



Instituto Politécnico Nacional



Escuela Superior de Cómputo

Diseño de Sistemas Distribuidos

Tarea 2 - Equipo 2

Docente:

Dr. Pineda Guerrero Carlos

Alumnos:

*Cazares Martínez Maximiliano
Chavarría Vázquez Luis Enrique
Cipriano Damián Sebastián*

Grupo: 4CV11

CDMX, 1 de marzo de 2022

Índice

Descripción del problema	3
Desarrollo	4
Conclusiones	8
Cazares Martínez Maximiliano	8
Chavarría Vázquez Luis Enrique	8
Cipriano Damián Sebastián	8

Índice de imágenes

Ilustración 1 Diagrama de la práctica	3
Ilustración 2 Primera parte certificado	4
Ilustración 3 Segunda parte certificado	5
Ilustración 4 Tercera parte certificado	5
Ilustración 5 Compilación del proyecto	6
Ilustración 6 Ejecución del programa inicio	7
Ilustración 7 Ejecución del programa final	7

Descripción del problema

Desarrollar un programa en Java, el cual implementará un token (un número entero de 16 bits) que se enviará de un nodo a otro nodo mediante sockets seguros, en una topología lógica de anillo.

El anillo constará de seis nodos:

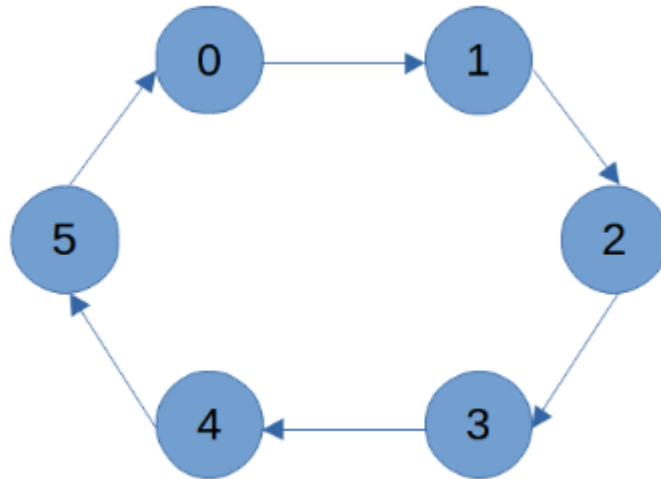


Ilustración 1 Diagrama de la práctica

- 1) Cada nodo será un cliente y un servidor
- 2) El cliente deberá implementar reintentos de conexión
- 3) Inicialmente el nodo 0 enviará el número 0 (token) al nodo 1
- 4) Cuando el nodo n reciba el token lo incrementará, lo desplegará y lo enviará al nodo: $(n + 1) \text{ módulo } 6$
- 5) Cuando en el nodo 0 el token sea mayor o igual a 500, el nodo 0 deberá terminar su ejecución.
- 6) El servidor en cada nodo deberá abrir un puerto diferente debido a que los seis nodos se ejecutan en la misma computadora.
- 7) Para ejecutar el programa en cada nodo se debe pasar como parámetros el número del nodo.
- 8) Se deberá probar el programa en una sola computadora utilizando seis ventanas de comandos de Windows o seis terminales de Linux o MacOS, cada ventana ejecutará una instancia del programa.

Desarrollo

Primero creamos el certificado auto firmado usando el programa **keytool** incluido en el JDK con el siguiente comando.

```
keytool -genkeypair -keyalg RSA -alias certificado_servidor -keystore  
keystore_servidor.jks -storepass 1234567
```

Nota: Las capturas que presentamos aquí fueron tomadas en la computadora de Chavarría Vázquez Luis Enrique, debido a que cuando creamos por primera vez nuestros certificados olvidamos tomar capturas, más sin embargo probando en otra computadora hemos repetido el proceso para ilustrarlo.

