



Instituto Politécnico Nacional



Escuela Superior de Cómputo

Desarrollo de Sistemas Distribuidos

Tarea 3. Chat Multicast

López Sánchez Kevin Ian

Grupo: 4CV12

Profesor: Pineda Guerrero Carlos

29 de marzo de 2023

Objetivo

Utilizando ChatGPT desarrollar un solo programa en Java en modo consola que implemente un chat utilizando comunicación multicast mediante datagramas.

Se deberá ejecutar el programa en una máquina virtual con Windows Server 2012 en Azure.

El programa deberá hacer lo siguiente:

- El programa creará un thread el cual recibirá los mensajes del resto de los nodos. Cada mensaje recibido será desplegado en la pantalla. El thread desplegará el mensaje que envía el mismo nodo.
- En el método main(), dentro de un ciclo infinito: Se desplegará el siguiente prompt: "Escribe tu mensaje: " (sin las comillas), entonces se leerá del teclado el mensaje.

Se deberá enviar el mensaje a los nodos que pertenecen al grupo identificado por la IP 239.0.0.0 a través del puerto 50000. El mensaje a enviar en el datagrama deberá tener la siguiente forma:

`nombre_usuario-->mensaje_ingresado`

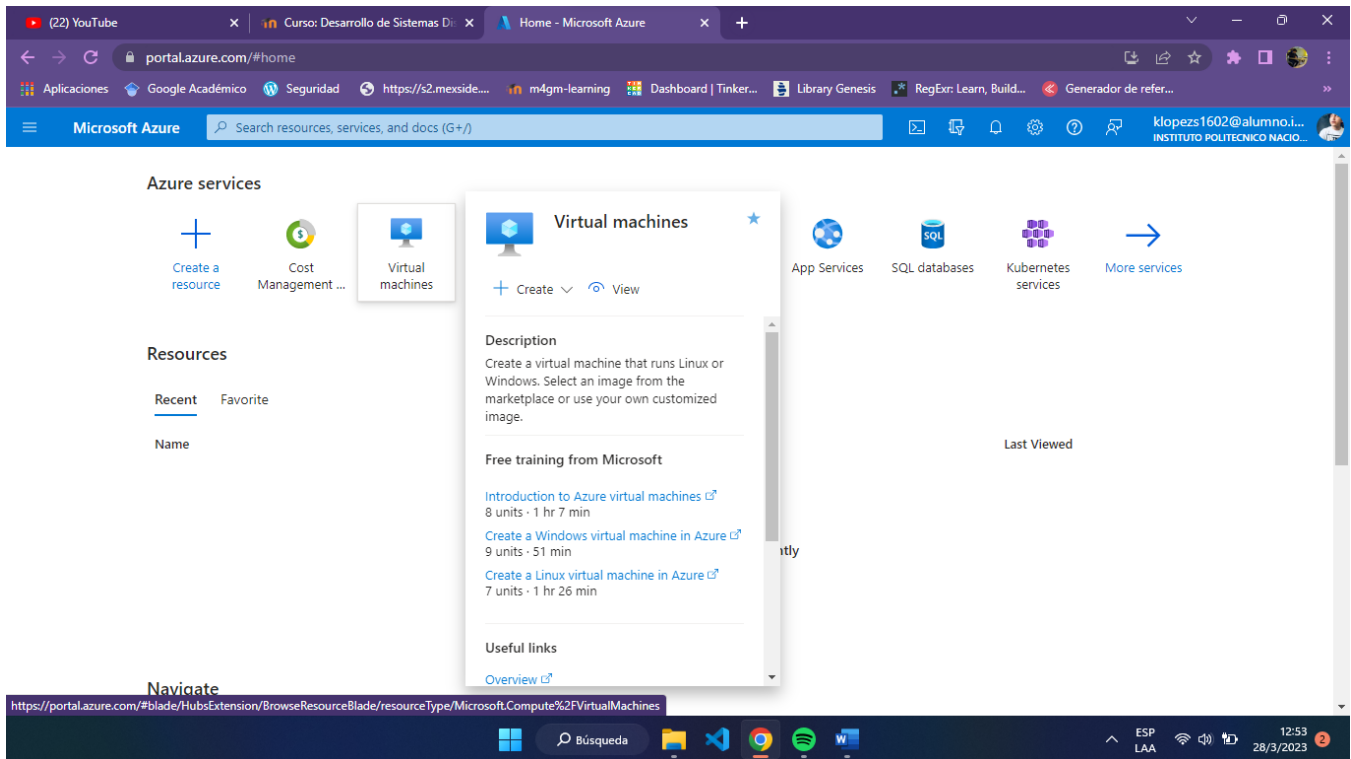
Debido a que el protocolo UDP no garantiza el orden en que se reciben los mensajes, no se deberá enviar la longitud del mensaje y luego el mensaje; se deberá utilizar una longitud de mensaje fija. Al desplegar el mensaje recibido se deberán eliminar los espacios adicionales a la derecha (si los hay).

