COMPRESIÓN Y SEGURIDAD
PRÁCTICA 1 - FASE 3

SISTEMA DE CIFRADO CON AES128 Y RSA

GRUPO 202:

Antonio Muñoz Torres
Pablo Guillén García
José Luis Segura Navarro
Ismael Cáceres Bernabeu

ÍNDICE:

1. INTRODUCCIÓN

Programa/sistema que permite cifrar archivos con RSA. Este programa permite encriptar archivos multimedia mediante la clave pública y privada del administrador y formar un fichero de texto con toda la información. El programa en sí encripta y desencripta los ficheros.

El funcionamiento del programa se explica detalladamente en el próximo punto.

2. PROTOCOLO DE SEGURIDAD

a. Notación Cifrado

- **Usuario u**
  - Clave privada RSA de u: k_u^{RSA}
  - Clave pública RSA de u: K_u^{RSA}
- **Usuario v**
  - Clave privada RSA de v: K_v^{RSA}
  - Clave pública RSA de v: K_v^{RSA}
- **Fichero File**
  - Clave AES para el fichero f: k_f
- **Password**
  - Contraseña del usuario u: p_u

b. Acción de Registro

- El usuario, para registrarse, necesita:
  - Un nombre de usuario
  - Un password de 16 caracteres
  - Un par de claves generadas por RSA
- Los pasos a seguir son los siguientes:
  - Una vez el servidor autorice al usuario, se genera un par de claves pública y privada: k_u^{RSA}, K_u^{RSA}

- El usuario u envía la clave pública
- El usuario u envía la clave privada encriptada con su contraseña:
  $E_{pu}^{AES}(k_u^{RSA})$

c. Subir un archivo

El usuario u quiere subir un archivo f y tiene que:

- Genera una clave AES que será única para ese archivo: k_f
- Cifra el fichero f con dicha clave: $E_{kf}^{AES}(f)$
- Cifra la clave del fichero con su clave pública: $E_{ku}^{RSA}{}_{RSA}(kf)$
- Y sube los dos elementos generados al servidor:
  $E_{kf}^{AES}(f)$, $E_{ku}^{RSA}{}_{RSA}(kf)$

d. Descargar archivo privado

Si el usuario u desea descargar archivo f

- Se descarga el archivo cifrado $E_{kf}^{AES}(f)$, y la clave AES de este cifrada con su clave pública $E_{ku}^{RSA}{}_{RSA}(kf)$
- Una vez descargado, descifra la clave AES:
  $$D_{ku}^{RSA}{}_{RSA}[E_{ku}^{RSA}{}_{RSA}(kf)] = k_f$$
- Y una vez tenga la clave AES del fichero, lo descifra
  $$D_{kf}^{AES}[E_{kf}^{AES}(f)] = f$$
- Y ya tendrá el fichero

e. Compartir archivos

- Escenario: El usuario u tiene un fichero f en servidor y quiere compartirlo con v
- u se descargar la clave AES del archivo f y recive:
  $$E_{ku}^{RSA}{}_{RSA}(kf)$$
- Como u tiene su clave privada RSA, lo descifra y obtiene la clave del archivo:
  $$D_{ku}^{RSA}{}_{RSA}[E_{ku}^{RSA}{}_{RSA}(kf)] = k_f$$
- Una vez ha obtenido kf, pide al servidor la clave pública del usuario v y cifra la clave del fichero con esta y lo sube al servidor.

- Cuando el usuario v quiera descargar el archivo, se descargará la clave que se ha generado a través de su clave pública y podrá conseguir la clave AES del archivo f:

$$D_{kv}^{RSA}{}_{RSA}[E_{kv}^{RSA}{}_{RSA}(kf)] = k_f$$

3. SOFTWARE

- JAVA:
  - Lenguaje de programación utilizado para implementar el programa de cifrado y descifrado.
- APACHE NETBEANS IDE:
  - Es un entorno de desarrollo integrado donde hemos desarrollado la práctica en java implementando también una interfaz gráfica.
- XAMPP:
  - Paquete que consiste en el sistema de gestión de base de datos mySQL utilizado para crear una base de datos y conectarla a nuestro proyecto.

4. LIBRERÍAS

- El programa está creado en Java y por tanto todas las librerías javas que hemos implementado serían las siguientes:

  - **java.io.BufferedWriter:** Clase que nos permite escribir texto en un Outputstream, utilizando un buffer para proporcionar una escritura eficiente de caracteres, arrays y strings.
  - **java.io.File:** La clase File representa un fichero o directorio de nuestro sistema de archivos.
  - **java.io.FileInputStream:** Un FileInputStream obtiene bytes de entrada de un archivo en un sistema de archivos.
  - **java.io.FileOutputStream:** Clase que sirve para volcar un stream de datos sobre un fichero.
  - **java.io.FileWriter:** Permite escribir en un fichero.
  - **java.io.InputStream:** declara los métodos para leer datos desde una fuente concreta y es la clase base de la mayor parte de los flujos de entrada en java.io.
  - **java.io.OutputStreamWriter:** Un OutputStreamWriter es un puente entre los flujos de caracteres y los flujos de bytes: los caracteres que se escriben en él se codifican en bytes utilizando un juego de caracteres especificado.
  - **java.security.key:** Proporciona las clases e interfaces para el marco de seguridad.
  - **java.security.NoSuchAlgorithmException:** Esta excepción se lanza cuando se solicita un algoritmo criptográfico particular, pero no está disponible en el entorno.

- **java.util.Base64:** Esta clase consta exclusivamente de métodos estáticos para obtener codificadores y decodificadores para el esquema de codificación Base64
- **java.util.Date:** La clase Date representa un instante específico en el tiempo, con precisión de milisegundos.
- **javax.crypto.Cipher:** Esta clase proporciona la funcionalidad de un cifrado criptográfico para cifrado y descifrado. Forma el núcleo del marco de Java Cryptographic Extension (JCE).
- **javax.crypto.KeyGenerator:** Esta clase proporciona la funcionalidad de un generador de claves secretas (simétricas).

