

Proyecto Final UT5



21-02-2020

Indice

UT5.-Proyecto Final	3
1.Creacion De Certificados	4
2.Cliente_SSL	5
2.Servidor_SSL	6
3.Main_Encriptar	7
Generador_RSA_claves.	7
Encriptar_archivo	8
Main	9
4.Main_Desencriptar	10
Desencriptar_archivo	10
Main	11
Pruebas de Ejecución	12
Cliente_SSL y Servidor_SSL	12
Main_Encriptar	15
Main_Desencriptar	16

UT5.-Proyecto Final

Está basado en el proyecto de la UT4, haciendo que la comunicación entre los clientes y el servidor sea segura, utilizando el protocolo SSL. Además, existirá la posibilidad de encriptar los ficheros generados en el servidor utilizando el cifrado asimétrico con el algoritmo RSA. La clave pública será guardada en un fichero para que pueda ser utilizada para desencriptar el fichero.

Consideraciones del proyecto:

- El proyecto se nombrará como UT5_ProyectoFinal_[NombreAlumnoApellido1]
- Toda la información generada y necesaria para la ejecución del proyecto, deberá estar ubicada dentro del mismo (certificados cliente y servidor para la comunicación SSL, clave pública, ficheros con la información de los clientes, fichero con los comandos keytool utilizados, ficheros encriptados y ficheros desencriptados)
- La ruta de la información a tener en cuenta será relativa al proyecto. Para ello, se podrá utilizar:

```
File dir=new File(".");  
String midir=dir.getCanonicalPath();
```

- Se creará un programa llamado **EncriptarFichero** que:

- Genere las claves públicas y privadas (RSA).
- Almacene la clave pública en un fichero llamado **ClavePublica** (que estará ubicado dentro del proyecto y será borrada si existía previamente)
- Pida el nombre del fichero a encriptar (en nuestro caso el generado en la conexión con los clientes, que estará ubicado dentro del proyecto llamado **fich1.txt**, **fich2.txt**, etc.)
- Genere un nuevo fichero encriptado con la clave privada que se llamará **encriptado[nombre_fichero]**. Ejemplo: **encriptadofich1.txt** (ubicado en el mismo lugar que el fichero origen que se borrará si existía previamente)
- Muestre por consola los mensajes de lo que va haciendo.

- Se creará un programa llamado **DesencriptarFichero** que:

- Obtenga la clave pública generada con el programa **EncriptarFichero** (hay que tener en cuenta que habrá una clave pública por cada fichero generado y que se supone se enviará al destinatario. En nuestro caso está ubicada en el mismo sitio, con el mismo nombre para verificar que funciona el proceso)
- Pida el nombre del fichero a desencriptar (un ejemplo sería **encriptadofich1.txt**)
- Genere el fichero desencriptado con la clave pública con el siguiente nombre: **desencriptado[nombrefichero]** (**desencriptadoencriptadofich1.txt** como ejemplo que se borrará si existía previamente)
- Muestre por consola los mensajes de lo que va haciendo.

- Se partirá de la documentación del proyecto UT4, añadiendo un apartado de seguridad en el que se explicarán las pruebas realizadas. Mejorar la documentación con lo que se estime oportuno.

PSP

1.Creacion De Certificados

Antes de todo tenemos que crear los Certificados correspondientes para nuestro proyecto.

Esto lo realizamos con ayuda de los comandos proporcionados.

Para poder crearlos primero ante todo tenemos que tener el PATH de nuestro Systema (Ordenador) la dirección del directorio bin de nuestro JDK de Java

```
(C) 2019 Microsoft Corporation. Todos los derechos reservados.  
C:\Users\klips>set path  
Path=C:\Oracle\18\bin;C:\Program Files (x86)\Common Files\Oracle\Java\javapath;C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Windows\System32\WindowsPowerShell\v1.0\;C:\Windows\System32\OpenSSH\;C:\Program Files (x86)\NVIDIA Corporation\PhysX\Common;C:\Program Files\Git\cmd;C:\Program Files\Microsoft SQL Server\130\Tools\Binn\;C:\Program Files\dotnet\;C:\Program Files\Microsoft SQL Server\Client SDK\ODBC\170\Tools\Binn\;C:\Program Files\Java\jdk1.8.0_231\bin;C:\Program Files\MySQL\MySQL Shell 8.0\bin\;C:\Users\klips\AppData\Local\Microsoft\WindowsApps;C:\Program Files\Java\jdk1.8.0_231\bin;  
PATHEXT=.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC  
C:\Users\klips>
```

Esto se puede realizar de la siguiente manera -set path:=%path%;[Direccion_ruta]

Después de incluirlo en el path, procedemos a ejecutar los siguientes comandos de forma ordenada.