Desarrollo

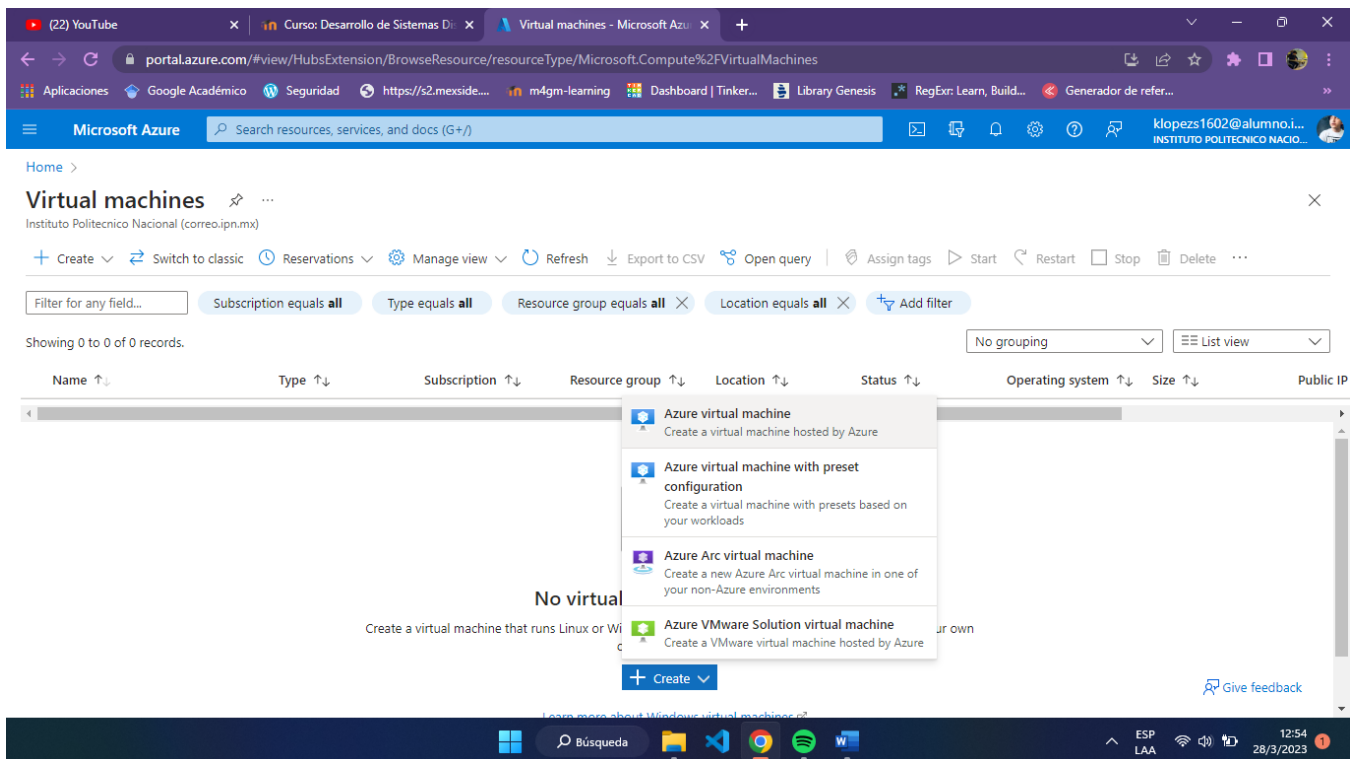
Creación de máquina virtual de Windows Server en Azure

Para realizar esta práctica es necesario crear una máquina virtual de Windows Server 2012 en nuestro portal de Azure y a continuación explicaremos los pasos para este proceso:

1. Entramos al portal de Azure y vamos al apartado de “Máquinas Virtuales”.



2. Seleccionamos la opción de “crear” y luego “Azure virtual machine”.



3. Seleccionamos el grupo de recursos o creamos uno nuevo, escribimos el nombre de nuestra máquina virtual y seleccionamos la opción de Windows Server 2012

(22) YouTube x Curso: Desarrollo de Sistemas D... x SISDIS-4CV12: Tarea 3. Chat multi... x Create a virtual machine - Micro... x +

portal.azure.com/#create/Microsoft.VirtualMachine-ARM

Aplicaciones Google Académico Seguridad https://s2.mexside... m4gm-learning Dashboard | Tinker... Library Genesis RegEx: Learn, Build... Generador de refer...

Microsoft Azure Search resources, services, and docs (G+)

Home > Virtual machines >

Create a virtual machine

⚠ Changing Basic options may reset selections you have made. Review all options prior to creating the virtual machine.

Subscription * ⓘ Azure for Students

Resource group * ⓘ (New) ESCOM
[Create new](#)

Instance details

Virtual machine name * ⓘ T3-2020630205 ✓

Region * ⓘ (US) East US

Availability options ⓘ No infrastructure redundancy required

Security type ⓘ Standard

Image * ⓘ Windows Server 2012 Datacenter - x64 Gen2
[See all images](#) | [Configure VM generation](#)

[Review + create](#) < Previous Next : Disks > [Give feedback](#)

Windows Taskbar: Búsqueda, File Explorer, VS Code, Chrome, Spotify, Word. System tray: ESP LAA, 13:06, 28/3/2023.

4. Seleccionamos 2 Gb para la memoria RAM de nuestra máquina e ingresamos el usuario y la contraseña para ingresar posteriormente.

(22) YouTube x Curso: Desarrollo de Sistemas D... x Create a virtual machine - Micro... x +

portal.azure.com/#create/Microsoft.VirtualMachine-ARM

Aplicaciones Google Académico Seguridad https://s2.mexside... m4gm-learning Dashboard | Tinker... Library Genesis RegEx: Learn, Build... Generador de refer...

Microsoft Azure Search resources, services, and docs (G+)

Home > Virtual machines >

Create a virtual machine

⚠ Changing Basic options may reset selections you have made. Review all options prior to creating the virtual machine.

ⓘ Arm64 is not supported with the selected image.

Run with Azure Spot discount ⓘ ☐

Size * ⓘ Standard_B1ms - 1 vcpu, 2 GiB memory (\$17.96/month)
[See all sizes](#)

Administrator account

Username * ⓘ kwindows ✓

Password * ⓘ ✓

Confirm password * ⓘ ✓

Inbound port rules

[Review + create](#) < Previous Next : Disks > [Give feedback](#)

Windows Taskbar: Búsqueda, File Explorer, VS Code, Chrome, Spotify, Word. System tray: ESP LAA, 13:03, 28/3/2023.

5. Seleccionamos el puerto de entrada por el que accederemos a la máquina virtual, en este caso el de RDP.

Home > Virtual machines >

Create a virtual machine

⚠ Changing Basic options may reset selections you have made. Review all options prior to creating the virtual machine.

Inbound port rules

Select which virtual machine network ports are accessible from the public internet. You can specify more limited or granular network access on the Networking tab.

Public inbound ports * ☐ None ☒ Allow selected ports

Select inbound ports *

⚠ This will allow all IP addresses to access your virtual machine. This is only recommended for testing. Use the Advanced controls in the Networking tab to create rules to limit inbound traffic to known IP addresses.

[Review + create](#) [< Previous](#) [Next : Disks >](#) [Give feedback](#)

6. Seleccionamos como almacenamiento un HDD estándar

Home > Virtual machines >

Create a virtual machine

ℹ Encryption at host is not registered for the selected subscription. [Learn more about enabling this feature](#)

OS disk

OS disk type *
The selected VM size supports premium disks. We recommend Premium SSD for high IOPS workloads. Virtual machines with Premium SSD disks qualify for the 99.9% connectivity SLA.

