

PAI-1. BYODSEC-BRING YOUR OWN DEVICE SEGURO PARA UNA ENTIDAD HOSPITALARIA USANDO ROAD WARRIOR VPN TLS



Araceli María Benítez Díaz

Paula Poley Ceballos

SCG, 3º ISA 2022



ÍNDICE

• <i>Cambiamos el path de cmd</i>	3
• <i>Creamos un KeyStore (Almacén de claves)</i>	6
• <i>Generación de un Socket TLS Servidor y un Socket TLS Cliente</i>	9
• <i>Compilar y ejecutar código cliente y servidor desde consola</i>	12
• <i>Comprobación del servidor VPN para soportar a los 300 empleados del hospital de forma concurrente.</i>	14

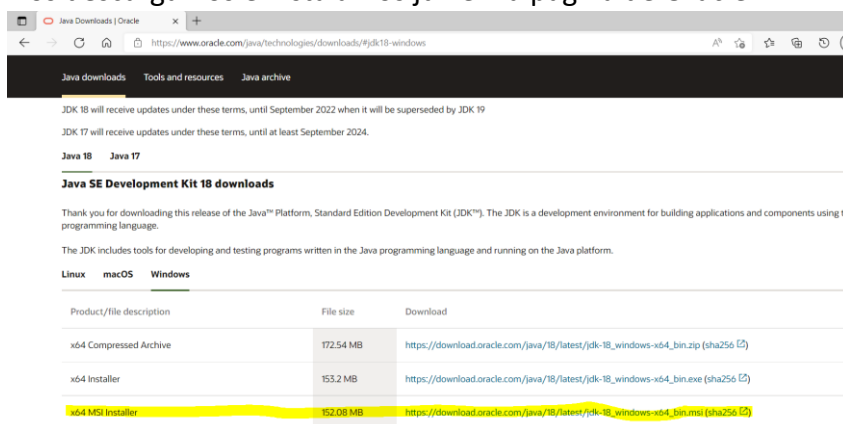
Vamos a comprobar login/Password de usuarios y un mensaje secreto con sockets seguros creando una infraestructura cliente-servidor. Hemos seguido una serie de pasos para crear la arquitectura segura.

Como Security Team elegimos usar el lenguaje Java para realizar el diseño.

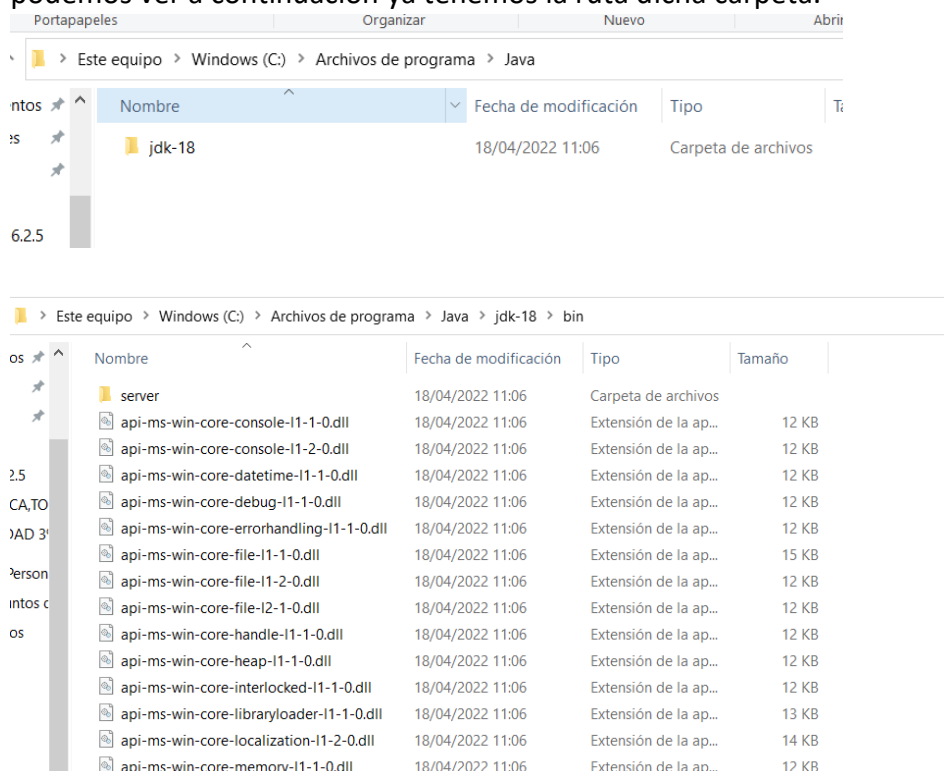
Para empezar:

- **Cambiamos el path de cmd**

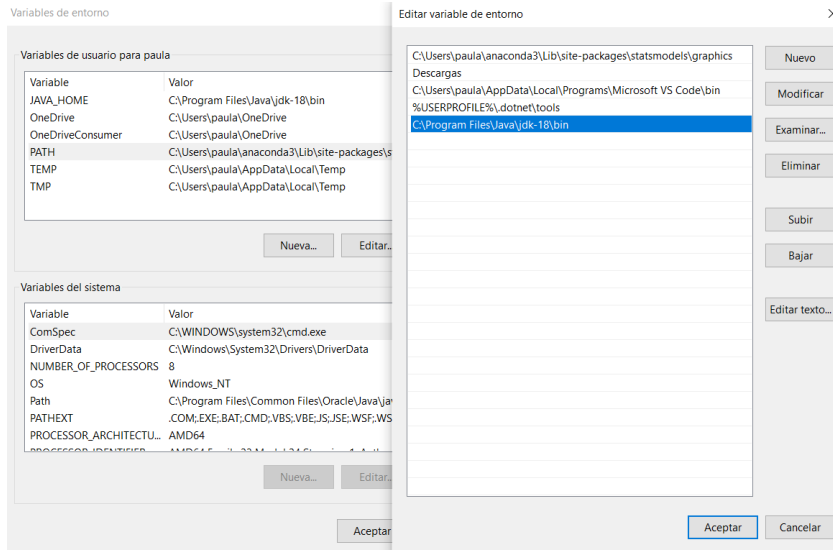
1. Nos descargamos e instalamos jdk en la página de Oracle.