EN EL SERVIDOR:

```
keytool -genkey -alias servidor -keyalg RSA -keystore AlmacenSrv -storepass 1234567
```

```
keytool -exportcert -alias servidor -keystore AlmacenSrv -storepass 1234567 -file CertificadoServ.cer
```

=====

EN EL CLIENTE:

```
keytool -importcert -trustcacerts -alias servidor -file CertificadoServ.cer -keystore CliCertConfianza -storepass 890123
```

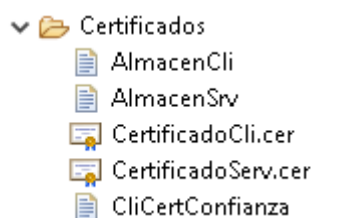
```
keytool -genkey -alias cliente -keyalg RSA -keystore AlmacenCli -storepass clavecli
```

```
keytool -exportcert -alias cliente -keystore AlmacenCli -storepass clavecli -file CertificadoCli.cer
```

Tendremos que rellenar nuestro certificados por CMD, debemos tener en cuenta que este proceso debe realizarse de forma correcta para que no nos de ningún problema en el proyecto.

Se debe insertar exactamente lo mismo para los dos.

El resultado debe ser exactamente el mismo:

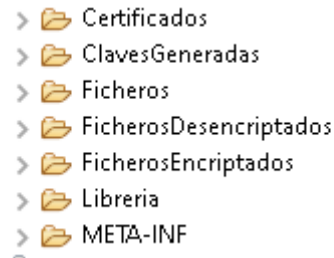


2.Cliente_SSL

Teniendo como base el proyecto anterior de la UT4 se a modificado para que utilice servidores SSL para que la comunicación sea segura.

Esto se realizara de la siguiente manera.

1.incluimos en nuestro proyecto una nueva carpeta con nuestros certificados, ademas añadimos otros directorios para los diferentes archivos que crearemos ,como pueden ser los archivos encriptados o desencriptados, dejándolo de esta manera.



2.Despues de incluir los certificados se a modificado nuestra Clase Cliente_SSL

```

0
1 //----- CAMBIOS PARA SSLsocket-----
2
3 //Socket Cliente = new Socket(Host, Puerto);
4
5 //variables a añadir.
6 //primero se opriene las certificaciones creadas previamente
7 String dir_cliente_certificado="\\.\\Certificados\\CliCertConfianza";
8 //La clave es la que se intridujo al general los certificados
9 String contraseña="890123";
10
11 System.out.println("Cliente Iniciado...");
12
13 //Establece la propiedad del sistema indicada por la clave especificada
14 System.setProperty("javax.net.ssl.trustStore",dir_cliente_certificado);
15 System.setProperty("javax.net.ssl.trustStorePassword", contraseña);
16
17 //creo un SSLSocketFactory llamada factory_ssl para mas tarde crear un SSLSocket
18 SSLSocketFactory factory_ssl = (SSLSocketFactory) SSLSocketFactory.getDefault();
19 SSLSocket cliente_ssl = (SSLSocket) factory_ssl.createSocket(Host, puerto);
20
21 //-----
22 //se crea un fuente de salida al servidor
23 PrintWriter fsalida = new PrintWriter (cliente_ssl.getOutputStream(), true);
24 //fuente de entrada al servidor
  
```

En primer lugar creo un variable de cadena de caracteres (String) que contenga la ruta de nuestro certificado llamado CliCertConfianza, ademas añadimos otra para nuestra contraseña.

Establecemos la propiedad del sistema para un comunicación segura.

```

4 System.setProperty("javax.net.ssl.trustStore",dir_cliente_certificado);
5 System.setProperty("javax.net.ssl.trustStorePassword", contraseña);
  
```

Se Genera un SSLSocketFactory que en este caso se llama factory_ssl , para despues crear un SSLSocket, pasando le el host y el puesto.

```

8 SSLSocketFactory factory_ssl = (SSLSocketFactory) SSLSocketFactory.getDefault();
9 SSLSocket cliente_ssl = (SSLSocket) factory_ssl.createSocket(Host, puerto);
10
  
```

2.Servidor_SSL

Los cambios que se realizaron al Servidor son muy similares a la del cliente, salgo en un par de puntos.

```

1 package mains;
2
3 import java.io.*;
4
5 public class Servidor_SSL {
6     public static void main(String args[]) throws IOException {
7         //ServerSocket servidor;
8         SSLSocket cliente_ssl = null;
9         int puerto = 5000;
10        String dir_servidor_certificado=".\\Certificados\\AlmacenSrv";
11        String contraseña_servidor="1234567";
12        //servidor = new ServerSocket(5000);
13        //----- CAMBIOS PARA SSLsocket-----
14        System.out.println("Servidor iniciado...");
15
16        System.setProperty("javax.net.ssl.keyStore", dir_servidor_certificado);
17        System.setProperty("javax.net.ssl.keyStorePassword", contraseña_servidor);
18
19        SSLServerSocketFactory factory = (SSLServerSocketFactory) SSLServerSocketFactory.getDefault();
20        SSLServerSocket servidor_ssl = (SSLServerSocket) factory.createServerSocket(puerto);
21
22        //----- CAMBIOS PARA SSLsocket-----
23        int i=0;
24
25        while (true) {
26            i++;
27            cliente_ssl = (SSLSocket) servidor_ssl.accept();
28            HiloServidor hilo = new HiloServidor(cliente_ssl);
29            hilo.run(i);
30
31        } //while
32    } //Main

```

El Servidor utiliza el archivo llamado AlmacenSrv que se encuentra en nuestro directorio de certificados

Cambio:

En primer lugar creamos dos variables para nuestra contraseña y la ruta de AlmacenSrv.