Delete with VM ☒

Key management

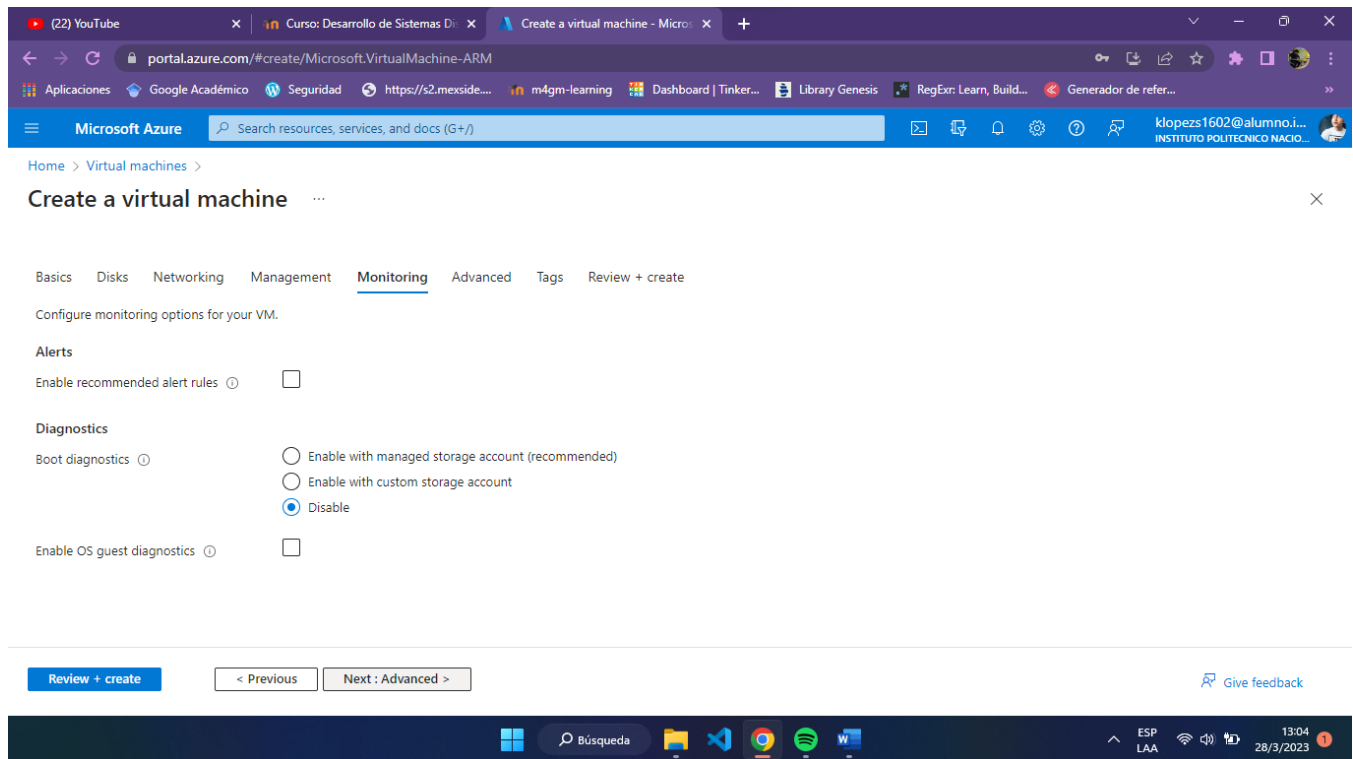
Enable Ultra Disk compatibility ☐
Ultra disk is supported in Availability Zone(s) 1,2,3 for the selected VM size Standard_B1ms.

Data disks

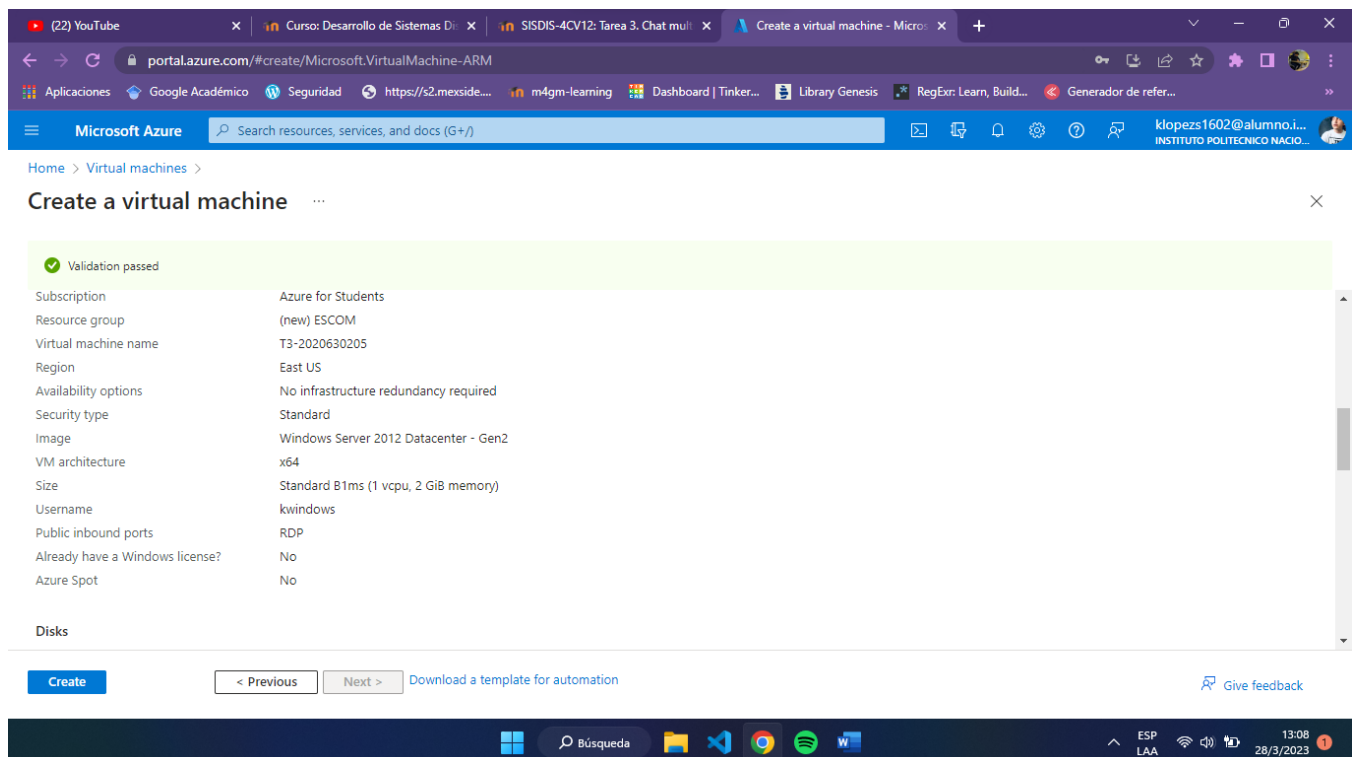
You can add and configure additional data disks for your virtual machine or attach existing disks. This VM also comes with a