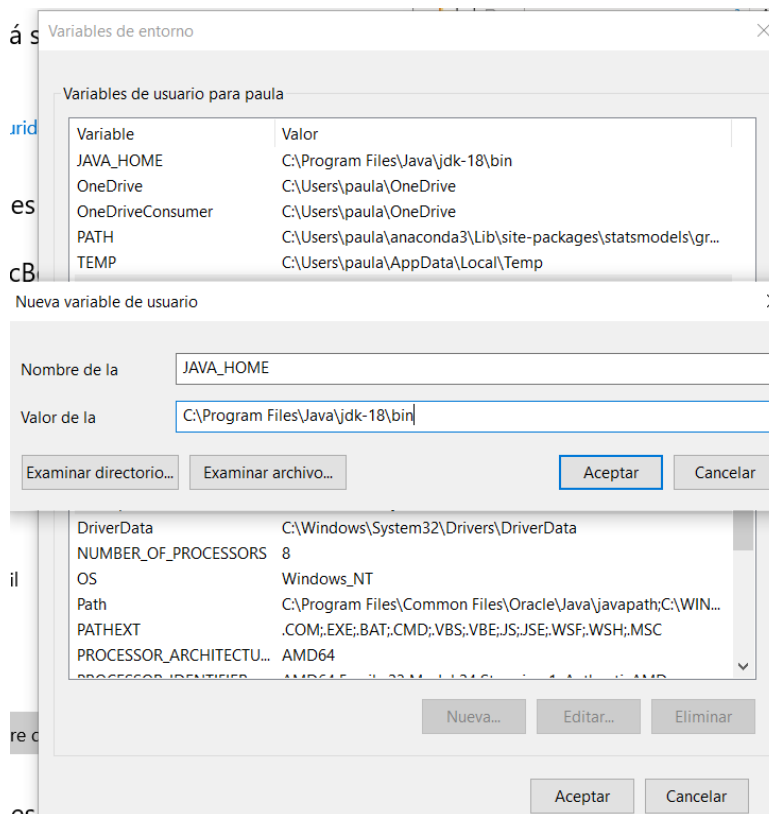
2. Buscamos la carpeta jdk-18 en nuestro equipo y dentro de ella la carpeta bin. Como podemos ver a continuación ya tenemos la ruta dicha carpeta.




- Una vez copiada la ruta de la carpeta bin cambiamos las propiedades de nuestro equipo. Buscamos en la configuración del equipo, editar las variables de entorno de esta cuenta . Seleccionamos PATH y lo editamos añadiendo la ruta de la carpeta y aceptamos.



Por último le damos a nueva y ponemos el nombre de la variable JAVA_HOME y el valor de ella que es la ruta de la carpeta bin.



4. Una vez hecho esto abrimos cmd y ponemos Keytool para ver si ya funciona. En la figura siguiente comprobamos que funciona correctamente.

 Símbolo del sistema

```
C:\Users\paula>Keytool
Key and Certificate Management Tool

Commands:

-certreq          Generates a certificate request
-changealias      Changes an entry's alias
-delete           Deletes an entry
-exportcert       Exports certificate
-genkeypair       Generates a key pair
-genseckey        Generates a secret key
-gencert          Generates certificate from a certificate request
-importcert       Imports a certificate or a certificate chain
-importpass       Imports a password
-importkeystore   Imports one or all entries from another keystore
-keypasswd        Changes the key password of an entry
-list             Lists entries in a keystore
-printcert        Prints the content of a certificate
-printcertreq     Prints the content of a certificate request
-printcrl         Prints the content of a CRL file
-storepasswd      Changes the store password of a keystore
-showinfo         Displays security-related information
-version          Prints the program version

Use "keytool -?, -h, or --help" for this help message
Use "keytool -command_name --help" for usage of command_name.
Use the -conf <url> option to specify a pre-configured options file.
```

5. Creamos un directorio llamado PA1 con el comando → mkdir PA1
6. Salimos con el comando cd .. que lo escribiremos dos veces.

```
C:\WINDOWS\system32>mkdir PA1

C:\WINDOWS\system32>cd ..

C:\Windows>cd ..
```

Atención: todo esto lo estamos creando en system32 y se creará como un archivo fuera pero queremos que se cree en una carpeta. Esto lo podemos hacer de dos maneras, de forma manual creando la carpeta o poniendo mkdir c:\PA1 . Llamando a la carpeta PA1.

- **Creamos un KeyStore (Almacén de claves)**

Creamos un KeyStore (repositorio de certificados de seguridad) para autenticar los servidores de los correspondientes certificados según el protocolo TLS. Los pasos que seguiremos serán:

1. Escribimos Keytool -genkey -keystore c:\SSLStore -keyalg RSA -alias SSLCertificate
Atención.
Sustituimos SSLStore por nuestro directorio creado, es decir, PA1 y al lado ponemos -keyalg RSA porque trabajamos con ello.
A continuación nos pregunta una serie de preguntas que contestaremos o podremos obviarlas.

```
C:\>Keytool -genkey -keystore c:\PA1 -keyalg RSA -alias SSLCertificate
Enter keystore password:
Re-enter new password:
What is your first and last name?
[no]: paula
What is the name of your organizational unit?
[us]:
What is the name of your organization?
[universidad]:
What is the name of your City or Locality?
[sevilla]:
What is the name of your State or Province?
[es]:
What is the two-letter country code for this unit?
[es]:
Is CN=paula, OU=us, O=universidad, L=sevilla, ST=es, C=es correct?
[no]: yes
Generating 2,048 bit RSA key pair and self-signed certificate (SHA256withRSA) with a validity of 90 days
for: CN=paula, OU=us, O=universidad, L=sevilla, ST=es, C=es
```

Podemos apreciar que la última línea nos dice que se ha generado correctamente y tendrá 90 días de validación. Toda esta información es para crear el certificado. Nos pide toda esa información para identificar la entidad que va a utilizarlo.

2. Abrimos Visual Studio y le damos a open folder y abrimos nuestro store .
Anteriormente hemos metido los dos socket proporcionados por el profesor.
(ClientSocket.java y SSocket.java)



3. Ahora tenemos dos formas de inicializarlo.

a. Desde visual studio

En el servidor le damos a botón derecho y a run java.

En la pestaña inferior nos sale Waiting for connection...

```

PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL

Windows PowerShell
Copyright (c) Microsoft Corporation. Todos los derechos reservados.

Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6

PS C:\PA1> & 'C:\Program Files\Java\jdk-18\bin\java.exe' '-enable-preview' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\paula\AppData\Roaming\Code\User\workspaceStorage\3bd0ba55ad98d86d55fc85982275b380\redhat.java\jdt_ws\PA1_759c96a8\bin' 'SSocket'
Waiting for connection...
  
