

Programación de Servicios y procesos

Tarea para PSP06.

Enunciado.

La tarea de la unidad está dividida en dos actividades.

Actividad 6.1. Crea una aplicación que realice los siguientes pasos:

- Solicita el nombre del usuario que va a utilizar la aplicación. El login tiene una longitud de 8 caracteres y está compuesto únicamente por letras minúsculas.
- Solicita al usuario el nombre de un fichero que quiere mostrar. El nombre del fichero es como máximo de 8 caracteres y tiene una extensión de 3 caracteres.
- Visualiza en pantalla el contenido del fichero.

Es importante tener en cuenta que se tiene que realizar una validación de los datos de entrada y llevar un registro de la actividad del programa.

Como en tareas anteriores, el material dispuesto en el tema es de gran ayuda para aportar las estructuras a utilizar en el archivo, por lo que se aprovechan en gran parte los ejemplos dados.

ValidacionRegistro.java

```
/*
 * TAREA PSP06. EJERCICIO 1.
 * Crea una aplicación que realice los siguientes pasos:
 * 1- Solicita el nombre del usuario que va a utilizar la aplicación.
 * El login tiene una longitud de 8 caracteres y está compuesto
únicamente
 * por letras minúsculas.
 * 2- Solicita al usuario el nombre de un fichero que quiere mostrar.
 * El nombre del fichero es como máximo de 8 caracteres y tiene una
 * extensión de 3 caracteres.
 * 3- Visualiza en pantalla el contenido del fichero.
 *
 * Es importante tener en cuenta que se tiene que realizar una
validación de
 * los datos de entrada y llevar un registro de la actividad del
programa.
 *
 * RECORDAR COMENTAR EL PACKAGE SI SE QUIERE COMPILAR FUERA DE
NETBEANS.
 */
package validacionregistro;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.logging.FileHandler;
import java.util.logging.Level;
import java.util.logging.Logger;
import java.util.logging.SimpleFormatter;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

/**
 * @author juang <juangmuelas@gmail.com>
 * @since 17/03/2021
 */
```

```

* @version 1
*/
public class ValidacionRegistro {

    /**
     * Seguimos el esquema propuesto en el Ejemplo para validar
     * entradas en el apartado 2.4 del temario.
     * @param cliente String de recogida de datos
     * @param fichero String de recogida de datos
     * @param patCliente objeto de la clase pattern para recogida
     * de las expresiones regulares.
     * @param matCliente objeto de la clase pattern para evaluar
     * las expresiones regulares.
     * @param patArchivo objeto de la clase pattern para recogida
     * de las expresiones regulares.
     * @param matArchivo objeto de la clase pattern para evaluar
     * las expresiones regulares.
     */

    public ValidacionRegistro() {
        String cliente = new String();
        String fichero = new String();
        Pattern patCliente = null;
        Matcher matCliente = null;
        Pattern patArchivo = null;
        Matcher matArchivo = null;

        /**
         * Seguimos el esquema propuesto en el Ejemplo para los
         * ficheros de registro en el apartado 2.6 del temario.
         * Aprovechamos por tanto parte de su código.
         * @param logger objeto de la clase Logger para controlar
         * los eventos.
         * @param fh para el manejador que asocia al fichero.
         */

        Logger logger = Logger.getLogger("MyLog");
        FileHandler fh;

        try {
            // Configuro el logger y establezco el formato
            fh = new FileHandler("MyLogFile.log", true);
            logger.addHandler(fh);
            logger.setLevel(Level.ALL); //ALL: Para registrar todos los
eventos

            SimpleFormatter formatter = new SimpleFormatter();
            fh.setFormatter(formatter);
        } catch (SecurityException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }

        /**
         * Tras revisar la documentación y comprobar varios ejemplos
         * como
         * el encontrado en el artículo de Medium:logs-en-java-con-
         * java-util
         * decido apartar los mensajes de registro de la vista
         */
        logger.setUseParentHandlers(false);
    }
}

