



Instituto Politécnico Nacional

Escuela Superior de Cómputo



Diseño de Sistemas Distribuidos

Tarea 5 – Equipo 2

Docente:

Dr. Pineda Guerrero Carlos

Alumnos:

Cazares Martínez Maximiliano

Chavarría Vázquez Luis Enrique

Cipriano Damián Sebastián

Grupo: 4CV11

CDMX, 18 de marzo de 2022

Índice

Descripción del problema.....	4
Desarrollo de la práctica.	6
• Implementación y creación de recursos en Azure.	6
Conclusiones	22
Cazares Martínez Maximiliano	22
Chavarría Vázquez Luis Enrique	22
Cipriano Damián Sebastián	23

Tabla de ilustraciones

Ilustración 1 Primera fase para creación de la máquina virtual.	6
Ilustración 2 Segunda fase para la creación de la máquina virtual.	7
Ilustración 3 Configuración de la sección de DISCOS y REDES	8
Ilustración 4 configuración de la sección de administración de las máquinas virtuales.....	9
Ilustración 5 revisión y creación de la máquina virtual.....	10
Ilustración 6 proceso de creación y revisión llevándose a cabo.	11
Ilustración 7 proceso concluido de la máquina virtual	12
Ilustración 8 elementos de las máquinas virtuales.	13
Ilustración 9 muestra principal de la consola para conexión inicial	13
Ilustración 10 instalación del jre dentro de la terminal.....	14
Ilustración 11 instalación del jdk dentro de la terminal.....	15
Ilustración 12 listado de máquinas virtuales que hemos creado.....	16
Ilustración 13 puesta de los archivos a nuestras máquinas virtuales.	16
Ilustración 14 update de los archivos para el programa.	17
Ilustración 15 archivos y muestras de las consolas de los nodos.....	18
Ilustración 16 compilación de los archivos y muestra de directorio	18
Ilustración 17 comando rmregistry& en las consolas periféricas.....	19
Ilustración 18 ejecución de los servidores	20
Ilustración 19 resultado del checksum.....	20
Ilustración 20 resultado checksum e impresión de la matriz	21

Descripción del problema

A continuación, podemos leer las especificaciones y requerimientos para la Tarea número 5. Es este cable que esta práctica estará dividida fundamentalmente por esta sección en donde estamos describiendo el problema, luego podremos abordar algunas situaciones referentes al desarrollo y muestra de cómo fue elaborada cada una de las partes de la práctica para finalmente llegar a las conclusiones del equipo.

Se deberá desarrollar un sistema que calcule el producto de dos matrices cuadradas utilizando Java RMI, tal como se explicó en clase.

Se deberá ejecutar dos casos:

N=8, se deberá desplegar las matrices A, B y C y el checksum de la matriz C.

N=4000, se deberá desplegar el checksum de la matriz C.

Los elementos de las matrices A, B y C serán de tipo float y el checksum será de tipo float.

Se deberá inicializar las matrices A y B de la siguiente manera:

$$A[i][j] = i + 2 * j$$

$$B[i][j] = 3 * i - j$$

Se deberá dividir las matrices A y B en cuatro partes, por tanto la matriz C estará dividida en 16 partes:

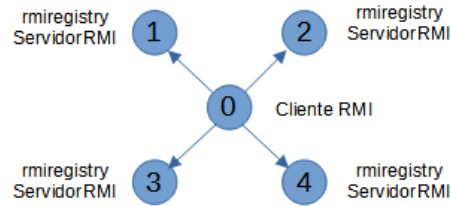
A1	X	B1	=	C1	C2	C3	C4
A2		B2		C5	C6	C7	C8
A3		B3		C9	C10	C11	C12
A4		B4		C13	C14	C15	C16

El cliente RMI ejecutará en una máquina virtual con Ubuntu en Azure (nodo 0).

El servidor RMI ejecutará en cuatro máquinas virtuales (nodo 1, nodo 2, nodo 3 y nodo 4) con Ubuntu en Azure.

El programa rmiregistry ejecutará en cada nodo donde ejecute el servidor RMI.

El nodo 1 calculará los productos C1, C2, C3 y C4, el nodo 2 calculará los productos C5, C6, C7 y C8, el nodo 3 calculará los productos C9, C10, C11 y C12, y el nodo 4 calculará los productos C13, C14, C15 y C16.



El cliente RMI inicializará las matrices A y B, realizará la transpuesta de la matriz B, obtendrá las matrices A1, A2, A3, A4, B1, B2, B3 y B4, invocará el método remoto `multiplica_matrices()`, obtendrá la matriz C a partir de las matrices C1 a la C16, y calculará el checksum de la matriz C. En el caso $N=8$ el cliente RMI deberá desplegar las matrices A, B y C (la matriz B antes de transponer).

Se **deberá** utilizar threads en el cliente RMI para invocar el método remoto `multiplica_matrices()` de manera que los servidores RMI calculen los productos en paralelo.

Se **deberá** subir a la plataforma el código fuente del sistema desarrollado (archivos texto) y un reporte de la tarea en formato PDF con portada, desarrollo y las conclusiones de cada integrante del equipo.

El archivo PDF deberá incluir las capturas de pantalla de la compilación y ejecución del programa, se **deberá incluir** la captura de pantalla correspondiente a **cada paso** de la creación de la primera máquina virtual (nodo 0).

No se admitirá la tarea si no incluye las pantallas correspondientes a cada paso del procedimiento de creación de la máquina virtual.

Cada captura de pantalla en el reporte deberá incluir una descripción.

El nombre de cada máquina virtual **deberá** ser: "Tarea-5-" concatenando el número de equipo, un guion y el número de nodo, por ejemplo, si el equipo es el 12, entonces el nodo 0 deberá llamarse: Tarea-5-12-0, el nodo 1 deberá llamarse Tarea-5-12-1, y así sucesivamente.

No se admitirá la tarea si los nodos no se nombran como se indicó anteriormente.

Valor de la tarea: 30% (2.1 puntos de la segunda evaluación parcial)

En la siguiente sección de la página de abajo procederemos a explicar los puntos fundamentales del desarrollo de la práctica.

Desarrollo de la práctica.

- **Implementación y creación de recursos en Azure.**

Primero que nada, debemos hacer la creación de nuestra máquina virtual, de modo tal que definamos el nombre de esta, como se indica en los requerimientos. Debemos evidentemente especificar el grupo de recursos que vamos a emplear, para lo cual hemos creado uno específicamente para la práctica y por otra parte también deberemos definir el nombre de nuestra máquina virtual, así como elegir el tipo de imagen o sistema operativo con la que correrá, para lo cual hemos elegido Ubuntu (esto de acuerdo con lo estipulado en la práctica).

Crear una máquina virtual - Microsoft Azure

https://portal.azure.com/?Microsoft_Azure_Education_correlationId=102197db68a64b6dbff706b146a396bb&Microsoft_Azure_Education_newA4E=true&Microsoft_Azure_Education...

Microsoft Azure

Inicio >

Crear una máquina virtual

Datos básicos | Discos | Redes | Administración | Opciones avanzadas | Etiquetas | Revisar y crear

Cree una máquina virtual que ejecute Linux o Windows. Seleccione una imagen de Azure Marketplace o use una imagen personalizada propia. Complete la pestaña Conceptos básicos y, después, use Revisar y crear para aprovisionar una máquina virtual con parámetros predeterminados o bien revise cada una de las pestañas para personalizar la configuración. [Más información](#)

Detalles del proyecto

Seleccione la suscripción para administrar recursos implementados y los costes. Use los grupos de recursos como carpetas para organizar y administrar todos los recursos.

Suscripción *

Grupo de recursos *
[Crear nuevo](#)

Detalles de instancia

Nombre de máquina virtual *

Región *

Opciones de disponibilidad

Tipo de seguridad

Imagen *
[Ver todas las imágenes](#) [Configurar la generación de máquinas virtuales](#)

Instancia de Azure de acceso puntual ☐

Tamaño *
[Ver todos los tamaños](#)

Cuenta de administrador

[Revisar y crear](#) [< Anterior](#) [Siguiente: Discos >](#)

26°C Soleado

6:41 PM 3/23/2022

ILUSTRACIÓN 1 PRIMERA FASE PARA CREACIÓN DE LA MÁQUINA VIRTUAL.

En la foto que aparece en la parte de autenticación la manera en que podemos validar los datos para la conexión al mismo, esto lo podemos hacer a través de establecer la contraseña y el nombre de usuario el cual a posteriori nos será de mucha utilidad para poder entablar la comunicación con nuestra máquina virtual, claro está todo este proceso depende de cómo es que elijamos realizar dicha conexión una vez que estén creadas todas las máquinas virtuales necesarias para cumplir con los requerimientos de la presente práctica.

ILUSTRACIÓN 2 SEGUNDA FASE PARA LA CREACIÓN DE LA MÁQUINA VIRTUAL.

Una vez concluido el proceso anterior, tenemos que irnos a la sección de discos en donde podemos elegir el tipo de almacenamiento que utilizará nuestra máquina virtual, al tratarse de una práctica que no requiere grandes prestaciones en la máquina podemos elegir el tipo de almacenamiento estándar o que es mejor conocido como un disco de almacenamiento duro. De igual modo podemos configurar el tipo de cifrado el cual precisamente por los requerimientos de la práctica se dejará como el tipo de cifrado predeterminado.