```

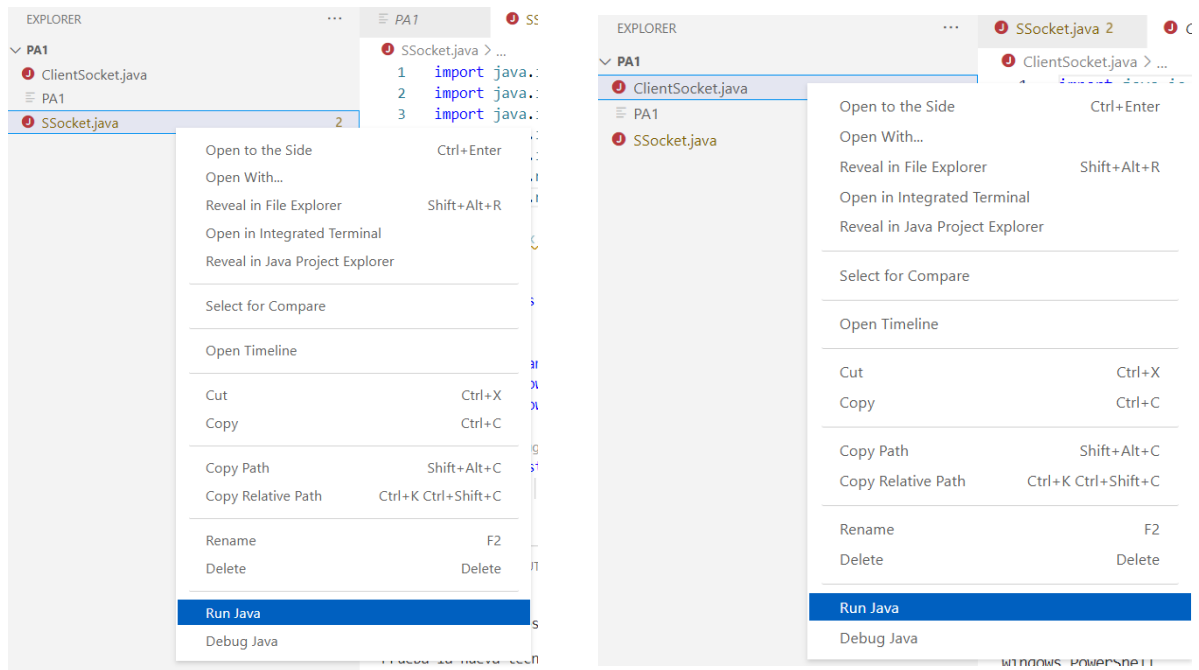
¿Qué es lo que está sucediendo?.

-De manera gráfica el servidor está detrás de una puerta esperando a que alguien le llame, está en espera.

Hacemos lo mismo con el cliente.

¿Y ahora que ocurre?.

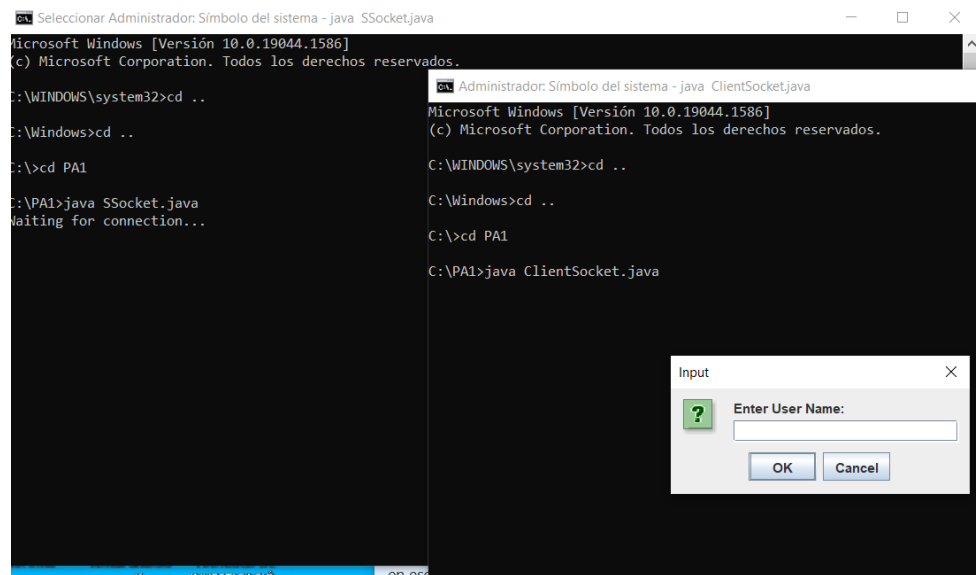
-El cliente llama a la puerta, el servidor la abre y pueden mantener una comunicación entre ellos.



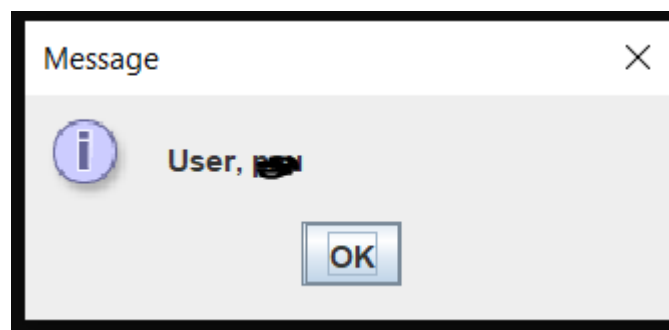
b. Desde cmd .

Entramos en nuestra carpeta PA1, ejecutamos primero el servidor y después el cliente.

Nos sale satisfactorio ya que nos pregunta el usuario y la contraseña.



