



**INSTITUTO POLITÉCNICO NACIONAL**  
**ESCUELA SUPERIOR DE CÓMPUTO**



-----Desarrollo de Sistemas Distribuidos-----

**TAREA 2:**

Transferencia de archivos utilizando sockets seguros

**Alumno:**

Meza Vargas Brandon David

**Grupo:**

4CV12

**Profesor:**

Pineda Guerrero Carlos

## Desarrollo

Primeramente se tuvo que crear un certificado auto firmado, esto se puede ver en la imagen 1.

```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Versión 10.0.19043.1889]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\PC>keytool -genkeypair -keyalg RSA -alias certificado_servidor -keystore keystore_servidor.jks -storepass 12345
error de herramienta de claves: java.lang.Exception: La contraseña del almacén de claves debe tener al menos 6 caracteres

C:\Users\PC>keytool -genkeypair -keyalg RSA -alias certificado_servidor -keystore keystore_servidor.jks -storepass 1234567
¿Cuáles son su nombre y su apellido?
[Unknown]: Brandon Meza
¿Cuál es el nombre de su unidad de organización?
[Unknown]: unidad
¿Cuál es el nombre de su organización?
[Unknown]: organizacion
¿Cuál es el nombre de su ciudad o localidad?
[Unknown]: CDMX
¿Cuál es el nombre de su estado o provincia?
[Unknown]: CDMX
¿Cuál es el código de país de dos letras de la unidad?
[Unknown]: MX
¿Es correcto CN=Brandon Meza, OU=unidad, O=organización, L=CDMX, ST=CDMX, C=MX?
[no]: si

Introduzca la contraseña de clave para <certificado_servidor>
(INTRO si es la misma contraseña que la del almacén de claves):

Warning:
El almacén de claves JKS utiliza un formato propietario. Se recomienda migrar a PKCS12, que es un formato estándar del sector que utiliza "keytool -importkeystore -srckeystore keystore_servi
dor.jks -destkeystore keystore_servidor.jks -deststoretype pkcs12".

C:\Users\PC>
```

Imagen 1. Creación del certificado auto firmado

Posteriormente, se obtiene el certificado contenido en el keystore como se puede ver en la imagen 2.

```
C:\WINDOWS\system32\cmd.exe
C:\Users\PC>keytool -exportcert -keystore keystore_servidor.jks -alias certificado_servidor -rfc -file certificado_servidor.pem
Introduzca la contraseña del almacén de claves:
Certificado almacenado en el archivo <certificado_servidor.pem>

Warning:
El almacén de claves JKS utiliza un formato propietario. Se recomienda migrar a PKCS12, que es un formato estándar del sector que utiliza "keytool -importkeystore -srcstore keystore_servi
dor.jks -deststoretype pkcs12".

C:\Users\PC>
```