```

```

/**
 * Tras el apartado de registros, seguimos aprovechando el
 * ejemplo del apartado 2.4
 * @param texto String que guarda los mensajes a mostrar.
 */
String texto ="";

// para leer del teclado
BufferedReader reader=new BufferedReader(new
InputStreamReader(System.in));

try{
/**
 * Solicitamos el login y cotejamos con un patrón.
 */

System.out.println("*****");
System.out.println("(solo con formato de 8 letras
minúsculas)");
System.out.println("LOGIN: ");
cliente=reader.readLine();
patCliente=Pattern.compile("[a-z]{8}$");
matCliente=patCliente.matcher(cliente);

/**
 * Comprobamos patrón correcto, se registra y sigue el
proceso
 */

if(matCliente.find()){
//Si cumple con el patrón, informamos, registramos y
continuamos
logger.log(Level.CONFIG,"LOGIN: " + cliente + ":
FORMATO CORRECTO.");
System.out.println("Bienvenido, "+cliente+" el LOGIN
es correcto.");

//Pedimos el nombre de archivo y comprobamos el patrón
System.out.println("*****");
System.out.println("(Formato de 8 letras+extension de
3 (Ejemplo: Ficheros.dox)");
System.out.println("INDIQUE LE ARCHIVO A OBTENER: ");

fichero=reader.readLine();
patArchivo=Pattern.compile("[a-zA-Z0-9]{1,8}.[a-zA-
Z]{3}");
matArchivo=patArchivo.matcher(fichero);
if(matArchivo.find()){
//Si cumple con el patrón, informamos, registramos
y continuamos
logger.log(Level.CONFIG,"FILE: " + fichero + ":
FORMATO CORRECTO.");
System.out.println("Obteniendo "+ fichero +".");
}else{
//Si no cumple con el patrón
logger.log(Level.CONFIG,"FILE " + fichero + ":
RECHAZADO CON FORMATO NO VALIDO.");
System.out.println("Atención: "+ fichero+" no
tiene un formato valido.");
}
}
}

```

```

    }

    File file= new File(fichero);
    /**
     * Si el archivo existe, registramos los eventos.
     */
    if(file.exists()){
        logger.log(Level.INFO,"Fichero: "+ fichero + "
solicitado existe.");
        //Se abre el archivo
        FileReader fr = new FileReader(fichero);
        BufferedReader br = new BufferedReader(fr);
        //mientras se recogan eventos, se imprimen.
        while((texto = br.readLine())!=null) {
            System.out.println(texto);
        }
        //Cerramos archivo
        br.close();
        fr.close();

        /**
         * Se registra el evento que nos muestra el
fichero

         */
        logger.log(Level.INFO,"Fichero: " + fichero + "
mostrado correctamente.");
    }else{
        //En caso de que el archivo no exista
        System.out.println("Fichero: " + fichero + " no
existe!");
        logger.log(Level.INFO,"Fichero: "+ fichero + "
solicitado no existe.");
    }

    }else{
        /**
         * Registrar e informar en caso de no cumplir con el
patrón

         */
        logger.log(Level.CONFIG,"LOGIN " + cliente + ":
RECHAZADO CON FORMATO NO VALIDO.");
        System.out.println("Atención: "+ cliente +" el LOGIN
no tiene un formato valido.");
    }

    } catch( Exception e ) {
        System.out.println( e.getMessage() );
    }
} //Fin constructor ValidacionRegistro

public static void main(String[] args) {
    // TODO code application logic here
    new ValidacionRegistro();
} //Fin de main
} //Fin clase ValidacionRegistro

```

Tras compilar se prueban las distintas entradas.

```

C:\Users\juang\Documents\MyDesk\DAM\PROGRAMACION DE SERVICIOS Y PROCESOS\garciamuelas_juanantonio_PSP06_Tarea>javac ValidacionRegistro.java
C:\Users\juang\Documents\MyDesk\DAM\PROGRAMACION DE SERVICIOS Y PROCESOS\garciamuelas_juanantonio_PSP06_Tarea>java ValidacionRegistro
(solo con formato de 8 letras minusculas)
LOGIN:
manuel
Atenci  n:  manuel el LOGIN no tiene un formato valido.
C:\Users\juang\Documents\MyDesk\DAM\PROGRAMACION DE SERVICIOS Y PROCESOS\garciamuelas_juanantonio_PSP06_Tarea>java ValidacionRegistro
(solo con formato de 8 letras minusculas)
LOGIN:
hanglong
Bienvenido,  hanglong el LOGIN es correcto.