The screenshot shows two Windows Command Prompt windows and an input dialog box. The left window, titled 'Seleccionar Administrador: Símbolo del sistema - java SSocket.java', shows the following commands and output:
Microsoft Windows [Versión 10.0.19044.1586]
(c) Microsoft Corporation. Todos los derechos reservados.
C:\WINDOWS\system32>cd ..
C:\Windows>cd ..
C:\>cd PA1
C:\PA1>java SSocket.java
Waiting for connection...
The right window, titled 'Administrador: Símbolo del sistema - java ClientSocket.java', shows the following commands and output:
Microsoft Windows [Versión 10.0.19044.1586]
(c) Microsoft Corporation. Todos los derechos reservados.
C:\WINDOWS\system32>cd ..
C:\Windows>cd ..
C:\>cd PA1
C:\PA1>java ClientSocket.java
An 'Input' dialog box is overlaid on the right window, with the title 'Input' and a close button. It contains a green question mark icon, the text 'Enter User Name:', a text input field, and 'OK' and 'Cancel' buttons.




```
// perpetually listen for clients
SSLServerSocketFactory socketFactory = (SSLServerSocketFactory)
SSLServerSocketFactory.getDefault();
ServerSocket = (ServerSocket) socketFactory.createServerSocket(7070);
```

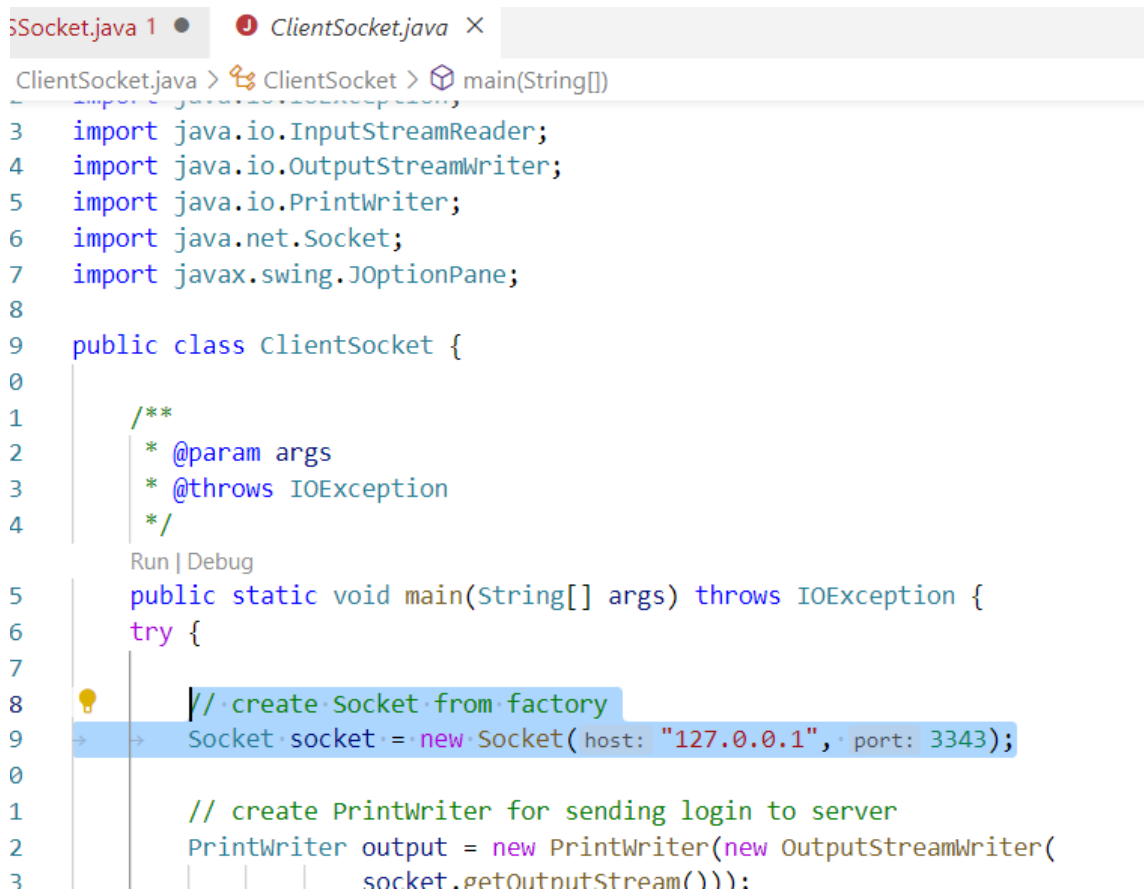
¿Qué ocurre? .Hay algunos fallos que habrá que cambiar.

Primero importamos SSLServerSocketFactory y SSLServerSocket . Luego creamos una variable local y cambiamos serverSocket por sSocket que es como inicialmente lo tenía escrito el profesor. En la captura siguiente vemos que se ha cambiado todo correctamente y no hay ningún error.

```
// perpetually listen for clients
SSLServerSocketFactory socketFactory = (SSLServerSocketFactory)
SSLServerSocketFactory.getDefault();
SSLServerSocket sSocket = (SSLServerSocket) socketFactory.createServerSocket(7070);
```

2. Cambiamos los subrayado también en el cliente por:

```
SSLSocketFactory socketFactory = (SSLSocketFactory) SSLSocketFactory.getDefault();
SSLSocket socket = (SSLSocket) socketFactory.createSocket("localhost", 7070);
```



```
ClientSocket.java 1 ClientSocket.java X
ClientSocket.java > ClientSocket > main(String[])
import java.io.*;
3 import java.io.InputStreamReader;
4 import java.io.OutputStreamWriter;
5 import java.io.PrintWriter;
6 import java.net.Socket;
7 import javax.swing.JOptionPane;
8
9 public class ClientSocket {
10
11     /**
12      * @param args
13      * @throws IOException
14      */
15     Run | Debug
16     public static void main(String[] args) throws IOException {
17         try {
18             // create Socket from factory
19             Socket socket = new Socket( host: "127.0.0.1", port: 3343);
20
21             // create PrintWriter for sending login to server
22             PrintWriter output = new PrintWriter(new OutputStreamWriter(
23                 socket.getOutputStream()));
```

Nos sale también errores, lo resolvemos importando SSLSocket y SSLSocketFactory

```
// create Socket from factory
SSLSocketFactory socketFactory = (SSLSocketFactory) SSLSocketFactory.getDefault();
SSLSocket socket = (SSLSocket) socketFactory.createSocket("localhost", 7070);
```

```
// create Socket from factory
SSLConnectionFactory socketFactory = (SSLConnectionFactory) SSLConnectionFactory.getDefault();
SSLSocket socket = (SSLSocket) socketFactory.createSocket("localhost", 7070);
```

- Una vez que ya hemos cambiado el servidor y el cliente lo inicializamos. Abrimos una ventana cmd para el servidor y pondremos:

```
C:\PA1>
C:\PA1>java SSocket.java
Waiting for connection...
```

