

Instituto Politécnico Nacional  
Escuela Superior de Computo

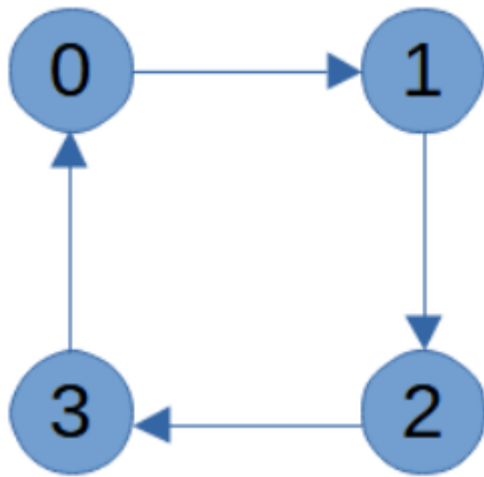
Desarrollo de Sistemas  
Distribuidos

Tarea 2. Implementación de  
un token-ring

Isaac Godínez Cortés  
4CV14

# Desarrollo.

En esta tarea se va a desarrollar un programa en Java, el cual implementará un token que se enviará de un nodo a otro mediante sockets seguros, en una topología lógica de anillo. El anillo constará de 4 nodos:



El token será un número entero de 64 bits.

1. Al principio, el nodo 0 enviará el token al nodo 1.
2. El nodo 1 recibirá el token y lo enviará al nodo 2.
3. El nodo 2 recibirá el token y lo enviará al nodo 3.
4. El nodo 3 recibirá el token y lo enviará al nodo 0.
5. El nodo 0 recibirá el token y lo enviará al nodo 1.
6. Ir al paso 2.

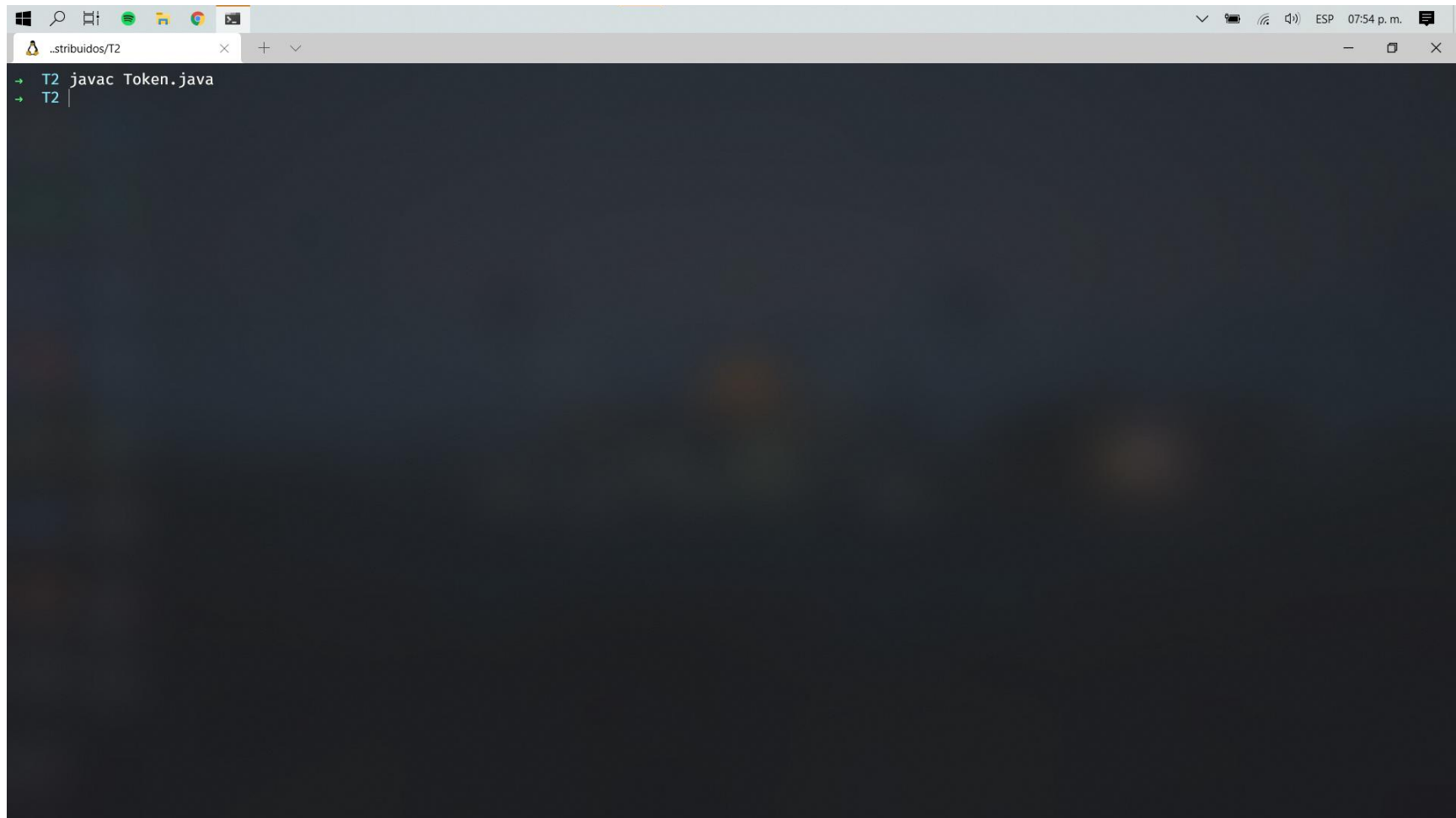
Cuando un nodo reciba el token, lo incrementará en 1 y desplegará el valor del token.