Imagen 2. Obteniendo el certificado del keystore

Una vez hecho lo anterior, pasamos a crear un keystore para que el cliente lo use, este será el repositorio de confianza, el cual contiene el certificado del servidor, esto lo hacemos en la imagen 3.

```
C:\WINDOWS\system32\cmd.exe
No se ha agregado el certificado al almacén de claves

C:\Users\PC>keytool -import -alias certificado_servidor -file certificado_servidor.pem -keystore keystore_cliente.jks -storepass 1234567
Propietario: CN=Brandon Meza, OU=unidad, O=organizacion, L=CDMX, ST=CDMX, C=MX
Emisor: CN=Brandon Meza, OU=unidad, O=organizacion, L=CDMX, ST=CDMX, C=MX
Número de serie: 2a2fbd1d
Válido desde: Thu Sep 08 19:33:38 CDT 2022 hasta: Wed Dec 07 18:33:38 CST 2022
Huellas digitales del certificado:
    MD5: 05:87:7D:45:A4:51:68:97:F0:68:DF:5E:76:80:92:33
    SHA1: 92:DD:30:A8:C5:90:D1:B9:58:02:C9:B5:D9:A8:6D:19:41:33:B3:80
    SHA256: E9:46:3A:E3:58:A3:EB:40:27:82:A4:A1:0A:87:C5:0B:0A:09:85:C9:2B:1F:F3:EF:10:4A:74:13:BF:7E:EA:BD
Nombre del algoritmo de firma: SHA256withRSA
Algoritmo de clave publica de asunto: Clave RSA de 2048 bits
Versión: 3

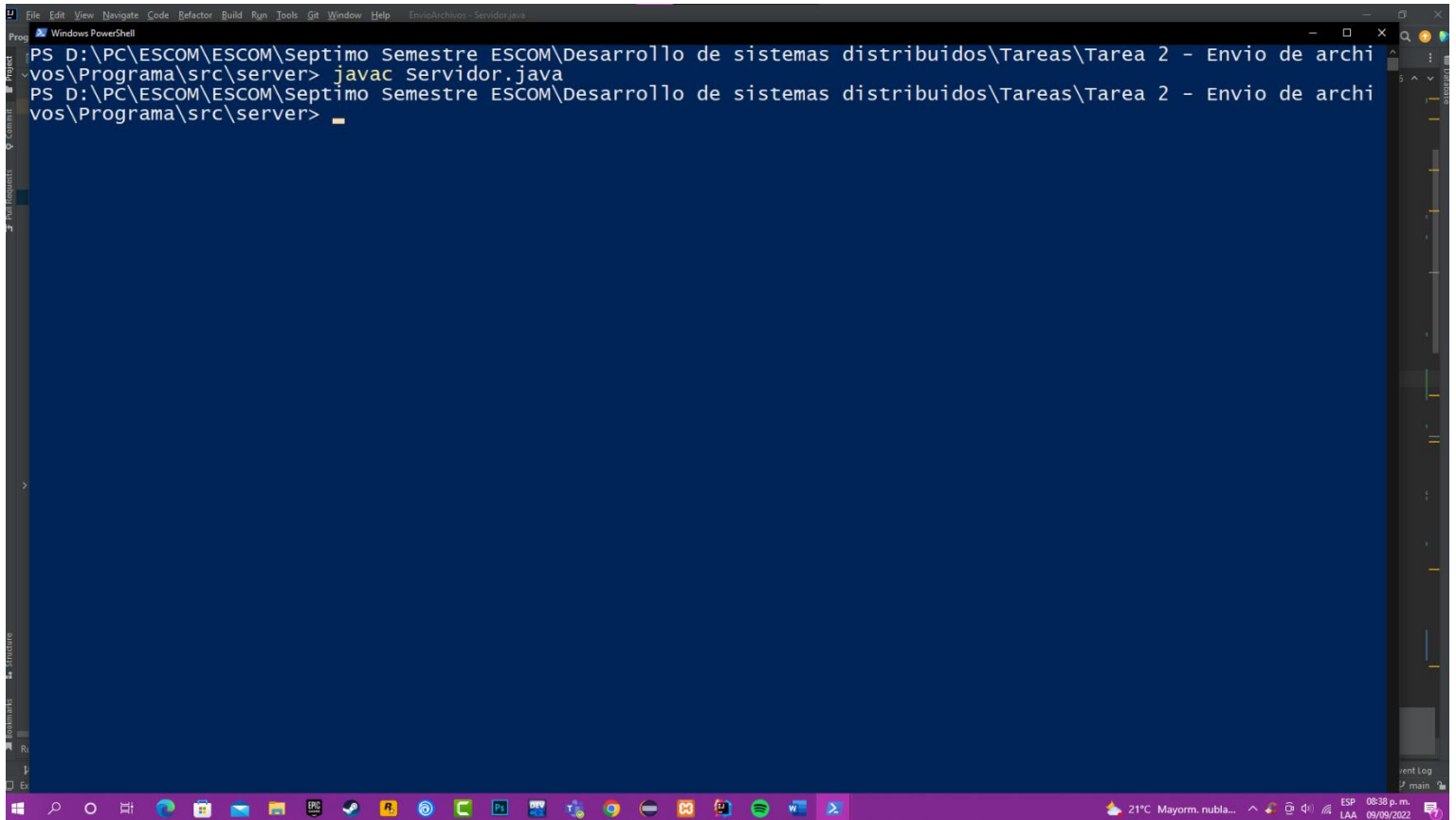
Extensiones:
#1: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: 4B B7 85 95 23 A6 53 03   E5 63 6B 38 A3 F1 AC DB   K...#.S..ck8....
0010: 52 F0 74 8E               R.t.
]
]

¿Confiar en este certificado? [no]: si
Se ha agregado el certificado al almacén de claves

C:\Users\PC>
```

Imagen 3. Creación del keystore para que lo pueda usar el cliente.