```

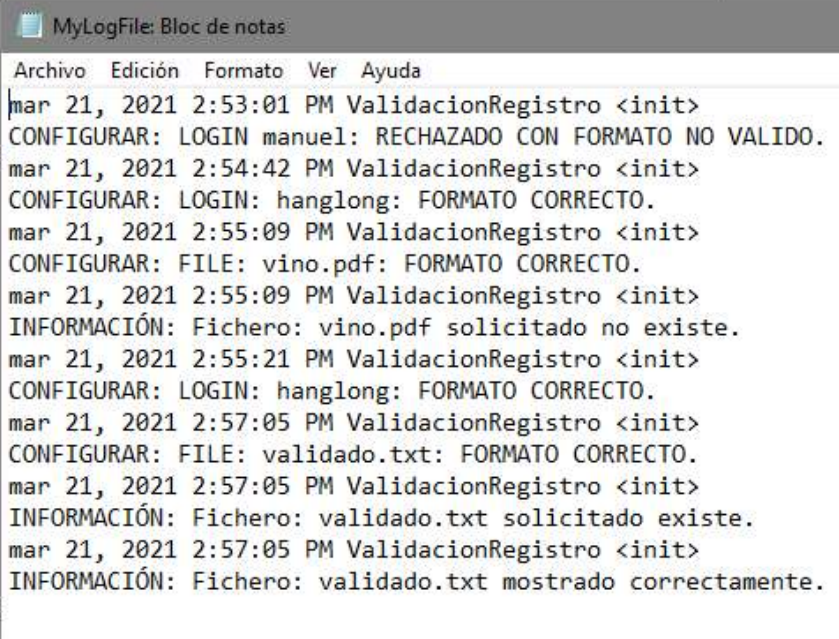
Tras pasar las primeras pruebas, comprobamos los archivos.

```

Bienvenido,  hanglong el LOGIN es correcto.
(solo con formato de 8 letras+extension de 3 (Ejemplo: Ficheros.dox)
INDIQUE LE ARCHIVO A OBTENER:
vino.pdf
Obteniendo vino.pdf.
Fichero: vino.pdf no existe!.
C:\Users\juang\Documents\MyDesk\DAM\PROGRAMACION DE SERVICIOS Y PROCESOS\garciamuelas_juanantonio_PSP06_Tarea>java ValidacionRegistro
(solo con formato de 8 letras minusculas)
LOGIN:
hanglong
Bienvenido,  hanglong el LOGIN es correcto.
(solo con formato de 8 letras+extension de 3 (Ejemplo: Ficheros.dox)
INDIQUE LE ARCHIVO A OBTENER:
validado.txt
Obteniendo validado.txt.
TAREA 06
Fichero de texto para prueba.
Comprobada entrada.
marzo 2021.
C:\Users\juang\Documents\MyDesk\DAM\PROGRAMACION DE SERVICIOS Y PROCESOS\garciamuelas_juanantonio_PSP06_Tarea>

```

Y si ha guardado correctamente los eventos:



```

Archivo  Edici  n  Formato  Ver  Ayuda
mar 21, 2021 2:53:01 PM ValidacionRegistro <init>
CONFIGURAR: LOGIN manuel: RECHAZADO CON FORMATO NO VALIDO.
mar 21, 2021 2:54:42 PM ValidacionRegistro <init>
CONFIGURAR: LOGIN: hanglong: FORMATO CORRECTO.
mar 21, 2021 2:55:09 PM ValidacionRegistro <init>
CONFIGURAR: FILE: vino.pdf: FORMATO CORRECTO.
mar 21, 2021 2:55:09 PM ValidacionRegistro <init>
INFORMACI  N: Fichero: vino.pdf solicitado no existe.
mar 21, 2021 2:55:21 PM ValidacionRegistro <init>
CONFIGURAR: LOGIN: hanglong: FORMATO CORRECTO.
mar 21, 2021 2:57:05 PM ValidacionRegistro <init>
CONFIGURAR: FILE: validado.txt: FORMATO CORRECTO.
mar 21, 2021 2:57:05 PM ValidacionRegistro <init>
INFORMACI  N: Fichero: validado.txt solicitado existe.
mar 21, 2021 2:57:05 PM ValidacionRegistro <init>
INFORMACI  N: Fichero: validado.txt mostrado correctamente.