Cuando la cuenta en el nodo 0 llegue a 1000, el nodo 0 deberá terminar su ejecución.

El programa se probará en una sola computadora utilizando cuatro ventanas de consola, cada ventana ejecutará una instancia del programa.

Notar que el programa es un cliente y un servidor.

# Capturas de pantalla.



A screenshot of a Windows terminal window. The title bar shows the window name as '..stribuidos/T2'. The terminal has a dark background with light blue text. The prompt 'T2' is followed by the command 'javac Token.java'. The cursor is positioned at the end of the command line.

```
T2 javac Token.java
T2 |
```

*Ilustración 1) Compilación del programa.*

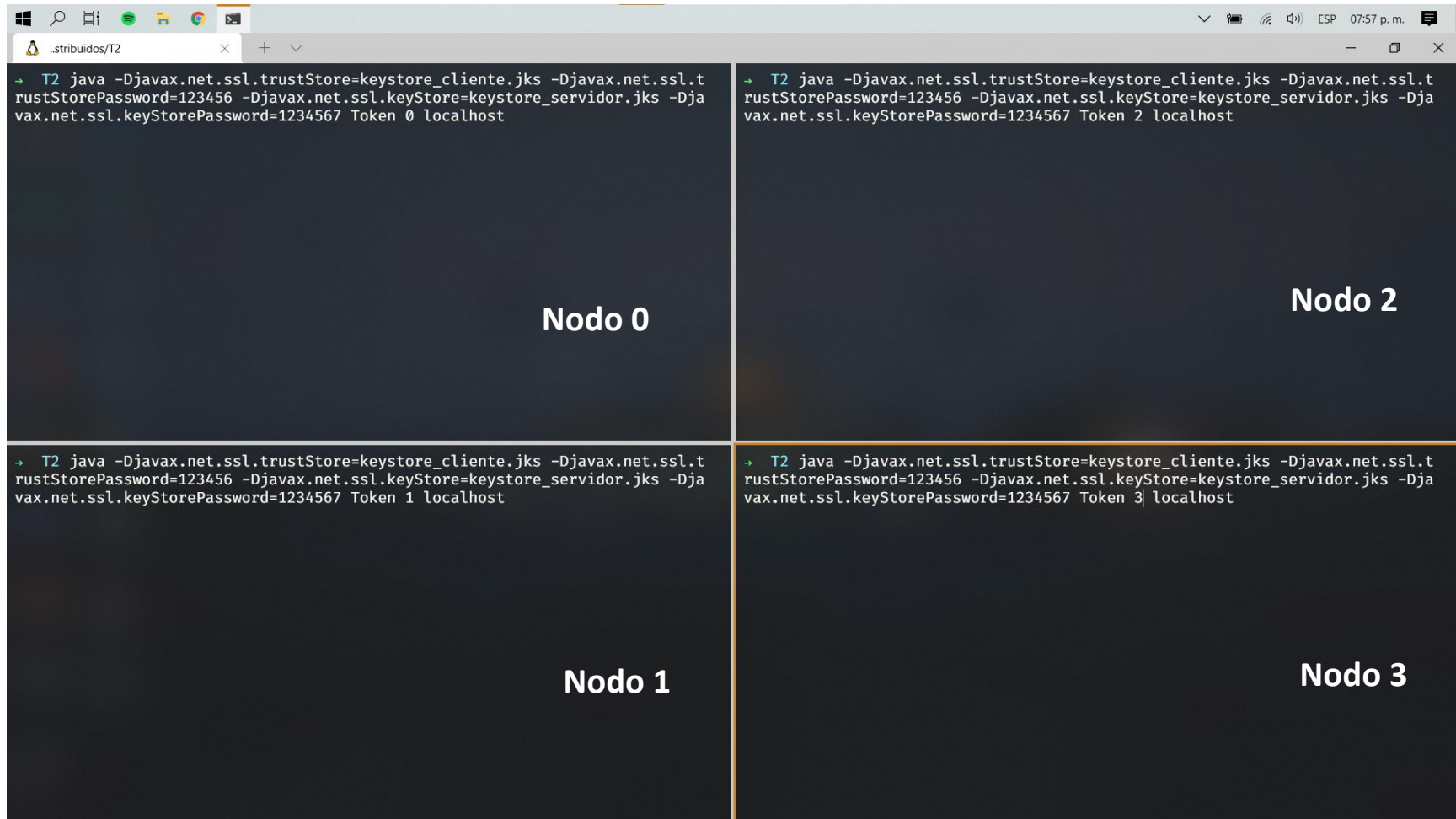
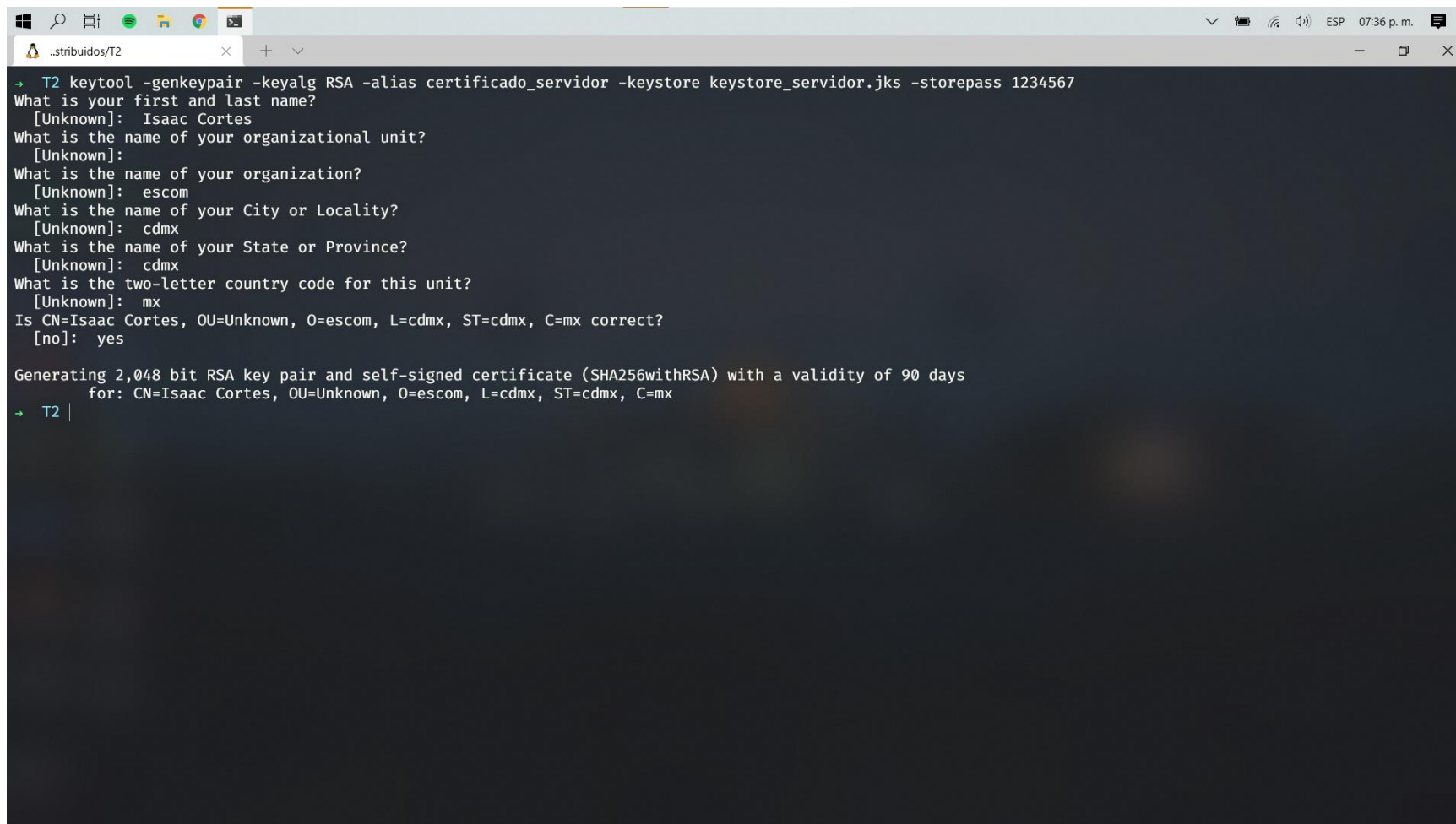