Le damos al Sistema la propiedad de una comunicación segura.

```

21        System.setProperty("javax.net.ssl.keyStore", dir_servidor_certificado);
22        System.setProperty("javax.net.ssl.keyStorePassword", contraseña_servidor);
23
24

```

Y generamos el SSL del servidor de la misma manera que en cliente salvo que en esta ocasión solo necesitamos el puerto.

```

SSLServerSocketFactory factory = (SSLServerSocketFactory) SSLServerSocketFactory.getDefault();
SSLServerSocket servidor_ssl = (SSLServerSocket) factory.createServerSocket(puerto);

```

Ademas incluimos en método accept() dentro de el while esperando la comunicación con un cliente.

```

        while (true) {
            i++;
            cliente_ssl = (SSLSocket) servidor_ssl.accept();
            HiloServidor hilo = new HiloServidor(cliente_ssl);
            hilo.run(i);

        } //while

```

PSP

A partir de este punto no se realizado mas cambios el programa base, ejecutándose de la misma manera que antes de un adaptación a SSL.

Siguiendo el siguiente orden.

Del while pasa al HiloServidor en donde se comprobara asta que se le envíe un (*) que el cliente introduzca por comandos (*), todo lo anterior sera guardado en un fichero x para después subirlo a <https://files.000webhost.com/> y enviarlo a nuestro email.

3.Main_Encriptar Generador_RSA_claves.

Antes de todo necesitamos una clase que nos ayude a generar el par de claves y guardar la publica, en este caso se la a llamado Generador_RSA_claves.

```
public class Generador_RSA_claves {  
  
    private KeyPair pares_claves_rsa;  
    String dir_claves=".\\ClavesGeneradas\\";  
    static String nuevo_nombre;
```

Para poder generar nuestras par de claves se a utilizado KeyPairGenerator de dentro del constructor publico de esta manera.

```
    //constructor de clase para claves RSA  
    public Generador_RSA_claves() {  
        try {  
            //de generan par de claves  
            KeyPairGenerator keyPairGenerator = KeyPairGenerator.getInstance("RSA");  
            pares_claves_rsa = keyPairGenerator.generateKeyPair();  
            System.out.println("Se han generado el par de claves");  
        } catch (Exception e) {  
            System.out.println("Error al generar las claves");  
            e.printStackTrace();  
        }  
    }  
}
```

También necesitamos un método get para poder recuperar nuestras claves generadas.

```
    //Método get para saber las claves RSA tanto publica como privada.  
    public KeyPair getClaves() {  
        return pares_claves_rsa;  
    }  
}
```

Para poder guardar la clave publica de a creado un método que tiene un parámetro de tipo String que se utiliza para dar nombre a nuestro archivo a crear.

Lo que realiza el método es recuperar la clave publica y escribirlo en un fichero.

Antes de esto se comprueba que exista y si existe se borra para dar paso al nuevo.

```
    //Metodo para guardar la clave  
    public void guardar_clave_publica(String nombre_archivo) {  
        try {  
            nuevo_nombre=dir_claves+"clave_"+nombre_archivo;  
  
            File file =new File(nuevo_nombre);  
            if (file.exists()) {  
                file.delete();  
            }  
  
            //factoria de claves con cifrado RSA  
            KeyFactory keyFactory = KeyFactory.getInstance("RSA");  
  
            //Creamos un Keyspec  
            RSAPublicKeySpec publicKeySpec = keyFactory.getKeySpec(pares_claves_rsa.getPublic(),RSAPublicKeySpec.class);  
            FileOutputStream stream = new FileOutputStream(nuevo_nombre);  
            PrintWriter pw = new PrintWriter(stream);  
            pw.println(publicKeySpec.getModulus());  
            pw.println(publicKeySpec.getPublicExponent());  
            pw.close();  
        } catch (Exception e) {  
            e.printStackTrace();  
        } finally {  
            System.out.println("Clave guardada");  
        }  
    }  
}
```

