



# 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

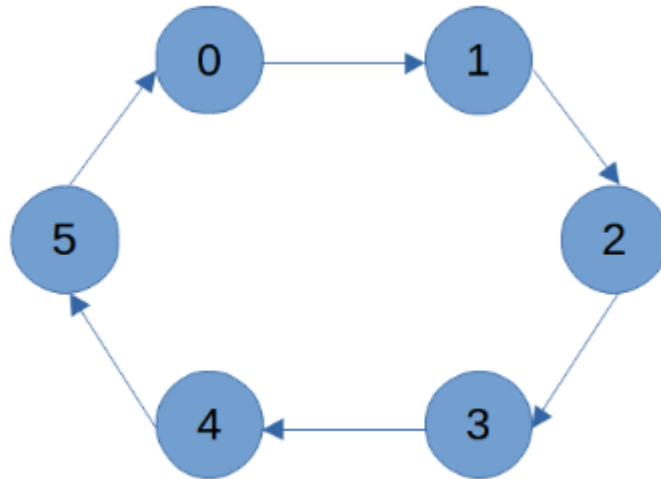
## Índice de imágenes

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:



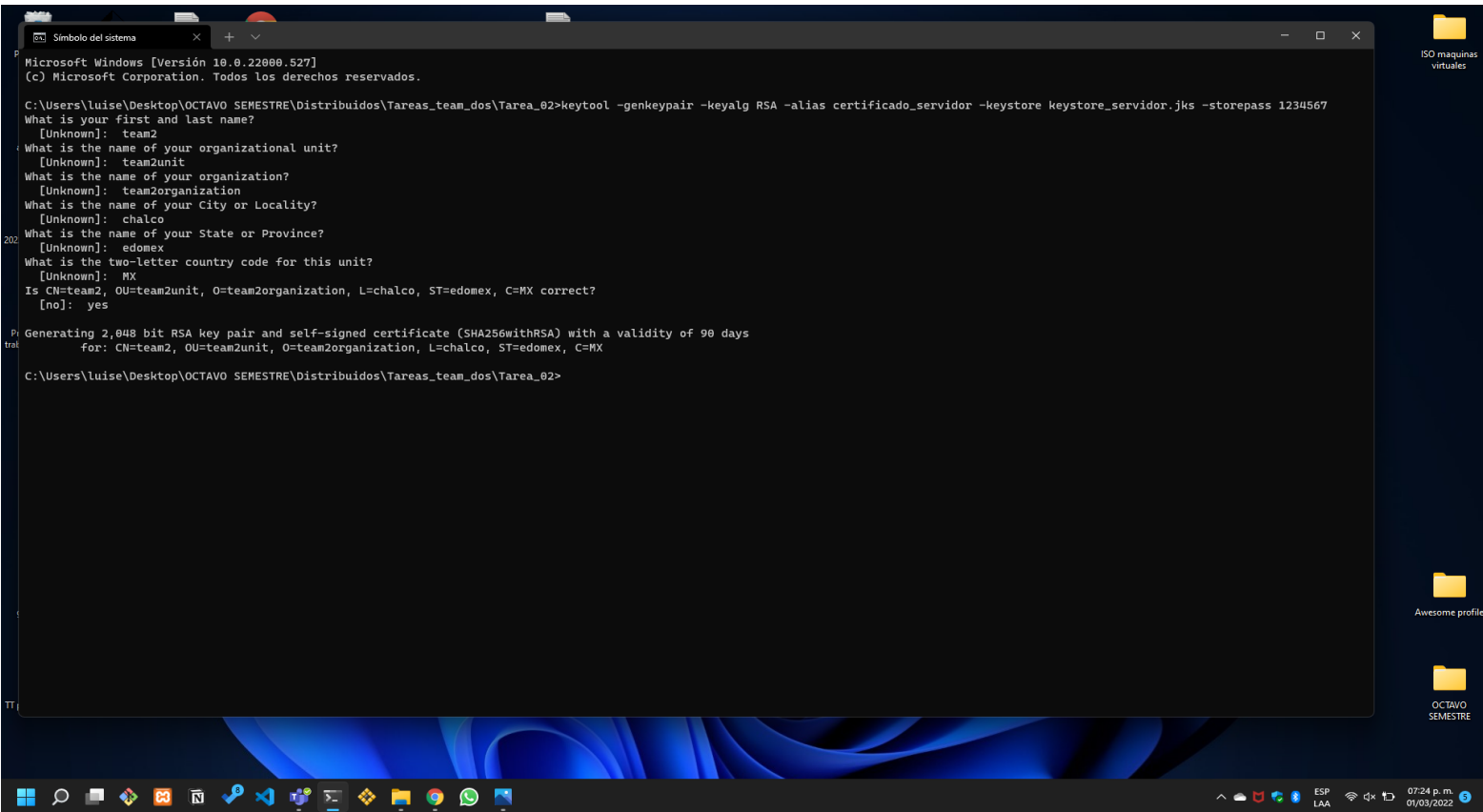
*Ilustración 1 Diagrama de la práctica*

- 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:  $(n + 1) \bmod 6$
- 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
```



*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
```

```
Símbolo del sistema
Microsoft Windows [Versión 10.0.22000.527]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\luise\Desktop\OCTAVO SEMESTRE\Distribuidos\Tareas_team_dos\Tarea_02>keytool -genkeypair -keyalg RSA -alias certificado_servidor -keystore keystore_servidor.jks -storepass 1234567
What is your first and last name?
[Unknown]: team2
What is the name of your organizational unit?
[Unknown]: team2unit
What is the name of your organization?
[Unknown]: team2organization
What is the name of your City or Locality?
[Unknown]: chalco
What is the name of your State or Province?
[Unknown]: edomex
What is the two-letter country code for this unit?