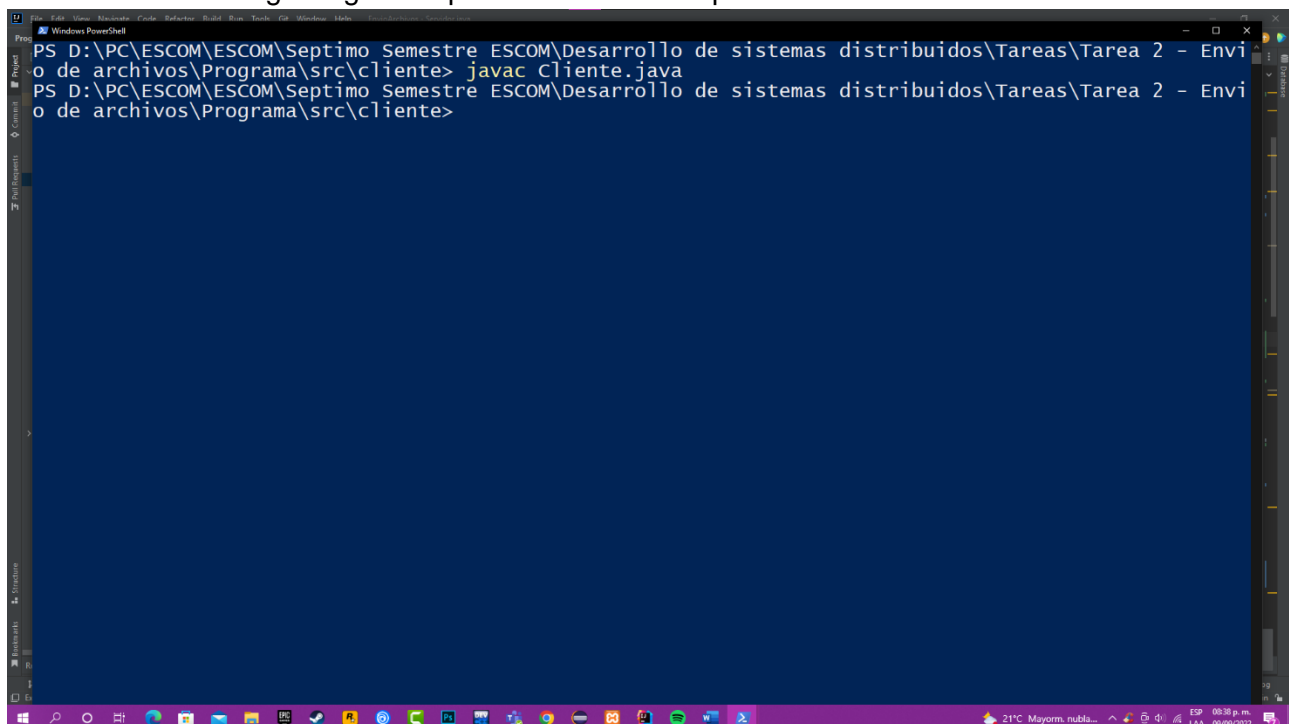
Una vez realizado todo lo anterior, ahora si procedemos con la compilación y ejecución de nuestro cliente y servidor, primeramente, como vemos en la imagen 4, tenemos la compilación del servidor.



```
PS D:\PC\ESCOM\ESCOM\Septimo Semestre ESCOM\Desarrollo de sistemas distribuidos\Tareas\Tarea 2 - Envio de archivos\Programa\src\server> javac Servidor.java
PS D:\PC\ESCOM\ESCOM\Septimo Semestre ESCOM\Desarrollo de sistemas distribuidos\Tareas\Tarea 2 - Envio de archivos\Programa\src\server>
```