Encriptar_archivo

Esta clase es la encargada de encriptar el objeto File que le pasemos.

```
public class Encriptar_archivo {
    String dir_encriptados="\\.\\FicherosEncriptados\\";
    File archivo_encriptado;

    public File getArchivoEncriptado() {
        return archivo_encriptado;
    }

    public void encriptar_fichero(File archivo){
        try {
            //genero las claves
            Generador_RSA_claves generador_RSA_claves = new Generador_RSA_claves();
            KeyPair keyPair_claves = generador_RSA_claves.getClaves();
            //Se guarda clave en un archivo.
            generador_RSA_claves.guardar_clave_publica(archivo.getName());
            //Se encripta el archivo SE DEFINE EL OBJETO Cipher
            Cipher cipher = Cipher.getInstance("RSA");
            //se pone en modo encriptar
            cipher.init(Cipher.ENCRYPT_MODE, keyPair_claves.getPrivate());
            File file = archivo;

            archivo_encriptado = new File(dir_encriptados+"Encrip_"+archivo.getName());

            if (archivo_encriptado.exists()) {
                archivo_encriptado.delete();
                System.out.println(archivo_encriptado+" existe pero a sido aliminado para el nuevo");
            } else {
                System.out.println("El fichero Encriptado no existe y sera creado");
                //System.exit(-1);
            }
        }
    }
}
```

Genero las claves
Se guarda la clave publica
Se define el objeto Cipher
Y se lo pone en modo de encriptacion.

Se comprueba que el nuevo archivo Encrip +[] Exista.

Se comprueba que el archivo a ser encriptado exista si no el programa se cierra, si existe comienza la encriptacion.

```
//Compruebo si existe o no existe.
if(archivo.exists()) {
    //fuente de entrada
    FileInputStream fileInputStream = new FileInputStream(file);
    //fuente de salida
    FileOutputStream fileOutputStream = new FileOutputStream(archivo_encriptado);

    byte[] bs_buffer = new byte[64];
    int bytes_leidos = fileInputStream.read(bs_buffer);
    while(bytes_leidos != -1) {
        byte[] c_buf = cipher.doFinal(bs_buffer, 0, bytes_leidos);
        fileOutputStream.write(c_buf);
        bytes_leidos = fileInputStream.read(bs_buffer);
    }
    fileInputStream.close();
    fileOutputStream.close();
} else {
    System.out.println("El fichero a Encriptar no existe");
    //si no existe el programa termina.
    System.exit(0);
}
System.out.println("Se ha encriptado "+archivo.getName());
} catch (Exception e) {
    System.out.println("Se ha producido un error al Encriptar ");
    e.printStackTrace();
} finally {
    System.out.println("El programa a finalizado");
}
}
```

Con ayuda de una fuente de entrada obtenemos la información a encriptar.

El proceso de encriptacion se realiza dentro de while utilizando la clave dada al cipher.

PSP

Main

Por lo que respecta al ejecutable el programa que encripta se ha realizado un pequeño menú.

```
1
2 public class Main_Encriptar {
3
4     public static void main(String[] args) {
5         String dir_ficheros = ".\\Ficheros\\";
6         String nombre_archivo;
7         Scanner scanner = new Scanner(System.in);
8         // TODO Auto-generated method stub
9         System.out.println("*****");
10        System.out.println("***** MAIN ENCRYPTAR *****");
11        System.out.println("*****\n");
12        System.out.println("Lista de ficheros\n");
13
14        File dir = new File(".\\Ficheros");
15        String[] ficheros = dir.list();
16        if (ficheros == null)
17            System.out.println("No hay ficheros en el directorio especificado");
18        else {
19            for (int x=0;x<ficheros.length;x++)
20                System.out.println(ficheros[x]);
21        }
22
23        System.out.print("\nIntroduce el nombre del archivo a Encriptar");
24        nombre_archivo = scanner.next();
25
26        File file = new File(dir_ficheros+nombre_archivo);
27
28        Encriptar_archivo archivo = new Encriptar_archivo();
29        archivo.encriptar_fichero(file);
30
31        scanner.close();
32    }
33 }
```

Lista los archivos de un directorio.

Creamos un objeto File con un nombre que recogemos con ayuda de un scanner.

Creamos un objeto de nuestra clase Encriptar_archivo y le pasamos a su metodo encriptar_archivo la file creada anteriormente.

4.Main_Desencriptar

Desencriptar_archivo

Para la creación de este programa se a creado como en el programa anterior una clase que nos ayudara a el cometido.