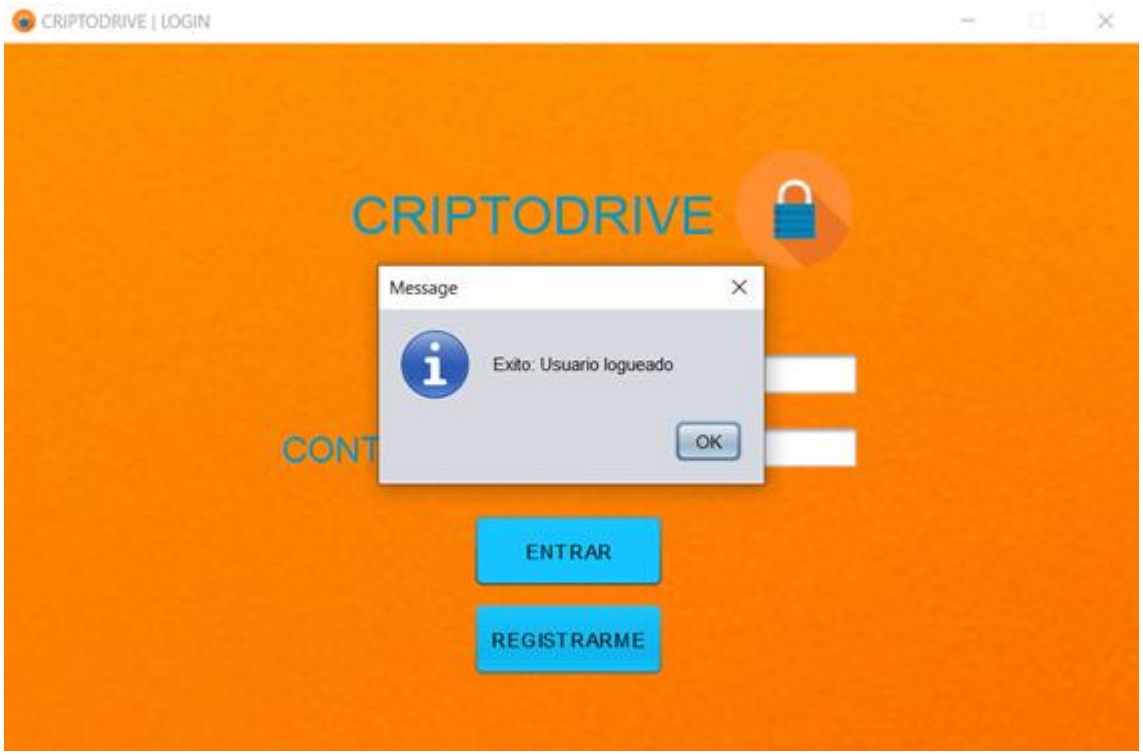
5. MANUAL DE USUARIO

En este apartado vamos a explicar cómo el usuario puede usar esta aplicación, definiendo sus funcionalidades y mostrando capturas de pantalla para que el usuario realice las operaciones de manera inequívoca.

Las ventanas constan, podríamos decir, a modo de encabezado de: logo y nombre de la aplicación, y posteriormente, un título de ventana donde se explicita qué es lo que se puede o debe hacer en esa ventana. En la ventana de registro de usuario, por ejemplo, se pondrá: "|REGISTRARSE|".

Primero, cuando se lanza la aplicación, podremos ver una interfaz donde el usuario puede loguearse si lo está, en caso contrario puede registrarse. Para la comodidad del usuario, habrá un enlace llamado "registrarme" en la parte inferior de la ventana para llevarlo a la parte de registro. Esta ventana la explicamos posteriormente.

Una vez nos hayamos logueado con el usuario y contraseña correctos, nos aparecerá una ventana pequeña informándonos del logueo exitoso, pudiendo así, entrar a la aplicación en sí.



Si los datos no son correctos, se le notificará al usuario.

Si no tenemos cuenta aún, podemos crearla, pinchando en el botón de registro.

En la ventana de registro, habrá tres campos:

- Uno para el nombre de usuario, donde se definirá el nombre que el usuario usará en esta aplicación.

- Otro para el de contraseña, donde el usuario definirá por primera vez la contraseña que quiera usar a partir de ahora en esta aplicación.

- Y, por último, habrá un campo donde el usuario deberá repetir la contraseña que haya escrito en el campo anterior, donde ambos campos deberán de coincidir para llevar a cabo el registro de manera exitosa.

Finalmente, esta ventana constará de un botón donde podremos registrarnos una vez completados los campos. Esto añadirá un nuevo usuario a la base de datos.

Una vez nos hayamos registrado nos aparecerá una pequeña ventana emergente, notificándonos que nos hemos registrado con éxito.

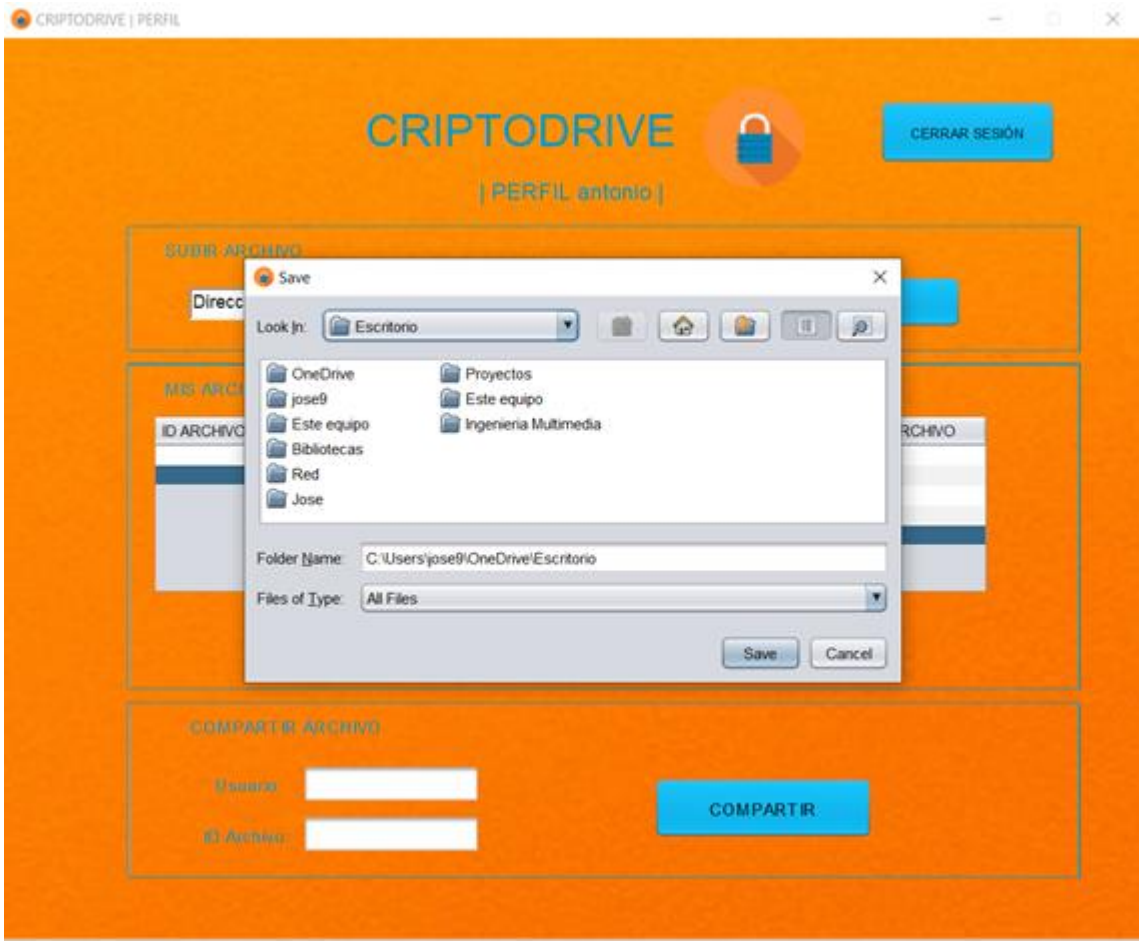
En caso de que no sean correctos los campos introducidos, también se le notificará al usuario.

Una vez realizado el registro, o en su defecto el logueo (inicio de sesión) podremos entrar a la funcionalidad principal de la aplicación, que trata de compartir archivos entre distintos usuarios.

Accediendo con éxito a la aplicación, nos aparecerá una ventana de perfil, nuestro perfil, donde podremos hacer varias acciones como son:

- **Subir un archivo**. En esta opción podremos mediante un botón (examinar) escoger de entre nuestros directorios el archivo que queramos subir. Una vez seleccionado, lo siguiente que debemos de hacer para completar la subida a la base de datos es simplemente clicar en el botón “subir”. Una vez realizado esto, podremos ver el archivo si recargamos la visualización de los archivos.