Imagen 4. Compilación del servidor

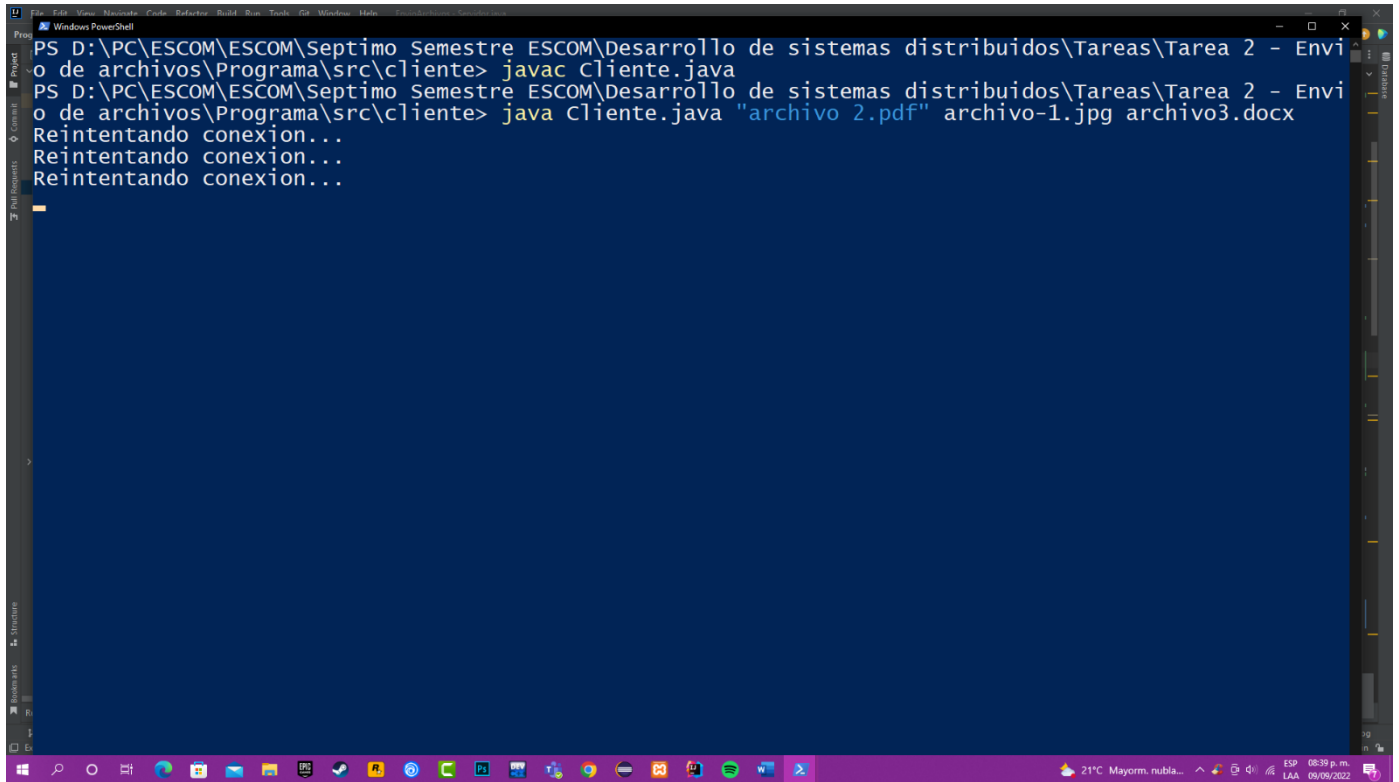
Ahora en la imagen siguiente podemos ver la compilación del cliente.



```
PS D:\PC\ESCOM\ESCOM\Septimo Semestre ESCOM\Desarrollo de sistemas distribuidos\Tareas\Tarea 2 - Envio de archivos\Programa\src\cliente> javac Cliente.java
PS D:\PC\ESCOM\ESCOM\Septimo Semestre ESCOM\Desarrollo de sistemas distribuidos\Tareas\Tarea 2 - Envio de archivos\Programa\src\cliente>
```