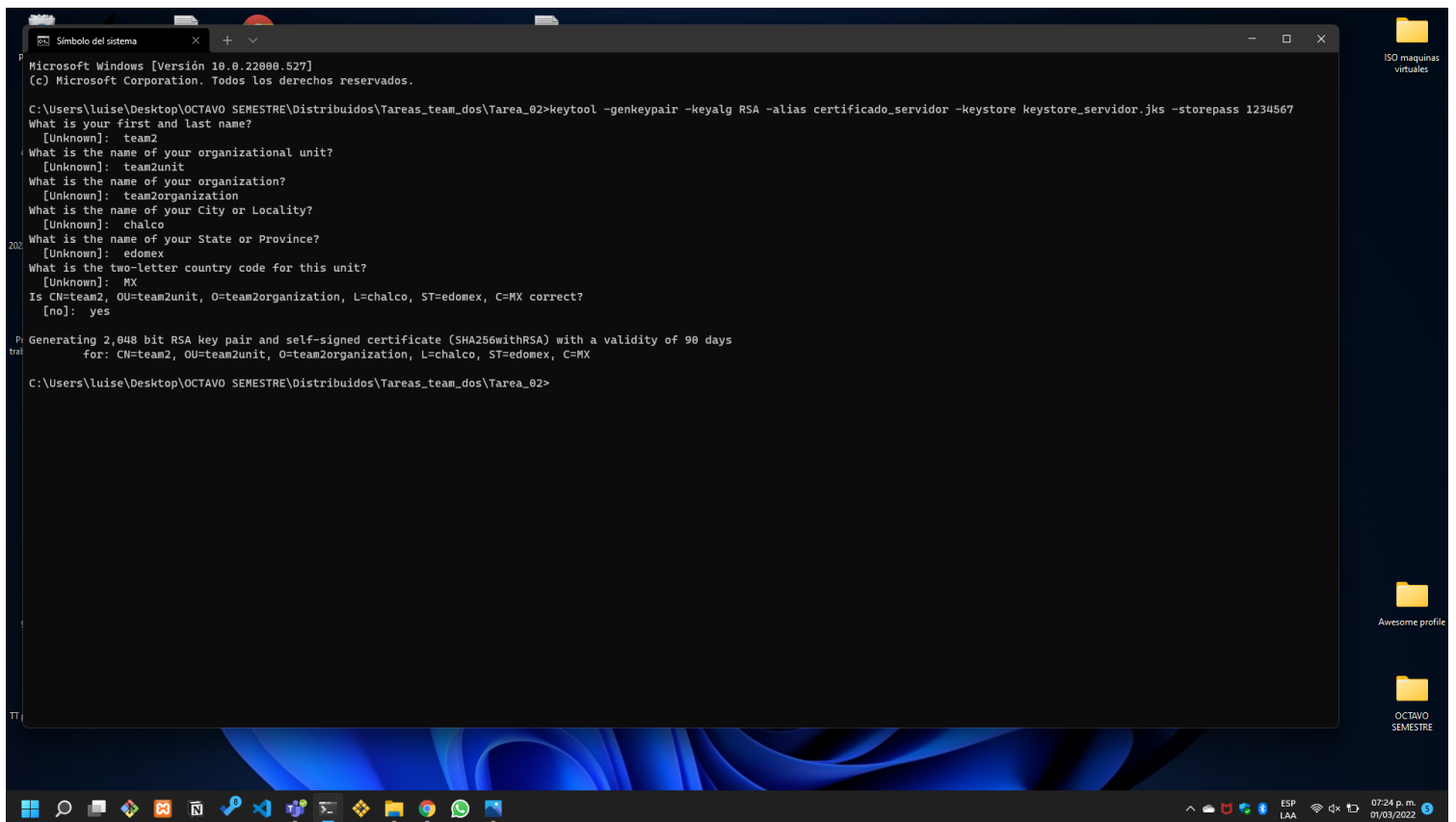


Ilustración 2 Primera parte certificado

Después, obtenemos el certificado contenido en el keystore.

```
keytool -exportcert -keystore keystore_servidor.jks -alias certificado_servidor -rfc -file  
certificado_servidor.pem -storepass 1234567
```