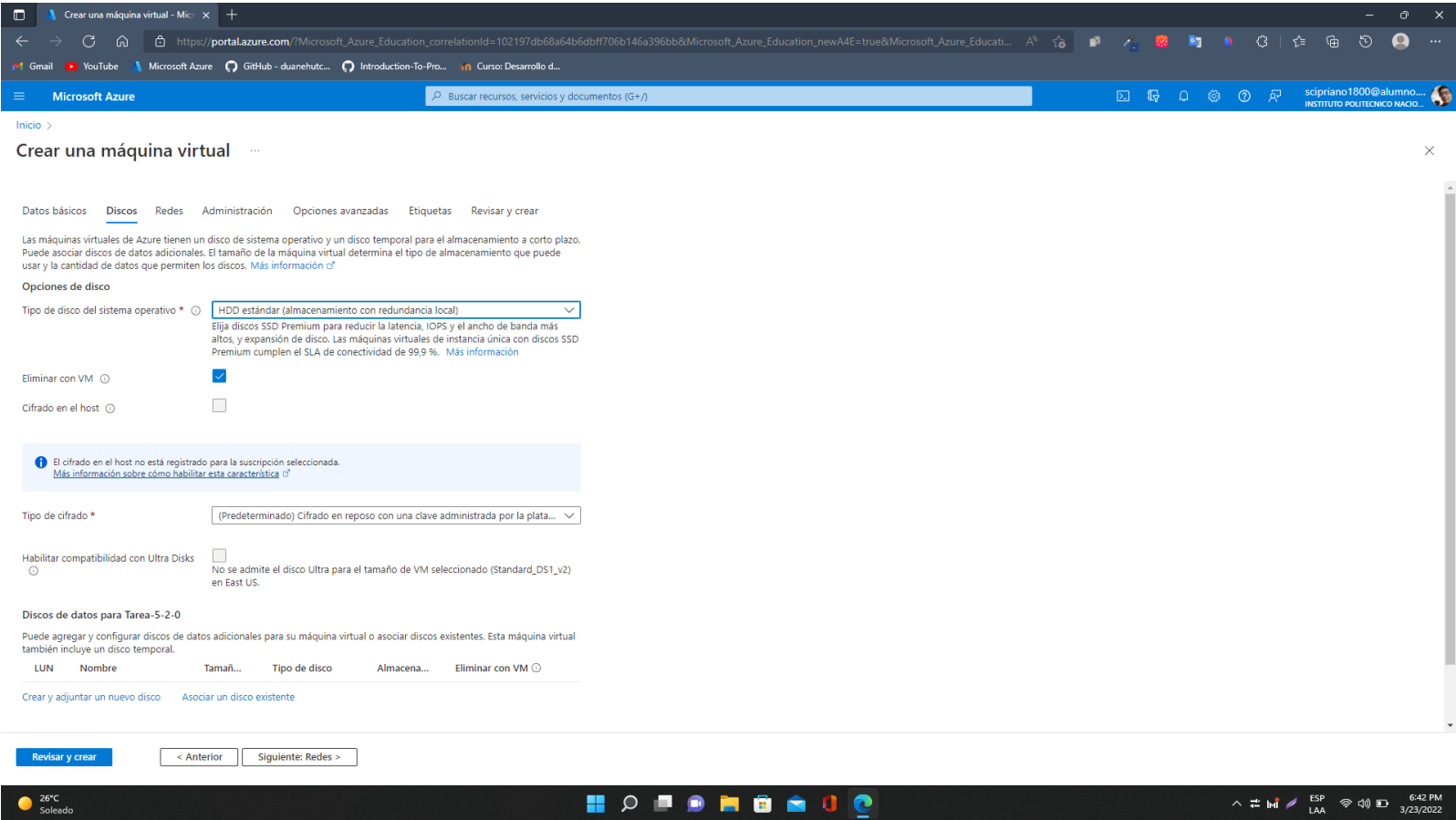


ILUSTRACIÓN 3 CONFIGURACIÓN DE LA SECCIÓN DE DISCOS Y REDES

Ya una vez que hemos configurado la parte de los discos también podemos configurar la parte de redes, podemos pasar inmediatamente a la parte de la configuración de la administración de dicha máquina virtual en donde para la parte del diagnóstico de arranque deberíamos dejarlo como deshabilitado, en adición a lo anteriormente mencionado las demás opciones debemos dejarlas tal cual y aparecen en un principio. Esto parece en la siguiente imagen de la siguiente página.

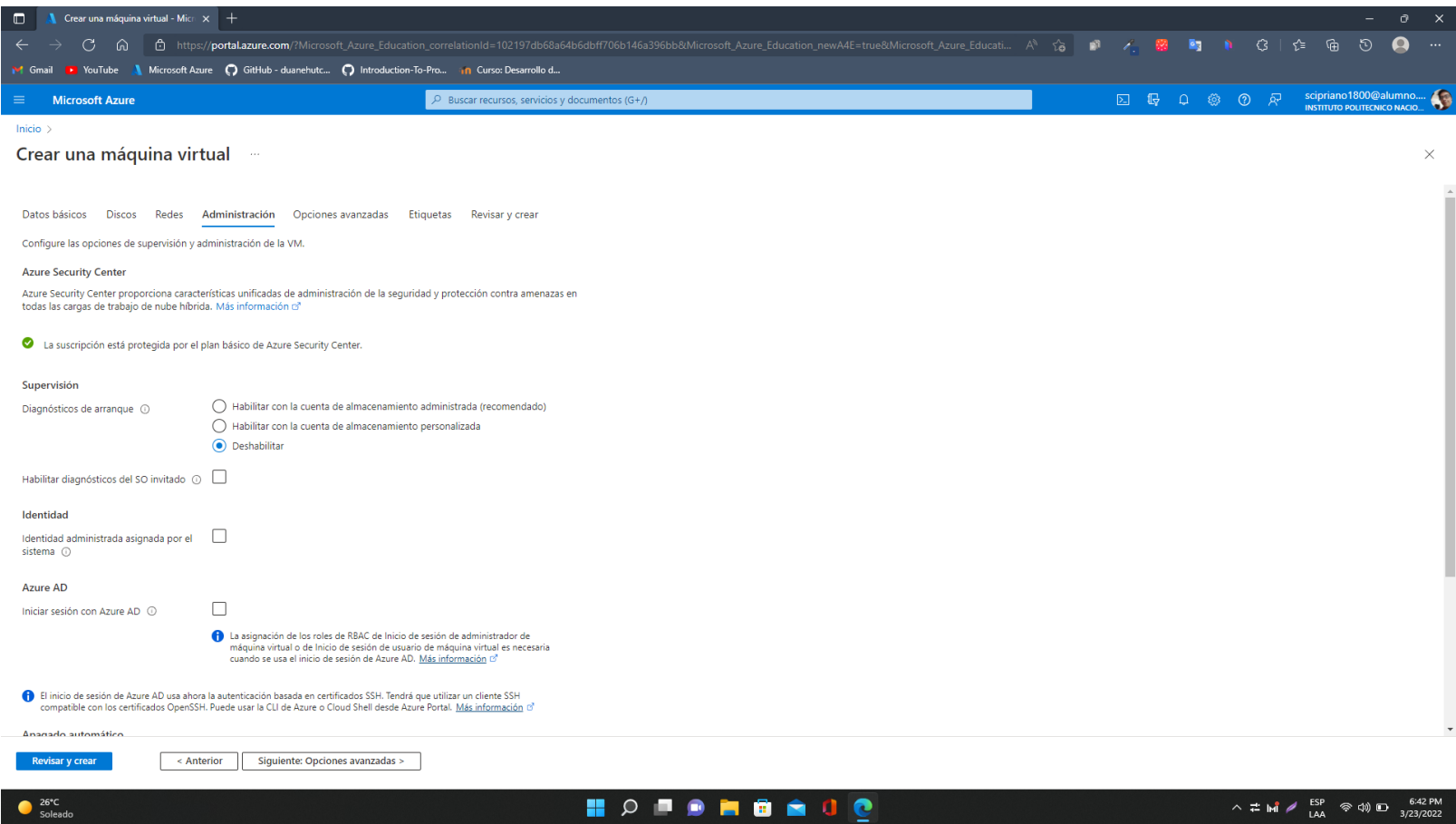


ILUSTRACIÓN 4 CONFIGURACIÓN DE LA SECCIÓN DE ADMINISTRACIÓN DE LAS MÁQUINAS VIRTUALES.

Ya finalmente en la parte de la revisión y la creación de la máquina virtual basta con que verifiquemos los datos y desde luego también hay que verificar la tarifa que nos están cobrando por el uso de dicha máquina virtual de esta manera podemos optimizar algunos de los recursos con los que contamos en nuestra cuenta institucional. Podemos ver también los datos del usuario que ha creado dicha máquina virtual basta con que comencemos la creación de cada una de las máquinas virtuales que necesitamos. Lo descrito anteriormente aparece en la imagen número 5.