Imagen 5. Compilación del cliente

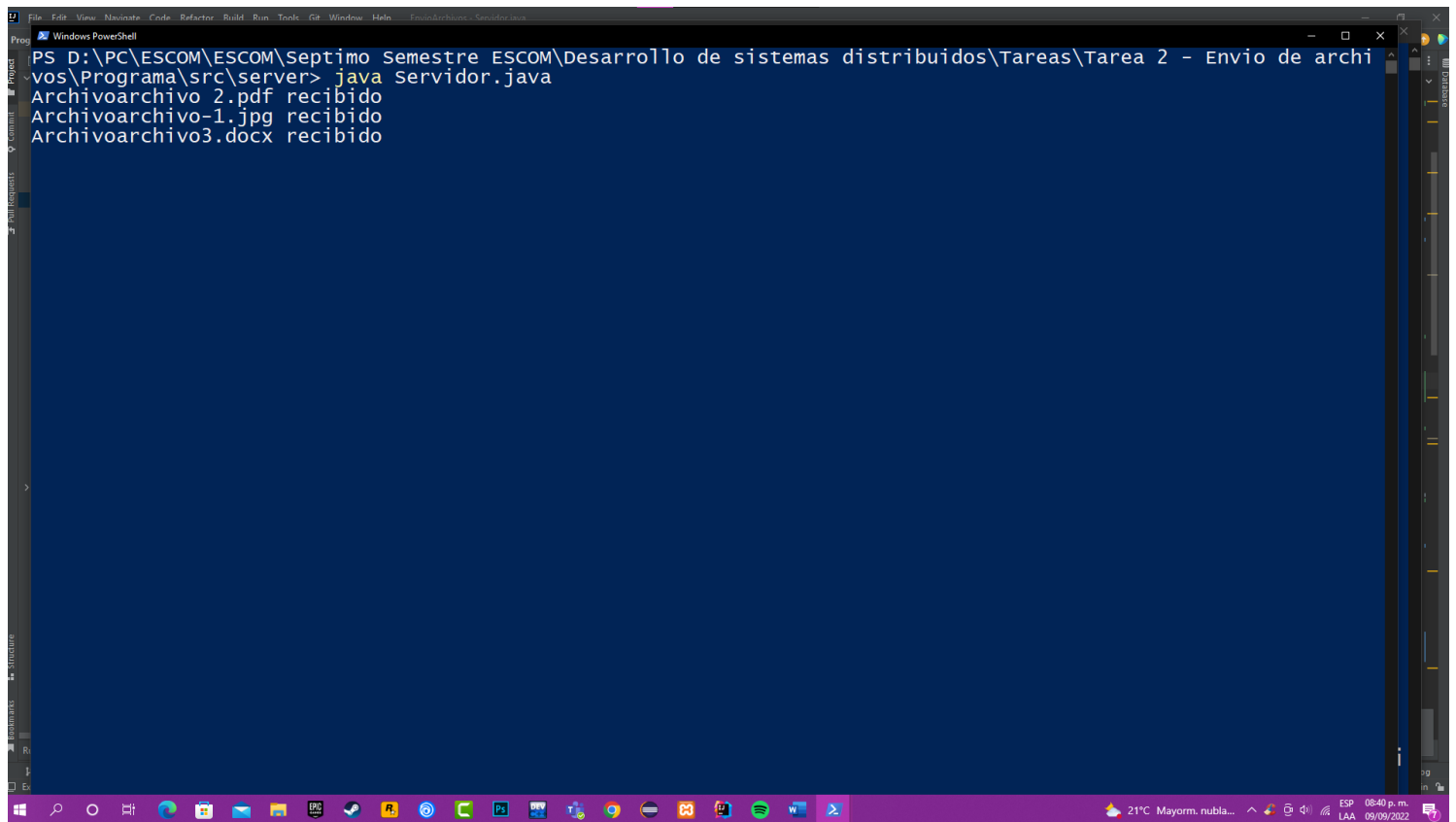
Ahora bien tenemos la ejecución de los programas, primeramente se ejecutó el cliente para verificar que se espera reintentando la conexión, de igual forma se puede ver que se están mandando 3 archivos como observamos en la imagen 6. Cabe resaltar que las propiedades para poder leer las llaves se encuentran en el código de los programas por lo que no es necesario incluir esta parte al ejecutarlos.



```
PS D:\PC\ESCOM\ESCOM\Septimo Semestre ESCOM\Desarrollo de sistemas distribuidos\Tareas\Tarea 2 - Envi  
o de archivos\Programa\src\cliente> javac Cliente.java  
PS D:\PC\ESCOM\ESCOM\Septimo Semestre ESCOM\Desarrollo de sistemas distribuidos\Tareas\Tarea 2 - Envi  
o de archivos\Programa\src\cliente> java cliente.java "archivo 2.pdf" archivo-1.jpg archivo3.docx  
Reintentando conexion...  
Reintentando conexion...  
Reintentando conexion...
```