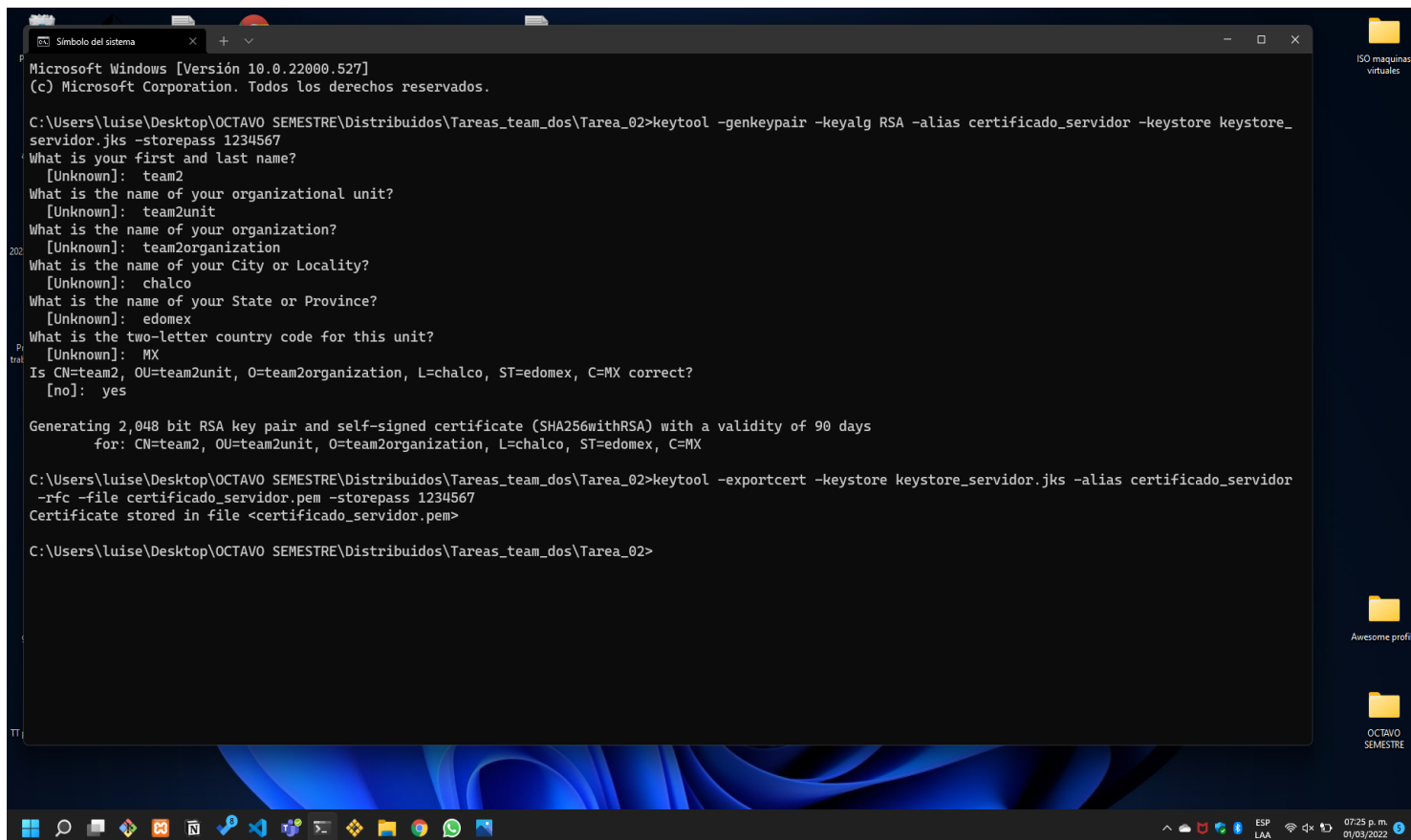


Ilustración 3 Segunda parte certificado

Finalmente, creamos un keystore que utilizará el cliente.