Ilustración 2) Comando para ejecutar el programa, debido a que es cliente y servidor, se deben pasar el keystore del cliente y del servidor para su ejecución.

Nodo 0		Nodo 2	
<pre>Nodo: 0 Token: 969 Nodo: 0 Token: 973 Nodo: 0 Token: 977 Nodo: 0 Token: 981 Nodo: 0 Token: 985 Nodo: 0 Token: 989 Nodo: 0 Token: 993 Nodo: 0 Token: 997 Nodo: 0 Token: 1001 → T2</pre>		<pre>Token: 995 Nodo: 2 Token: 999 Exception in thread "main" java.net.SocketException: Connection reset     at java.base/sun.nio.ch.NioSocketImpl.implRead(NioSocketImpl.java:323)     at java.base/sun.nio.ch.NioSocketImpl.read(NioSocketImpl.java:350)     at java.base/sun.nio.ch.NioSocketImpl\$1.read(NioSocketImpl.java:803)     at java.base/java.net.Socket\$SocketInputStream.read(Socket.java:976)     at java.base/sun.security.ssl.SSLSocketInputRecord.read(SSLSocketInputRecord.java:478)     at java.base/sun.security.ssl.SSLSocketInputRecord.readHeader(SSLSocketInputRecord.java:472)     at java.base/sun.security.ssl.SSLSocketInputRecord.bytesInCompletePacket(SSLSocketInputRecord.java:70)     at java.base/sun.security.ssl.SSLSocketImpl.readApplicationRecord(SSLSocketImpl.java:1454)     at java.base/sun.security.ssl.SSLSocketImpl\$AppInputStream.read(SSLSocketImpl.java:1060)</pre>	
Nodo 1		Nodo 3	
<pre>Nodo: 1 Token: 978 Nodo: 1 Token: 982 Nodo: 1 Token: 986 Nodo: 1 Token: 990 Nodo: 1 Token: 994 Nodo: 1 Token: 998 Exception in thread "main" java.io.EOFException     at java.base/java.io.DataInputStream.readFully(DataInputStream.java:203)     at java.base/java.io.DataInputStream.readLong(DataInputStream.java:422)     at Token.main(Token.java:81) → T2</pre>		<pre>Token: 992 Nodo: 3 Token: 996 Nodo: 3 Token: 1000 Exception in thread "main" java.net.SocketException: Connection reset     at java.base/sun.nio.ch.NioSocketImpl.implRead(NioSocketImpl.java:323)     at java.base/sun.nio.ch.NioSocketImpl.read(NioSocketImpl.java:350)     at java.base/sun.nio.ch.NioSocketImpl\$1.read(NioSocketImpl.java:803)     at java.base/java.net.Socket\$SocketInputStream.read(Socket.java:976)     at java.base/sun.security.ssl.SSLSocketInputRecord.read(SSLSocketInputRecord.java:478)     at java.base/sun.security.ssl.SSLSocketInputRecord.readHeader(SSLSocketInputRecord.java:472)     at java.base/sun.security.ssl.SSLSocketInputRecord.bytesInCompletePacket(SSLSocketInputRecord.java:70)     at java.base/sun.security.ssl.SSLSocketImpl.readApplicationRecord(SSLSocketImpl.java:1454)</pre>	