[Unknown]: MX
Is CN=team2, OU=team2unit, O=team2organization, L=chalco, ST=edomex, C=MX correct?
[no]: yes

Generating 2,048 bit RSA key pair and self-signed certificate (SHA256withRSA) with a validity of 90 days
for: CN=team2, OU=team2unit, O=team2organization, L=chalco, ST=edomex, C=MX

C:\Users\luise\Desktop\OCTAVO SEMESTRE\Distribuidos\Tareas_team_dos\Tarea_02>keytool -exportcert -keystore keystore_servidor.jks -alias certificado_servidor -rfc -file certificado_servidor.pem -storepass 1234567
Certificate stored in file <certificado_servidor.pem>

C:\Users\luise\Desktop\OCTAVO SEMESTRE\Distribuidos\Tareas_team_dos\Tarea_02>
```

*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*

```
Símbolo del sistema
Microsoft Windows [Versión 10.0.22000.527]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\luise\Desktop\OCTAVO SEMESTRE\Distribuidos\Tareas_team_dos\Tarea_02>keytool -import -alias certificado_servidor -file certificado_servidor.pem -keystore keystore_cliente.jks -storepass 123456
Owner: CN=team2, OU=team2unit, O=team2organization, L=chalco, ST=edomex, C=MX
Issuer: CN=team2, OU=team2unit, O=team2organization, L=chalco, ST=edomex, C=MX
Serial number: 2f1df3428219f30
Valid from: Tue Mar 01 19:24:13 CST 2022 until: Mon May 30 20:24:13 CDT 2022
Certificate fingerprints:
    SHA1: 80:10:82:D6:55:2B:2A:20:FD:45:3B:13:30:C2:3E:DD:2C:9C:DD:82
    SHA256: EF:81:D3:31:0A:2A:F7:97:76:AE:23:6A:F6:8A:D9:EE:9C:23:5D:C2:27:CB:09:AB:73:D1:EE:FC:55:FC:75:09
Signature algorithm name: SHA256withRSA
Subject Public Key Algorithm: 2048-bit RSA key
Version: 3

Extensions:

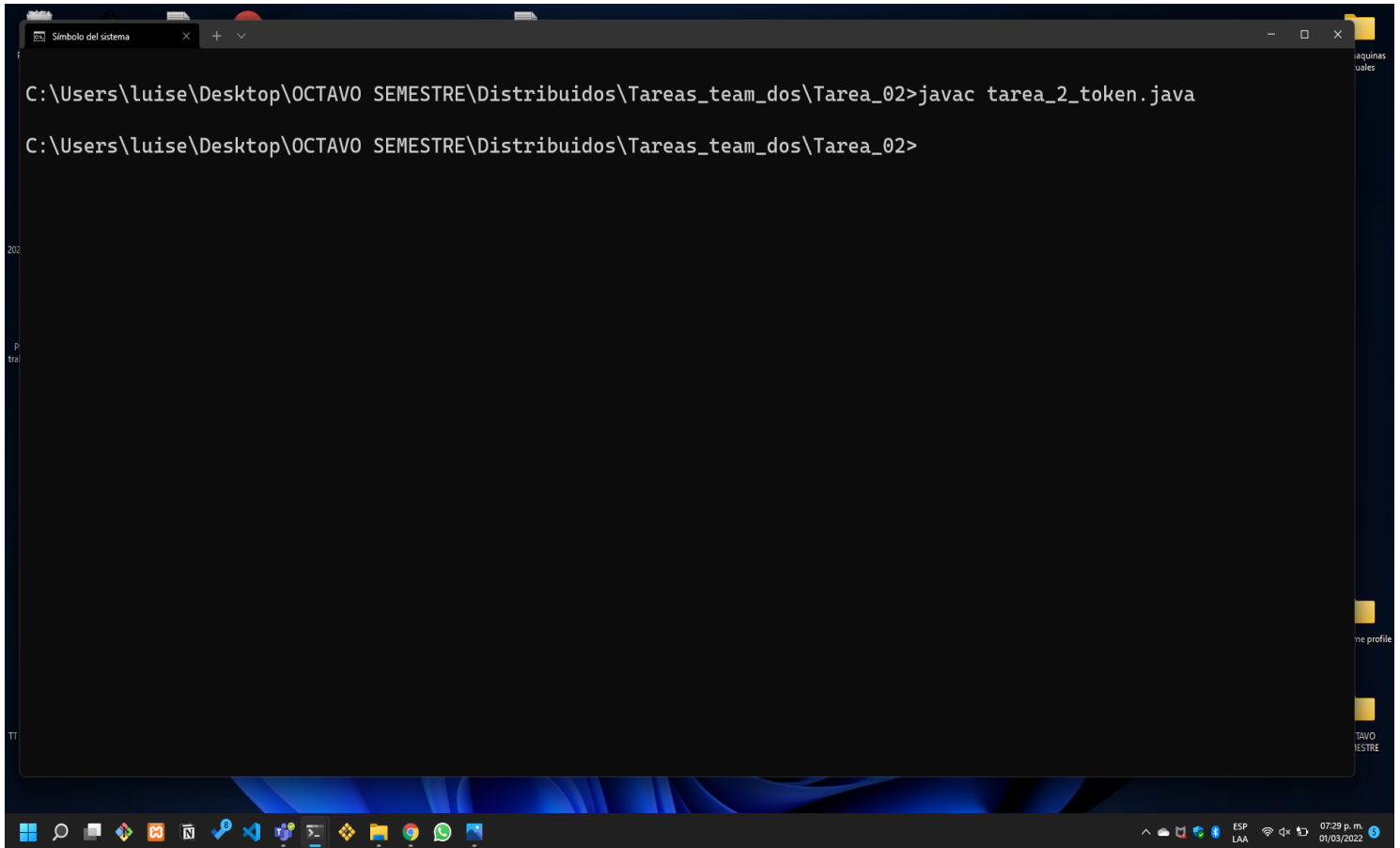
#1: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: E8 87 B0 40 24 D6 DD 10    55 A3 97 DD 85 D1 5C 83    ...@$...U....\
0010: 35 7C 17 EE                5...
]
]

Trust this certificate? [no]: yes
Certificate was added to keystore

C:\Users\luise\Desktop\OCTAVO SEMESTRE\Distribuidos\Tareas_team_dos\Tarea_02>
```

*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 "C:\Users\luise\Desktop\OCTAVO SEMESTRE\Distribuidos\Tareas\_team\_dos\Tarea\_02>". The taskbar at the bottom shows various application icons and the system clock indicating 07:28 p.m. on 01/03/2022.