keytool -import -alias certificado_servidor -file certificado_servidor.pem -keystore keystore_cliente.jks -storepass 123456

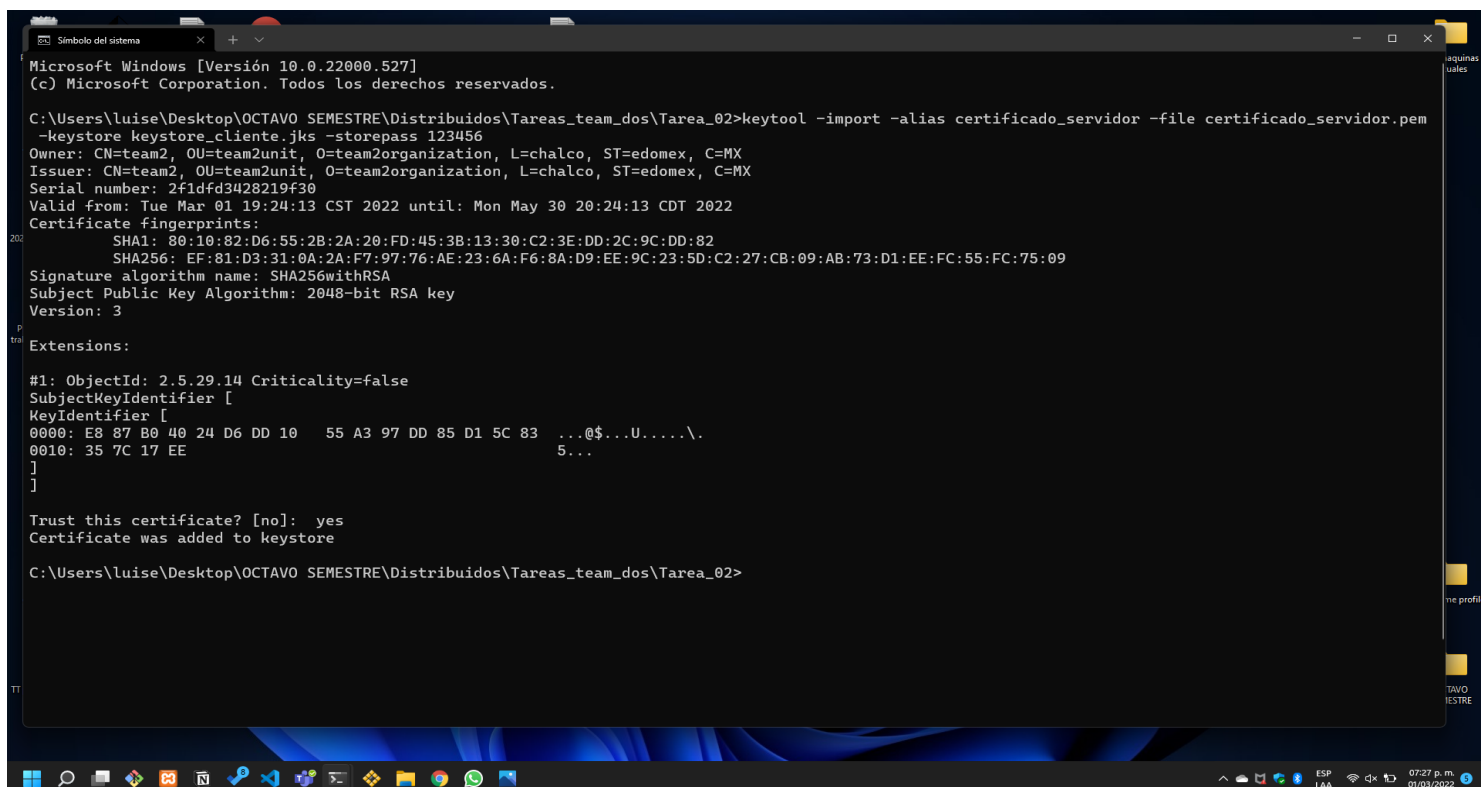