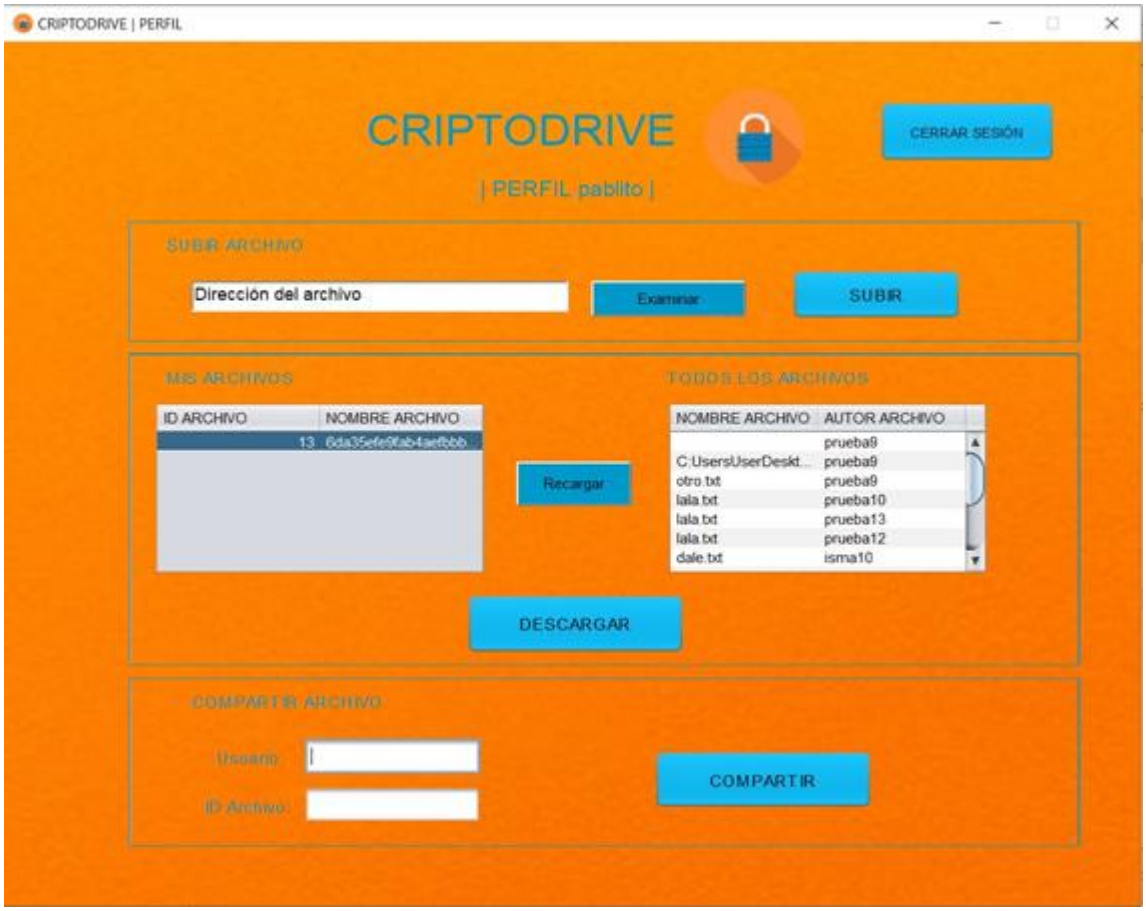
- **Ver todos los archivos, los míos y los compartidos para mí.** Aquí podremos ver nuestros archivos (tanto los que hemos subido nosotros como los que nos han compartido otros usuarios), como también todos los archivos que existen y su respectivo usuario (persona que lo ha subido). Tendremos también un botón de descargar, para poder bajarnos un archivo. Para hacer esto deberemos seleccionar el archivo deseado en “Mis archivos” y clicar en el botón descargar. Si pulsamos nos saldrá una ventana para elegir el directorio donde queremos guardar el archivo.

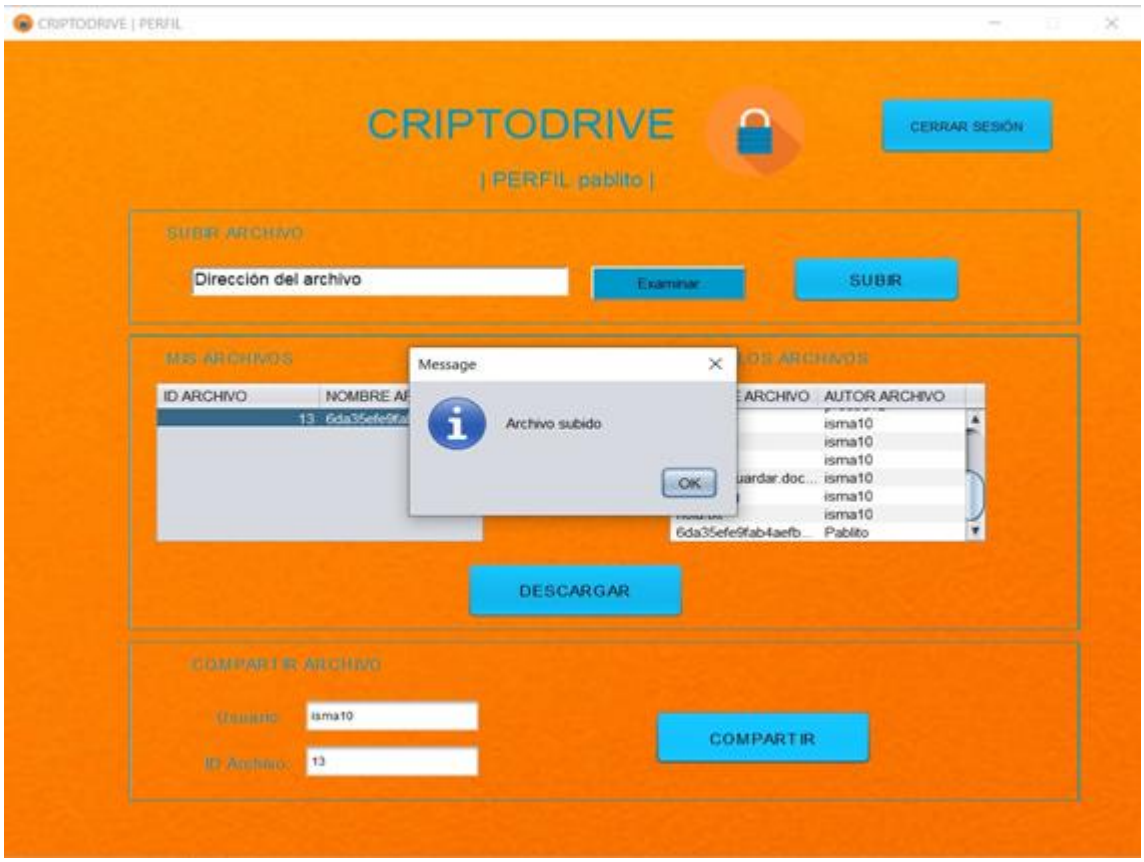


En “Mis archivos” saldrá un par de columnas informativas, una es el id de archivo que solo lo podrá ver el usuario al que le han compartido tal archivo o el dueño de éste, y otra que es el nombre del archivo.

Tenemos un botón recargar que, para visualizar la subida de un archivo, por ejemplo, podremos darle y ver que efectivamente se encuentra subido.

· **Compartir un archivo**. En esta sección tendremos dos campos: uno para introducir el usuario, y otro para introducir el id del archivo, como es lógico solo podremos compartir nuestros archivos, y podremos poner el id para hacerlo ya que solo nosotros sabemos el id de nuestros archivos. Completados los campos, tendremos un botón “compartir” donde si pulsamos, compartiremos el archivo con el id que hayamos especificado al usuario que hayamos elegido. Este archivo le aparecerá a dicho usuario en su ventana perfil -> mis archivos. Pongamos un ejemplo.