```
Símbolo del sistema
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.

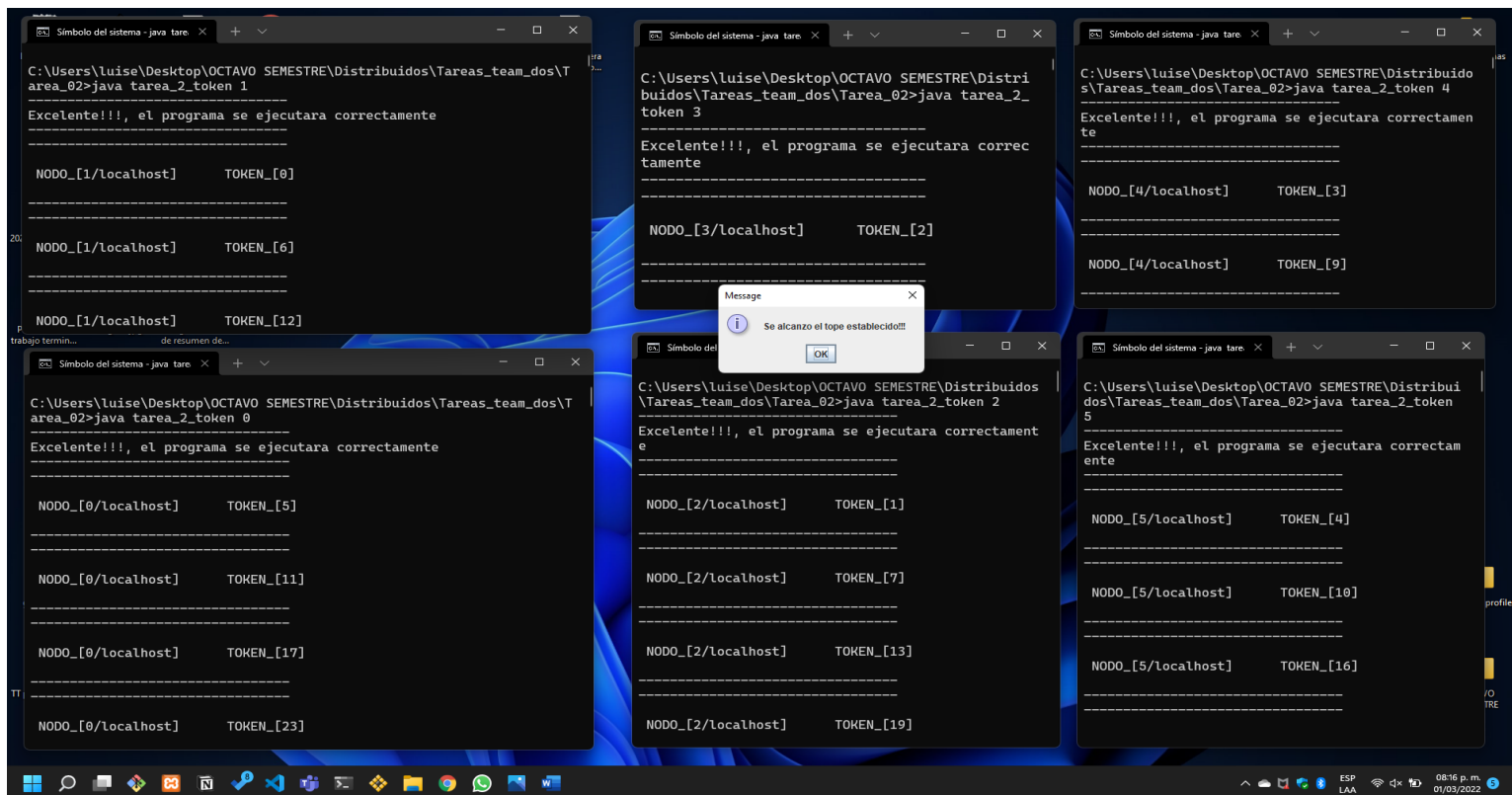


Ilustración 6 Ejecución del programa inicio

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