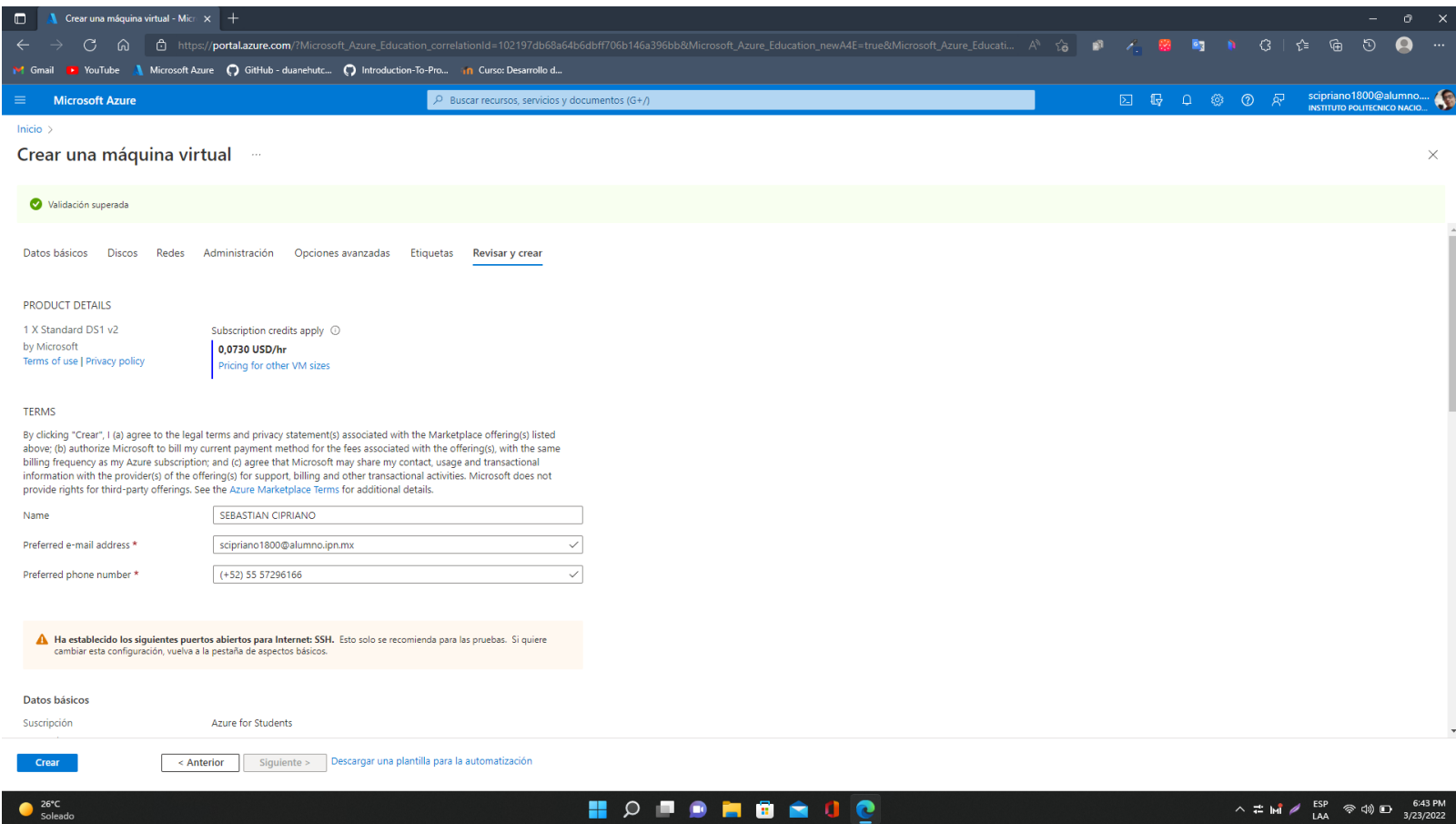


ILUSTRACIÓN 5 REVISIÓN Y CREACIÓN DE LA MÁQUINA VIRTUAL

En la siguiente captura podemos ver cómo la implementación de la máquina virtual está en curso, simplemente es necesario que esperemos a que el proceso termine de implementarse, podemos ver desde luego el estado de la máquina virtual en las opciones que nos da en donde nos indica que ya ha sido creada pero más sin en cambio la implementación tomará un tiempo. Esto lo podemos ver en la siguiente página en donde tenemos precisamente lo descrito en la imagen número 6.

Microsoft Azure portal interface showing the implementation status of a virtual machine. The page title is "CreateVm-canonical.0001-com-ubuntu-server-focal-2-20220323184018 | Información general". The status is "La implementación está en curso". A table shows the resource "Tarea520nsg565" with type "Microsoft.Network/networkSecurityGroups" and state "Created". A notification bubble says "Implementación en curso... Se está realizando la implementación en el grupo de recursos 'Tarea-5-2'."

Recurso	Tipo	Estado	Detalles de la operación
Tarea520nsg565	Microsoft.Network/networkSecurityGroups	Created	Detalles de la operación

ILUSTRACIÓN 6 PROCESO DE CREACIÓN Y REVISIÓN LLEVÁNDOSE A CABO.

Una vez que el proceso haya sido concluido por el sistema nos aparecerá la siguiente pantalla, en donde se nos indicará que el proceso ya sea concluido, del mismo modo podemos ver algunas opciones en donde podemos descargar los detalles de la implementación de nuestra máquina virtual sólo en caso de que queramos echarles un vistazo, se puede apreciar también que aparece una opción donde nos indicarán los pasos siguientes en donde también podemos presionar el botón para poder ir a los recursos de dicha máquina. Esto puede verse en la siguiente página, para ser más específico en la página 7.

The screenshot displays the Microsoft Azure portal interface. The main heading is "CreateVm-canonical.0001-com-ubuntu-server-focal-2-20220323184018 | Información general". A green checkmark icon indicates "Se completó la implementación". Below this, details are provided: "Nombre de implementación: CreateVm-canonical.0001-com-ubunt...", "Suscripción: Azure for Students", and "Grupo de recursos: Tarea-5-2". The "Pasos siguientes" section lists recommended actions: "Configurar el apagado automático", "Supervisar el estado, el rendimiento y las dependencias de red de la máquina virtual", and "Ejecutar un script dentro de la máquina virtual". A right-hand sidebar contains links for "Cost Management", "Microsoft Defender for Cloud", and "Tutoriales gratuitos de Microsoft". A top-right notification states "Implementación correcta". The bottom of the image shows a Windows taskbar with the date "3/23/2022" and time "6:45 PM".

ILUSTRACIÓN 7 PROCESO CONCLUIDO DE LA MÁQUINA VIRTUAL

Si vemos la información de nuestra máquina virtual podemos ver el grupo de recursos al que pertenece como podemos ver también el estado el cual en este momento está en ejecución y por supuesto podemos observar algunos datos fundamentales como lo son el tipo de red con la que opera y también los recursos con los que se dispone para dicha máquina virtual como lo es el tamaño de la máquina virtual, esto es de vital importancia puesto que entender estas características nos dará la pauta para poder saber qué tipo de implementaciones podemos realizar a posteriori, más sin embargo para el desarrollo de esta práctica realmente no se requieren grandes características con grandes prestaciones en la máquina virtual, lo cual a decir verdad es bastante afortunado debido a la optimización de costes. Los datos anteriormente mencionados pueden ser vistos en la imagen que sigue en la otra página.

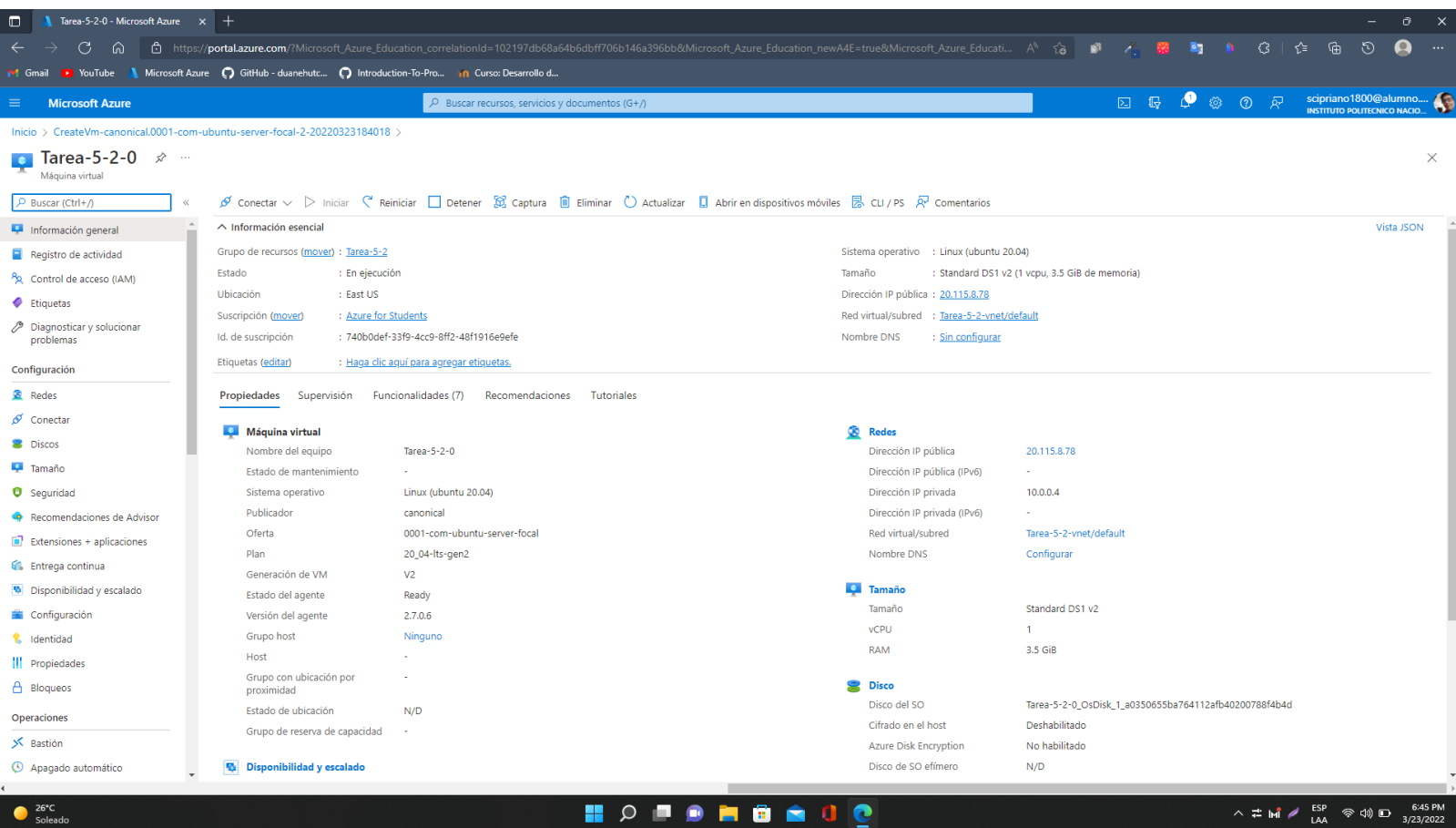


ILUSTRACIÓN 8 ELEMENTOS DE LAS MÁQUINAS VIRTUALES.

Es posible realizar la conexión a cualquiera de nuestras máquinas virtuales, a continuación, podemos ver cómo se tiene la consola perteneciente a lo que es el nodo número cero comas y en el encabezado se puede apreciar también el grupo de recursos al que pertenece. Esta consola y en general todas las que vayamos creando son de vital importancia puesto que nos permitirán instalar algunos recursos y también poder mostrar algunos de los resultados.