[Review + create](#) [< Previous](#) [Next : Networking >](#) [Give feedback](#)

7. Desactivamos el diagnóstico de inicio.



8. Nos aseguramos de que todo se haya configurado correctamente y creamos la máquina virtual



9. Esperamos a que nuestra máquina termine de crearse.

The screenshot shows the Azure portal interface with the deployment 'CreateVm-MicrosoftWindowsServer.WindowsServer-201-20230328125629' in progress. The deployment details table lists the following resources:

Resource	Type	Status	Operation details
T3-2020630205	Microsoft.Compute/virtua...	Created	Operation details
t3-2020630205742	Microsoft.Network/netwo...	Created	Operation details
T3-2020630205	Microsoft.Network/virtual...	OK	Operation details
T3-2020630205-nsg	Microsoft.Network/netwo...	OK	Operation details
T3-2020630205-ip	Microsoft.Network/public...	OK	Operation details

Deployment summary: Deployment name: CreateVm-MicrosoftWindowsServer.WindowsSe..., Subscription: Azure for Students, Resource group: ESCOM, Start time: 3/28/2023, 1:09:39 PM, Correlation ID: c8ee3e49-bdea-4a34-ba0d-63d.

The screenshot shows the Azure portal interface with the deployment 'CreateVm-MicrosoftWindowsServer.WindowsServer-201-20230328125629' completed. The deployment details table lists the following resources:

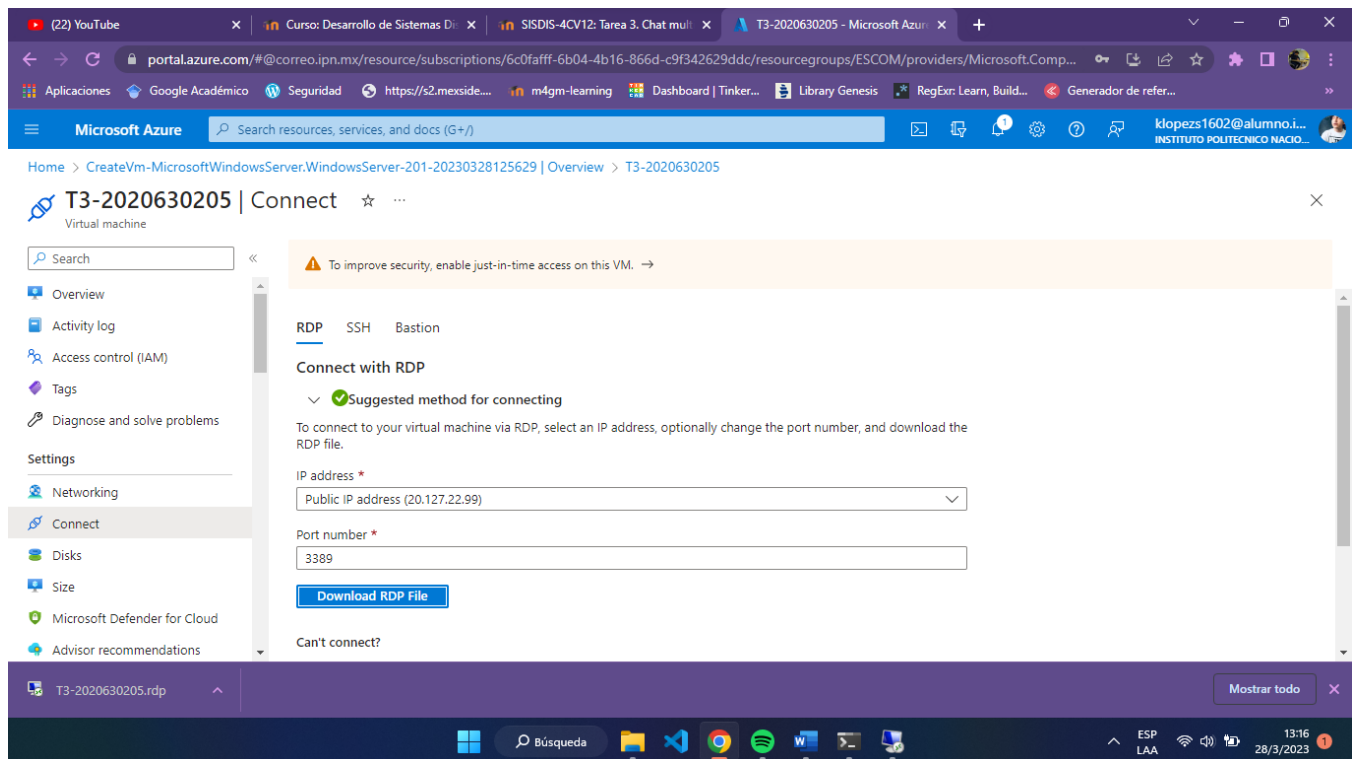
Resource	Type	Status	Operation details
T3-2020630205	Microsoft.Compute/virtua...	Created	Operation details
t3-2020630205742	Microsoft.Network/netwo...	Created	Operation details
T3-2020630205	Microsoft.Network/virtual...	OK	Operation details
T3-2020630205-nsg	Microsoft.Network/netwo...	OK	Operation details
T3-2020630205-ip	Microsoft.Network/public...	OK	Operation details

Deployment summary: Deployment name: CreateVm-MicrosoftWindowsServer.WindowsSe..., Subscription: Azure for Students, Resource group: ESCOM, Start time: 3/28/2023, 1:09:39 PM, Correlation ID: c8ee3e49-bdea-4a34-ba0d-63d.