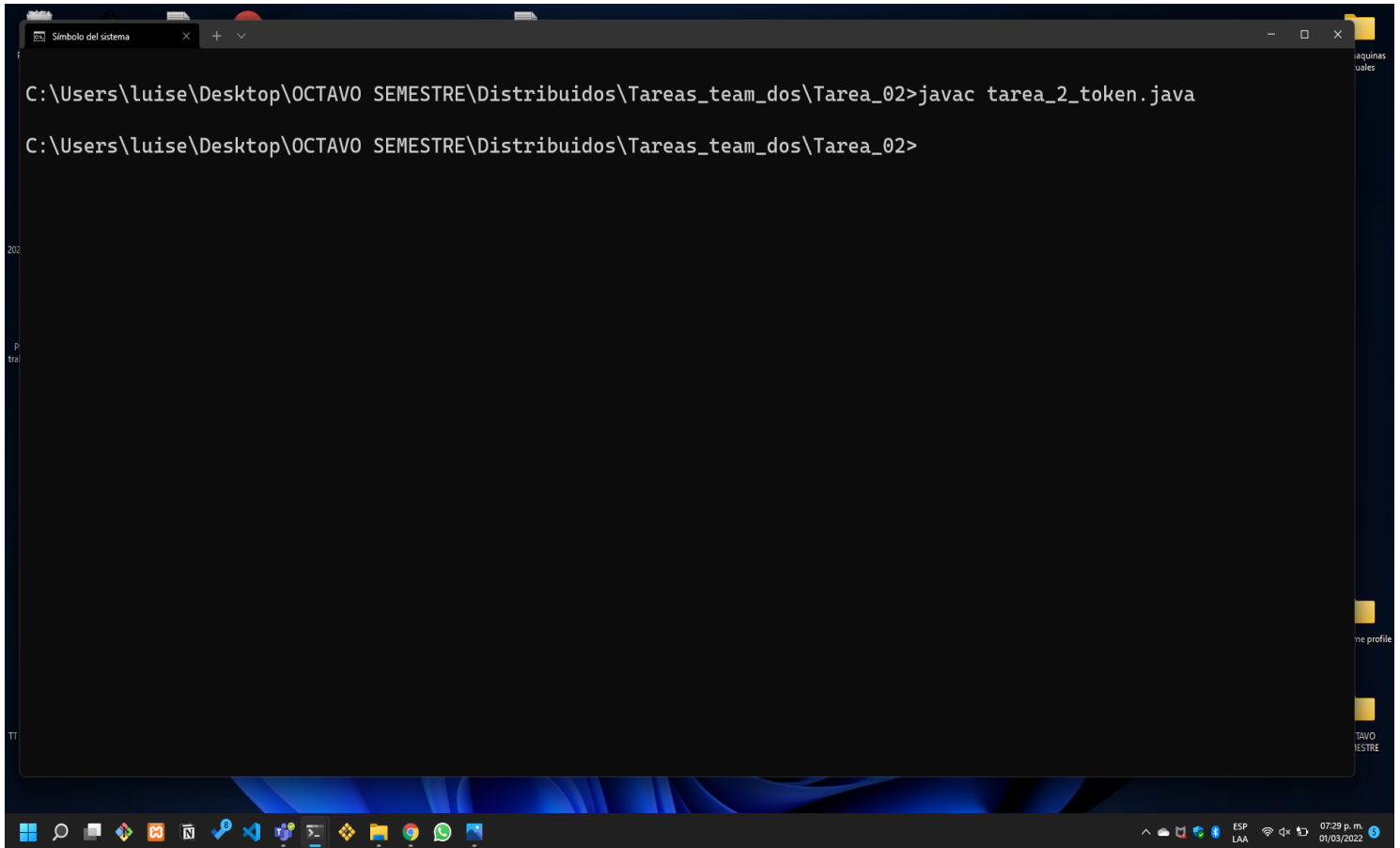


Ilustración 4 Tercera parte certificado

Posteriormente, procedemos a compilar el programa “*tarea_2_token.java*”.