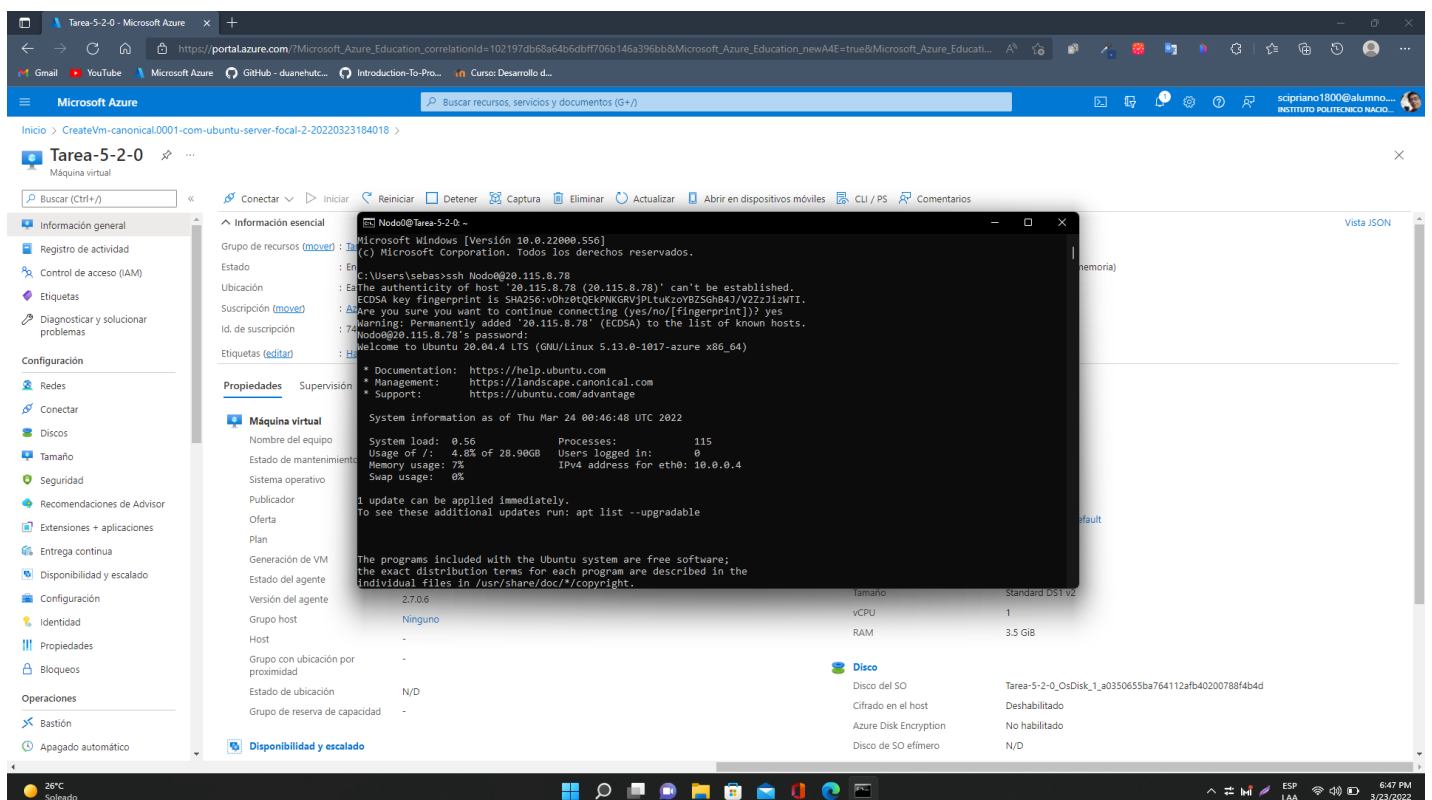


ILUSTRACIÓN 9 MUESTRA PRINCIPAL DE LA CONSOLA PARA CONEXIÓN INICIAL

Una vez explicado lo anterior hemos de instalar los recursos necesarios para la práctica de entre los cuales se destaca el jre que es vital para poder ejecutar nuestros programas. Esto lo podemos ver en la siguiente imagen donde vemos todo el proceso completo en la ilustración número 10.

The screenshot displays the Microsoft Azure portal interface. The main window shows the configuration for a virtual machine named "Tarea-5-2-0". The "Máquina virtual" tab is active, showing details such as the operating system (Ubuntu Server), size (Standard D1s v2), and disks. A terminal window is open, showing the command `sudo apt install default-jre` and its output, which lists various additional packages to be installed along with their sizes and dependencies.

Terminal Output:

```
Nodo0@Tarea-5-2-0:~$ sudo apt install default-jre
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  at-spi2-core ca-certificates-java default-jre-headless fontconfig-config fonts-dejavu-core fonts-dejavu-extra
  java-common libatk-bridge2.0-0 libatk-wrapper-java libatk-wrapper-java-jni libatk1.0-0 libatk1.0-data libatspi2.0-0
  libavahi-client3 libavahi-common-data libavahi-common3 libcups2 libdrm-amdgpu1 libdrm-intel1 libdrm-nouveau2
  libdrm-radeon1 libfontconfig1 libfontenc1 libgl1 libgl1-mesa-dri libglapi-mesa libglvnd0 libglx-mesa0
  libglx0 libgraphite2-3 libharfbuzz0b libice6 libjpeg-turbo8 libjpeg8 liblcms2-2 liblvm2 libpciaccess0 libpccsclite1
  libpsensors5 libpsensors-config libpsensors5 libsm6 libvulkan1 libwayland-client0 libx11-xcb1 libxaw7 libxcb-dri2-0 libxcb-dri3-0
  libxcb-glx0 libxcb-present0 libxcb-randr0 libxcb-shape0 libxcb-shm0 libxcb-sync1 libxcb-xfixes0 libxcomposite1
  libxfixes3 libxft2 libx11-xcb1 libxinerama1 libxkbfile1 libxmu6 libxpm4 libxrandr2 libxrender1 libxshmfence1 libxt6
  libxtst6 libxv1 libxxf86dga1 libxxf86vm1 mesa-vulkan-drivers openjdk-11-jre openjdk-11-jre-headless x11-common
  x11-utils
Suggested packages:
  cups-common liblcms2-utils pccscd lm-sensors libnss-mdns fonts-ipafont-gothic fonts-ipafont-mincho fonts-wqy-microhei
  fonts-wqy-zenhei fonts-indic mesa-utils
The following NEW packages will be installed:
  at-spi2-core ca-certificates-java default-jre default-jre-headless fontconfig-config fonts-dejavu-core
  fonts-dejavu-extra java-common libatk-bridge2.0-0 libatk-wrapper-java libatk-wrapper-java-jni libatk1.0-0
  libatk1.0-data libatspi2.0-0 libavahi-client3 libavahi-common-data libavahi-common3 libcups2 libdrm-amdgpu1
  libdrm-intel1 libdrm-nouveau2 libdrm-radeon1 libfontconfig1 libfontenc1 libgl1 libgl1-mesa-dri libglapi-mesa
  libglvnd0 libglx-mesa0 libglx0 libgraphite2-3 libharfbuzz0b libice6 libjpeg-turbo8 libjpeg8 liblcms2-2 liblvm2
  libpciaccess0 libpccsclite1 libpsensors-config libpsensors5 libsm6 libvulkan1 libwayland-client0 libx11-xcb1 libxaw7
  libxcb-dri2-0 libxcb-dri3-0 libxcb-glx0 libxcb-present0 libxcb-randr0 libxcb-shape0 libxcb-shm0 libxcb-sync1
  libxcb-xfixes0 libxcomposite1 libxfixes3 libxft2 libx11-xcb1 libxinerama1 libxkbfile1 libxmu6 libxpm4 libxrandr2
  libxrender1 libxshmfence1 libxt6 libxtst6 libxv1 libxxf86dga1 libxxf86vm1 mesa-vulkan-drivers openjdk-11-jre
  openjdk-11-jre-headless x11-common x11-utils
```

ILUSTRACIÓN 10 INSTALACIÓN DEL JRE DENTRO DE LA TERMINAL

Dentro del mismo también tenemos que hacer la instalación del JDK que también nos será de vital importancia para poder realizar el desarrollo de la aplicación implementación que se solicita en esta práctica el lenguaje de programación JAVA. Esto que hemos descrito lo podemos ver en la imagen número 11 en donde se presenta todo el proceso completo en la terminal.

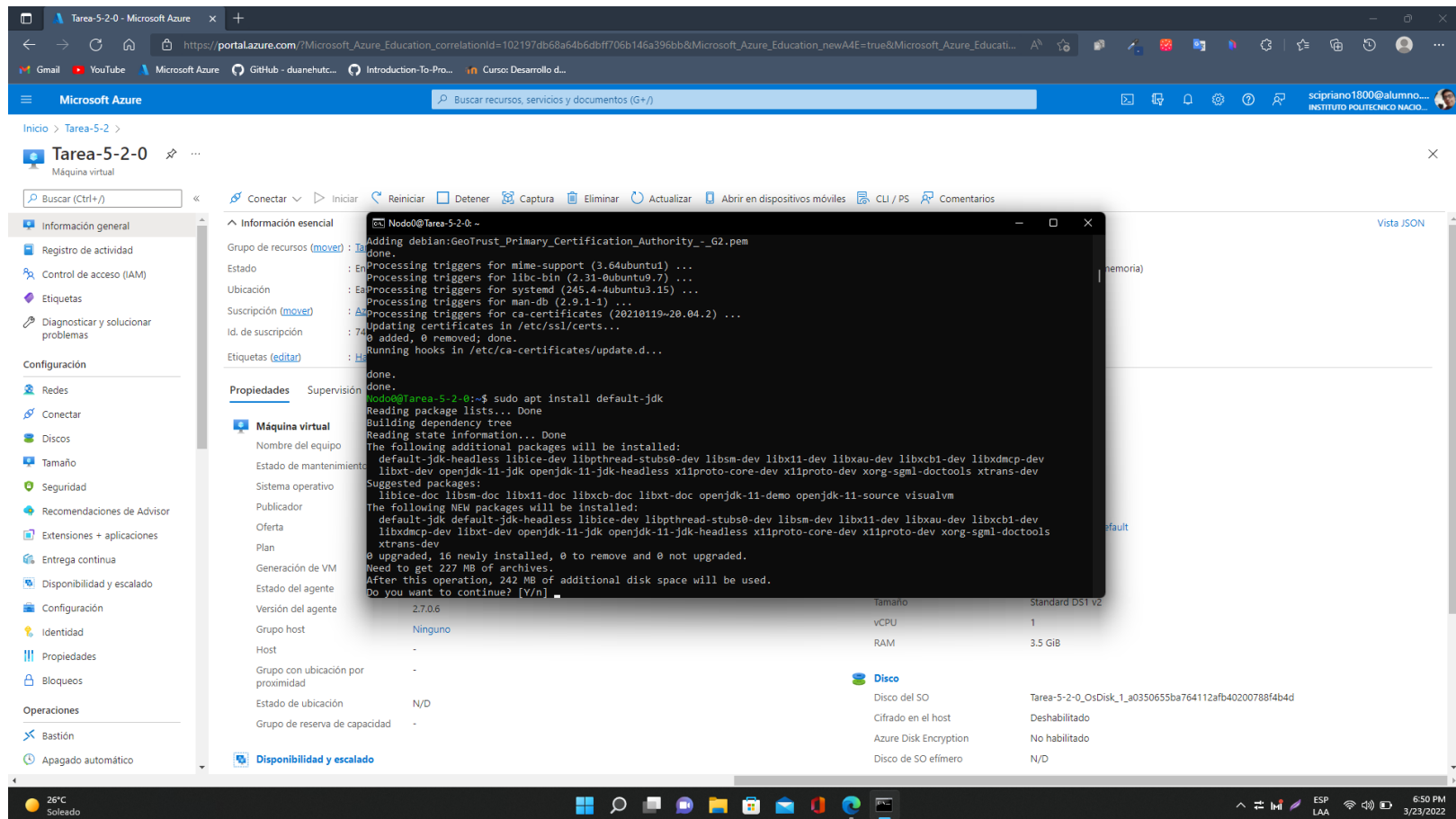


ILUSTRACIÓN 11 INSTALACIÓN DEL JDK DENTRO DE LA TERMINAL

En la siguiente captura que aparece en la parte de abajo podemos ver todo el listado de los recursos o mejor dicho todo el listado de las máquinas virtuales que han sido creadas con el fin o propósito de realizar esta práctica número 5, vemos que tenemos numeradas en orden ascendente cada una de las máquinas virtuales creadas de la cero hasta la máquina virtual número 5, de acuerdo con los requerimientos de la práctica algunas de éstas cumplirán el rol de servidores y tendremos una de ellas que cumplirá el rol de cliente. El listado mencionado con anterioridad lo podemos ver en la siguiente imagen número como la “Ilustración 12”.