*Imagen 6. Ejecución del cliente*

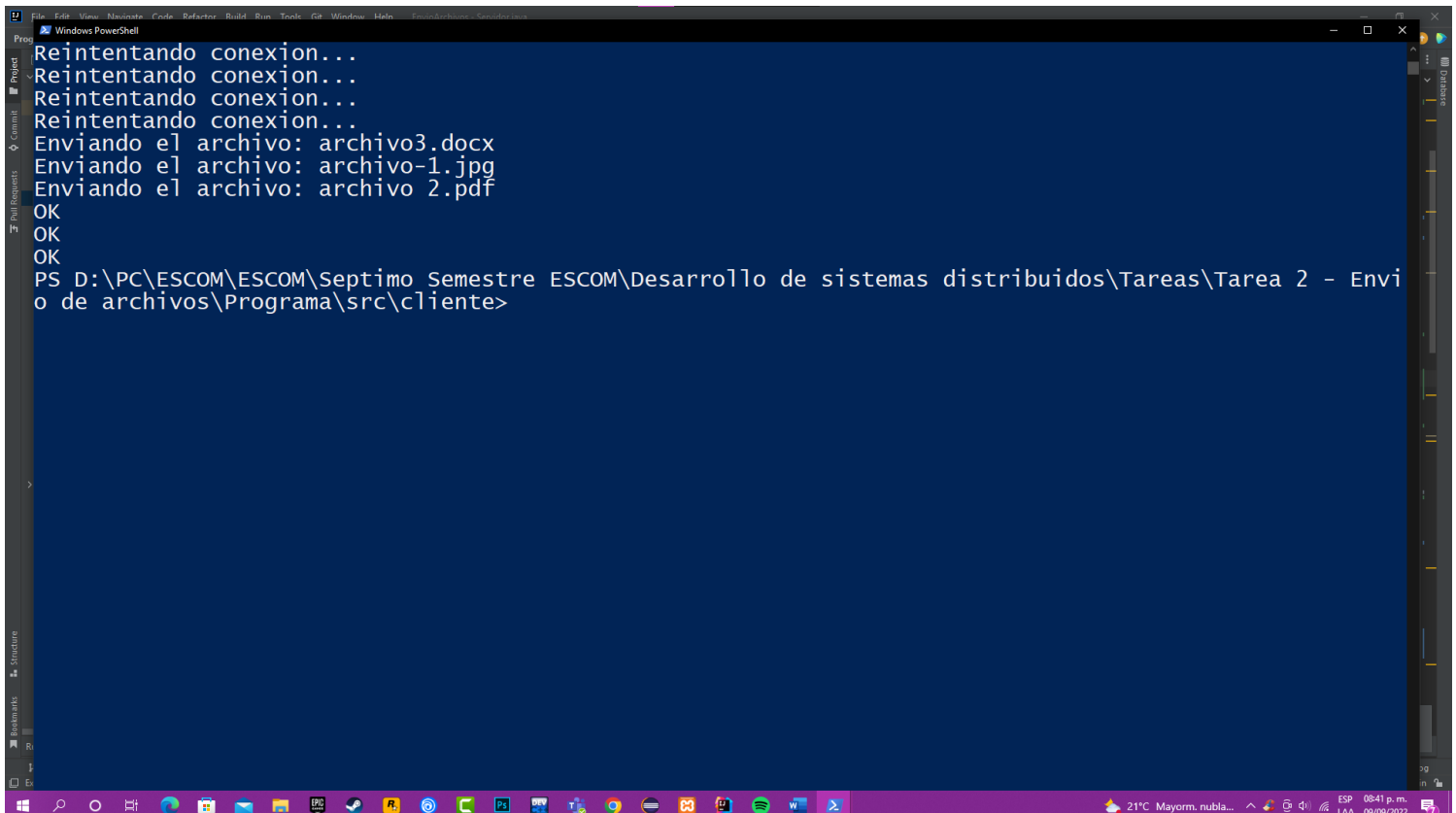
Después al ejecutar el servidor, inmediatamente el cliente hace la conexión y manda los archivos, esto lo vemos en la imagen 7.



```
PS D:\PC\ESCOM\ESCOM\Septimo Semestre ESCOM\Desarrollo de sistemas distribuidos\Tareas\Tarea 2 - Envio de archivos\Programa\src\server> java Servidor.java
Archivoarchivo 2.pdf recibido
Archivoarchivo-1.jpg recibido
Archivoarchivo3.docx recibido
```

*Imagen 7. Ejecución del servidor*

Por último, el cliente recibe las respuestas del servidor como se ve en la siguiente imagen.



```
Reintentando conexion...
Reintentando conexion...
Reintentando conexion...
Reintentando conexion...
Enviando el archivo: archivo3.docx
Enviando el archivo: archivo-1.jpg
Enviando el archivo: archivo 2.pdf
OK
OK
OK
PS D:\PC\ESCOM\ESCOM\Septimo Semestre ESCOM\Desarrollo de sistemas distribuidos\Tareas\Tarea 2 - Envio de archivos\Programa\src\cliente>
```

En la siguiente imagen podemos ver los archivos del cliente y en el servidor, verificando que se pasaron correctamente. La ventana de la derecha corresponde al servidor, y la de la izquierda al cliente.