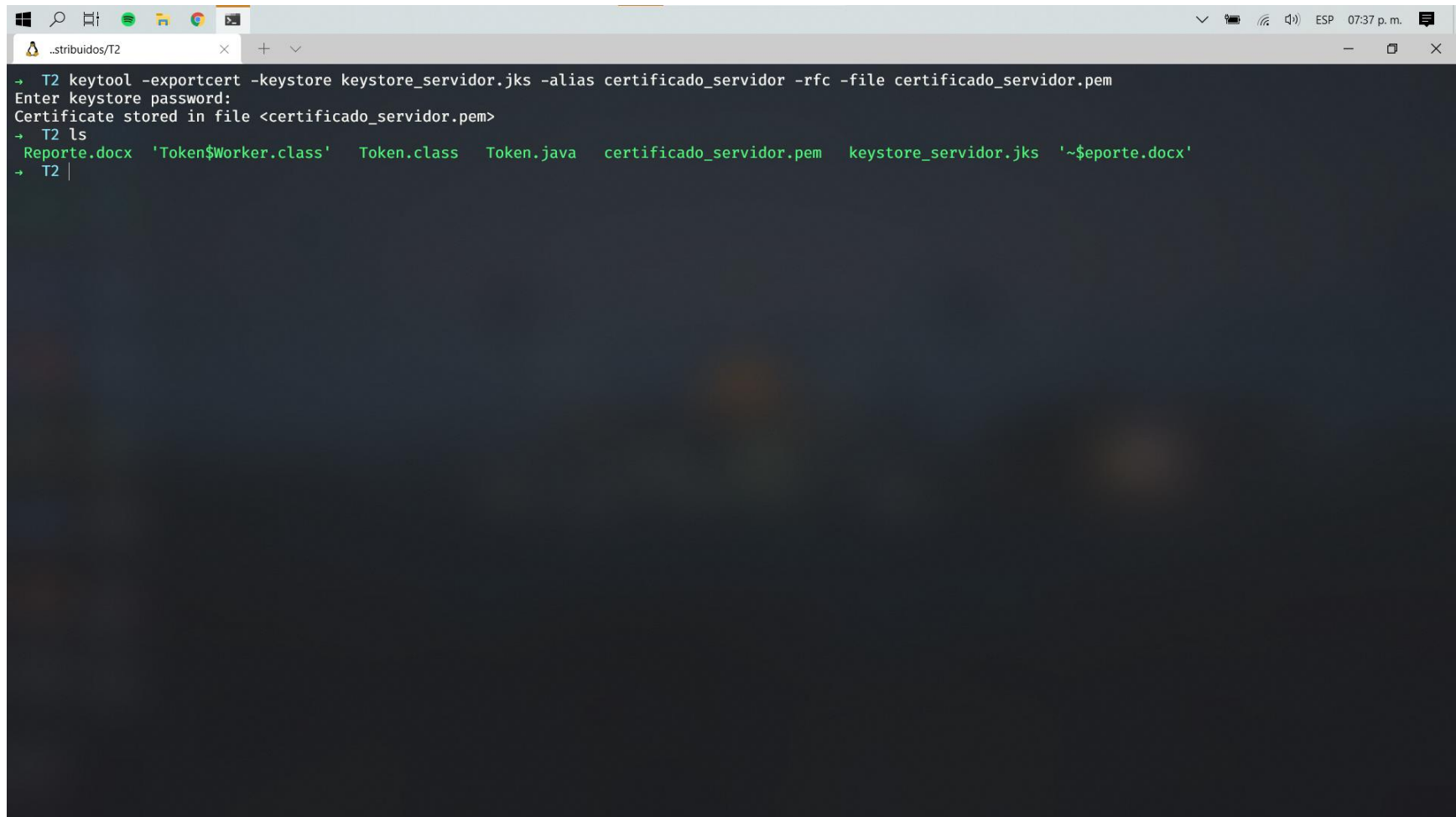
Ilustración 3) Ejecución y salida del programa.



```
→ T2 keytool -genkeypair -keyalg RSA -alias certificado_servidor -keystore keystore_servidor.jks -storepass 1234567
What is your first and last name?
[Unknown]: Isaac Cortes
What is the name of your organizational unit?
[Unknown]:
What is the name of your organization?
[Unknown]: escom
What is the name of your City or Locality?
[Unknown]: cdmx
What is the name of your State or Province?
[Unknown]: cdmx
What is the two-letter country code for this unit?
[Unknown]: mx
Is CN=Isaac Cortes, OU=Unknown, O=escom, L=cdmx, ST=cdmx, 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=Isaac Cortes, OU=Unknown, O=escom, L=cdmx, ST=cdmx, C=mx
→ T2 |
```