Y otra ventana para el cliente:

```
Administrador: Símbolo del sistema
Microsoft Windows [Versión 10.0.19044.1586]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\WINDOWS\system32>cd ..
C:\Windows>cd ..
C:\>cd PA1

C:\PA1>java ClientSocket.java
java.net.SocketException: Socket is closed
    at java.base/sun.security.ssl.SSLSocketImpl.getInputStream(SSLSocketImpl.java:883)
    at ClientSocket.main(ClientSocket.java:46)
    at java.base/jdk.internal.reflect.DirectMethodHandleAccessor.invoke(DirectMethodHandleAccessor.java:104)
    at java.base/java.lang.reflect.Method.invoke(Method.java:577)
    at jdk.compiler/com.sun.tools.javac.launcher.Main.execute(Main.java:421)
    at jdk.compiler/com.sun.tools.javac.launcher.Main.run(Main.java:192)
    at jdk.compiler/com.sun.tools.javac.launcher.Main.main(Main.java:132)
```

Al escribir java CClientSocket.java, que podremos escribirlo o simplemente dándole al tabulador se escribe solo, veremos cómo nos sale error en el cliente (captura anterior) y error en el servidor (captura siguiente)

```
C:\PA1>
C:\PA1>java SSocket.java
Waiting for connection...
javax.net.ssl.SSLHandshakeException: No available authentication scheme
    at java.base/sun.security.ssl.Alert.createSSLException(Alert.java:131)
    at java.base/sun.security.ssl.Alert.createSSLException(Alert.java:117)
    at java.base/sun.security.ssl.TransportContext.fatal(TransportContext.java:358)
    at java.base/sun.security.ssl.TransportContext.fatal(TransportContext.java:314)
    at java.base/sun.security.ssl.TransportContext.fatal(TransportContext.java:305)
    at java.base/sun.security.ssl.CertificateMessage$T13CertificateProducer.onProduceCertificate(CertificateMessage.java:972)
    at java.base/sun.security.ssl.CertificateMessage$T13CertificateProducer.produce(CertificateMessage.java:961)
    at java.base/sun.security.ssl.SSLHandshake.produce(SSLHandshake.java:440)
    at java.base/sun.security.ssl.ClientHello$T13ClientHelloConsumer.goServerHello(ClientHello.java:1246)
    at java.base/sun.security.ssl.ClientHello$T13ClientHelloConsumer.consume(ClientHello.java:1182)
    at java.base/sun.security.ssl.ClientHello$ClientHelloConsumer.onClientHello(ClientHello.java:840)
    at java.base/sun.security.ssl.ClientHello$ClientHelloConsumer.consume(ClientHello.java:801)
    at java.base/sun.security.ssl.SSLHandshake.consume(SSLHandshake.java:396)
    at java.base/sun.security.ssl.HandshakeContext.dispatch(HandshakeContext.java:480)
    at java.base/sun.security.ssl.HandshakeContext.dispatch(HandshakeContext.java:458)
    at java.base/sun.security.ssl.TransportContext.dispatch(TransportContext.java:201)
    at java.base/sun.security.ssl.SSLTransport.decode(SSLTransport.java:172)
    at java.base/sun.security.ssl.SSLSocketImpl.decode(SSLSocketImpl.java:1500)
    at java.base/sun.security.ssl.SSLSocketImpl.readHandshakeRecord(SSLSocketImpl.java:1415)
    at java.base/sun.security.ssl.SSLSocketImpl.startHandshake(SSLSocketImpl.java:450)
    at java.base/sun.security.ssl.SSLSocketImpl.ensureNegotiated(SSLSocketImpl.java:915)
    at java.base/sun.security.ssl.SSLSocketImpl$AppInputStream.read(SSLSocketImpl.java:1006)
    at java.base/sun.nio.cs.StreamDecoder.readBytes(StreamDecoder.java:270)
    at java.base/sun.nio.cs.StreamDecoder.implRead(StreamDecoder.java:313)
    at java.base/sun.nio.cs.StreamDecoder.read(StreamDecoder.java:188)
    at java.base/java.io.InputStreamReader.read(InputStreamReader.java:176)
    at java.base/java.io.BufferedReader.fill(BufferedReader.java:162)
    at java.base/java.io.BufferedReader.readLine(BufferedReader.java:329)
    at java.base/java.io.BufferedReader.readLine(BufferedReader.java:396)
    at SSocket.main(SSocket.java:43)
    at java.base/jdk.internal.reflect.DirectMethodHandleAccessor.invoke(DirectMethodHandleAccessor.java:104)
    at java.base/java.lang.reflect.Method.invoke(Method.java:577)
    at jdk.compiler/com.sun.tools.javac.launcher.Main.execute(Main.java:421)
    at jdk.compiler/com.sun.tools.javac.launcher.Main.run(Main.java:192)
    at jdk.compiler/com.sun.tools.javac.launcher.Main.main(Main.java:132)
```

Este error es bueno, es decir, estamos utilizando una comunicación segura pero no se entienden el servidor y el cliente, no está configurado correctamente.

Un ejemplo sería que el servidor ya le ha abierto la puerta al cliente pero el servidor se comunica en italiano y el cliente se comunica en español. Entre ellos no se entienden, entonces tendrá que hablar en un idioma común como es el inglés. Pues eso mismo es lo que vamos a hacer a continuación.

- **Compilar y ejecutar código cliente y servidor desde consola**

1. Creamos un archivo.class (con el código interpretable por la JVM de Java) en la consola con permiso de administración siguiendo el siguiente comando:

```
javac BYODServer.java
```

que en este caso cambiaremos BYODServer por nuestra clase servidor SSocket. Todo lo haremos desde nuestro directorio.