Esta clase solo tiene en su interior un método que tiene como parámetro un objeto File y un String para el nombre del fichero que contiene la clave publica necesaria para desencriptar.

```

public class Desencriptar_archivo {
    String dir_desencrip="..\\FicherosDesencriptados\\";

    public void desencriptar_fichero(File archivo,String clave_p) {
        try {
            //ClavesGeneradas\\clave_fichero1.txt
            String clave_publica = "..\\ClavesGeneradas\\"+clave_p;
            String dir_archivo = dir_desencrip+"Desencript_"+archivo.getName();

            //Recupero la clave publica
            BufferedReader bufferedReader = new BufferedReader(new FileReader(clave_publica));

            BigInteger bigInteger = new BigInteger(bufferedReader.readLine());
            BigInteger bigInteger2 = new BigInteger(bufferedReader.readLine());

            RSAPublicKeySpec rsaPublicKeySpec = new RSAPublicKeySpec(bigInteger, bigInteger2);

            KeyFactory keyFactory = KeyFactory.getInstance("RSA");
            Key clave = keyFactory.generatePublic(rsaPublicKeySpec);
            //Se genera el cipher y le damos la clave publica que estaba en el fichero encriptado
            //para despues desencriptarlo.
            Cipher cipher = Cipher.getInstance("RSA");
            //lo ponemos en modo desencriptar.
            cipher.init(Cipher.DECRYPT_MODE,clave);

            File file = archivo;

            File archivo_desencrip = new File(dir_archivo);

            if (archivo_desencrip.exists()) {
                archivo_desencrip.delete();
                System.out.println("El fichero Desencriptado existe y se borra para dejar paso al nuevo");
            }else {
                System.out.println("El fichero Desencriptado no existe por lo tanto se creara");
                //System.exit(-1);
            }

            if(!file.exists()) {
                System.out.println("Fichero no existe...");
                System.exit(0);
            }else {
                FileInputStream fileInputStream = new FileInputStream(file);
                FileOutputStream fileOutputStream = new FileOutputStream(dir_archivo);
                //se a probado uno de 1000 , recomendado 256
                byte[] buffer = new byte[1000];

                int bytes = fileInputStream.read(buffer);
                while(bytes != -1) {
                    fileOutputStream.write(cipher.doFinal(buffer,0, bytes));

                    bytes = fileInputStream.read(buffer);
                }
                fileInputStream.close();
                fileOutputStream.close();
                bufferedReader.close();
            }
        }
    }
}

```

Lo primero que se hace es recuperar la clave publica con ayuda de un bufferReader

Una ves recuperada generamos un cipher que nos ayudara a desencriptar , poniendolo en modo Decrypt y dándole la clave.

Se comprueba si existe o no.

La desencripcion del fichero se realiza en el while.

En muy parecido con el proceso de Encriptacion.

PSP

Main

El main se asemeja mucho al de encriptar salvo que en esta ocasión también es necesario dar el nombre del archivo que contiene la clave pública generada.

Esto se realiza de la misma forma que el programa Main_Encriptar

```
public class Main_Desenciptar {  
  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
  
        String dir_ficherosDes = ".\\FicherosEncriptados\\";  
        String nombre_archivo;  
        String clave_p = null;  
        Scanner scanner = new Scanner(System.in);  
        // TODO Auto-generated method stub  
        System.out.println("*****");  
        System.out.println("***** MAIN DESENCRIPTAR *****");  
        System.out.println("*****\n");  
        System.out.println("Lista de ficheros Encriptados\n");  
  
        //Se pide el fichero encriptado  
        File dir = new File(".\\FicherosEncriptados");  
        String[] ficheros = dir.list();  
        if (ficheros == null)  
            System.out.println("No hay ficheros en el directorio especificado");  
        else {  
            for (int x=0;x<ficheros.length;x++)  
                System.out.println(ficheros[x]);  
        }  
  
        System.out.print("\nIntroduce el nombre del archivo a Desenciptar");  
        nombre_archivo = scanner.next();  
  
        System.out.print("\n-----\n");  
        System.out.print("Lista de ficheros claves\n\n");  
  
        //Se pide la clave  
        try {  
            File file2 = new File(".\\ClavesGeneradas");  
            String[] ficheros2 = file2.list();  
            if (ficheros2 == null)  
                System.out.println("No hay ficheros en el directorio especificado");  
            else {  
                for (int x=0;x<ficheros2.length;x++)  
                    System.out.println(ficheros2[x]);  
            }  
        }  
        catch (Exception e) {  
            // TODO: handle exception  
        }  
  
        System.out.print("\nIntroduce el nombre del fichero clave\n");  
        clave_p = scanner.next();  
  
        File file = new File(dir_ficherosDes+nombre_archivo);  
        Desenciptar_archivo archivo2 = new Desenciptar_archivo();  
        archivo2.desenciptar_fichero(file,clave_p);  
  
        scanner.close();  
    }  
}
```