Ilustración 4) Creación de un certificado autofirmado utilizando el programa keytool incluido en JDK.



```
→ T2 keytool -exportcert -keystore keystore_servidor.jks -alias certificado_servidor -rfc -file certificado_servidor.pem
Enter keystore password:
Certificate stored in file <certificado_servidor.pem>
→ T2 ls
Reporte.docx 'Token$Worker.class' Token.class Token.java certificado_servidor.pem keystore_servidor.jks '~$eporte.docx'
→ T2 |
```

*Ilustración 5) Obtención del certificado contenido en el keystore, generando el archivo 'certificado\_servidor.pem'.*

```
→ T2 keytool -import -alias certificado_servidor -file certificado_servidor.pem -keystore keystore_cliente.jks -storepass 123456
Owner: CN=Isaac Cortes, OU=Unknown, O=escom, L=cdmx, ST=cdmx, C=mx
Issuer: CN=Isaac Cortes, OU=Unknown, O=escom, L=cdmx, ST=cdmx, C=mx
Serial number: 5472f5a01bf25029
Valid from: Thu Sep 09 19:32:03 CDT 2021 until: Wed Dec 08 18:32:03 CST 2021
Certificate fingerprints:
    SHA1: 09:7C:6A:27:E5:B2:A1:B2:C8:52:B8:1A:EF:0F:61:35:99:3A:31:EB
    SHA256: F2:A8:3D:51:27:93:0B:FB:37:E7:05:3F:66:99:FE:61:C5:10:46:59:77:2C:DA:CE:D5:CB:BE:C8:FA:DB:4D:AA
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: 88 0C BE 4A 44 7A 09 55    86 D5 1A F6 96 75 5C 07    ... JDz.U.....u\
0010: E6 54 09 5E                .T.^
]
]

Trust this certificate? [no]: yes
Certificate was added to keystore
→ T2 |
```

Ilustración 6) Creación del keystore que utilizara el cliente, que debe contener el certificado del servidor.



# Conclusiones.

En esta práctica se implementó exitosamente un token ring, donde si un nodo se cae, automáticamente se caen todos, ya que se rompe este anillo que hay entre los nodos, por ese mismo motivo, al terminar la ejecución del programa, se muestra un error y es debido a que el nodo 0 acaba bruscamente y sin esperar a los demás nodos, entonces los nodos detectan esto y muestran un error.