```

Actividad 6.2. Utilizando la aplicaci  n desarrollada en la actividad anterior, configura las pol  ticas de acceso para:

- Firmar digitalmente la aplicaci  n.
- Que s  lo pueda leer los datos del directorio **c:/datos**.

Creamos una nueva aplicaci  n con el c  digo anterior.

Compilamos y comprobamos los permisos de acceso.

Luego creamos el archivo jar, mediante el comando:

```
jar cvf ValidacionRegistro.jar ValidacionRegistro.class
```

```
C:\Users\juang\Documents\MyDesk\DAM\PROGRAMACION DE SERVICIOS Y PROCESOS\garciamuelas_juanantonio_PSP06_Tarea>
java -Djava.security.manager ValidacionRegistro
java.security.AccessControlException: access denied ("java.util.logging.LoggingPermission" "control")
    at java.security.AccessControlContext.checkPermission(Unknown Source)
    at java.security.AccessController.checkPermission(Unknown Source)
    at java.lang.SecurityManager.checkPermission(Unknown Source)
    at java.util.logging.LogManager.checkPermission(Unknown Source)
    at java.util.logging.Handler.checkPermission(Unknown Source)
    at java.util.logging.FileHandler.<init>(Unknown Source)
    at ValidacionRegistro.<init>(ValidacionRegistro.java:75)
    at ValidacionRegistro.main(ValidacionRegistro.java:185)
*****
(solo con formato de 8 letras minusculas)
LOGIN:
hanglong
Bienvenido, hanglong el LOGIN es correcto.
*****
(Formato de 8 letras+extension de 3 (Ejemplo: Ficheros.dox)
INDIQUE EL ARCHIVO A OBTENER:
validado.txt
Obteniendo validado.txt.
mar 21, 2021 7:06:06 PM ValidacionRegistro <init>
INFORMACIÖN: Fichero: validado.txt solicitado existe.
TAREA 06
Fichero de texto para prueba.
Comprobada entrada.
marzo 2021.
mar 21, 2021 7:06:06 PM ValidacionRegistro <init>
INFORMACIÖN: Fichero: validado.txt mostrado correctamente.

C:\Users\juang\Documents\MyDesk\DAM\PROGRAMACION DE SERVICIOS Y PROCESOS\garciamuelas_juanantonio_PSP06_Tarea>
jar cvf ValidacionRegistro.jar ValidacionRegistro.class
manifiesto agregado
agregando: ValidacionRegistro.class(entrada = 3640) (salida = 2009)(desinflado 44%)
```

Preparamos la firma siguiendo las pautas indicadas en el tema:

```
Keytool -genkey -alias firmar -keypass hola00 -keystore DAM -storepass distancia
```

Tras esto, nos pide unos datos (Ojo con el país, para que no dé error)

```
C:\Users\juang\Documents\MyDesk\DAM\PROGRAMACION DE SERVICIOS Y PROCESOS\garciamuelas_juanantonio_PSP06_Tarea>
keytool -genkey -alias firmar -keypass hola00 -keystore DAM -storepass distancia
¿Cuáles son su nombre y su apellido?
[Unknown]: Juan Antonio García
¿Cuál es el nombre de su unidad de organización?
[Unknown]: DAM
¿Cuál es el nombre de su organización?
[Unknown]: MEC
¿Cuál es el nombre de su ciudad o localidad?
[Unknown]: LEON
¿Cuál es el nombre de su estado o provincia?
[Unknown]: LEON
¿Cuál es el código de país de dos letras de la unidad?
[Unknown]: SP
¿Es correcto CN=Juan Antonio García, OU=DAM, O=MEC, L=LEON, ST=LEON, C=SP?
[no]: si

Warning:
El almacén de claves JKS utiliza un formato propietario. Se recomienda migrar a PKCS12, que es un formato estándar del sector que utiliza "keytool -importkeystore -srckeystore DAM -destkeystore DAM -deststoretype pkcs12"
.

C:\Users\juang\Documents\MyDesk\DAM\PROGRAMACION DE SERVICIOS Y PROCESOS\garciamuelas_juanantonio_PSP06_Tarea>
jarsigner -keystore DAM -signedjar sValidacionRegistro.jar ValidacionRegistro.jar firmar
Enter Passphrase for keystore:
jarsigner error: java.lang.RuntimeException: keystore load: Keystore was tampered with, or password was incorrect
```