Lista directorio de ficheros Encriptados

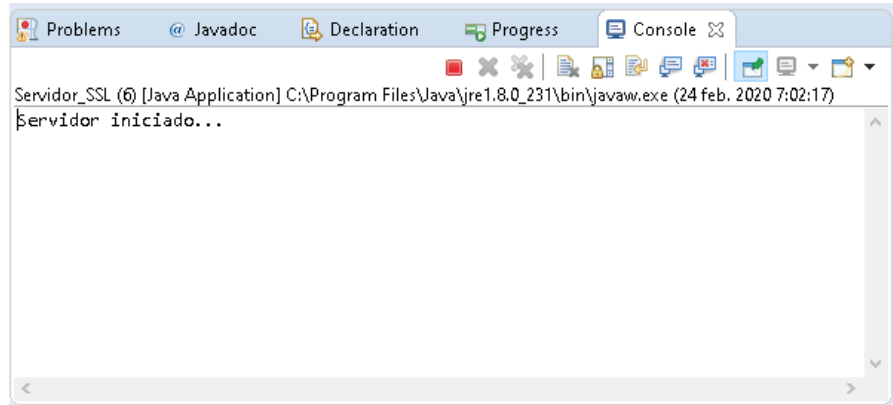
Lista directorio de ficheros con la clave

Se recoge los datos con ayuda de un scanner y los pasamos al metodo desenciptar_fichero de un objeto Desenciptar_archivo

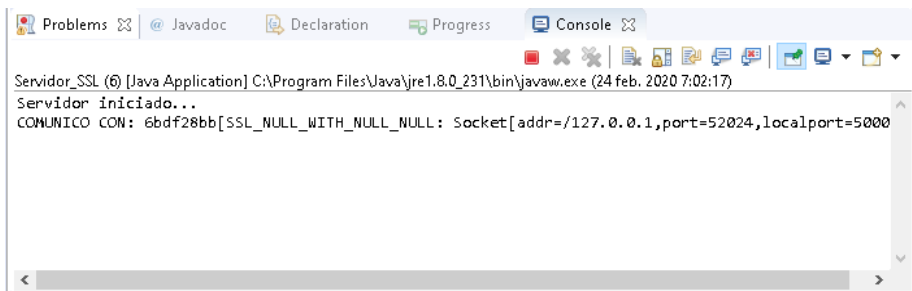
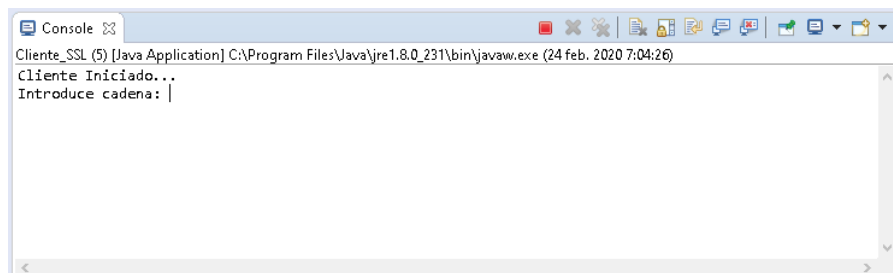
PSP

Pruebas de Ejecución Cliente_SSL y Servidor_SSL

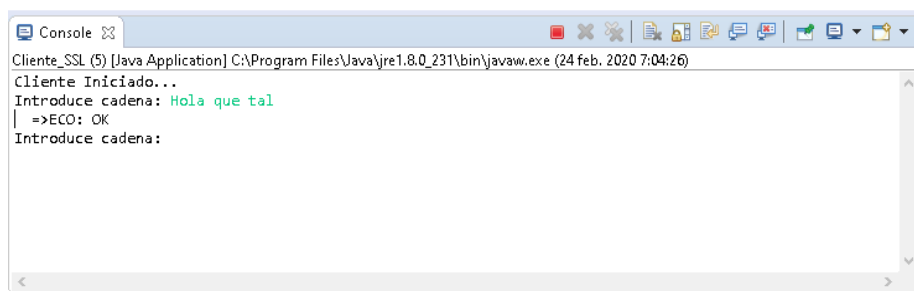
En la siguientes pruebas se a realizado con dos clientes.



Primero se inicia el servidor y después los clientes.



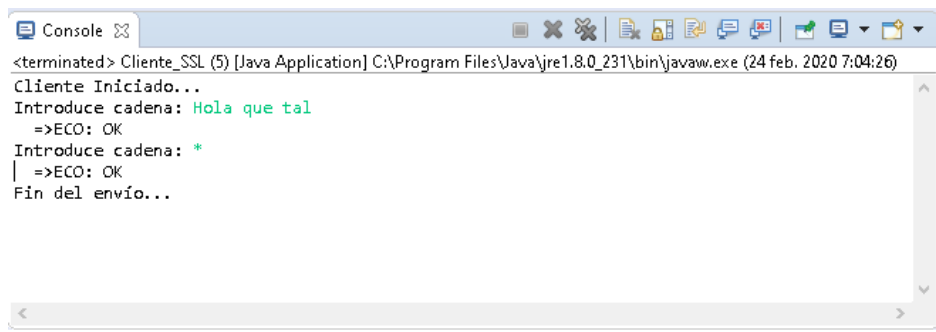
Cuando encuentra un cliente.



