Tarea Tema 5. PSP

Contenido

1 Introducción

En esta tarea entregaremos un ejercicio realizado como **obligatorio**, para empezar. **Realizaremos un CHAT TCP aplicando Sockets SSL** siguiendo el **patrón de diseño observer**

2 Duración

4 horas.

3 Objetivos

Utilización de seguridad y encriptación en nuestras aplicaciones Java

4 Tarea.

Realizaremos la Actividad 5.3 del libro para **manejo de sistemas criptográficos y claves**

Construimos un **CHAT TCP usando el patrón Observer** y la **tecnología de SSLSockets con TSL**. El **resultado lo vamos definiendo** en los siguientes apartados.

4.1 Primera parte

En esta **parte entregareis el un proyecto con la actividad 5.3 del libro más el Ejemplo 8 y el Ejemplo 9 funcionando correctamente**. El objetivo de esta parte **es crear y almacenar una clave privada y publica para aplicar el algoritmo DSASHA256** en **la firma electrónica**. Firmaremos un **Fichero Fichero.DAT en ejemplo 8 y en ejemplo 9 comprobaremos que el fichero enviado firmado es correcto** y su firma es correcta.

4.1.1 Entrega de parte I

Se **incluirá todo en una carpeta Parte I**. Junto **a la carpeta Parte II** se incluirá en un zip **TareaTema5NombreYApellidos**.zip. Ver el **apartado de entrega de la tarea II**.

Adjuntais un PDF con los pantallazos demostrando que funciona.

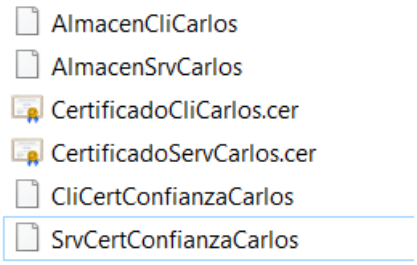
El nombre del proyecto será **ProyectoParteINombreApellidos**

El nombre del PDF será **ProyectoParteINombreApellidos.pdf**

4.2 Segunda parte

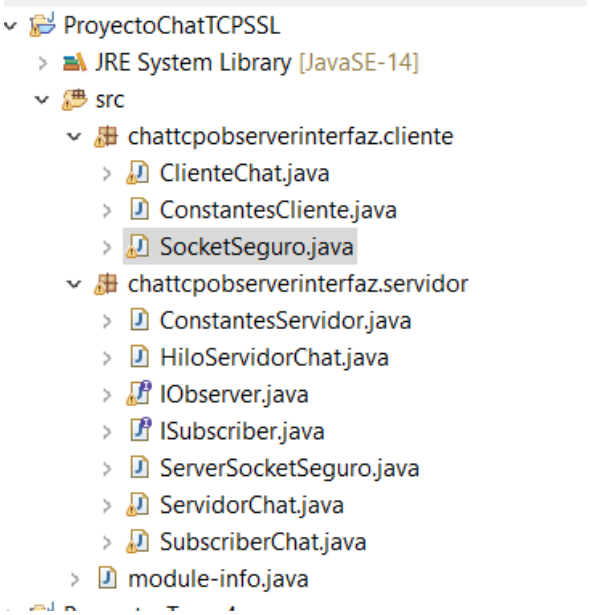
4.2.1 Creamos un Chat TCP con Sockets SSL

Usaremos los almacenes para realizar el handshake entre **el socket cliente con el Servidor**. Todos **los clientes usaran el mismo Certificado y almacen Trust**. Elegid **vosotros el nombre de los almacenes y de los certificados**. Debéis **entregarlos junto al proyecto**. Colocadlos en el **directorio c:\tarea5** . En total son 4 Almacenes y 2 certificados como podéis ver en la figura



4.2.2 Estructura del proyecto

La estructura del proyecto será la siguiente:



La clase **SocketSeguro** ofrecerá un método estático para crear sockets cliente SSL llamado `creaSocketSeguro`.