Como podemos ver, si introducimos mal algún dato nos dará error. Probamos de nuevo con los datos correctos (distancia/hola00)


```
CA: Símbolo del sistema
C:\Users\juang\Documents\MyDesk\DAM\PROGRAMACION DE SERVICIOS Y PROCESOS\garciamuelas_juanantonio_PSP06_Tarea>
jarsigner -keystore DAM -signedjar sValidacionRegistro.jar ValidacionRegistro.jar firmar
Enter Passphrase for keystore:
Enter key password for firmar:
jar signed.

Warning:
The signer's certificate is self-signed.

C:\Users\juang\Documents\MyDesk\DAM\PROGRAMACION DE SERVICIOS Y PROCESOS\garciamuelas_juanantonio_PSP06_Tarea>
```

Exportamos la llave pública con la clave guardada anteriormente (distancia)

Keytool -export -keystore DAM -alias firmar -file JuanAntonio.cert

```
CA: Símbolo del sistema
C:\Users\juang\Documents\MyDesk\DAM\PROGRAMACION DE SERVICIOS Y PROCESOS\garciamuelas_juanantonio_PSP06_Tarea>
keytool -export -keystore DAM -alias firmar -file JuanAntonio.cert
Introduzca la contraseña del almacén de claves:
Certificado almacenado en el archivo <JuanAntonio.cert>

Warning:
El almacén de claves JKS utiliza un formato propietario. Se recomienda migrar a PKCS12, que es un formato estándar del sector que utiliza "keytool -importkeystore -srckeystore DAM -destkeystore DAM -deststoretype pkcs12"

C:\Users\juang\Documents\MyDesk\DAM\PROGRAMACION DE SERVICIOS Y PROCESOS\garciamuelas_juanantonio_PSP06_Tarea>
```

Ahora como indica el punto 3.6, aunque esté importado nuestro certificado, este puede dar problemas de seguridad. Para solventarlo ejecutamos el siguiente comando:

Keytool -import -alias Juan -file JuanAntonio.cert -keystore DAM

```
CA: Símbolo del sistema
C:\Users\juang\Documents\MyDesk\DAM\PROGRAMACION DE SERVICIOS Y PROCESOS\garciamuelas_juanantonio_PSP06_Tarea>
keytool -import -alias Juan -file JuanAntonio.cert -keystore DAM
Introduzca la contraseña del almacén de claves:
El certificado ya existe en el almacén de claves con el alias <firmar>
¿Aún desea agregarlo? [no]: si
Se ha agregado el certificado al almacén de claves

Warning:
El almacén de claves JKS utiliza un formato propietario. Se recomienda migrar a PKCS12, que es un formato estándar del sector que utiliza "keytool -importkeystore -srckeystore DAM -destkeystore DAM -deststoretype pkcs12"

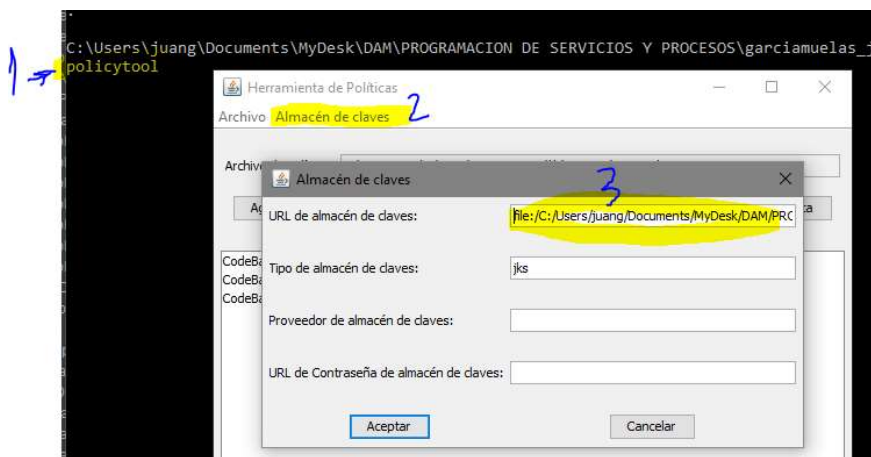
C:\Users\juang\Documents\MyDesk\DAM\PROGRAMACION DE SERVICIOS Y PROCESOS\garciamuelas_juanantonio_PSP06_Tarea>
```