2. Abrimos un blog de notas y escribimos :

```
java -Djavax.net.ssl.keyStore=C:\SSLStore -
```

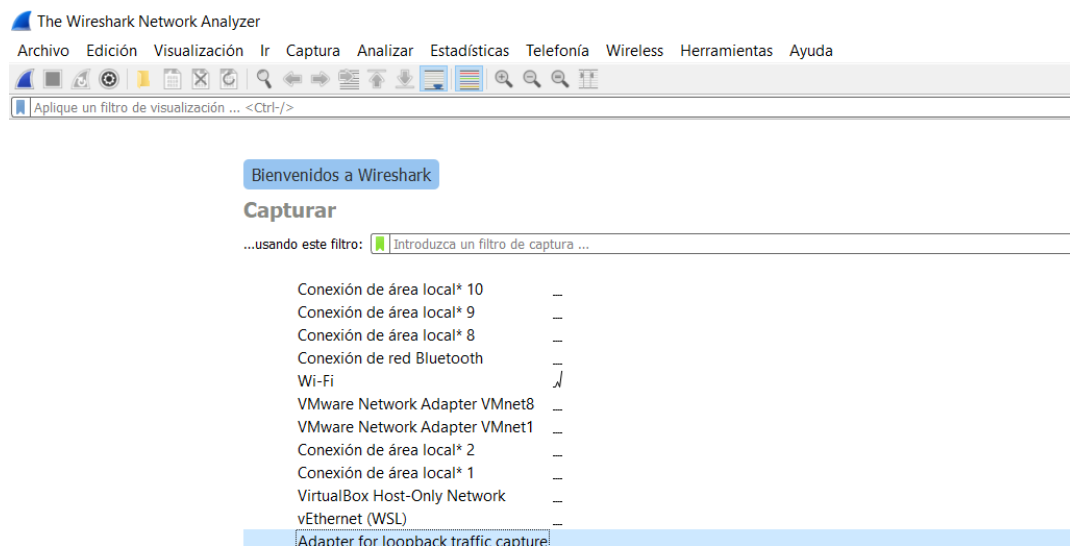
```
Djavax.net.ssl.keyStorePassword=PASSWORD BYODServer
```

```
java -Djavax.net.ssl.trustStore=C:\SSLStore -Djavax.net.ssl.
```

```
trustStorePassword=PASSWORD BYODCliente
```

Cambiamos el nombre de nuestra clase. BYODServer por SSocket , BYODCliente por ClientSocket. Nuestra contraseña y SSLStore por PA1.

3. En el siguiente paso nos descargaremos wireshark y capturamos tráfico entre cliente y servidor. Wireshark es un sniffers , es decir se interponen en los canales de comunicación entre el servidor y cliente y capturan toda la información que navega por ellos. Seleccionamos la búsqueda en “Adapter for loopback traffic capture” (es una red para realizar conexiones entre los Hosts)



- Le damos al símbolo de aleta de tiburón para capturar tráfico. Inicializamos primero el servidor y después el cliente mediante cmd como ya hemos hecho anteriormente. Una vez que ya hemos puesto el usuario y la contraseña, paramos el tráfico dándole al cuadrado rojo de la barra superior y veremos lo siguiente.

Adapter for loopback traffic capture

Archivo Edición Visualización Ir Captura Analizar Estadísticas Telefonía Wireless Herramientas Ayuda

Aplicar un filtro de visualización: <Ctrl>/>

No.	Time	Source	Destination	Protocol	Length	Info
9	46.695961	127.0.0.1	127.0.0.1	TLSv1.3	50	Change Cipher Spec
10	46.696001	127.0.0.1	127.0.0.1	TCP	44	7070 → 58844 [ACK] Seq=128 Ack=459 Win=2619648 Len=0
11	46.697183	127.0.0.1	127.0.0.1	TLSv1.3	50	Change Cipher Spec
12	46.697251	127.0.0.1	127.0.0.1	TCP	44	58844 → 7070 [ACK] Seq=459 Ack=134 Win=2619648 Len=0
13	46.702541	127.0.0.1	127.0.0.1	TLSv1.3	114	Application Data
14	46.702610	127.0.0.1	127.0.0.1	TCP	44	58844 → 7070 [ACK] Seq=459 Ack=204 Win=2619392 Len=0
15	46.707022	127.0.0.1	127.0.0.1	TLSv1.3	84	Application Data
16	46.707080	127.0.0.1	127.0.0.1	TCP	44	58844 → 7070 [ACK] Seq=459 Ack=244 Win=2619392 Len=0
17	46.708748	127.0.0.1	127.0.0.1	TCP	44	7070 → 58844 [FIN, ACK] Seq=244 Ack=459 Win=2619648 Len=0
18	46.708797	127.0.0.1	127.0.0.1	TCP	44	58844 → 7070 [ACK] Seq=459 Ack=245 Win=2619392 Len=0
19	46.710147	127.0.0.1	127.0.0.1	TCP	44	58844 → 7070 [FIN, ACK] Seq=459 Ack=245 Win=2619392 Len=0
20	46.710246	127.0.0.1	127.0.0.1	TCP	44	7070 → 58844 [ACK] Seq=245 Ack=460 Win=2619648 Len=0

> Frame 1: 128 bytes on wire (1024 bits), 128 bytes captured (1024 bits) on interface \Device\NPF_{...}, id 0

> Null/Loopback

> Internet Protocol Version 4, Src: 192.168.222.1, Dst: 192.168.222.1

> Internet Control Message Protocol

> NetBIOS Name Service

```

0000 02 00 00 00 45 00 00 7c 02 27 00 00 00 01 00 00  ....E...
0010 c0 a8 de 01 c0 a8 de 01 03 01 e2 0b 00 00 00 00  E..U...
0020 45 00 00 00 55 de 00 00 80 11 00 00 c0 a8 de 01  E..U...
0030 c0 a8 de 02 00 00 00 00 00 4c 9c 1a 9d a3 60 00  ....L...
0040 00 01 00 00 00 00 00 01 20 46 48 45 50 46 43 45  ....FH...
0050 4c 45 48 46 43 45 50 46 46 46 41 43 41 43 41 43  LEHFCPEP FFACACAC
0060 41 43 41 43 41 43 41 41 41 00 00 20 00 01 c0 0c  ACACACAA A...
0070 00 20 00 01 00 04 93 c0 00 06 e0 00 c0 a8 de 01  ....

```

Como vemos ha capturado muchos paquetes que pasan por la interfaz que hemos seleccionado. Para que nos salga solo en la pantalla aquellos que utilizan el protocolo TCP añadimos un filtro en la barra superior y ponemos tcp.stream eq 0 .