La clase **ServerSocketSeguro** ofrecerá un método estático para crear sockets Servidor SSL llamado `creaSocketSeguro.`

HiloServerChat controlará la conexión entre el **servidor y cada cliente** subscrito al chat con un hilo **HiloForkJoin paralelo.** Comunicará **a SubscriberChat** de que ha llegado un nuevo mensaje por esa conexión.

Subscriber Chat se encarga de almacenar todas los subscriptores, y **notificará a todos cuando llegue un nuevo mensaje.**

El modelo **de dato ISubscriber el interfaz implementado por subscriberChat**

```
public interface ISubscriber {

public  void subscribirse(IObserver observer) throws Exception;

	public void eliminarSubscripcion(IObserver observer);


}
```

IObserver es el **interfaz implementado por HiloServidorChat**, que será **el nexo de unión entre el Cliente, el observer**, y **el Subscriber** en la **parte de servidor**.

```
public interface IObserver {

	public void modificarChat(String mensaje) ;


}
```

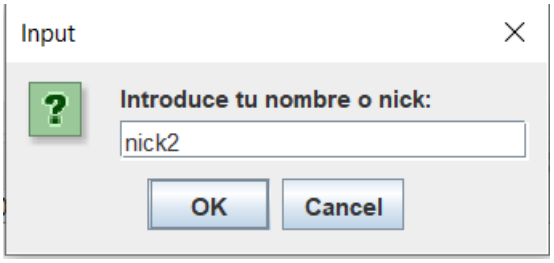
4.2.3 Funcionamiento

Controlaremos el número de subscripciones del **lado del servidor**, cada vez que un cliente se **subscriba o se quite del registro controlaremos el número de subscripciones**.

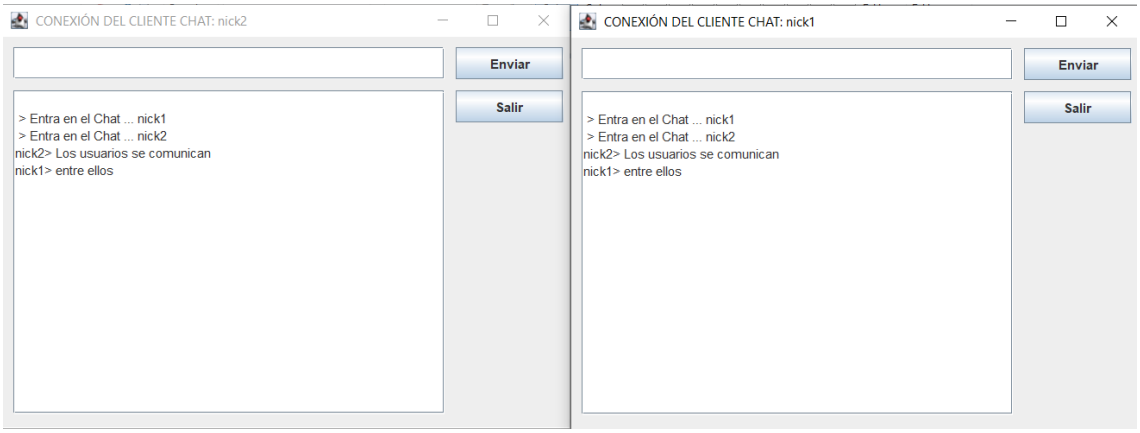
```
Servidor iniciado...
```

```
NUMERO DE Subscripciones ACTUALES: 1
NUMERO DE Subscripciones ACTUALES: 2
```

Del **lado del cliente , usuario del chat, al entrar introduciremos un Nick** que será nuestro **identificador de conexión**.



De esta manera varios clientes del chat se enviaran datos de unos a otros, usando la siguiente interfaz gráfica:



4.2.4 Entregaremos

Un pdf llamado **ParteIINombreyApellidos con los pantallazos mostrando** que **el programa funciona.**

El **proyecto preferiblemente con nombre ProyectoChatTCPSSLNombreyApellidos**.

Todo en una **carpeta Parte II.**

La **carpeta Parte I y Parte II se incluirá en un zip** llamado **TareaTema5NombreyApellidos**.