CRIPTODRIVE | PERFIL
CRIPTODRIVE
CERRAR SESIÓN
| PERFIL pablito |
SUBIR ARCHIVO
Dirección del archivo
Examinar
SUBIR
MIS ARCHIVOS
ID ARCHIVO
NOMBRE ARCHIVO
13 6da35efe9fab4aefbbb
Recargar
TODOS LOS ARCHIVOS
NOMBRE ARCHIVO
AUTOR ARCHIVO
prueba9
C:\Users\User\Deskt...
prueba9
otro.txt
prueba9
lala.txt
prueba10
lala.txt
prueba13
lala.txt
prueba12
dale.txt
isma10
DESCARGAR
COMPARTIR ARCHIVO
Usuario
ID Archivo
COMPARTIR

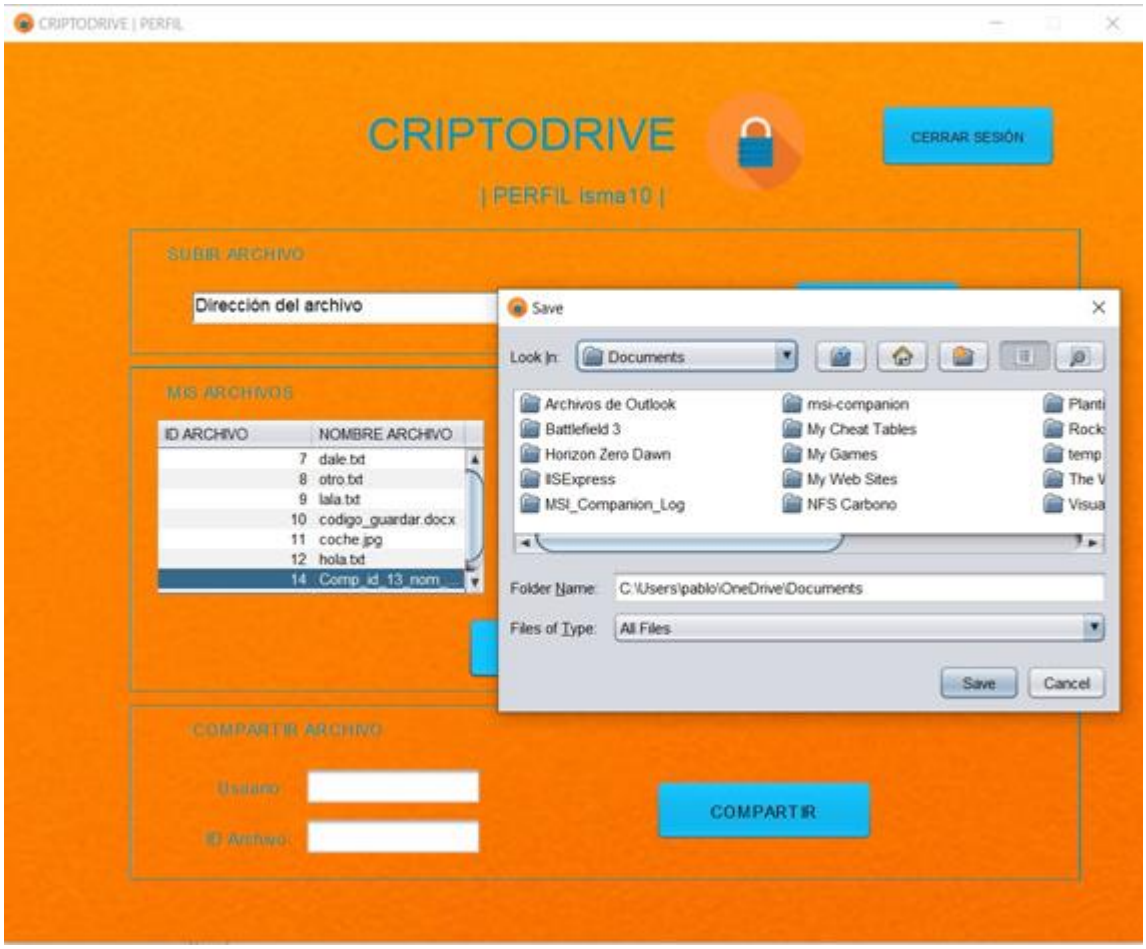

CRIPTODRIVE
CERRAR SESIÓN
| PERFIL pablito |
SUBIR ARCHIVO
Dirección del archivo
Examinar
SUBIR
MIS ARCHIVOS
Message
Archivo subido
OK
ID ARCHIVO
AUTOR ARCHIVO
isma10
isma10
isma10
isma10
isma10
isma10
6da35efe9fab4aefb...
Pablito
DESCARGAR
COMPARTIR ARCHIVO
Usuario
isma10
ID Archivo
13
COMPARTIR

Como vemos, el usuario Pablito ha seleccionado un archivo para compartirlo con otro usuario (se ha de seleccionar el archivo, si no, el botón no hace nada), en este caso “isma10”, y ha puesto el id de ese archivo.

Si isma10 entra en su perfil, verá un archivo compartido, este archivo se mostrará en “Mis archivos”.

El archivo compartido le saldrá a isma10 con el nombre “Comp_” + id del archivo original_+ “nom_” + nombre de archivo original. También saldrá donde se encuentra la lista de todos los archivos. El archivo compartido para isma10 ahora tiene un id, y es el numero consecutivo al ultimo id que tenga el último archivo subido.

Ahora ima10 puede descargarse este archivo que le ha compartido el usuario Pablito, seleccionándolo y presionando el botón de descargar.

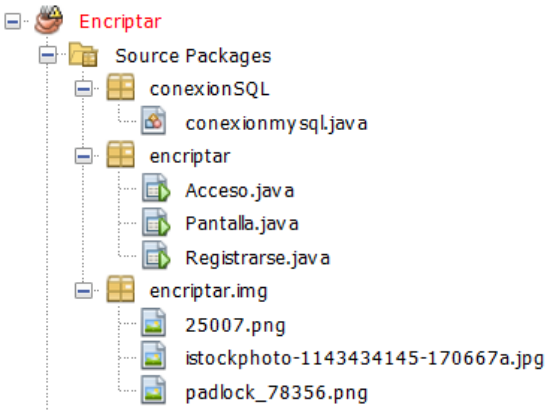


A partir de aquí, se abrirá un explorador de archivos para que pueda descargárselo en el directorio que el usuario elija.

· **Cerrar sesión.** Para cerrar nuestra sesión, salir, solo tendremos que clicar este botón que se encuentra en la parte superior izquierda de la ventana.