Añadido el alias, verificamos las políticas de acceso para que lea sólo el directorio que indica la tarea (**c:/datos**).

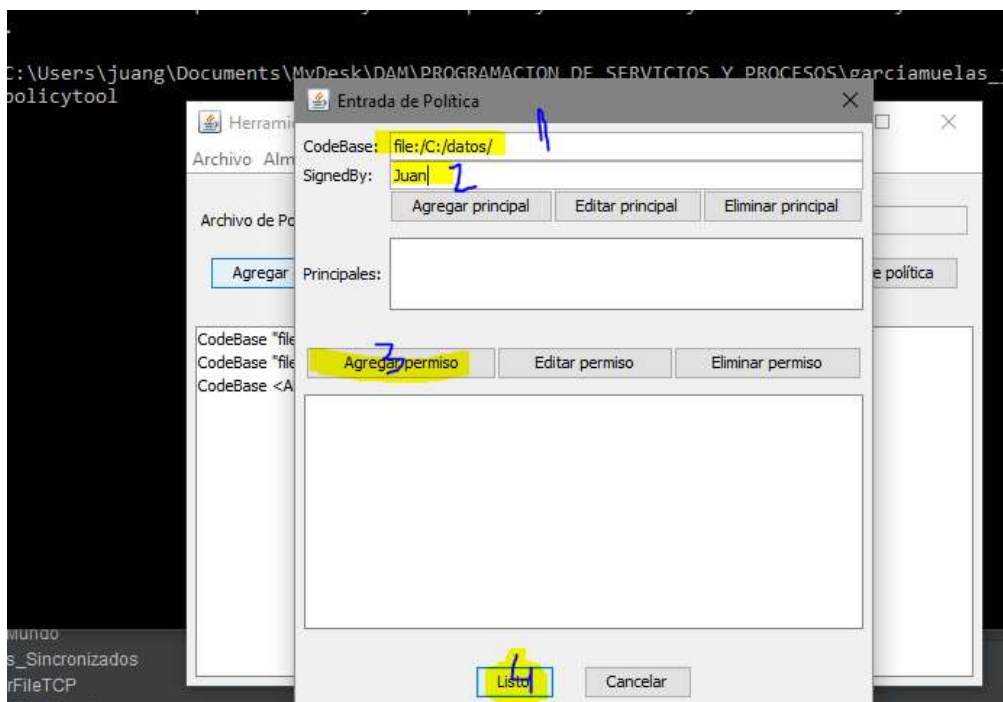
Accedemos a la herramienta de Políticas mediante el comando **policytool** (si no estábamos como administradores, mejor iniciar nueva consola para evitar errores al guardar datos).

En Archivo de Política, debemos indicar el directorio de nuestro archivo **java.policy**

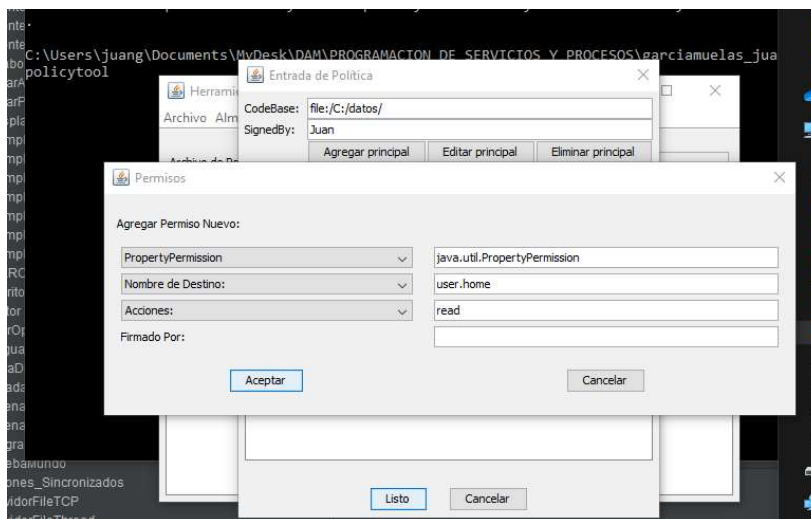
En Almacén de datos, pulsamos sobre Editar y en la url añadimos la ubicación de nuestras claves.

