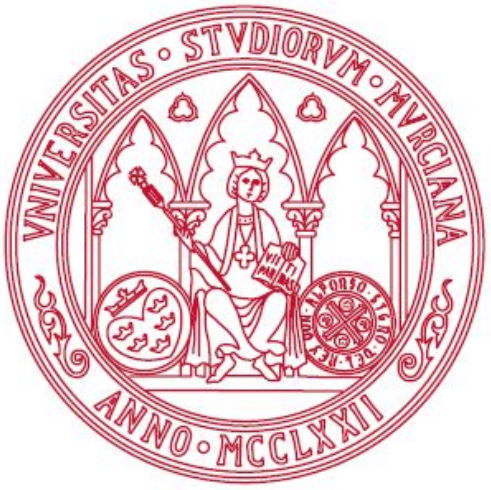

Universidad de Murcia

Grado en Ingeniería Informática

Curso 2020/2021

Programación para las comunicaciones

Profesor: Humberto Martínez Barberá

Práctica IV

Jaime Ortiz Aragón - 49196689B - jaime.ortiza@um.es

- **ESCENARIO**

El objetivo de esta práctica es mantener la funcionalidad del carrito de compra de la práctica 1, con los recursos pedidos al cliente. Pero en este caso, vamos a securizar el acceso al cliente web por medio de SSL. Para realizar esto, es necesario hacer uso de una autoridad de certificación (CA), que certifique tanto a cliente como servidor. Esta CA tendrá que ponerse como confiable por el cliente para que la comunicación esté realmente protegida.

- **CÓDIGO JAVA**

En cuanto a la implementación de código en Java, se cuenta con el mismo diagrama de clases con el que contábamos para la primera práctica. Los cambios a adoptar han sido los nuevos objetos *Socket* y *ServerSocket*. En este caso, al tratarse de comunicación bajo SSL, estos objetos deben ser *SSLSocket* y *SSLServerSocket*. Estas clases son proporcionadas por la librería *Java Secure Socket Extension* (JSSE). Para la implementación de los sockets seguros es necesario hacer uso de las factorías *SSLSocketFactory* y *SSLServerSocketFactory*. Estas obtienen una instancia de su clase con las que se inicializan los sockets.

El mayor de los cambios viene por cómo obtiene el objeto ServerSocket. Para esto se hace uso del método *getServer()*. Aquí obtenemos la instancia del keystore en el que se encuentran las claves. El método de seguridad concretamente es TLS, que para inicializarse en Java su contexto con el cual obtenemos la factoría necesita además del manager del keystore, el manager de la autoridad de certificación. Este último se obtiene a través del certificado confiable de la CA, *cacert.pem*.

- **GENERACIÓN DE CERTIFICADOS**

Para la generación de los certificados, se han seguido los pasos indicados en el pdf de *openssl*. Con *mkdir* se crea el directorio demoCA y dentro de él newcerts. Los certificados que se irán generando serán a partir del índice indicado en el fichero *serial*. De este modo, el primer certificado que se cree lo encontraremos en newcerts como *01.pem* y se incrementará en 1 en índice.

El primer paso a realizar es la generación de las claves de la autoridad de certificación, con el comando *openssl req -x509 -newkey rsa:2048 -keyout cakey.pem -days 3650 -out cacert.pem*, e introduciendo el DN : cn=ca,o=umu, ou=ppc, c=es. Para que no dé problemas al generar las demás claves, he cambiado en el fichero de configuración de openssl las políticas para que los campos que se dejan en blanco no hagan match, ya que por defecto al generarlos se crean con valores distintos.

```
stateOrProvinceName     = optional
localityName            = optional
```

A continuación, se crea una clave privada, que se almacena en el keystore con el alias servidor. Finalmente, hay que generar el proceso de firma de los certificados. El primer paso es crear la solicitud, la cual se hace con el comando *keytool -certreq -file servidor.csr*

-keystore servidor.ks. En caso del cliente, la solicitud será *cliente.csr* y el almacén generado habrá sido *cliente.ks*. A continuación, firmamos el la solicitud con la CA, mediante el comando *openssl ca -keyfile cakey.pem -in servidor.csr -out servidor.pem*. Este paso es igual para el cliente.

```
Certificate is to be certified until Dec  2 16:34:28 2021 GMT (365 days)
Sign the certificate? [y/n]:y
```

Importamos al keystore el certificado de la CA como confiable mediante el comando *keytool -import -trustcacerts -alias CA -file cacert.pem -keystore servidor.ks* a ambos almacenes.

```
¿Confiar en este certificado? [no]:  si
Se ha agregado el certificado al almacén de claves
imjaimeortiz@imjaimeortiz-msi:~/demoCA$
```

Una vez hecho esto, también incluimos en el almacén el certificado que se ha solicitado.
Finalmente, sólo nos queda configurar la confianza en el cliente de navegador. Para esto introducimos como confiable el certificado de la CA, para que lo tome como una autoridad de certificación confiable.
Para que se confíe en el cliente que intenta acceder, se ha generado el PKCS12 con el software de *portecle*. Este archivo .p12 se introduce al navegador en el apartado de mis certificados. Cuando se realice una petición nos pedirá lo siguiente para confiar y finalmente podremos obtener la funcionalidad que teníamos sobre HTTP, pero con la seguridad que ofrece TLS.