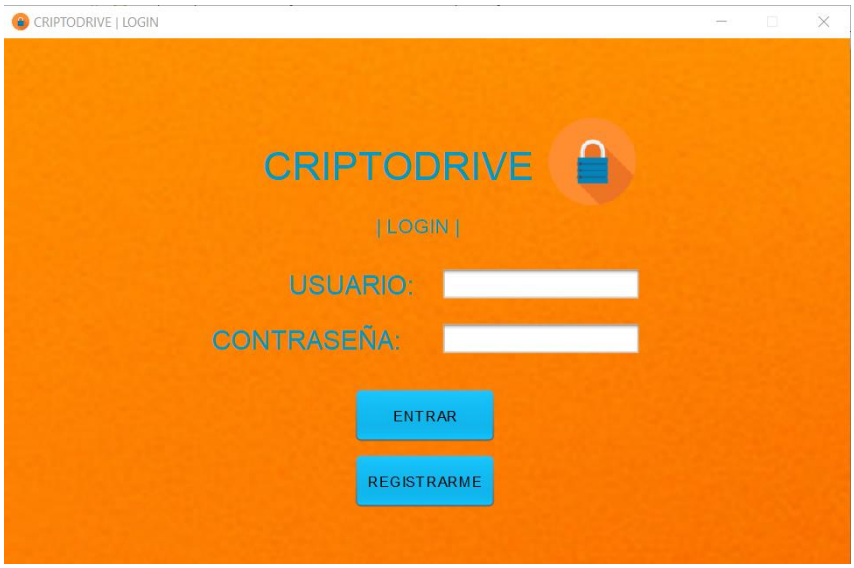
6. DETALLES DE IMPLEMENTACIÓN

Este programa se ha desarrollado en un entorno llamado Apache Netbeans. Por ello vamos a ver la estructura de carpetas principal y a explicar el funcionamiento de cada paquete. La carpeta más importante es “Source Packages”. En ella tenemos la conexión SQL, muy importante dado que toda la información se debe guardar en una base de datos, tenemos el paquete “encriptar” donde residen los archivos principales que le dan vida a nuestro programa, los cuales comentaremos en profundidad, y por último el paquete “encriptar.img” donde guardamos las imágenes que utilizamos para el diseño de la interfaz de nuestro programa.



Ahora iremos viendo las diferentes interfaces de nuestro programa, y comentando que hacen los botones y opciones que tenemos en cada una de ellas y su código correspondiente.

Empecemos por la interfaz de Acceso. En Acceso solo tenemos dos bloques de texto para rellenar tanto el usuario como la contraseña de una cuenta ya creada que se aloja en nuestra base de datos. En caso de introducir dichos datos tenemos justo debajo el botón de entrar y en caso de no tener cuenta tenemos el botón de registrarse.



En cuanto al código, en la declaración de la clase Acceso declaramos dos atributos que servirán para la conexión con la base de datos y declaramos un constructor de clase donde inicializamos los componentes de la clase e introducimos las imágenes para la parte estética de la interfaz.

```
public class Acceso extends javax.swing.JFrame {

    conexionmysql cc= new conexionmysql();
    Connection con= cc.conexion();

    public Acceso() {
        initComponents();
        setIconImage(new ImageIcon(getClass().getResource("img/padlock_78356.png")).getImage());
    }
```

En el botón Entrar, cogemos los dos textos (usuario y contraseña) que el usuario ha introducido y los utilizamos para realizar una consulta sql a nuestra base de datos. De esta forma comprobamos si el usuario y contraseña introducidos corresponden a algún usuario existente. En caso de que sea así mostramos un mensaje en el que se expresa que se ha logueado con éxito y creamos un objeto de tipo Pantalla (el cual veremos posteriormente), lo hacemos visible y cerramos el Acceso actual. Por el contrario, si los datos no corresponden a ningún usuario mostramos un mensaje informando de ello.

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    int resultado =0;

    try{
        String usuario = txtUsuario.getText();
        String pass= String.valueOf(txtpass.getPassword());

        String sql= "select * from usuario where nombre_usuario='"+usuario+"' and password=SHA('"+pass+"');";

        java.sql.Statement st= con.createStatement();
        ResultSet rs= st.executeQuery(sql);

        if(rs.next()){
            resultado=1;
            JOptionPane.showMessageDialog(null, "Exito: Usuario logueado");
            Pantalla login= new Pantalla(usuario,pass);
            login.setVisible(true);
            this.dispose();
        }
        else{
            JOptionPane.showMessageDialog(null, "Usuario no encontrado");
        }

    }catch(Exception e){
        JOptionPane.showMessageDialog(null, "Error: "+e.getMessage());
    }


}
```

El botón Registrarme simplemente crea un objeto de tipo Registrarse, lo hace visible y cierra la vista Acceso actual.