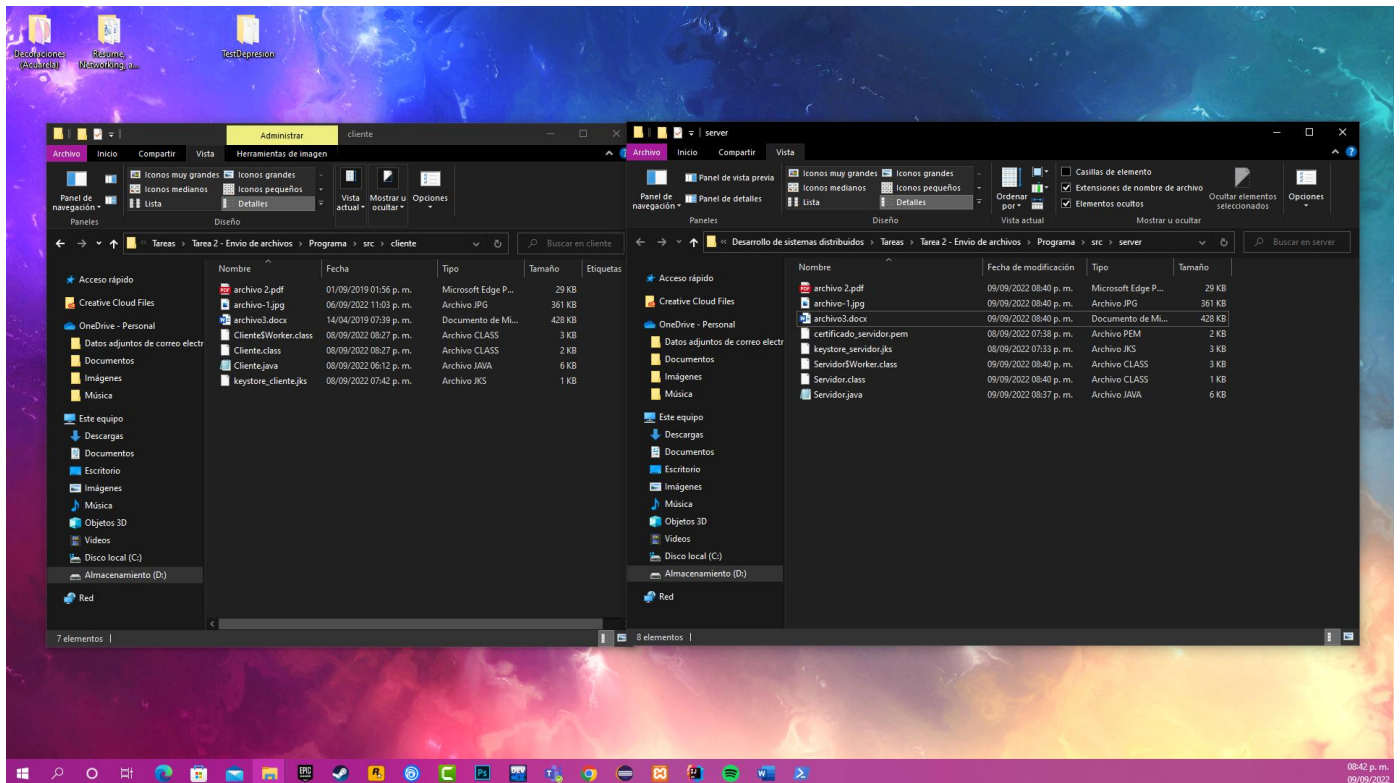


Imagen 8. Archivos del cliente y servidor

De igual forma se muestran en la ventana de comandos usando el comando dir a continuación.