Microsoft Azure portal showing a list of virtual machines (Máquinas virtuales) under the 'Inicio' tab. The table displays columns for Name, Type, Subscription, Resource Group, Location, State, Operating System, Size, Public IP Address, and Disks. The list shows five virtual machines, all in 'En ejecución' (Running) state, with various sizes and public IP addresses.

Nombre	Tipo	Suscripción	Grupo de recursos	Ubicación	Estado	Sistema operativo	Tamaño	Dirección IP pública	Discos
Tarea-5-2-0	Máquina virtual	Azure for Students	Tarea-5-2	East US	En ejecución	Linux	Standard_DS1_v2	20.115.8.78	1
Tarea-5-2-1	Máquina virtual	Azure for Students	Tarea-5-2	East US	En ejecución	Linux	Standard_DS1_v2	104.45.194.106	1
Tarea-5-2-2	Máquina virtual	Azure for Students	Tarea-5-2	East US	En ejecución	Linux	Standard_DS1_v2	20.25.102.200	1
Tarea-5-2-3	Máquina virtual	Azure for Students	Tarea-5-2	East US	En ejecución	Linux	Standard_DS1_v2	52.188.121.125	1
Tarea-5-2-4	Máquina virtual	Azure for Students	Tarea-5-2	East US	En ejecución	Linux	Standard_B1ms	20.127.129.108	1

ILUSTRACIÓN 12 LISTADO DE MÁQUINAS VIRTUALES QUE HEMOS CREADO.

Si bien ahora podemos ver cómo se ha realizado la conexión con nuestro entorno local y nuestros entornos remotos, mejor dicho, se ha permitido el acceso algunos de los recursos y ahora también podemos acceder por medio de la consola para poder subir los archivos que son necesarios para la ejecución del programa, esto es desde luego obviamente muy importante porque necesitamos tener la capacidad de ejecutar los códigos necesarios para la implementación del RMI. Esto aparece en la imagen 13.

Microsoft Azure portal showing the details of a virtual machine named 'Tarea-5-2-0'. The 'Propiedades' (Properties) tab is selected, displaying information such as the resource group, location, subscription, and operating system. A terminal window is overlaid on the screen, showing the execution of commands to upload files to the virtual machine using 'sftp'.

```
C:\Windows\System32\cmd.exe - sftp Node0@20.115.8.78
Microsoft Windows [Versión 10.0.22000.556]
(c) Microsoft Corporation. Todos los derechos reservados.
C:\Users\sebas\Desktop>Tarea 5>sftp Node0@20.115.8.78
Node0@20.115.8.78's password:
sftp> Connected to 20.115.8.78.
sftp> put InterfaceMatricesRMI.java
Uploading InterfaceMatricesRMI.java to /home/Node0/InterfaceMatricesRMI.java
100% 217 2.5KB/s 00:00
sftp> put ClienteMatricesRMI.java
Uploading ClienteMatricesRMI.java to /home/Node0/ClienteMatricesRMI.java
100% 7645 91.5KB/s 00:00
sftp> put ClaseMatricesRMI.java
Uploading ClaseMatricesRMI.java to /home/Node0/ClaseMatricesRMI.java
100% 673 8.7KB/s 00:00
sftp> put ServidorMatricesRMI.java
Uploading ServidorMatricesRMI.java to /home/Node0/ServidorMatricesRMI.java
100% 1019 14.6KB/s 00:00
sftp>
```

ILUSTRACIÓN 13 PUESTA DE LOS ARCHIVOS A NUESTRAS MÁQUINAS VIRTUALES.

Hemos de hacer lo mismo con cada una de las máquinas virtuales que hemos creado a fin de poder disponer de los recursos en cada una de ellas. Podemos verlo en la imagen número 14.

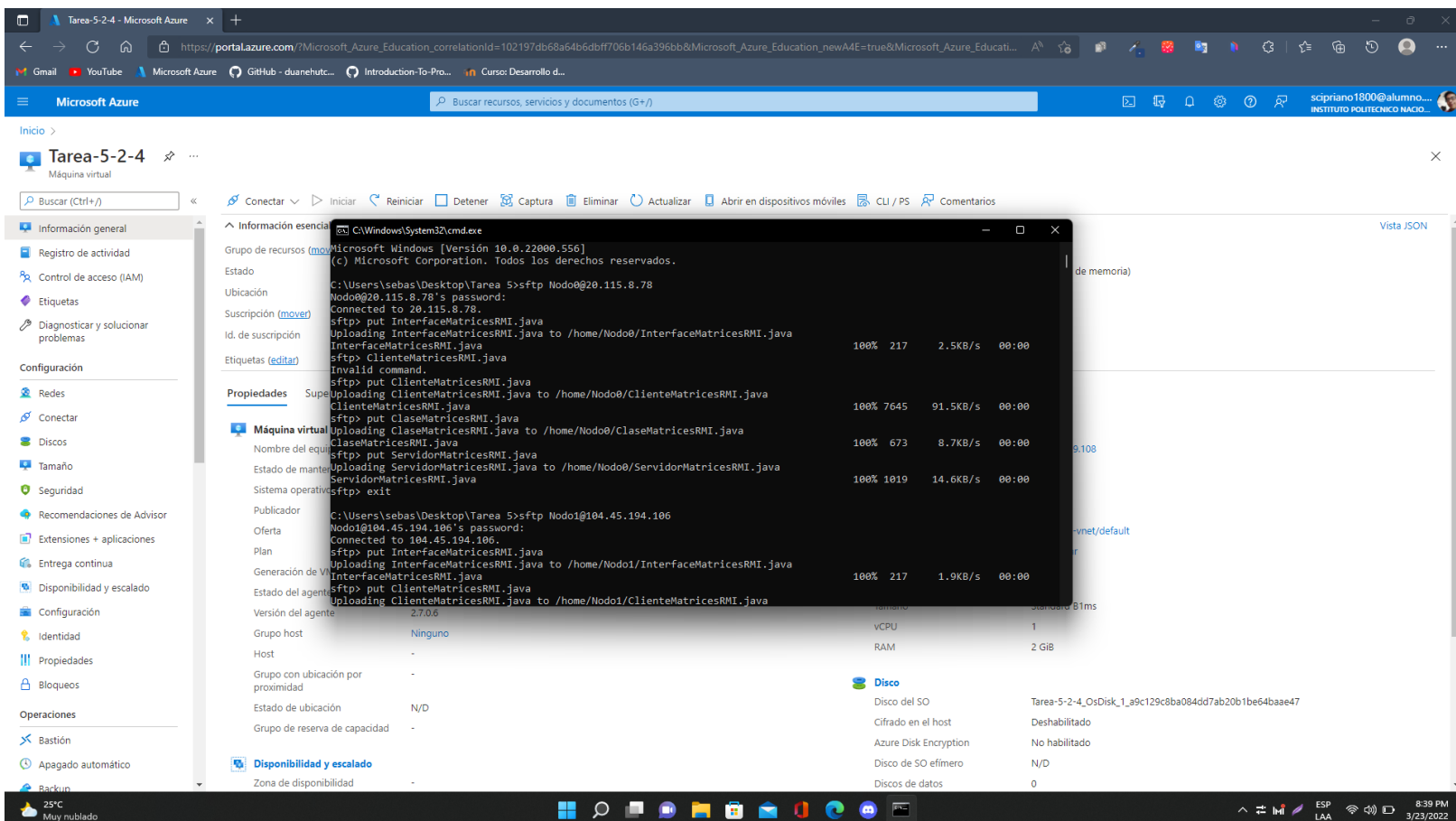


ILUSTRACIÓN 14 UPDATE DE LOS ARCHIVOS PARA EL PROGRAMA.

En la siguiente página estaremos mostrando muchas de las capturas con los resultados de nuestra práctica, por cuestiones del espacio en la pantalla se ha decidido mostrar las consolas de modo tal que se pueda ver en el centro a nuestro cliente y en la periferia se pueden ver a los servidores requeridos para la implementación y solución del problema.

Ya he mencionado lo anterior podemos ver cómo ahora tenemos nuestras consolas abiertas , cada una de estas consolas tiene en su interior el acceso a cada uno de los archivos que son necesarios para ejecutar el programa, se puede ver desde luego que tenemos el archivo para las clases de la matriz , tenemos el archivo para el cliente , tenemos el archivo de la interfaz y finalmente tenemos el archivo para el trabajo con el servidor de las matrices. Esto puede apreciarse en la imagen número 15.