```
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    Registrarse regis= new Registrarse();
    regis.setVisible(true);
    this.dispose();// TODO add your handling code here:
}
```

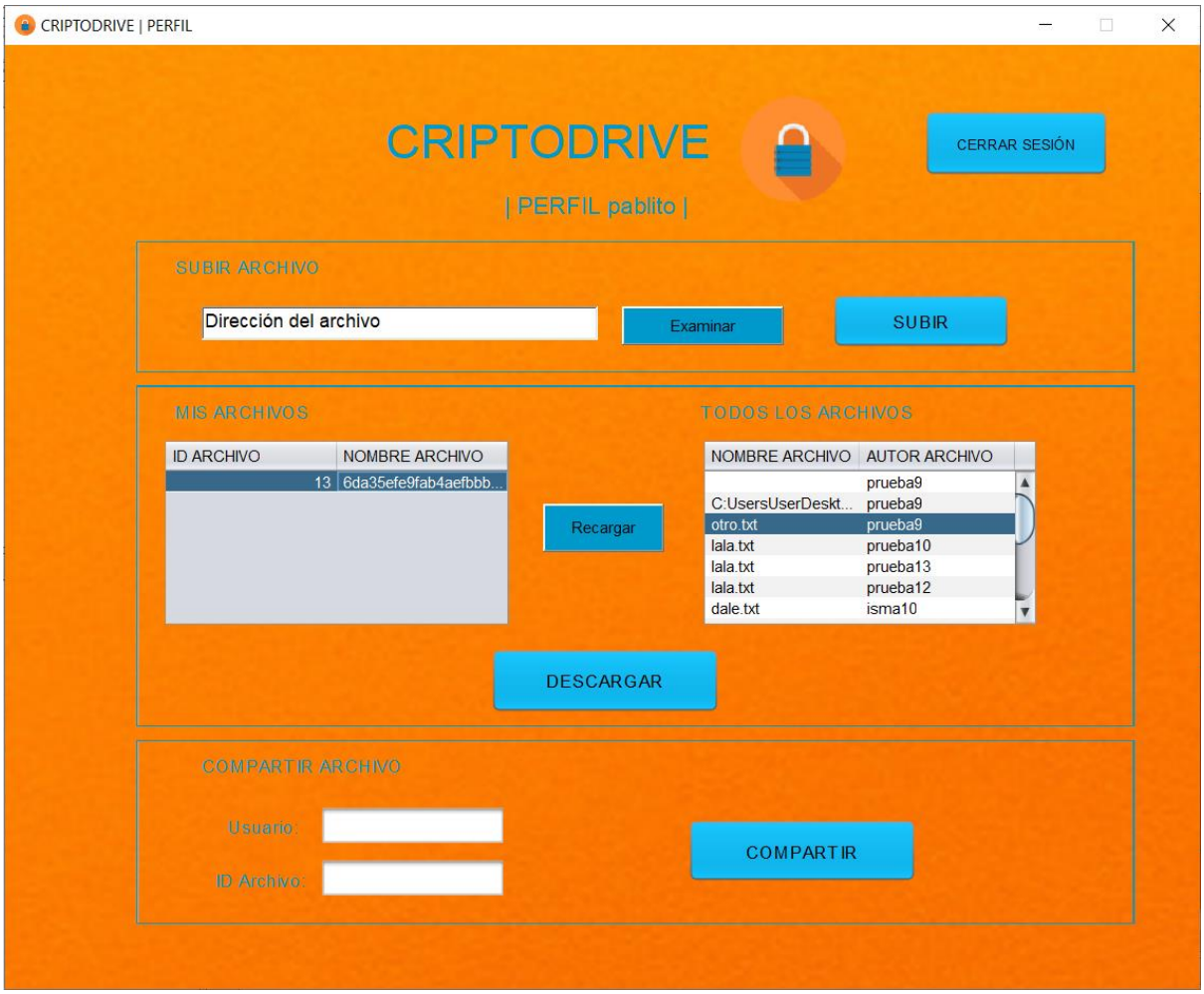
Vayamos con la clase Registrarse. Los atributos de la clase y el constructor son iguales que en la clase Acceso.

Aquí tenemos tres campos para rellenar con información, el nombre de usuario, la contraseña que queremos para dicho usuario y un campo para volver a poner la contraseña como método de seguridad clásico por si nos hemos equivocado al escribirla la primera vez. Por último, tenemos el botón Registrarse.

En el botón registrarse recogemos toda la información introducida y comprobamos que sea válida, es decir, que no sean cadenas vacías, que las contraseñas sean iguales y de 16 caracteres. Si esto se cumple entonces generamos un par de claves RSA, una pública y una privada y encriptamos la privada con AES. La pública se manda tal cual y la privada se manda encriptada. Ahora, insertamos en la base de datos el usuario con su contraseña cifrada y sus claves RSA pública y privada. Si se realiza correctamente la inserción, mostramos por mensaje emergente que el usuario ha sido registrado, abrimos la interfaz de Acceso y cerramos Registrarse. En caso contrario mostramos un mensaje de error.

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    boolean accede= true;
    String nombre= txtNuevoUsuario.getText();
    String pass= String.valueOf(txtNuevoPass.getPassword());
    String pass2= String.valueOf(txtRepePass.getPassword());

    if(nombre.equals("") || nombre.equals(" ") || nombre.length()==0){
        accede=false;
    }

    if(pass.length()!=16){
        accede=false;
    }
    else if(!pass.equals(pass2)){
        accede=false;
    }

    if(accede){

        String publica=null;
        String privada= null;

        try {
            //GENERO PAR DE CLAVES
            KeyPairGenerator generadorRSA= KeyPairGenerator.getInstance("RSA");
            generadorRSA.initialize(1024);
            KeyPair claves = generadorRSA.genKeyPair();
            publica= Base64.getEncoder().encodeToString(claves.getPublic().getEncoded());
            //ENCRIPTO RSA PRIVADA CON LA CONTRASEÑA MEDIANTE AES
            SecretKey passMontada= new SecretKeySpec(pass.getBytes(("UTF-8")),"AES");
            Cipher en= Cipher.getInstance("AES");
            en.init(Cipher.ENCRYPT_MODE,passMontada);
            byte[] subir_privada= en.doFinal(claves.getPrivate().getEncoded());
            privada= Base64.getEncoder().encodeToString(subir_privada);

        } catch (NoSuchAlgorithmException ex) {
            Logger.getLogger(Registrarse.class.getName()).log(Level.SEVERE, null, ex);
        } catch (UnsupportedEncodingException ex) {
            Logger.getLogger(Registrarse.class.getName()).log(Level.SEVERE, null, ex);
        } catch (NoSuchPaddingException ex) {
            Logger.getLogger(Registrarse.class.getName()).log(Level.SEVERE, null, ex);
        } catch (InvalidKeyException ex) {
            Logger.getLogger(Registrarse.class.getName()).log(Level.SEVERE, null, ex);
        } catch (IllegalBlockSizeException ex) {
            Logger.getLogger(Registrarse.class.getName()).log(Level.SEVERE, null, ex);
        } catch (BadPaddingException ex) {
            Logger.getLogger(Registrarse.class.getName()).log(Level.SEVERE, null, ex);
        }


        String sql="INSERT INTO usuario (nombre_usuario, password, publica, privada_AES) VALUES ('"+nombre+"',SHA('"+p

        try{
            java.sql.PreparedStatement pst= con.prepareStatement(sql);
            pst.execute();
            JOptionPane.showMessageDialog(null, "Usuario registrado");
            Acceso ac= new Acceso();
            ac.setVisible(true);
            this.dispose();
        }
    catch(Exception e){
        JOptionPane.showMessageDialog(null, "Error "+e.getMessage());
    }
}
else{
    JOptionPane.showMessageDialog(null, "Parametros mal introducidos");
}
```

Llegamos a la última clase, Pantalla. Esta es la clase más compleja y detallada ya que es donde suceden todas las funcionalidades de la aplicación en sí. Dividimos la interfaz en tres bloques según las funcionalidades principales. Primero tenemos el bloque subir archivo, donde tenemos un botón Examinar para seleccionar un archivo de cualquier tipo de nuestra PC, y una vez seleccionado tenemos un botón subir para colocarlo en nuestra base de datos a nombre del usuario con el que hayamos entrado.
Después tenemos el bloque Mis Archivos, donde tenemos a la izquierda una tabla donde podemos seleccionar un archivo de los que tenemos en nuestra base de datos y darle al botón descargar para bajarlo a nuestra PC. También tenemos a la derecha una tabla no interactiva donde se muestran todos los archivos de todos los usuarios registrados en nuestra base de datos. Además, en el centro tenemos un botón Recargar por si subimos o compartimos algún archivo para que las dos tablas se actualicen y muestren la información más reciente.
Por último, tenemos el bloque Compartir Archivo donde introducimos un usuario de la base de datos y el ID de uno de los archivos que tengamos subidos y pulsamos el botón compartir para mandarle ese archivo al usuario indicado.
Como funcionalidad base arriba a la derecha tenemos un botón para cerrar sesión en caso de que queramos entrar con un usuario distinto al que estamos utilizando.



En esta clase tenemos una declaración distinta. A parte de hacer lo mismo que en las otras dos, creamos dos String, usuario y password, que asignaremos en el constructor a los valores pasados por parámetro que corresponden a los datos del usuario con el que nos hemos logueado previamente en Acceso.

```
public class Pantalla extends javax.swing.JFrame {
    private String usuario;
    private String password;

    conexionmysql cc= new conexionmysql();
    Connection con= cc.conexion();
    /**
     * Creates new form Pantalla
     */
    public Pantalla(String usu, String pass) {
        usuario = usu;
        password= pass;
        initComponents();
        this.setLocationRelativeTo(null);
        setIconImage(new ImageIcon(getClass().getResource("img/padlock_78356.png")).getImage());
        recargar();
        nombre();
    }
```

Empecemos por el apartado de Subir Archivo. El botón de examinar simplemente nos abre el explorador de archivos para que elijamos uno y guardar la ruta para poder acceder a él. En el botón Subir cogemos la ruta, a partir de ella extraemos el archivo y nos lo guardamos para pasarlo a bytes. Con el archivo en bytes, una clave AES generada anteriormente y el nombre del archivo, llamamos al método Encriptar.
En el método Encriptar, cogemos la clave pública mediante otro método auxiliar llamado cogerClave donde se devuelve la clave pública del usuario que estemos utilizando obtenida de la base de datos y lista para ser utilizada. Con esta clave, la cifraremos con RSA y después la pasamos de bytes a String(cadena) para guardarla. Entonces se realiza el insert en la base de datos del archivo con su clave en el usuario correspondiente y notificamos por pantalla que el archivo se ha subido.