Cliente 1

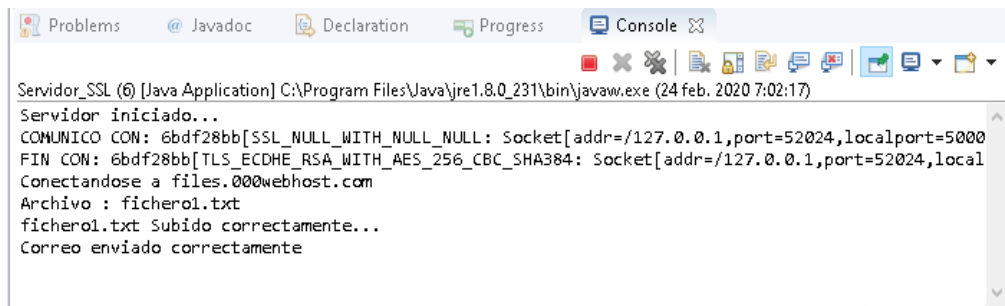
PSP

Cuando se introduce (*) en cliente para.



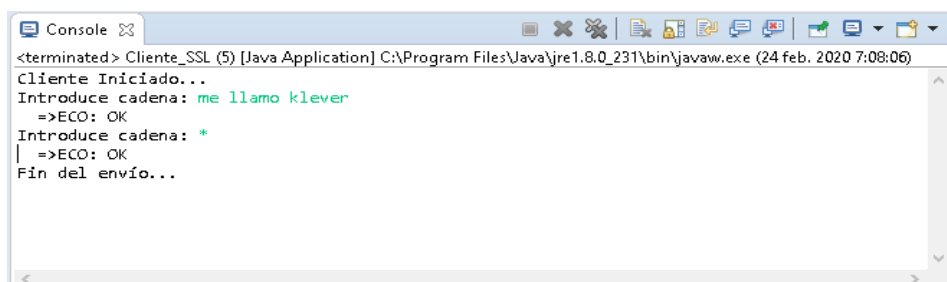
```
<terminated> Cliente_SSL (5) [Java Application] C:\Program Files\Java\jre1.8.0_231\bin\javaw.exe (24 feb. 2020 7:04:26)
Cliente Iniciado...
Introduce cadena: Hola que tal
=>ECO: OK
Introduce cadena: *
| =>ECO: OK
Fin del envío...
```

Y el servidor hace su trabajo.

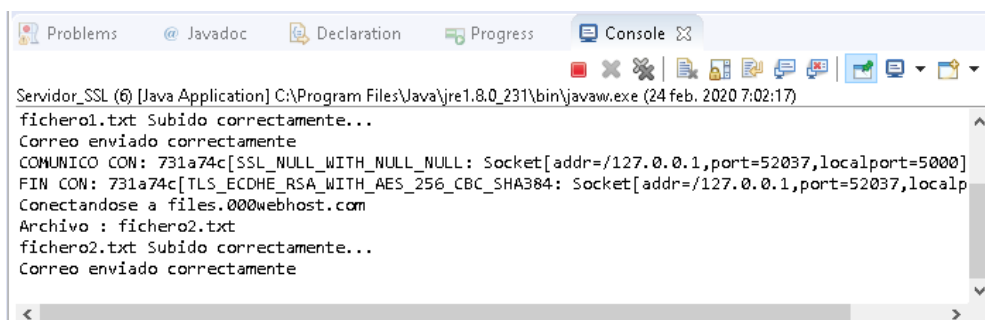


```
Problems @ Javadoc Declaration Progress Console
Servidor_SSL (6) [Java Application] C:\Program Files\Java\jre1.8.0_231\bin\javaw.exe (24 feb. 2020 7:02:17)
Servidor iniciado...
COMUNICO CON: 6bdf28bb[SSL_NULL_WITH_NULL_NULL: Socket[addr=/127.0.0.1,port=52024,localport=5000]
FIN CON: 6bdf28bb[TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384: Socket[addr=/127.0.0.1,port=52024,localport=5000]
Conectandose a files.000webhost.com
Archivo : fichero1.txt
fichero1.txt Subido correctamente...
Correo enviado correctamente
```

Cliente 2



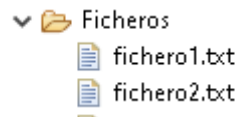
```
<terminated> Cliente_SSL (5) [Java Application] C:\Program Files\Java\jre1.8.0_231\bin\javaw.exe (24 feb. 2020 7:08:06)
Cliente Iniciado...
Introduce cadena: me llamo klever
=>ECO: OK
Introduce cadena: *
| =>ECO: OK
Fin del envío...
```



```
Problems @ Javadoc Declaration Progress Console
Servidor_SSL (6) [Java Application] C:\Program Files\Java\jre1.8.0_231\bin\javaw.exe (24 feb. 2020 7:02:17)
fichero1.txt Subido correctamente...
Correo enviado correctamente
COMUNICO CON: 731a74c[SSL_NULL_WITH_NULL_NULL: Socket[addr=/127.0.0.1,port=52037,localport=5000]
FIN CON: 731a74c[TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384: Socket[addr=/127.0.0.1,port=52037,localport=5000]
Conectandose a files.000webhost.com
Archivo : fichero2.txt
fichero2.txt Subido correctamente...
Correo enviado correctamente
```

PSP

Ahora se comprueba que este todo lo pedido se ha realizado con éxito.