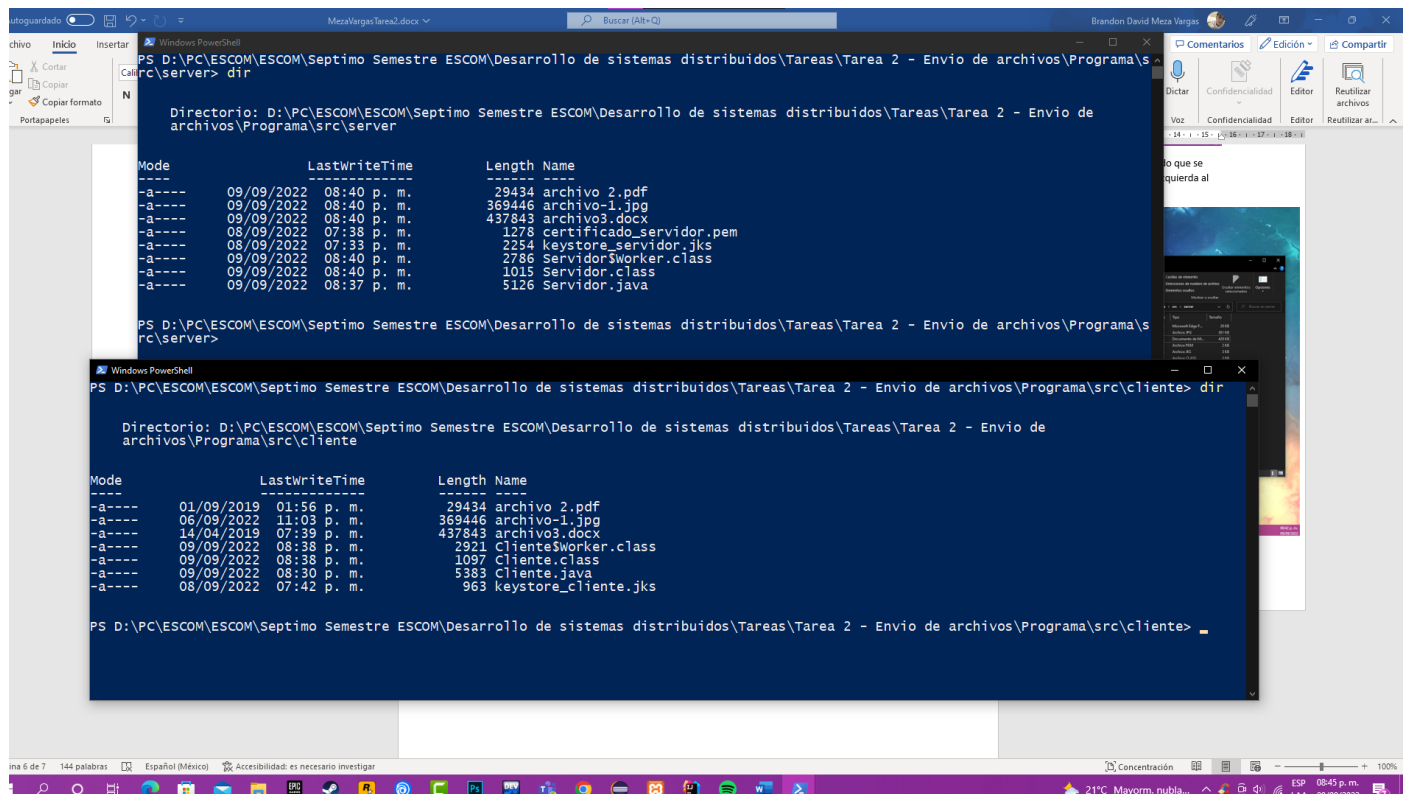


Imagen 9. Archivos del cliente y servidor en la ventana de comandos

## **Conclusiones**

Esta práctica fue muy interesante, pues personalmente, antes había visto conexión con sockets no seguros pero nunca con sockets seguros y no fue hasta esta práctica que se implementaron los sockets seguros que comprendí como funcionan de mejor forma.

Puedo concluir que fue una buena práctica para poner en marcha lo que vimos en clases sobre sockets seguros y todo lo que conllevan, como crear un certificado auto firmado, obtener certificados, etc.

Es importante saber que con este tipo de conexiones podemos transferir información, en este caso archivos, de una manera completamente segura entre un cliente y un servidor, lo mejor de todo es que no hace falta modificar nuestro código implementado con sockets no seguros de una manera drástica, pues solo es cuestión de modificar y agregar unas cuantas líneas de código extra.