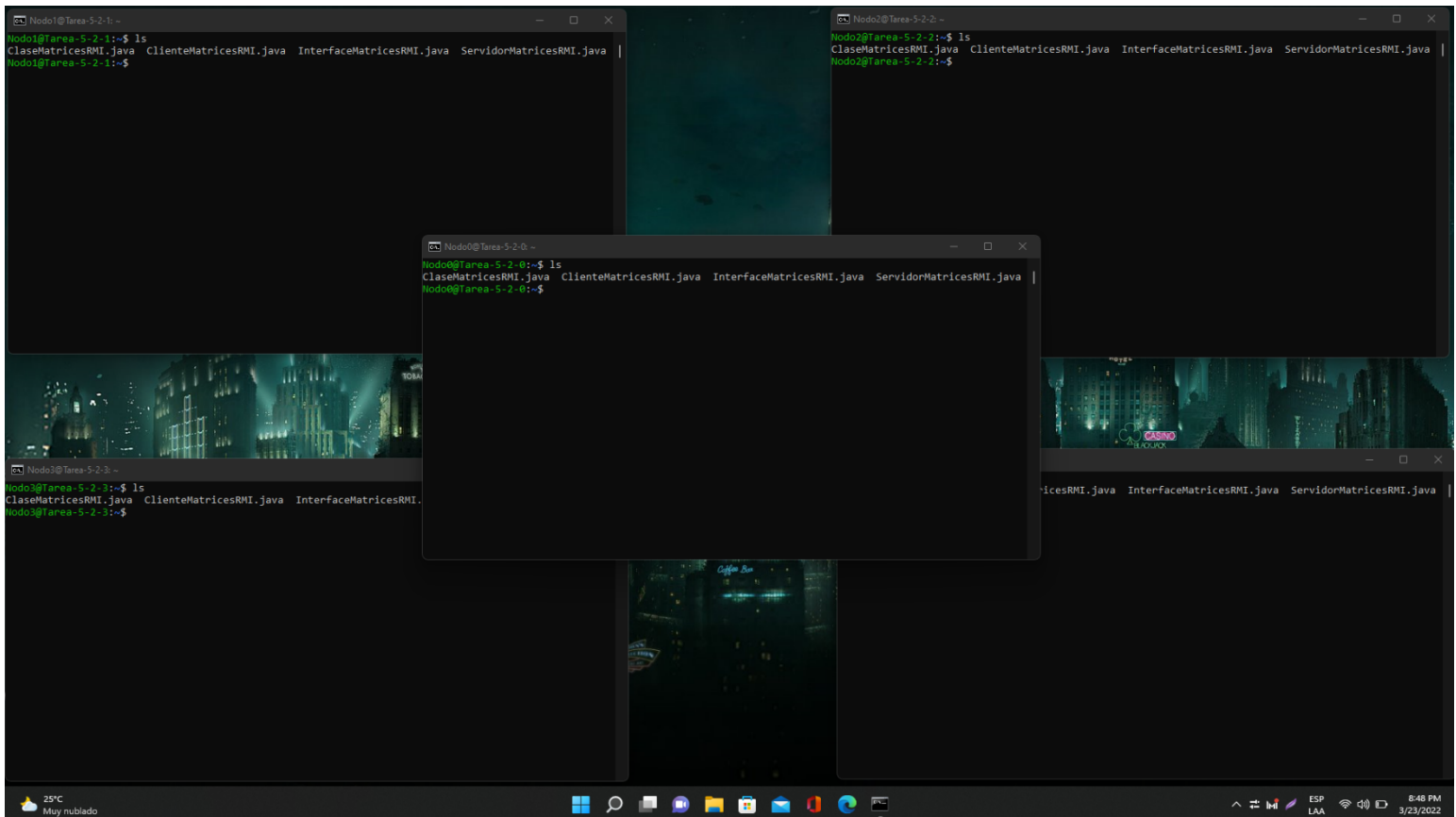


ILUSTRACIÓN 15 ARCHIVOS Y MUESTRAS DE LAS CONSOLAS DE LOS NODOS

Esta captura es necesario tenerla en cuenta porque estamos haciendo la compilación de todos los archivos que requerimos para lo cual usamos el comando “javac *.java” que tomará todos los archivos con extensión .java y procederá a hacer la compilación, lo único que falta es mostrar los resultados de dicha compilación para comprobar que ha sido totalmente exitosa para lo cual mostraremos qué es lo que ha resultado en nuestro directorio.

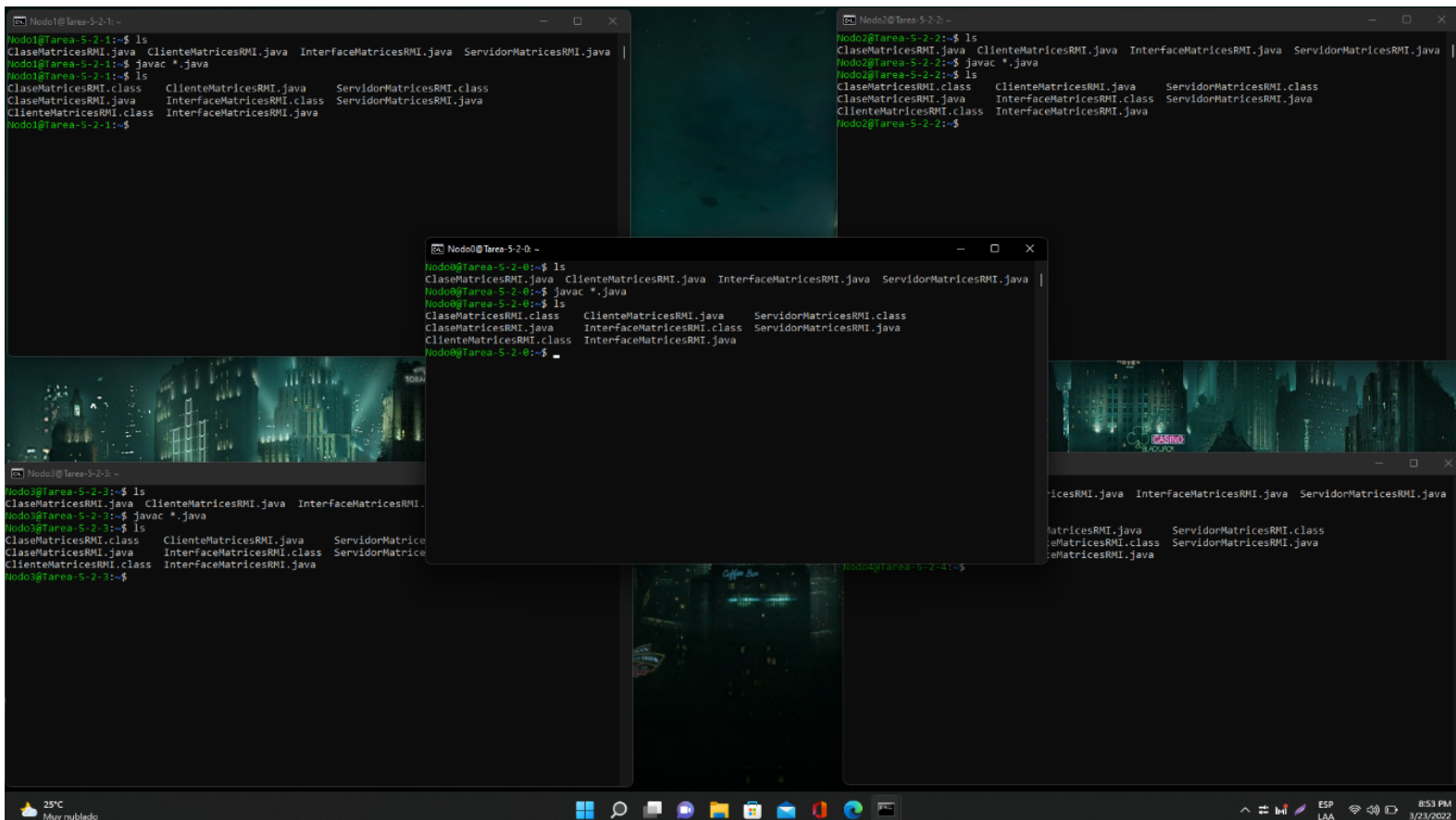


ILUSTRACIÓN 16 COMPILACIÓN DE LOS ARCHIVOS Y MUESTRA DE DIRECTORIO

Basta con ejecutar la instrucción `rmiregistry&` en donde podemos apreciar en cada una de las consolas los distintos resultados que se tienen para el nodo uno, el nodo 2, el nodo 3 y finalmente el nodo número cuatro; ahora bien, ¿qué es lo que pasa respecto al nodo número cero?, en líneas generales este debería estar a la espera de que en este caso los otros nodos estén por ponerlo en términos simples activos y a la espera, recordando el rol que cada uno de este cumple. Lo descrito lo podemos ver en la página número 17.

The image shows five terminal windows, each representing a different node in a distributed system. The windows are titled 'Nodo1@Tarea-5-2-1', 'Nodo2@Tarea-5-2-2', 'Nodo3@Tarea-5-2-3', 'Nodo4@Tarea-5-2-4', and 'Nodo0@Tarea-5-2-0'. Each window displays the following sequence of commands and output:

```

Nodo1@Tarea-5-2-1:~$ ls
ClaseMatricesRMI.java  ClienteMatricesRMI.java  InterfaceMatricesRMI.java  ServidorMatricesRMI.java
Nodo1@Tarea-5-2-1:~$ javac *.java
Nodo1@Tarea-5-2-1:~$ ls
ClaseMatricesRMI.class  ClienteMatricesRMI.class  InterfaceMatricesRMI.class  ServidorMatricesRMI.class
Nodo1@Tarea-5-2-1:~$ rmiregistry&
[1] 6220
Nodo1@Tarea-5-2-1:~$

Nodo2@Tarea-5-2-2:~$ ls
ClaseMatricesRMI.java  ClienteMatricesRMI.java  InterfaceMatricesRMI.java  ServidorMatricesRMI.java
Nodo2@Tarea-5-2-2:~$ javac *.java
Nodo2@Tarea-5-2-2:~$ ls
ClaseMatricesRMI.class  ClienteMatricesRMI.class  InterfaceMatricesRMI.class  ServidorMatricesRMI.class
Nodo2@Tarea-5-2-2:~$ rmiregistry&
[1] 5936
Nodo2@Tarea-5-2-2:~$

Nodo3@Tarea-5-2-3:~$ ls
ClaseMatricesRMI.java  ClienteMatricesRMI.java  InterfaceMatricesRMI.java  ServidorMatricesRMI.java
Nodo3@Tarea-5-2-3:~$ javac *.java
Nodo3@Tarea-5-2-3:~$ ls
ClaseMatricesRMI.class  ClienteMatricesRMI.class  InterfaceMatricesRMI.class  ServidorMatricesRMI.class
Nodo3@Tarea-5-2-3:~$ rmiregistry&
[1] 6013
Nodo3@Tarea-5-2-3:~$

Nodo4@Tarea-5-2-4:~$ rmiregistry&
[1] 6407
Nodo4@Tarea-5-2-4:~$

Nodo0@Tarea-5-2-0:~$
  
```

The background of the terminal windows features a cityscape at night. The Windows taskbar is visible at the bottom, showing the date and time as 8:58 PM on 3/23/2022.