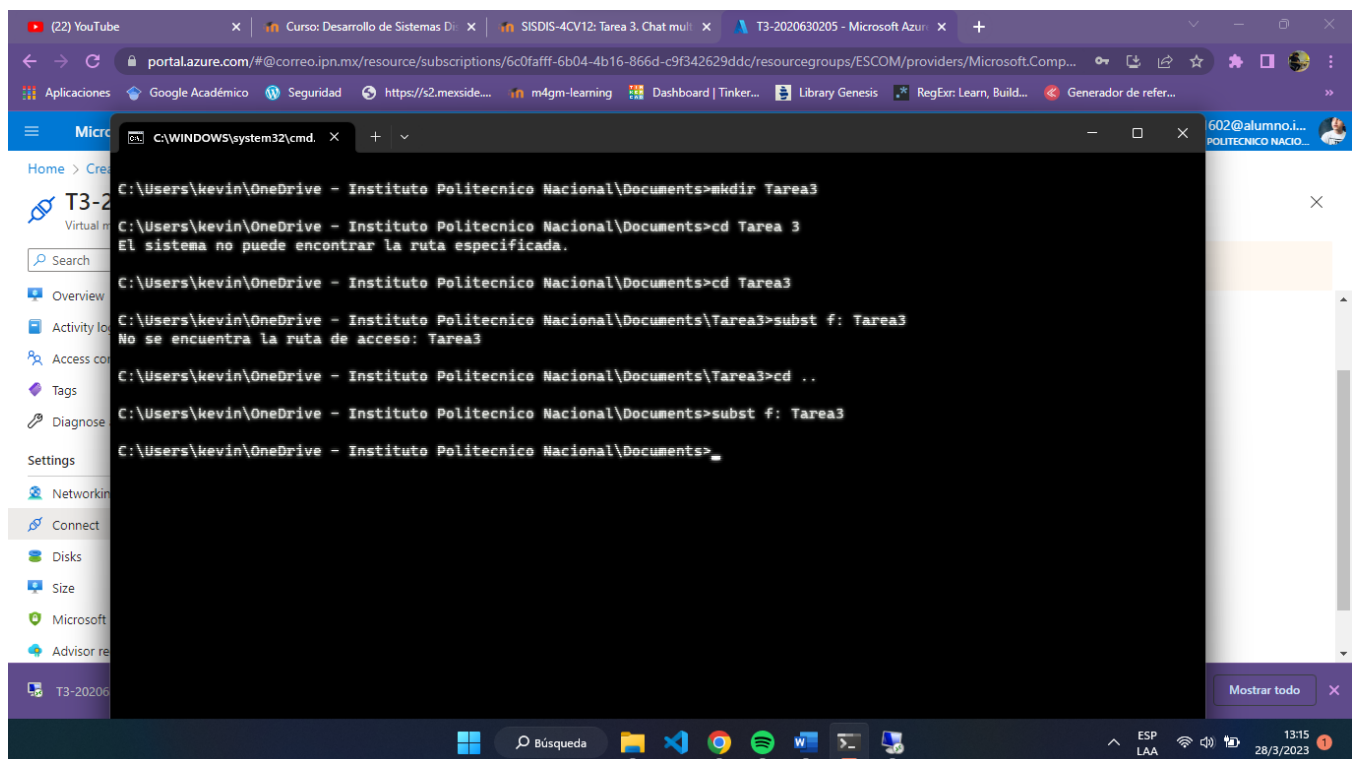
Next steps: Setup auto-shutdown Recommended, Monitor VM health, performance and network dependencies Recommended, Run a script inside the virtual machine Recommended.

Buttons: Go to resource, Create another VM.

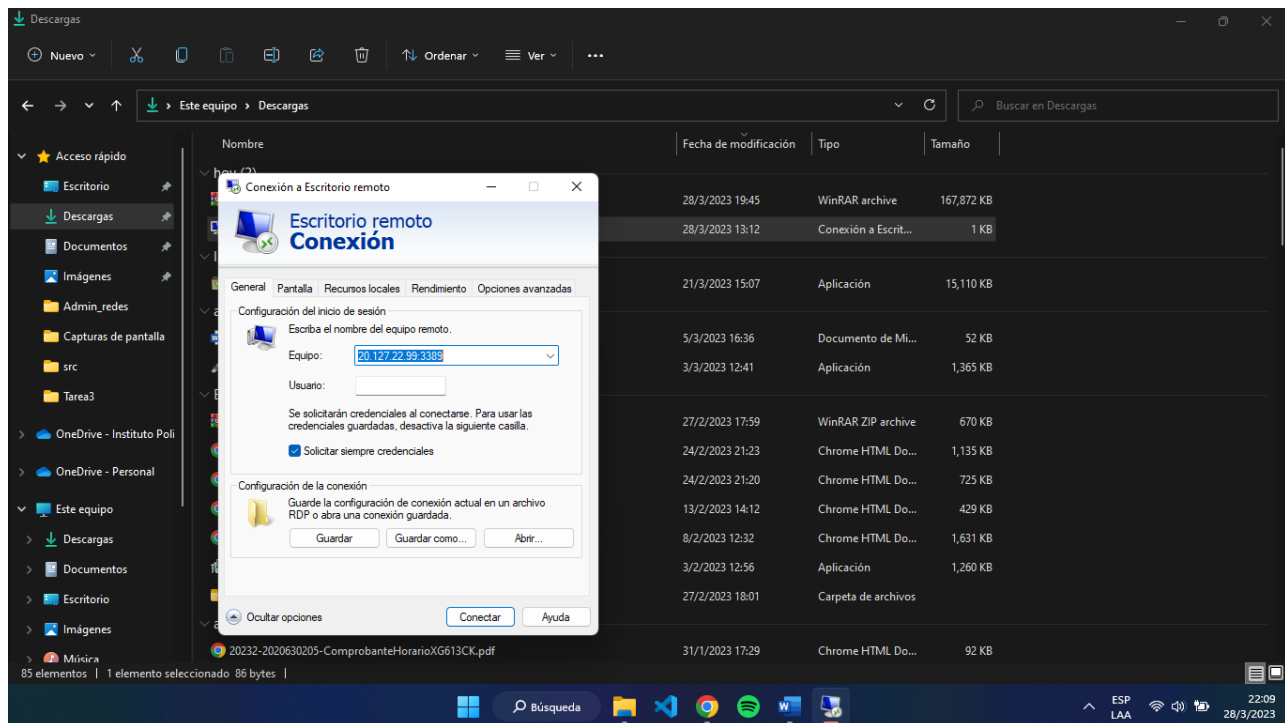
10. Una vez creada, descargamos el archivo RDP para la conexión.