Adapter for loopback traffic capture

Archivo Edición Visualización Ir Captura Analizar Estadísticas Telefonía Wireless Herramientas Ayuda

Aplicar un filtro de visualización: tcp.stream eq 0

No.	Time	Source	Destination	Protocol	Length	Info
2	39.997052	127.0.0.1	127.0.0.1	TCP	56	58844 → 7070 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM=1
3	39.997261	127.0.0.1	127.0.0.1	TCP	56	7070 → 58844 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM=1
4	39.997367	127.0.0.1	127.0.0.1	TCP	44	58844 → 7070 [ACK] Seq=1 Ack=1 Win=2619648 Len=0
5	46.624063	127.0.0.1	127.0.0.1	TLSv1.3	496	Client Hello
6	46.624138	127.0.0.1	127.0.0.1	TCP	44	7070 → 58844 [ACK]
7	46.675211	127.0.0.1	127.0.0.1	TLSv1.3	171	Server Hello
8	46.675283	127.0.0.1	127.0.0.1	TCP	44	58844 → 7070 [ACK]
9	46.695961	127.0.0.1	127.0.0.1	TLSv1.3	50	Change Cipher Spec
10	46.696001	127.0.0.1	127.0.0.1	TCP	44	7070 → 58844 [ACK]
11	46.697183	127.0.0.1	127.0.0.1	TLSv1.3	50	Change Cipher Spec
12	46.697251	127.0.0.1	127.0.0.1	TCP	44	58844 → 7070 [ACK]
13	46.702541	127.0.0.1	127.0.0.1	TLSv1.3	114	Application Data

> Frame 5: 496 bytes on wire (3968 bits), 496 bytes captured (3968 bits) on interface \Device\NPF_{...}, id 0

> Null/Loopback

> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1

> Transmission Control Protocol, Src Port: 58844, Dst Port: 7070, Seq: 1, Ack: 1, Len: 452

> Transport Layer Security

Marcar/Desmarcar paquete Control+M

Ignorar/No ignorar paquete Control+D

Establecer/Anular referencia de tiempo Control+T

Modificar horario... Control+Mayúsculas+T

Comentarios de paquete

Editar nombre resuelto

Aplicar como filtro

Preparar as Filter

Filtro de conversación

Colorar conversación

SCTP

Seguir

Copiar

Protocol Preferences

Decode As...

Mostrar paquete en nueva ventana

Flujo TCP Control+Alt+Mayúsculas+T

Flujo UDP Control+Alt+Mayúsculas+U

DCCP Stream Control+Alt+Mayúsculas+E

Flujo TLS Control+Alt+Mayúsculas+S

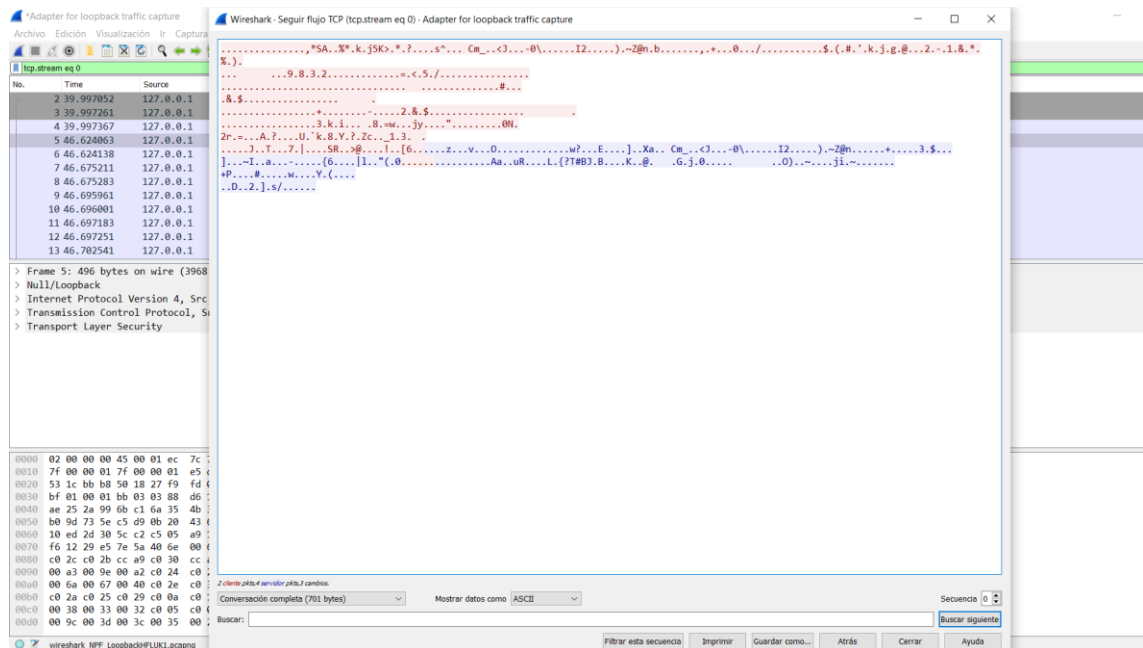
Flujo HTTP Control+Alt+Mayúsculas+H

Flujo HTTP/2

Flujo QUIC

SIP Call

Como podemos comprobar en la captura anterior el TLS utilizado es TLSv1.3. Hacemos click derecho a la conexión seleccionada (Client Hello) y le damos a seguir flujo TCP.



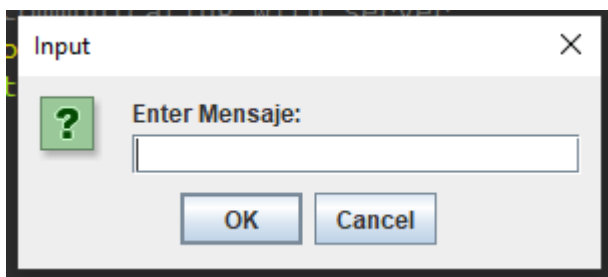
¿Qué quiere decir la captura anterior?

Lo que demuestra es que el canal es seguro. No podemos ver ni obtener el nombre de usuario ni la contraseña ... no podemos ver nada porque la información está cifrada. Como se pueden comunicar entre ellos utilizan el mismo keyStore en Cliente y Servidor, si se utilizara dos máquinas distintas estarían también en el Servidor y en el Cliente.