ILUSTRACIÓN 17 COMANDO RMIREGISTRY& EN LAS CONSOLAS PERIFÉRICAS

Podemos ver como en cada una de las consolas basta con correr el archivo del servidor, en donde es necesario elegir desde luego el número del nodo del cual se trata lo cual en cada una de estas ejecutaremos el correspondiente como un parámetro que se le pasará por medio de la línea de comandos, de resultar exitoso este proceso debería aparecernos un mensaje que diga que se está iniciando el servidor correspondiente. Lo podemos apreciar en la imagen número 18.

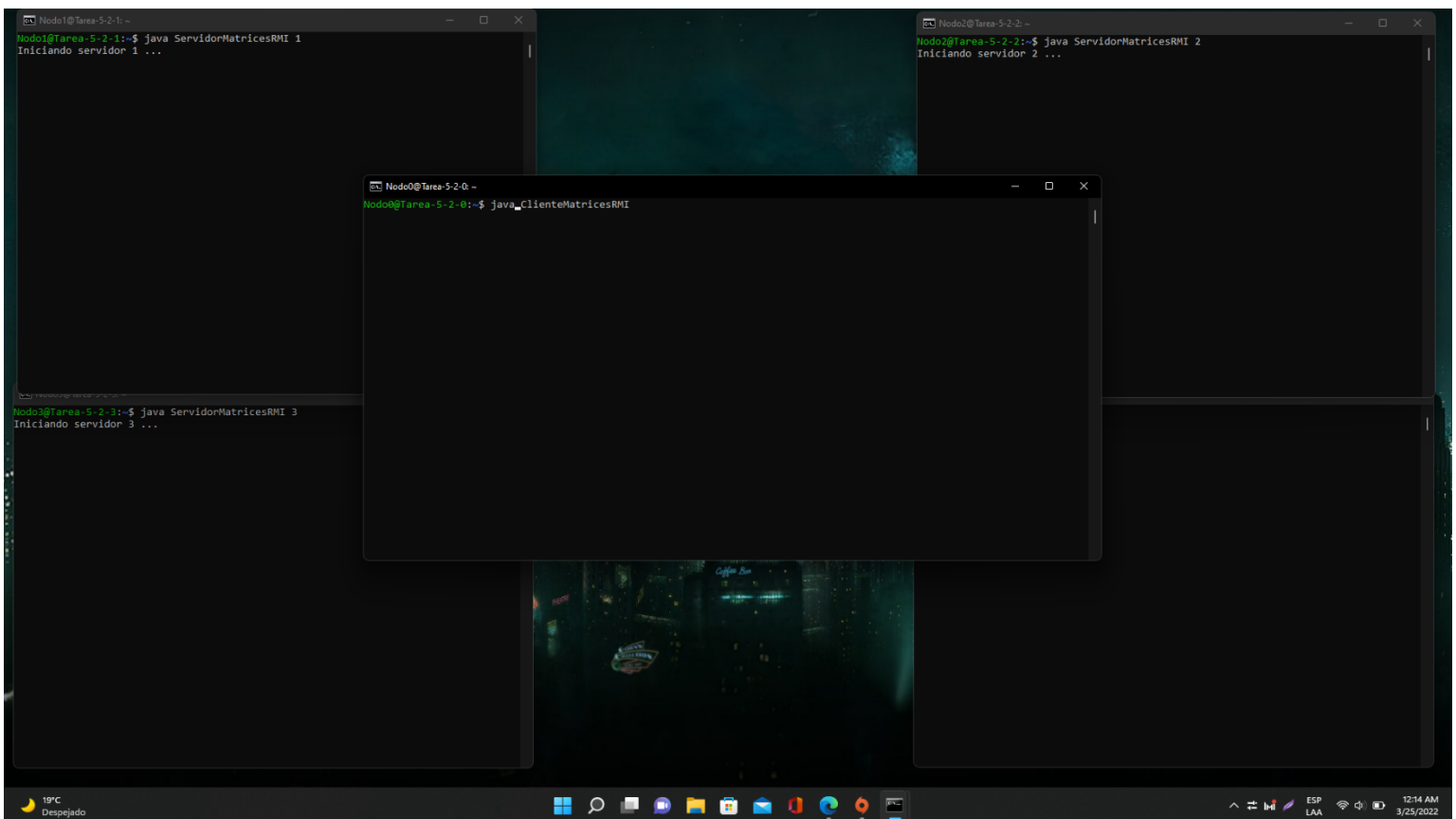


ILUSTRACIÓN 18 EJECUCIÓN DE LOS SERVIDORES

Si ejecutamos en nuestro nodo cero, o mejor dicho en el nodo central que pertenece al cliente recibiremos como retorno precisamente el resultado de nuestro checksum, el cual resulta bastante útil si queremos comprobar que efectivamente el procedimiento con las matrices se haya realizado de manera adecuada y según lo estipulado. La ejecución del primer caso lo podemos ver en la consola de la parte central de la siguiente imagen.

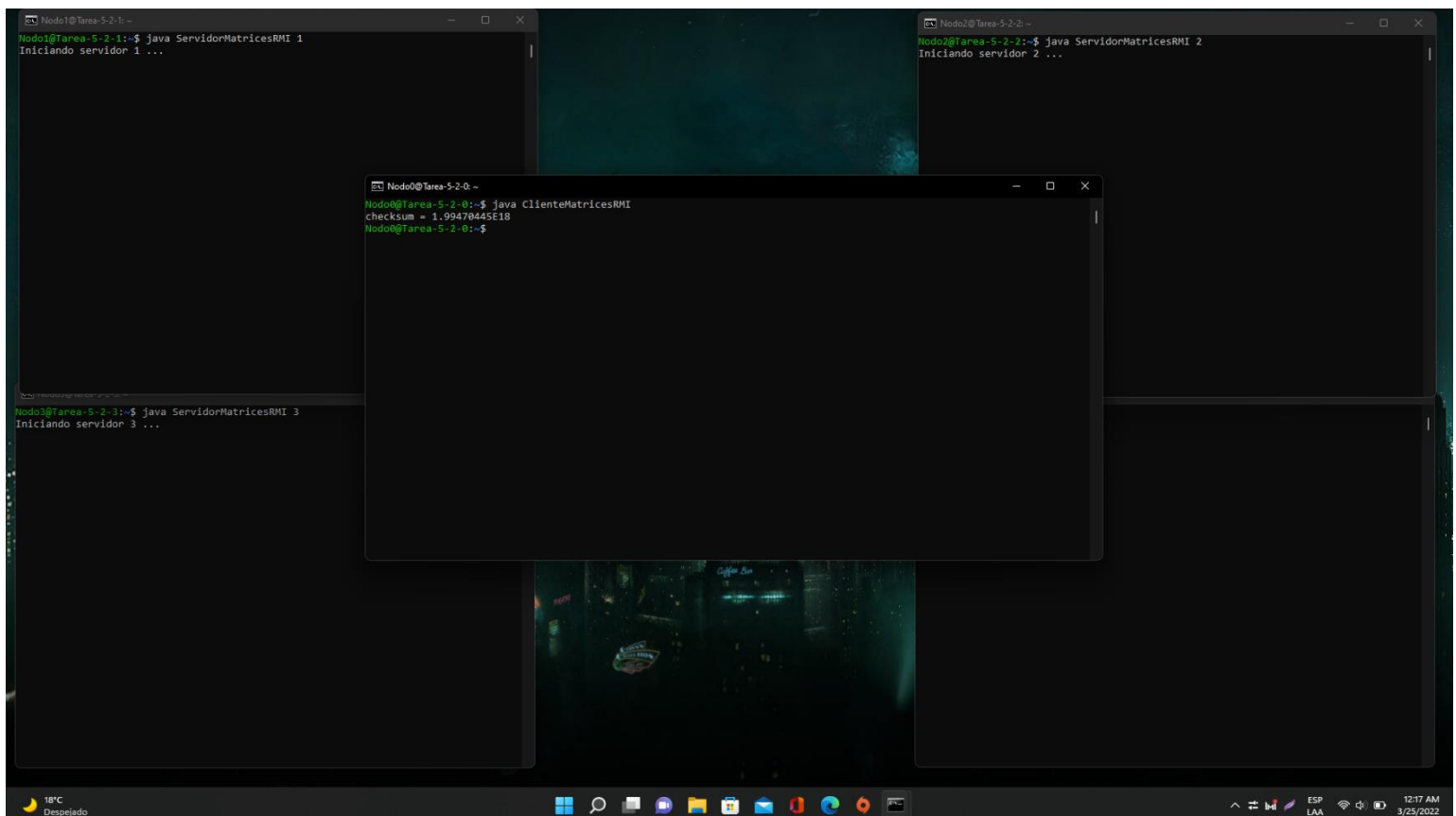
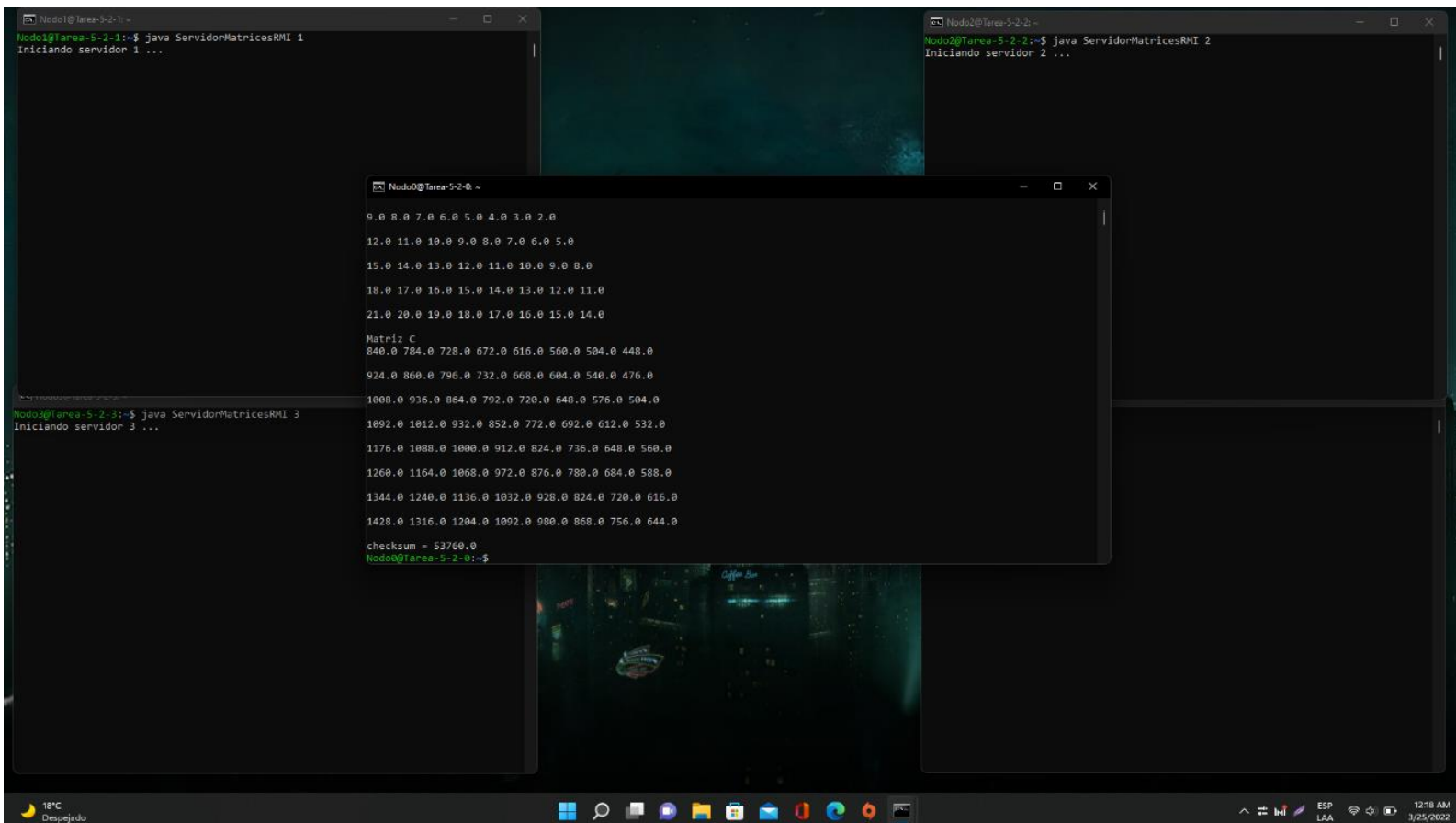