Los Archivos fueron creados.

	Nombre	Tamaño	Fecha	Permisos
<input type="checkbox"/>	archivo1.txt	0,1 kB	2020-02-24 06:06:00	-rw-r--r--
<input type="checkbox"/>	archivo2.txt	0,1 kB	2020-02-24 06:09:00	-rw-r--r--

Se lograron subir.

/Klever/fichero1.txt

1 Hola que tal

Principal

Social **50 nuevos**
YouTube

Promociones **50 nuevos**
Pinterest, Wish, Vigilancia el...

☐ prumail2 2

Actividad servidor Receptor de Mensajes - Se ha enviado el fichero .\Ficheros...

7:09

Y los Email fueron recibidos

Actividad servidor Receptor de Mensajes **RECIBIDOS 25**

prumail2@gmail.com
para mí

7:06 (hace 4 minutos)

☆ ↶ ⋮

Se ha enviado el fichero .\Ficheros\fichero1.txt con los datos generados en la comunicación 6bdf28bb[TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384: Socket[addr=/127.0.0.1,port=52024,localport=5000]]al servidor FTP.
...

[Mensaje recortado] [Ver todo el mensaje](#)

prumail2@gmail.com
para mí

7:09 (hace 2 minutos)

☆ ↶ ⋮

Se ha enviado el fichero .\Ficheros\fichero2.txt con los datos generados en la comunicación 731a74c[TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384: Socket[addr=/127.0.0.1,port=52037,localport=5000]]al servidor FTP.
...

[Mensaje recortado] [Ver todo el mensaje](#)

PSP

Main_Encriptar

En las pruebas de ejecución se va utilizar uno de los ficheros generados en la prueba anterior de SSL.

```
Console Console
Main_Encriptar [Java Application] C:\Program Files\Java\jre1.8.0_231\bin
*****
***** MAIN ENCRYPTAR *****
*****
```

Lista de ficheros

fichero1.txt
fichero2.txt
fichero3.txt
fichero4.txt
fichero5.txt

Incluye el nombre del archivo a Encriptar

```
*****
***** MAIN ENCRYPTAR *****
*****
```

Lista de ficheros

fichero1.txt
fichero2.txt
fichero3.txt
fichero4.txt
fichero5.txt

Incluye el nombre del archivo a Encriptar

fichero1.txt

Se han generado el par de claves

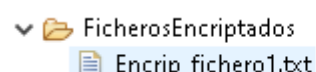
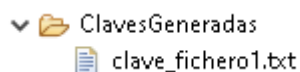
Clave guardada

.\FicherosEncriptados\Encrip_fichero1.txt existe pero a sido aliminado para el nuevo

Se ha encriptado fichero1.txt

El programa a finalizado

Como se puede ver me ha generado un fichero encriptado y su clave para poder desencriptarlo.



PSP

Main_Descriptar

En las pruebas de ejecución se va utilizar uno de los ficheros generados en la prueba anterior de SSL.

```
Console Console
Main_Descriptar [Java Application] C:\Program Files\Java\jre1.8.0_231\bin\javaw.exe (24 feb. 2020 7:33:
*****
***** MAIN DESENCRIPTAR *****
*****

Lista de ficheros Encriptados

Encrip_fichero1.txt
Encrip_fichero2.txt
Encrip_fichero3.txt
Encrip_fichero4.txt
Encrip_fichero5.txt

Incluye el nombre del archivo a Desencriptar
```

```
Console Console
<terminated> Main_Descriptar [Java Application] C:\Program Files\Java\jre1.8.0_231\bin\javaw.exe (24 feb. 20
*****
***** MAIN DESENCRIPTAR *****
*****

Lista de ficheros Encriptados

Encrip_fichero1.txt
Encrip_fichero2.txt
Encrip_fichero3.txt
Encrip_fichero4.txt
Encrip_fichero5.txt

Incluye el nombre del archivo a Desencriptar
Encrip_fichero1.txt

-----
Lista de ficheros claves

clave_fichero1.txt
clave_fichero2.txt
clave_fichero3.txt
clave_fichero4.txt
clave_fichero5.txt

Incluye el nombre del fichero clave
clave_fichero1.txt
El fichero Desencriptado existe y se borra para dejar paso al nuevo
Se a Desencriptado Encrip_fichero1.txt
Ha finalizado el programa
```

Como se puede ver me ha generado un fichero desencriptado con el contenido esperado.