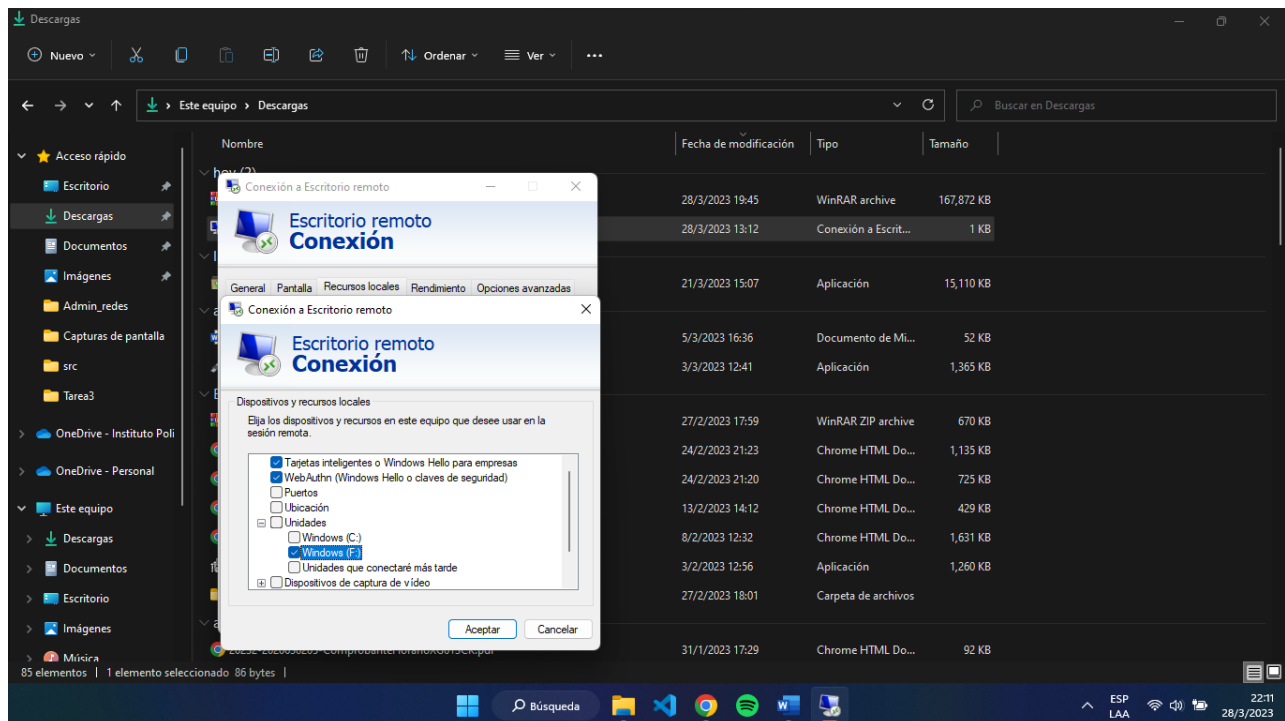
11. Creamos un apartado en nuestro disco por el cual compartiremos archivos a la máquina virtual



12. En el archivo descargado damos clic derecho y modificar, nos aparecerá la siguiente pantalla:



13. Seleccionamos la “partición del disco” que se creó cuando creamos la carpeta en el apartado recursos locales.



14. Y ya podemos entrar a nuestra máquina virtual con nuestras credenciales anteriormente creadas.

Conversación con Chat GPT

Yo:

Requiero escribir un código que implemente un chat multicast multihilo en el lenguaje de programación Java en el que se deberá pasar como parámetro al momento de su ejecución el nombre del usuario

ChatGPT:

Aquí te dejo un ejemplo de código en Java que implementa un chat multicast multihilo y recibe el nombre del usuario como parámetro:

```
import java.net.*;

import java.io.*;

public class MulticastChat implements Runnable {

    private MulticastSocket socket;

    private InetAddress group;

    private int port = 1234; // Puerto del grupo multicast

    private String username;

    public MulticastChat(String username) {

        try {

            this.username = username;

            socket = new MulticastSocket(port);

            group = InetAddress.getByName("230.0.0.0"); // Dirección del grupo multicast

            socket.joinGroup(group);

        } catch(IOException ex) {
```

```
        ex.printStackTrace();
    }
}
```

```
public void run() {
    while(true) {
        try {
            byte[] buffer = new byte[1000];

            DatagramPacket packet = new DatagramPacket(buffer, buffer.length);

            socket.receive(packet);

            String message = new String(packet.getData(), 0, packet.getLength());

            System.out.println(message);
        } catch(IOException ex) {
            ex.printStackTrace();
        }
    }
}
```

```
public void send(String message) {
    try {
        message = "[" + username + ": " + message;

        byte[] buffer = message.getBytes();

        DatagramPacket packet = new DatagramPacket(buffer, buffer.length, group, port);

        socket.send(packet);
    }
}
```

```
    } catch(IOException ex) {  
        ex.printStackTrace();  
    }  
}
```

```
public static void main(String[] args) {  
    if(args.length != 1) {  
        System.out.println("Debe proporcionar el nombre de usuario como argumento.");  
        return;  
    }  
}
```

```
MulticastChat chat = new MulticastChat(args[0]);  
Thread thread = new Thread(chat);  
thread.start();
```

```
BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));  
while(true) {  
    try {  
        String message = reader.readLine();  
        chat.send(message);  
    } catch(IOException ex) {  
        ex.printStackTrace();  
    }  
}
```

```
}  
}
```

Explicación:

La clase MulticastChat implementa la interfaz Runnable para que pueda ser ejecutada en un hilo aparte.

En el constructor se crea un MulticastSocket y se une al grupo multicast con la dirección 230.0.0.0.

En el método run, el hilo se mantiene escuchando los mensajes que llegan al grupo multicast y los imprime por pantalla.

El método send se encarga de enviar un mensaje al grupo multicast, incluyendo el nombre de usuario proporcionado al principio.

En el método main, se crea una instancia de MulticastChat con el nombre de usuario proporcionado como argumento y se inicia un hilo con esa instancia.

A continuación, se lee continuamente la entrada del usuario desde la consola y se llama al método send para enviar cada mensaje al grupo multicast.

Para ejecutar el programa, se debe compilar el archivo Java y ejecutar el archivo .class generado, pasando el nombre de usuario como argumento:

```
javac MulticastChat.java
```

```
java MulticastChat nombre_de_usuario
```

Yo:

Y si deseo mantener una longitud del mensaje fija, para no tener que mandar la longitud del paquete y rellenar a la derecha con espacios.

ChatGPT:

Si deseas mantener una longitud fija para los mensajes, una opción es definir un tamaño máximo para los mensajes que se envían y reciben. Luego, puedes rellenar los mensajes con caracteres adicionales (por ejemplo, espacios en blanco) para que todos los mensajes tengan la misma longitud.