ILUSTRACIÓN 19 RESULTADO DEL CHECKSUM

Si bien es cierto el anterior fue un caso muy concreto, pero ahora en el siguiente caso podemos ver el despliegue de las matrices y en la parte inferior podemos apreciar el despliegue de nuestro checksum que del mismo modo nos ayudará a comprobar que efectivamente hayamos llegado al resultado correcto y que de manera interna todo el proceso de las operaciones matemáticas está bien implementado. Lo descrito se puede ver en la siguiente imagen con la impresión de la matriz y precisamente la parte del checksum.



The image shows a Windows desktop environment with three terminal windows open. The background is a dark-themed wallpaper featuring a space scene with a planet and stars. The taskbar at the bottom shows the Windows logo, search icon, and several application icons. The system tray on the right indicates the temperature is 18°C, the system is 'Despejado' (Clear), and the time is 12:18 AM on 3/25/2022.

The three terminal windows are as follows:

- Top Left Window:** Shows the command `java ServidorMatricesRMI 1` and the output `Iniciando servidor 1 ...`.
- Top Right Window:** Shows the command `java ServidorMatricesRMI 2` and the output `Iniciando servidor 2 ...`.
- Bottom Center Window:** Shows the command `java ServidorMatricesRMI 3` and the output `Iniciando servidor 3 ...`. Below this, it displays a 10x10 matrix of floating-point numbers, followed by the text `Checksum = 53760.0`.

```
Nodo1@Tarea-5-2-1:~$ java ServidorMatricesRMI 1
Iniciando servidor 1 ...

Nodo2@Tarea-5-2-2:~$ java ServidorMatricesRMI 2
Iniciando servidor 2 ...

Nodo0@Tarea-5-2-0:~$ java ServidorMatricesRMI 3
Iniciando servidor 3 ...

9.0 8.0 7.0 6.0 5.0 4.0 3.0 2.0
12.0 11.0 10.0 9.0 8.0 7.0 6.0 5.0
15.0 14.0 13.0 12.0 11.0 10.0 9.0 8.0
18.0 17.0 16.0 15.0 14.0 13.0 12.0 11.0
21.0 20.0 19.0 18.0 17.0 16.0 15.0 14.0

Matriz C
840.0 784.0 728.0 672.0 616.0 560.0 504.0 448.0
924.0 868.0 812.0 756.0 699.0 644.0 588.0 532.0
1008.0 936.0 864.0 792.0 720.0 648.0 576.0 504.0
1092.0 1012.0 932.0 852.0 772.0 692.0 612.0 532.0
1176.0 1088.0 1000.0 912.0 824.0 736.0 648.0 560.0
1260.0 1164.0 1068.0 972.0 876.0 780.0 684.0 588.0
1344.0 1240.0 1136.0 1032.0 928.0 824.0 720.0 616.0
1428.0 1316.0 1204.0 1092.0 980.0 868.0 756.0 644.0

Checksum = 53760.0
Nodo0@Tarea-5-2-0:~$
```

ILUSTRACIÓN 20 RESULTADO CHECKSUM E IMPRESIÓN DE LA MATRIZ

Conclusiones

Cazares Martínez Maximiliano

Trabajar con objetos distribuidos es una forma mucho más fácil que usar sockets, ya que no nos tenemos que preocupar por serializar los datos y recibirlos. Únicamente nos concentramos en crear métodos que hagan lo que requerimos hacer con dichos datos y así de esta forma cumplir con los requisitos solicitados.

Esta práctica fue relativamente fácil, ya que muchas de las funciones usadas ya las teníamos escritas desde la práctica 3 así que lo único que hicimos fue implementarlas con RMI. Sin embargo, la parte de los hilos se nos complicó un poco, puesto que la lógica inicial no funciona bien así que tuvimos que hacer unos ajustes para que la práctica fuera totalmente funcional.

Chavarría Vázquez Luis Enrique

Para esta práctica puedo concluir que si bien fue un tanto complejo entender algunos de los conceptos que fueron necesarios para abordar los requerimientos solicitados, al final se simplificó un poco debido a que muchos de los aspectos que ya habíamos tratado se parecían bastante a los de la práctica número 3 con lo cual si bien esto no simplificó del todas las cosas sí me permitió tener una mejor idea de cómo poder aportar al equipo algunas de las soluciones que nos podrían llevar a los resultados esperados, si bien tengo que decir que uno de los aspectos que no me gustaron tanto de la práctica es que la aplicación fue un tanto abstracta en el sentido de qué realmente yo no soy muy fan de algunos problemas que abordan temas relacionados a cuestiones matemáticas para entender o ejemplificar algunos aspectos que podrían ser presentados de manera totalmente práctica en el sentido más cotidiano de la palabra; esta práctica difirió en el sentido anteriormente mencionado de la número cuatro que a decir verdad fue una de mis favoritas puesto que se trabajó directamente en un chat que permitía ver la ejecución del programa en un ambiente que resulta de utilidad para cualquiera de los casos o ideas que pudieran llegar de repente a plantearse sobre la mesa en el desarrollo o la solución de un problema al que pudiéramos enfrentarnos en un futuro.

Ahora bien un aspecto que quisiera destacar es el que tiene que ver directamente con los aspectos técnicos, en este ámbito sí que pude aprender mucho más referente a cómo utilizar el ya conocido como RMI, con lo cual ahora puedo ver un poco mejor cómo pudiera llevar este tipo de infraestructura a determinadas aplicaciones o desarrollos futuros, lo interesante de esta práctica es que teníamos toda una serie de servidores funcionando y esperando aquí un cliente central realizará una petición en concreto, lo cual si bien puede verse como un proceso sencillo en el papel al final sí que tiene su gracia hacer la implementación en código

y más específicamente en este lenguaje de programación que estamos usando en el curso, mencionó esto puesto que la mayor parte de las cosas que yo suelo trabajar las hago casi siempre en Python y si bien para mí trabajar en Java representa una curva de aprendizaje adicional y muchas veces a mi consideración hace tedioso el trabajo con algunos conceptos que son un tanto abstractos. Mas sin embargo en esta práctica en concreto logré ver con mucha claridad cómo es que teníamos que realizar el trabajo y cómo es que podemos proponer soluciones a mayor escala empleando precisamente este tópico. Esta práctica en general ayudó a trabajar con objetos distribuidos de maneras mucho más sencillas que como lo habíamos hecho de manera tradicional entonces se pudo evitar el envío de mensajes de modo tal que ya no era necesario realización de los datos y pues ya únicamente uno se iba enfocando en la estructura de los requerimientos, en general todos los aspectos que he mencionado anteriormente se quedan pequeños respecto a las grandes ventajas que implica trabajar con esta técnica ya que si bien al inicio fue un poco difícil ya una vez acostumbrándose lo hace mucho más sencillo.

Cipriano Damián Sebastián

La práctica que hemos desarrollado fue en realidad algo fácil en términos de que ahora el trabajo y la implementación fue mucho más compacta y de alguna manera también fue mucho más modular al momento de lidiar con los ficheros de nuestro código, en la anterior práctica habíamos estado trabajando por medio de la comunicación por mensajes lo cual si bien era algo sencillo de entender al final sí que complican poco las cosas tener que estar realizando los datos para poder recibirlos, en esta ocasión simplemente fue necesario que nos concentráramos en como cumplir con los requisitos solicitados y también seguir los lineamientos que establece esta técnica para el uso de objetos distribuidos.

Si bien la experiencia de las prácticas anteriores nos fue de utilidad, cabe destacar que manejar los conceptos que ya traíamos de la práctica número 3 sí que fue de ayuda para poder simplificar el proceso de implementación ya que había bastantes cosas que si bien eran diferentes al final del día compartían bastantes semejanzas con lo ya visto anteriormente.

Sin duda es una práctica en la que se han podido reafirmar algunos de los conceptos vistos en clase y en la que se han puesto a prueba nuestros conocimientos.