;
4.
5. public class Chat{
6.
7.   private static final String HOST = "230.0.0.0";
8.   private static final int PUERTO = 5000;
9.   private static final int MAX_LEN = 1024;
10.       private static String nombre;
11.       private static final String FIN_DE_TEXTO = "$FIN";
12.       private static int maxima_longitud;
13.
14.       static class Worker extends Thread{
15.         public void run(){
16.           try{
17.           InetAddress grupo = InetAddress.getByName( HOST );
18.           MulticastSocket socket = new MulticastSocket( PUERTO
   );
19.           socket.joinGroup( grupo );
20.           String mensaje;
21.           int indice;
22.           while( true ){
23.                 mensaje = new String( recibe_mensaje_multicast(
   socket, MAX_LEN ));
24.                 indice = mensaje.indexOf( FIN_DE_TEXTO );
25.                 mensaje = mensaje.substring(0, indice);
```

```
26.                 System.out.println("\r" + mensaje + "
   \n" );
27.                   System.out.print("\r"+ "Ingresa el mensaje a
   enviar"+ " > ");
28.           }
29.           }catch( IOException ioex ){
30.             System.out.println( "Error al recibir mensaje" );
31.           }
32.         }
33.
34.         static byte[] recibe_mensaje_multicast(MulticastSocket
   socket, int longitud_mensaje) throws IOException{
35.           byte[] buffer = new byte[ longitud_mensaje ];
36.           DatagramPacket paquete = new DatagramPacket( buffer,
   buffer.length );
37.           socket.receive( paquete );
38.           return paquete.getData();
39.         }
40.       }
41.
42.       public static void main( String[] args ){
43.         if( args.length == 0 ){
44.           System.out.println("Debes pasar como argumento el
   nombre del usuario");
45.           System.out.println("Ejemplode uso: java Chat <nombre-
   usuario>");
46.           System.exit(0);
47.         }else{
48.           try{
49.           Worker w = new Worker();
50.           w.start();
51.
52.           nombre = args[0];
53.           maxima_longitud = MAX_LEN + nombre.length() +
   FIN_DE_TEXTO.length() - 2;
54.           Scanner sc = new Scanner(System.in, "UTF-16");
55.           String mensaje;
56.           while( true ){
57.               System.out.print("\r" + "Ingresa el mensaje a
   enviar" + " > ");
58.               mensaje = System.console().readLine();
59.               if ( mensaje.length() > maxima_longitud ){
60.                 System.out.println("El mensaje no puede exceder
   los " + maxima_longitud + "caracteres" );
61.               }else{
```

```
62.                 mensaje = nombre + ": " + mensaje + FIN_DE_TEXTO;
63.                 envia_mensaje_multicast( mensaje.getBytes(),
   HOST, PUERTO );
64.               }
65.           }
66.           }catch( IOException ioex ){
67.             System.out.println("Error al enviar mensaje");
68.           }
69.         }
70.       }
71.
72.       static void envia_mensaje_multicast(byte[] buffer, String
   ip, int puerto) throws IOException{
73.         DatagramSocket socket = new DatagramSocket();
74.         socket.send( new DatagramPacket( buffer, buffer.length,
   InetAddress.getByName( ip ), puerto ));
75.         socket.close();
76.       }
77.     }
78.
```

Conclusión.

después de haber trabajado con sockets de flujo es conveniente entender el funcionamiento de los sockets multicast y sobre todo las facilidades que pueden presentar en ciertos escenarios de desarrollo, donde puede ser más conveniente su uso en lugar de los sockets de flujo.

Adicionalmente, la practica me obligo a poner en práctica el uso de escritorios remotos, que pocas veces había utilizado para conectarme a una computadora, sin embargo, es una de las razones por las que comencé a interesarme por la carrera de sistemas computacionales, pues en una ocasión hace años me sorprendió presenciar que un técnico de la compañía de teléfonos que me proporciona internet haya arreglado un problema en mi conexión a internet a través de una conexión de este estilo.