Aquí te muestro cómo puedes modificar los métodos `enviarMensaje` y `recibirMensajes` para que funcionen con mensajes de longitud fija:

```
public void enviarMensaje(String mensaje, int longitudMaxima) throws IOException {  
    // Rellenar el mensaje con espacios en blanco para que tenga la longitud máxima  
    mensaje = String.format("%-" + longitudMaxima + "s", mensaje);  
  
    byte[] mensajeBytes = mensaje.getBytes("UTF-8");  
  
    DatagramPacket paquete = new DatagramPacket(mensajeBytes, mensajeBytes.length,  
grupo, puerto);  
  
    socket.send(paquete);  
}  
  
public String recibirMensajes(int longitudMaxima) throws IOException {  
    byte[] buffer = new byte[longitudMaxima];  
  
    DatagramPacket paquete = new DatagramPacket(buffer, buffer.length);  
  
    socket.receive(paquete);  
  
    // Convertir los bytes recibidos a una cadena y eliminar los espacios en blanco al final  
    String mensaje = new String(paquete.getData(), 0, paquete.getLength(), "UTF-8").trim();  
  
    return mensaje;  
}
```

En el método `enviarMensaje`, utilizamos el método `String.format` para rellenar el mensaje con espacios en blanco hasta que tenga la longitud máxima especificada. Luego, convertimos el mensaje a bytes y lo enviamos en un paquete UDP.

En el método recibirMensajes, creamos un buffer de bytes con la longitud máxima especificada. Cuando recibimos un paquete UDP, convertimos los bytes recibidos a una cadena utilizando el constructor `String(byte[] bytes, int offset, int length, String charsetName)` y eliminamos los espacios en blanco al final de la cadena utilizando el método `trim()`.

Ten en cuenta que si el mensaje original es más largo que la longitud máxima especificada, se recortará el mensaje y se perderá la parte final del mensaje. Por lo tanto, es importante elegir una longitud máxima que sea lo suficientemente grande para acomodar la mayoría de los mensajes que se esperan recibir en el chat.

Yo:

Impleméntalo en el código que antes has desarrollado con una longitud fija de 1024 bytes.

ChatGPT:

Por supuesto, aquí te muestro cómo modificar el código para que el mensaje tenga una longitud fija de 1024 caracteres:

```
import java.net.*;
```

```
import java.io.*;
```

```
public class ChatMulticast implements Runnable {
```

```
    private MulticastSocket socket;
```

```
    private InetAddress group;
```

```
    private int port = 50000; // Puerto del grupo multicast
```

```
    private String username;
```

```
    public ChatMulticast(String username) {
```

```
        try {
```

```
            this.username = username;
```

```
            socket = new MulticastSocket(port);
```

```
            group = InetAddress.getByName("239.0.0.0"); // Dirección del grupo multicast
```

```
        socket.joinGroup(group);
    } catch(IOException ex) {
        ex.printStackTrace();
    }
}
```

```
public void run() {
    while(true) {
        try {
            byte[] buffer = new byte[1024];
            DatagramPacket packet = new DatagramPacket(buffer, buffer.length);
            socket.receive(packet);

            String message = new String(packet.getData(), "UTF-8").trim(); // elimina los
espacios vacíos al final

            System.out.println(message);
            System.out.print("Escribe tu mensaje: ");
        } catch(IOException ex) {
            ex.printStackTrace();
        }
    }
}
```

```
public void send(String message) {
    try {
        message = username + " ---> " + message;
```



```

        byte[] buffer = new byte[1024];

        byte[] bytes = message.getBytes("UTF-8");

        System.arraycopy(bytes, 0, buffer, 0, bytes.length); // copia los bytes en el buffer

        DatagramPacket packet = new DatagramPacket(buffer, buffer.length, group, port);

        socket.send(packet);

    } catch(IOException ex) {

        ex.printStackTrace();

    }

}

```

```

public static void main(String[] args) {

    if(args.length != 1) {

        System.out.println("Debe proporcionar el nombre de usuario como argumento.");

        return;

    }

```

```

    ChatMulticast chat = new ChatMulticast(args[0]);

    Thread thread = new Thread(chat);

    thread.start();

```

```

    BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));

    System.out.print("Escribe tu mensaje: ");

    while(true) {

        try {

```

```

        String message = reader.readLine();

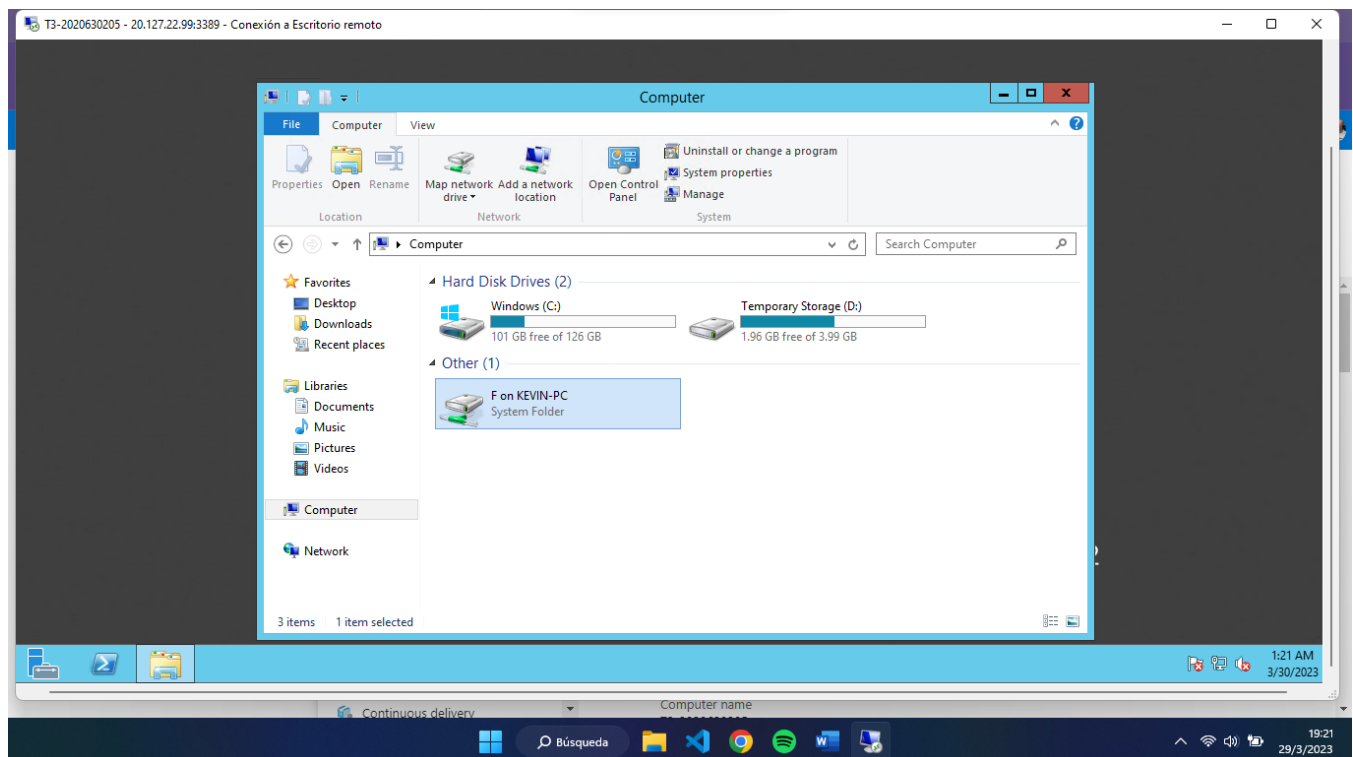
        chat.send(message);
    } catch(IOException ex) {
        ex.printStackTrace();
    }
}
}
}