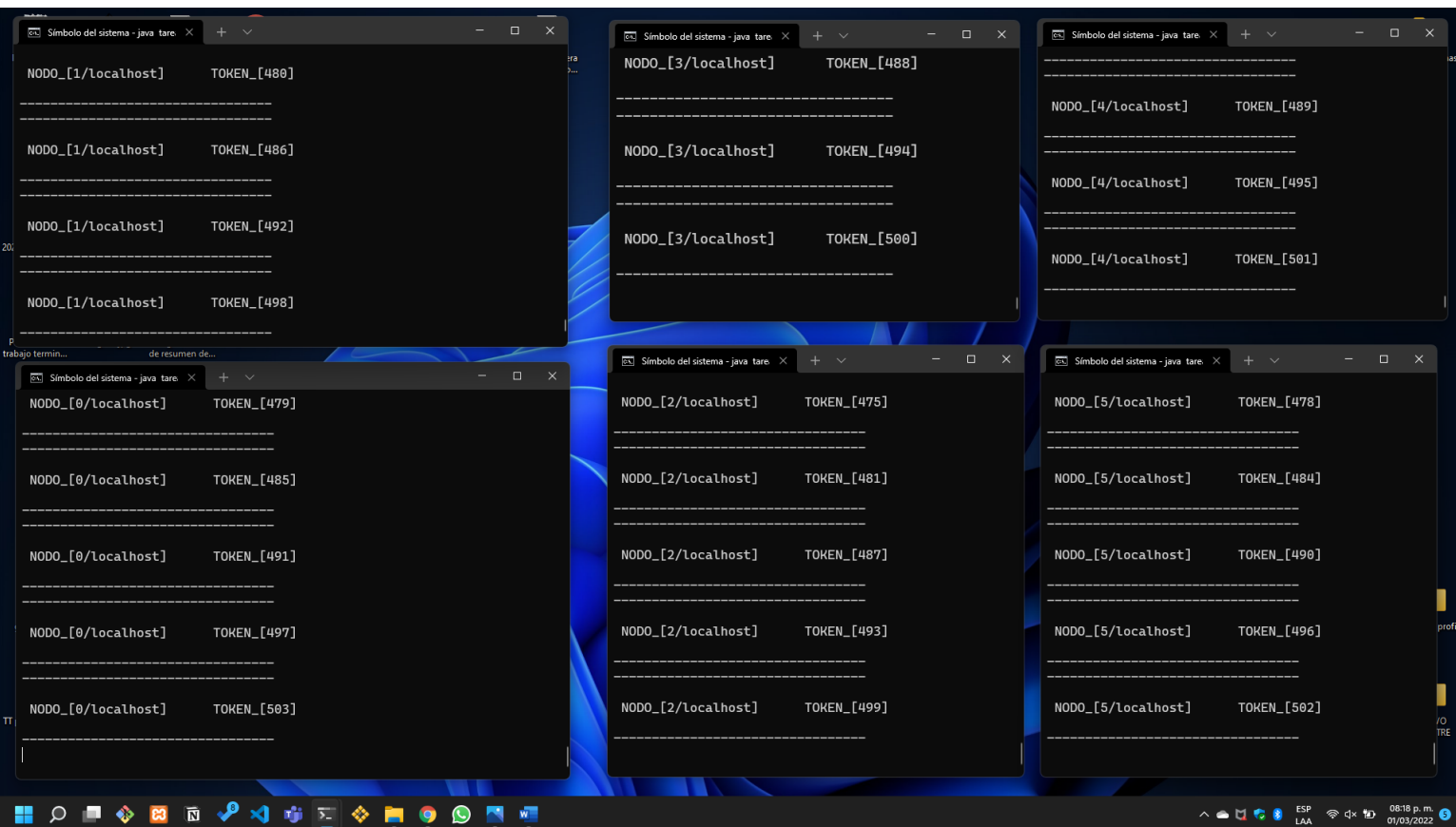


Ilustración 7 Ejecución del programa final

## **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 empecemos 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.