A screenshot of a Windows command prompt window. The title bar at the top reads "Símbolo del sistema". The command prompt shows the current directory as "C:\Users\luise\Desktop\OCTAVO SEMESTRE\Distribuidos\Tareas_team_dos\Tarea_02" and the command "javac tarea_2_token.java" has been executed. The prompt is now waiting for the next command. The Windows taskbar is visible at the bottom with various application icons and a system tray on the right showing the date and time as 07:28 p.m. on 01/03/2022.

```
C:\Users\luise\Desktop\OCTAVO SEMESTRE\Distribuidos\Tareas_team_dos\Tarea_02>javac tarea_2_token.java
C:\Users\luise\Desktop\OCTAVO SEMESTRE\Distribuidos\Tareas_team_dos\Tarea_02>
```

Ilustración 5 Compilación del proyecto

Ahora, dentro de 6 “**consolas de Windows o Linux**” ejecutamos los nodos del 0 al 5.

Nota: Las siguientes capturas de la ejecución fueron tomadas en la computadora de Damián Sebastián Cipriano, como lo mencionamos las siguientes capturas son de otra computadora, porque olvidamos tomar capturas de la primera vez que hicimos los certificados, pero son exactamente los mismo que hicimos antes.

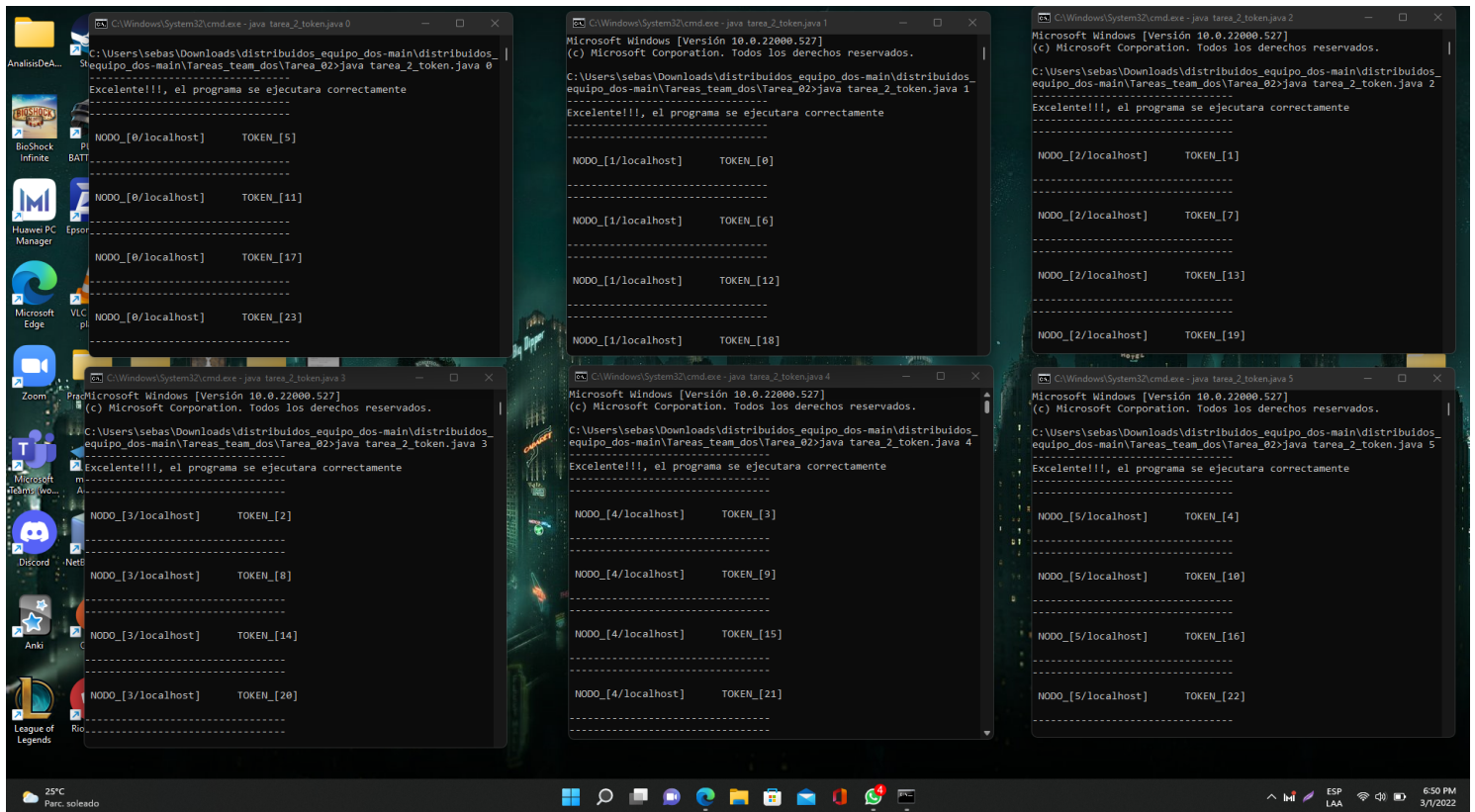


Ilustración 6 Ejecución del programa inicio

Finalmente, tenemos el resultado del token llegando al límite establecido en el nodo