```

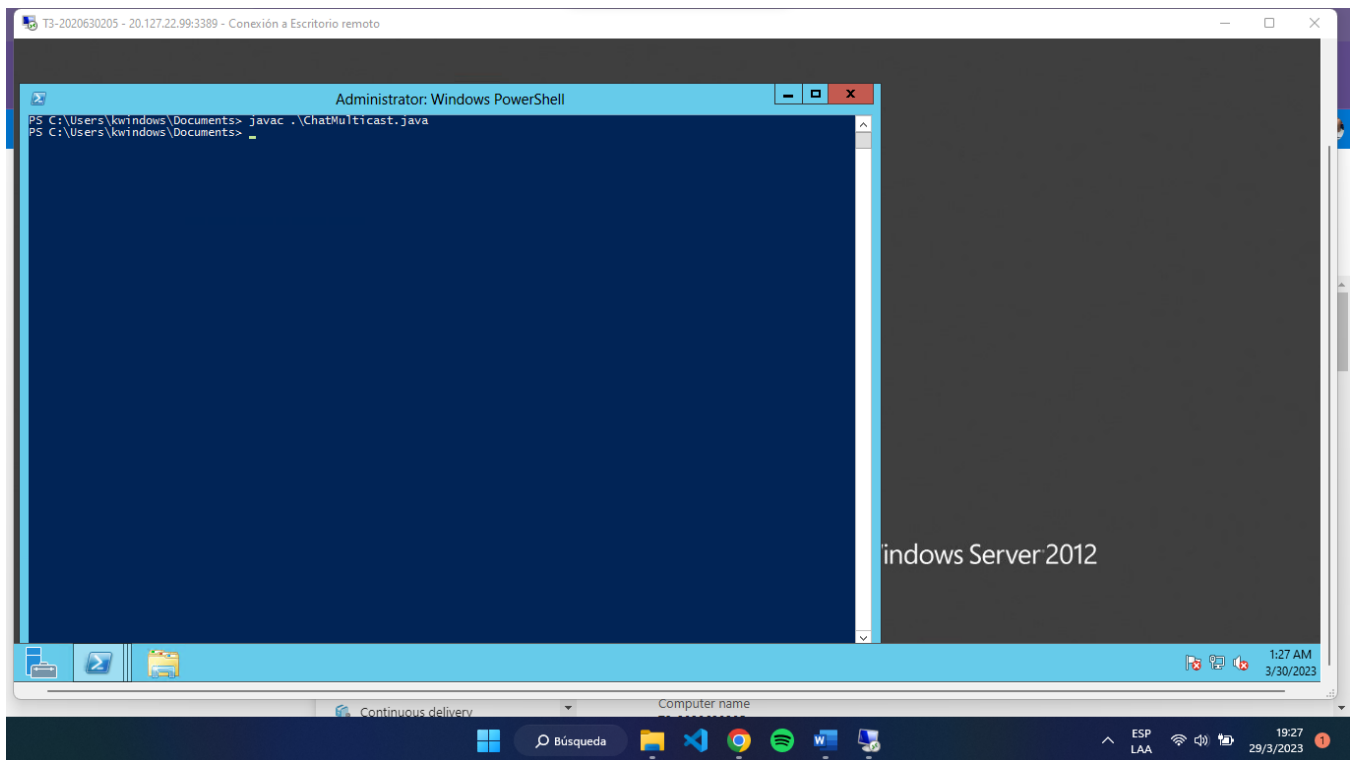
Con esta modificación, el mensaje se convertirá a un arreglo de bytes de longitud 1024, y si el mensaje es más corto, los bytes restantes se llenarán con ceros. Si el mensaje es más largo, se truncará a 1024 caracteres. De esta forma, no es necesario enviar la longitud del paquete en cada envío de mensaje.

Implementación

Una vez que nos encontramos dentro de nuestra máquina virtual, sólo queda transferir los archivos del código e instalar java para compilar y ejecutar nuestro programa.



Ingresamos a la terminal de la máquina virtual de Windows Server 2012 y compilamos el programa



Para ejecutar abrimos 3 ventanas de la terminal y ejecutamos el programa agregando el nombre de los usuarios de la prueba como parámetro. Hugo, Paco y Luis.

Conclusiones

A través de realizar la presente tarea, se puede comprender la complejidad de trabajar con sockets, especialmente con sockets multicast. Fue necesario entender cómo funcionan los grupos multicast, las direcciones IP y los puertos en este tipo de comunicación. También se necesitó tener un conocimiento sólido sobre la programación en hilos, ya que el chat multicast en Java debía manejar múltiples hilos para permitir tanto la recepción como el envío de mensajes simultáneamente.

En cuanto a la ejecución en una máquina virtual de Windows Server 2012, se evidenció la importancia de considerar la configuración de la codificación de caracteres de la terminal para poder visualizar adecuadamente caracteres especiales como acentos y signos de interrogación. Además, se pudo observar que las diferentes versiones de Windows pueden tener configuraciones diferentes que pueden afectar el funcionamiento del programa.

En general, la tarea de desarrollar y ejecutar un chat multicast en Java en una máquina virtual de Windows Server 2012 permitió poner en práctica conceptos teóricos y adquirir conocimientos prácticos sobre programación en red, comunicación de sockets, y configuraciones de terminal, lo que puede resultar útil para futuros proyectos y desarrollos en este ámbito.