Botones Examinar y Subir:

```
    private void button1ActionPerformed(java.awt.event.ActionEvent evt) {
        JFileChooser exam = new JFileChooser();
        exam.showOpenDialog(this);
        File archivo=exam.getSelectedFile();
        if(archivo != null){
            ruta.setText(archivo.getAbsolutePath());
        }
    }

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        if(!"".equals(ruta.getText())){
            KeyGenerator generarClave = null;
            SecretKey clave=null;
            try {
                generarClave = KeyGenerator.getInstance("AES");
                generarClave.init(128);
                clave = generarClave.generateKey();
            } catch (NoSuchAlgorithmException ex) {
                Logger.getLogger(Pantalla.class.getName()).log(Level.SEVERE, null, ex);
            }
            byte[] contenido=null;

            String nom= ruta.getText();
            String[] nomarchi= ruta.getText().split("\\\\");
            String apa=nomarchi[nomarchi.length-1];


            File arch = new File(ruta.getText());
            InputStream lee = null;
            try {
                lee = new FileInputStream(arch);
            } catch (FileNotFoundException e2) {
                e2.printStackTrace();
            }
            try {
                contenido = new byte[lee.available()];
            } catch (IOException e1) {
                e1.printStackTrace();
            }
            try{
                lee.read(contenido);
            }catch(IOException e){
                e.printStackTrace();
            }
            encriptar(clave, contenido, apa);//encripto contenido con clave aes
        }
    }
```

Función Encriptar:

```
public void encriptar(SecretKey clave, byte[] contenido, String nom){
    Cipher cipher;
    byte[] archivo_en = null;
    String archivo_subir= null;
    String subir_clave= null;
    try{
        cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
        cipher.init(Cipher.ENCRYPT_MODE, clave);
        archivo_en= cipher.doFinal(contenido);
        archivo_subir= Base64.getEncoder().encodeToString(archivo_en);


        //obtengo la clave publica lista para usar
        PublicKey pub= cogerClave();

        //encripto clave aes con pública
        byte[] arc= enc(pub,clave.getEncoded());

        //paso de byte a string para guardar
        subir_clave=Base64.getEncoder().encodeToString(arc);

        String sql="INSERT INTO archivo (nombre_archivo, nombre_usuario, archivo_cifrado, aes_conrsa) VALUES ('"+nom+

        java.sql.PreparedStatement pst= con.prepareStatement(sql);
        pst.execute();
        JOptionPane.showMessageDialog(null, "Archivo subido");
        recargar();
    }
    catch(Exception e){
        e.printStackTrace();
    }
}
```

Métodos cogerClave y enc:

```
public PublicKey cogerClave() throws SQLException, InvalidKeySpecException, NoSuchAlgorithmException{
    String contra=null;
    String sql= "select * from usuario where nombre_usuario='"+usuario+"'";
    PublicKey po= null;

    java.sql.Statement st= con.createStatement();
    ResultSet rs= st.executeQuery(sql);

    if(rs.next()){
        contra= rs.getString("publica");
        byte[] publics= Base64.getDecoder().decode(contra);
        X509EncodedKeySpec keys= new X509EncodedKeySpec(publics);
        KeyFactory keyFactor= KeyFactory.getInstance("RSA");
        po= keyFactor.generatePublic(keys);
    }
    else{
        JOptionPane.showMessageDialog(null, "Usuario no encontrado");
    }
    return po;
}

public byte[] enc(PublicKey cla, byte[] cont) throws NoSuchAlgorithmException, NoSuchPaddingException, InvalidKeyExcep
    Cipher cifraRSA= Cipher.getInstance("RSA/ECB/PKCS1Padding");
    cifraRSA.init(Cipher.ENCRYPT_MODE, cla);
    byte[] claveAESencr= cifraRSA.doFinal(cont);
    return claveAESencr;
}
```

Pasamos al apartado Mis Archivos. Aquí tenemos dos botones que realizan dos funciones totalmente diferentes. El botón Descargar funciona cuando se selecciona un archivo de la tabla donde aparecen los archivos que tiene subidos a la base de datos el usuario. entonces extraemos el id del archivo en cuestión y llamamos a la función auxiliar formarArchivo.

En formarArchivo creamos dos consultas sql, una para encontrar el archivo en cuestión mediante el id y otra para la información del usuario. Cogemos los datos del archivo que conseguimos de la primera consulta y los guardamos. Ahora con la segunda consulta cogemos la clave RSA privada del usuario que está encriptada con AES, la desencriptamos y la montamos. Con la clave privada ya montada la utilizamos para desencriptar y montar la

clave AES que utilizamos para descifrar el archivo. Una vez descifrado lo montamos para poder descargarlo y lo notificamos por pantalla.

Botón Descargar y función auxiliar formarArchivo:

```
    private void descargarMisActionPerformed(java.awt.event.ActionEvent evt) {
        if(tablaMis.getSelectedRow()>-1){
          DefaultTableModel tm= (DefaultTableModel) tablaMis.getModel();
            int fila= tablaMis.getSelectedRow();
            String valor = (String) tm.getValueAt(fila, 0);

            formarArchivo(valor);
        }

    }
```

```
public void formarArchivo(String id){
    String sql="select * from archivo where id_archivo="+id;
    String sql2="select * from usuario where nombre_usuario='"+usuario+"'";

    try{
       java.sql.Statement st= con.createStatement();
       ResultSet rs= st.executeQuery(sql);
       String[] mete= new String[4];
       if(rs.next()){
           mete[0]=rs.getString("id_archivo");
           mete[1]=rs.getString("nombre_archivo");
           mete[2]=rs.getString("archivo_cifrado");
           mete[3]=rs.getString("aes_conrsa");
       }
       java.sql.Statement stu= con.createStatement();
       ResultSet rsu= stu.executeQuery(sql2);
       String clave=null;
       String pas=null;
       PrivateKey po=null;
       if(rsu.next()){
           clave=rsu.getString("privada_AES");//la tengo encriptada aes con su contraseña

           SecretKey passMon= new SecretKeySpec(password.getBytes("UTF-8"),"AES");
           byte[] ayuda= Base64.getDecoder().decode(clave.getBytes("UTF-8"));

           Cipher prim= Cipher.getInstance("AES");
           prim.init(Cipher.DECRYPT_MODE,passMon);
           byte[] rsadesen=prim.doFinal(ayuda);

           //MONTAMOS CLAVE PRIVADA RSA
           KeyFactory kf= KeyFactory.getInstance("RSA");
           po =kf.generatePrivate(new PKCS8EncodedKeySpec(rsadesen));
```

```
            KeyFactory kf= KeyFactory.getInstance("RSA");
            po =kf.generatePrivate(new PKCS8EncodedKeySpec(rsadesen));
        }
        //DESENCRIPTAMOS AES CON LA PRIVADA
        byte[] aes=Base64.getDecoder().decode(mete[3].getBytes("UTF-8"));
        Cipher desc= Cipher.getInstance("RSA/ECB/PKCS1Padding");
        desc.init(Cipher.DECRYPT_MODE, po);
        byte[] aesDes=desc.doFinal(aes);
        String hola= Base64.getEncoder().encodeToString(aesDes);

        //MONTAMOS AES
        SecretKey claveAES = new SecretKeySpec(aesDes,"AES");

        //Desciframos archivo
        byte[] archivo= Base64.getDecoder().decode(mete[2].getBytes("UTF-8"));
        Cipher desi=Cipher.getInstance("AES/ECB/PKCS5Padding");
        desi.init(Cipher.DECRYPT_MODE, claveAES);
        byte[] archivoListo= desi.doFinal(archivo);
        String ar= Base64.getEncoder().encodeToString(archivoListo);

        //Montamos archivo
        try {
            JFileChooser jf= new JFileChooser();
            jf.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
            int sele=jf.showSaveDialog(this);
            if(sele==JFileChooser.APPROVE_OPTION){
                File fichero= jf.getSelectedFile();
                String fa=fichero.getAbsolutePath();
                String[] separar=mete[1].split("\\.");
                String nomFinal= fa+"\\"+separar[0]+"_E_."+separar[1];
                FileOutputStream montar = new FileOutputStream(nomFinal);
                 montar.write(archivoListo);
                 montar.close();
                 JOptionPane.showMessageDialog(null, "Archivo descargado");
            }

         }catch(Exception e){
             e.printStackTrace();
         }


    }
    catch(Exception e){
        e.printStackTrace();
    }
}
```

Para el botón Recargar hacemos un par sql una para cada tabla y realizamos un par de bucles, en el primero rellenando la tabla de la izquierda solo con los archivos del usuario con el que hemos iniciado sesión, y en el segundo rellenando la tabla de la derecha con todos los archivos de todos los usuarios de la base de datos.