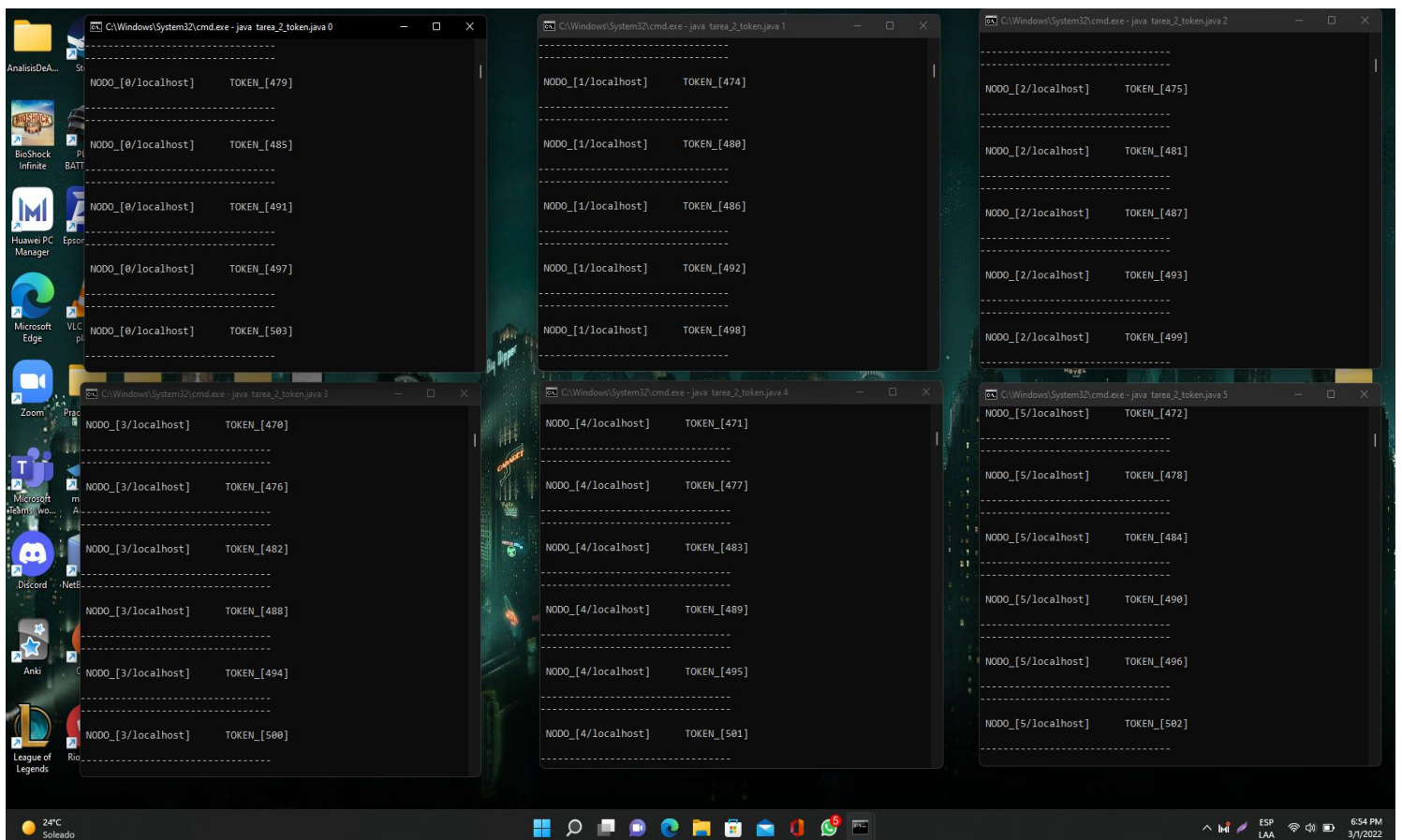


Ilustración 7 Ejecución del programa final

“#nodo”.

Conclusiones

Cazares Martínez Maximiliano

Una topología de anillo hace posible prescindir de un nodo central encargado de gestionar la conectividad de los demás nodos. De esta forma podemos desarrollar sistemas donde cada nodo se comporta como cliente y servidor para realizar una sola tarea como en el caso de esta práctica. Resulta interesante las posibilidades que se pueden plantear haciendo uso de una topología como está, donde el siguiente nodo continúa con la tarea del anterior y no siendo demasiado complejo como para implementarlo.

Chavarría Vázquez Luis Enrique

La práctica a decir verdad para mí fue un desafío porque se nos cruzaron varias actividades relacionadas con nuestro trabajo terminal, pero al final del día logramos conseguir el resultado, además de que pude investigar un poco más sobre el tema de la topología de anillo y la importancia del uso de nodos que de alguna manera centralizan la gestión de la conectividad. Además de que tiene el agregado de trabajar con sockets seguros, que si bien son firmados por nosotros mismos ya es un gran avance poder ver cómo implementarlos desde JAVA. Honestamente no me imagine que el proceso fuera así, casi siempre yo pagaba a servicios de terceros para que me pusieran mis certificados SSL, pero ya con esto puedo ver mucho mejor como implementarlo en programas propietarios.

Estoy entusiasmado porque empezamos a mezclar ya más la parte de Azure para poder poner a prueba nuestros programas y ver cómo podríamos nosotros mismos implementar nuestros servicios.

Cipriano Damián Sebastián

Esta práctica fue esencial para reforzar los temas vistos en clase, como lo son los sockets seguros y la creación de certificados autoafirmados. Además, el ejercicio realizado en esta práctica fue muy útil para comprender el uso y funcionamiento de la topología de anillo. Por último, se reforzó el concepto de cliente-servidor Multithreading, ya que en este ejercicio los nodos actúan como nodo cuando recibe el token y después actúa como servidor al enviar el token al siguiente nodo.