- **Comprobación del servidor VPN para soportar a los 300 empleados del hospital de forma concurrente.**

Creación de una pantalla de mensaje:

```
String mensaje = JOptionPane.showInputDialog(null,
    "Enter Mensaje:");
output.println(mensaje);
```



Algunas de las soluciones que hemos pensado han sido:

Bucle for:

Queríamos realizar un bucle que enviara 10 usuarios al servidor a la vez.

```
public static void main(String[] args) throws IOException {
    int i=1;
    for(i=1; i<=10;i++) {
    try {

        // create Socket from factory
        SSLSocketFactory socketFactory = (SSLSocketFactory) SSLSocketFactory.getDefault();
        SSLSocket socket = (SSLSocket) socketFactory.createSocket("localhost", 7070);

        //Socket socket = new Socket("127.0.0.1", 7070);

        // create PrintWriter for sending login to server

        PrintWriter output = new PrintWriter(new OutputStreamWriter(
            socket.getOutputStream()));

        // prompt user for user name
        String userName = JOptionPane.showInputDialog(null,
            "Enter User Name:");

        // send user name to server
        output.println(userName);
    }
    }
```

Bucle while:

Al igual, que con el bucle for, mientras que se cumpliera la condición, se enviaran 10 usuarios a la vez al servidor.

```
public static void main(String[] args) throws IOException {
    int i=1;
    while(i<=10) {
    try {

        // create Socket from factory
        SSLSocketFactory socketFactory = (SSLSocketFactory) SSLSocketFactory.getDefault();
        SSLSocket socket = (SSLSocket) socketFactory.createSocket("localhost", 7070);

        //Socket socket = new Socket("127.0.0.1", 7070);

        // create PrintWriter for sending login to server

        PrintWriter output = new PrintWriter(new OutputStreamWriter(
            socket.getOutputStream()));

        // prompt user for user name
        String userName = JOptionPane.showInputDialog(null,
            "Enter User Name:");

        // send user name to server
        output.println(userName);
    }
    }
```

Creación de una segunda clase con un bucle for para ejecutar la clase ClientSocket:

Con nuestra clase ClientSocket creada, con una segunda clase ClientesSocket creamos un bucle for para ejecutar la clase al menos 100 veces, para que así se envíen 100 clientes al servidor.


```
public static void main(String[] args) throws IOException {
    try {

        // create Socket from factory
        SSLSocketFactory socketFactory = (SSLSocketFactory) SSLSocketFactory.getDefault();
        SSLSocket socket = (SSLSocket) socketFactory.createSocket("localhost", 7070);
        //Socket socket = new Socket("127.0.0.1", 7070);

        // create PrintWriter for sending login to server

        PrintWriter output = new PrintWriter(new OutputStreamWriter(
            socket.getOutputStream()));

        // prompt user for user name
        String userName = JOptionPane.showInputDialog(null,
            "Enter User Name:");

        // send user name to server
        output.println(userName);

        // prompt user for password
        String password = JOptionPane.showInputDialog(null,
            "Enter Password:");
        output.println(password);
        // prompt user for password
        String mensaje = JOptionPane.showInputDialog(null,
            "Enter Mensaje:");
        output.println(mensaje);
        /// send password to server
    }
}
```

```
public class ClientesSocket {

    public static void main(String[] args) throws IOException {
        try {

            // create Socket from factory
            SSLSocketFactory socketFactory = (SSLSocketFactory) SSLSocketFactory.getDefault();
            SSLSocket socket = (SSLSocket) socketFactory.createSocket("localhost", 7070);

            for (int i = 0; i < 100; i++)
            {
                ClientSocket.writeUTF("Este es el cliente número " + (i+1) + "\n");
            }

            socket.close();
        }
    }
}
```

```
Waiting for connection...
javax.net.ssl.SSLHandshakeException: Remote host terminated the handshake
    at java.base/sun.security.ssl.SSLSocketImpl.handleEOF(SSLSocketImpl.java:1661)
    at java.base/sun.security.ssl.SSLSocketImpl.decode(SSLSocketImpl.java:1470)
    at java.base/sun.security.ssl.SSLSocketImpl.readHandshakeRecord(SSLSocketImpl.java:1370)
    at java.base/sun.security.ssl.SSLSocketImpl.startHandshake(SSLSocketImpl.java:437)
    at java.base/sun.security.ssl.SSLSocketImpl.ensureNegotiated(SSLSocketImpl.java:878)
    at java.base/sun.security.ssl.SSLSocketImpl$AppInputStream.read(SSLSocketImpl.java:969)
    at java.base/sun.nio.cs.StreamDecoder.readBytes(StreamDecoder.java:297)
    at java.base/sun.nio.cs.StreamDecoder.implRead(StreamDecoder.java:339)
    at java.base/sun.nio.cs.StreamDecoder.read(StreamDecoder.java:188)
    at java.base/java.io.InputStreamReader.read(InputStreamReader.java:181)
    at java.base/java.io.BufferedReader.fill(BufferedReader.java:161)
    at java.base/java.io.BufferedReader.readLine(BufferedReader.java:326)
    at java.base/java.io.BufferedReader.readLine(BufferedReader.java:392)
    at SSocket.main(SSocket.java:40)
Caused by: java.io.EOFException: SSL peer shut down incorrectly
```

Con hilos

Otra idea sería crear un hilo, en otra clase “Con hilos”

```
String atributo;  
public ConHilos(int i)  
{  
    atributo = "clientes" + i;  
}  
  
public static void main(String[] args)  
{  
    for (int i = 0; i < 4; i++)  
    {  
        Thread socket = new Thread(new ConHilos(i));  
        socket.start();  
    }  
}
```

Poner el bucle for en el servidor:

La última opción era crear un bucle for en el servidor, con eso conseguimos que en una misma conexión el servidor soportase i clientes, pero de uno en uno, no todos a la vez. Así que tuvimos que desear esa idea.

```
public class SSocket {  
    public static void main(String[] args) throws IOException,  
        InterruptedException {  
  
        SSLServerSocketFactory socketFactory = (SSLServerSocketFactory)  
            SSLServerSocketFactory.getDefault();  
        SSLServerSocket sSocket = (SSLServerSocket) socketFactory.createServerSocket(7070);  
        // perpetually listen for clients  
        //ServerSocket sSocket = new ServerSocket(3343);  
        while (true) {  
            int i;  
            // wait for client connection and check login information  
            try {  
                for (i=0;i<=10;i++) {  
                    System.err.println("Waiting for connectionn...");  
                    Socket socket = sSocket.accept();  
                    System.err.println("Sirvo al cliente: " + i );  
                    socket.close();  
                }  
            }  
        }  
    }  
}
```

```
C:\PA1>java -Djavax.net.ssl.keyStore=.\SSLStore -Djavax.net.ssl.keyStorePassword=SeguridaD SSocket  
Waiting for connectionn...  
Sirvo al cliente 0  
Sirvo al cliente 1  
Sirvo al cliente 2  
Sirvo al cliente 3  
Sirvo al cliente 4  
Sirvo al cliente 5  
Sirvo al cliente 6  
Sirvo al cliente 7  
Sirvo al cliente 8  
Sirvo al cliente 9  
Waiting for connectionn...  
Sirvo al cliente 0  
Sirvo al cliente 1
```