Botón Recargar:

```
private void button2ActionPerformed(java.awt.event.ActionEvent evt) {
    String sql="select * from archivo where nombre_usuario='"+usuario+"'";
    DefaultTableModel tm= (DefaultTableModel) tablaMis.getModel();
    tm.setRowCount(0);
    String sql2="select * from archivo";
    DefaultTableModel tm2= (DefaultTableModel) tablaTodos.getModel();
    tm2.setRowCount(0);

    try{
        java.sql.Statement st= con.createStatement();
        ResultSet rs= st.executeQuery(sql);

        while(rs.next()){
            String[] mete= new String[2];
            mete[0]=rs.getString("id_archivo");
            mete[1]=rs.getString("nombre_archivo");
            tm.addRow(mete);
        }

        java.sql.Statement st2= con.createStatement();
        ResultSet rs2= st2.executeQuery(sql2);


        while(rs2.next()){
            String[] mete= new String[2];
            mete[0]=rs2.getString("nombre_archivo");
            mete[1]=rs2.getString("nombre_usuario");
            tm2.addRow(mete);
        }
    }
    catch(Exception e){
          e.printStackTrace();
```

Por último, vamos con la funcionalidad de Compartir Archivo. Aquí tenemos un botón llamado Compartir en el que recogemos el nombre del usuario a quien queremos compartir un archivo y el nombre de dicho archivo. Si no son cadenas vacías llamamos a la función formarArchivo2. La función formarArchivo2 hace lo mismo que la función formarArchivo que hemos comentado antes con la diferencia que esta vez no queremos descargar el archivo en cuestión montado completamente en nuestro PC queremos hacer una transición y volver a subirlo, pero en el usuario al que queremos mandarlo. Para ello realizamos los mismos pasos hasta llegar a tener nuestro archivo en bytes. entonces obtengo los datos que necesito del usuario, su clave pública para encriptar una clave AES que hemos creado con ella, la pasamos a String y realizamos la inserción en la base de datos de dicho archivo con su nombre, el usuario al que va dirigido, el archivo como tal, y su clave, y lo notificamos por pantalla.

Botón Compartir:

```
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    String usuario=usu_comp.getText();
    String archivo=arch_comp.getText();

    if(!usuario.equals("") && !archivo.equals("")){
        formarArchivo2(archivo,usuario);
    }
    else{
        JOptionPane.showMessageDialog(null, "Parametros vacios");
    }
}
```

Función fomarArchivo2:

```
public void formarArchivo2(String nom, String usu){
    String sql="select * from archivo where id_archivo="+nom+" and nombre_usuario='"+usuario+"'";
    String sql2="select * from usuario where nombre_usuario='"+usuario+"'";

    try{
       java.sql.Statement st= con.createStatement();
       ResultSet rs= st.executeQuery(sql);
       String[] mete= new String[4];
       if(rs.next()){
           mete[0]=rs.getString("id_archivo");
           mete[1]=rs.getString("nombre_archivo");
           mete[2]=rs.getString("archivo_cifrado");
           mete[3]=rs.getString("aes_conrsa");
       }
       java.sql.Statement stu= con.createStatement();
       ResultSet rsu= stu.executeQuery(sql2);
       String clave=null;
       String pas=null;
       PrivateKey po=null;
       if(rsu.next()){
           clave=rsu.getString("privada_AES");//la tengo encriptada aes con su contraseña

           SecretKey passMon= new SecretKeySpec(password.getBytes("UTF-8"),"AES");
           byte[] ayuda= Base64.getDecoder().decode(clave.getBytes("UTF-8"));

           Cipher prim= Cipher.getInstance("AES");
           prim.init(Cipher.DECRYPT_MODE,passMon);
           byte[] rsadesen=prim.doFinal(ayuda);

           //MONTAMOS CLAVE PRIVADA RSA
           KeyFactory kf= KeyFactory.getInstance("RSA");
           po =kf.generatePrivate(new PKCS8EncodedKeySpec(rsadesen));
       }
       //DESENCRIPTAMOS AES CON LA PRIVADA
       byte[] aes=Base64.getDecoder().decode(mete[3].getBytes("UTF-8"));
       Cipher desc= Cipher.getInstance("RSA/ECB/PKCS1Padding");
       desc.init(Cipher.DECRYPT_MODE, po);
       byte[] aesDes=desc.doFinal(aes);
       String hola= Base64.getEncoder().encodeToString(aesDes);

       //MONTAMOS AES
       SecretKey claveAES = new SecretKeySpec(aesDes,"AES");

       //Desciframos archivo
       byte[] archivo= Base64.getDecoder().decode(mete[2].getBytes("UTF-8"));
       Cipher desi=Cipher.getInstance("AES/ECB/PKCS5Padding");
       desi.init(Cipher.DECRYPT_MODE, claveAES);
       byte[] archivoListo= desi.doFinal(archivo);

       //SUBIMOS ARCHIVO
       KeyGenerator generarClave = null;
       SecretKey clave2=null;
        try {
            generarClave = KeyGenerator.getInstance("AES");
            generarClave.init(128);
            clave2 = generarClave.generateKey();
        } catch (NoSuchAlgorithmException ex) {
            Logger.getLogger(Pantalla.class.getName()).log(Level.SEVERE, null, ex);
        }
        Cipher cipher;
        byte[] archivo_en = null;
```

```
        String archivo_subir= null;
        String subir_clave= null;

        try{
        cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
        cipher.init(Cipher.ENCRYPT_MODE, clave2);
        archivo_en= cipher.doFinal(archivoListo);
        archivo_subir= Base64.getEncoder().encodeToString(archivo_en);

        //obtengo la clave publica lista para usar
        String contra=null;
        String sql4= "select * from usuario where nombre_usuario='"+usu+"'";
        PublicKey pub= null;

        java.sql.Statement st3= con.createStatement();
        ResultSet rs3= st3.executeQuery(sql4);

        if(rs3.next()){
            contra= rs3.getString("publica");
            byte[] publics= Base64.getDecoder().decode(contra);
            X509EncodedKeySpec keys= new X509EncodedKeySpec(publics);
            KeyFactory keyFactor= KeyFactory.getInstance("RSA");
            pub= keyFactor.generatePublic(keys);
        }
        else{
            JOptionPane.showMessageDialog(null, "Usuario no encontrado");
        }

        //encripto clave aes con pública
        byte[] arc= enc(pub,clave2.getEncoded());
        //paso de byte a string para guardar
        subir_clave=Base64.getEncoder().encodeToString(arc);

        String nombre="Comp_id_"+nom+"_nom_"+mete[1];

        String sql3="INSERT INTO archivo (nombre_archivo, nombre_usuario, archivo_cifrado, aes_conrsa) VALUES ('"+nomb

        java.sql.PreparedStatement pst= con.prepareStatement(sql3);
        pst.execute();
        JOptionPane.showMessageDialog(null, "Archivo subido");
        }
        catch(Exception e){
            e.printStackTrace();
        }


    }
    catch(Exception e){
        e.printStackTrace();
    }
}
```