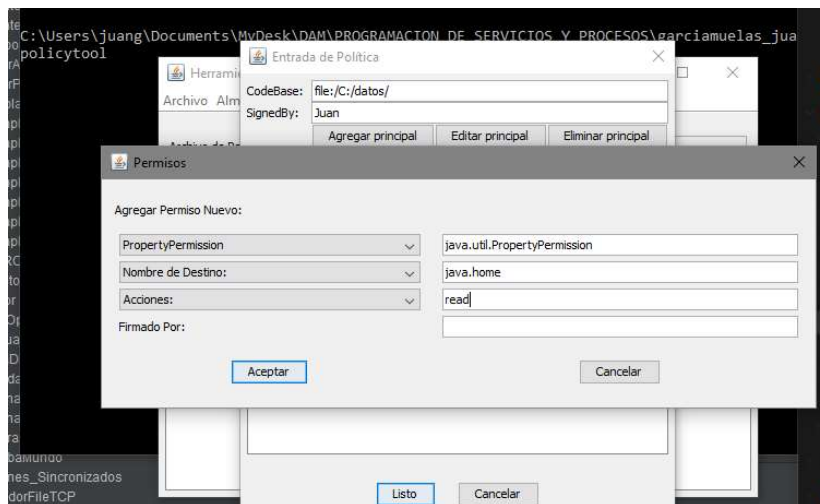


Agregamos la nueva entrada

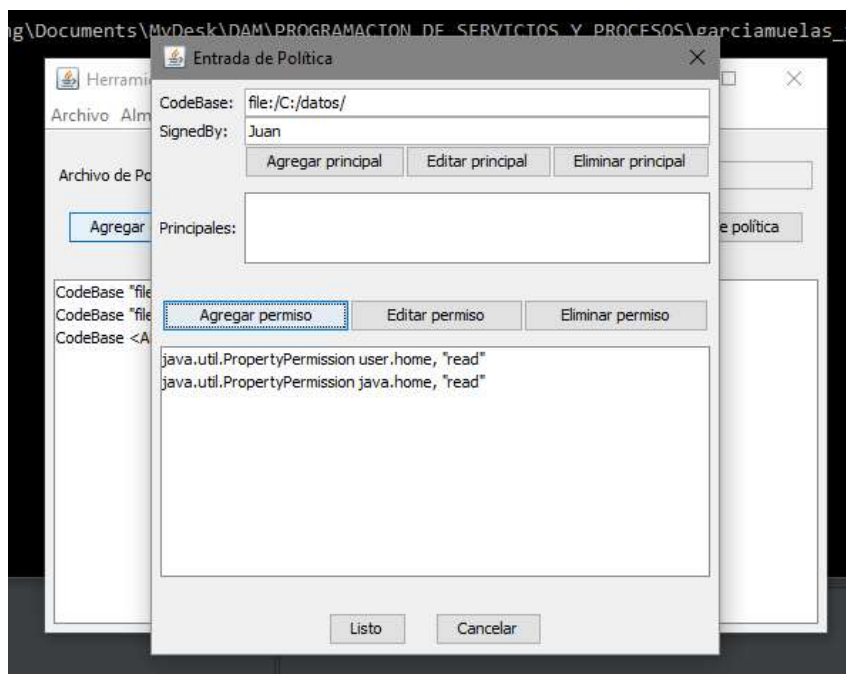


Al agregar debemos indicar los permisos según nos indica el temario en el 3.4

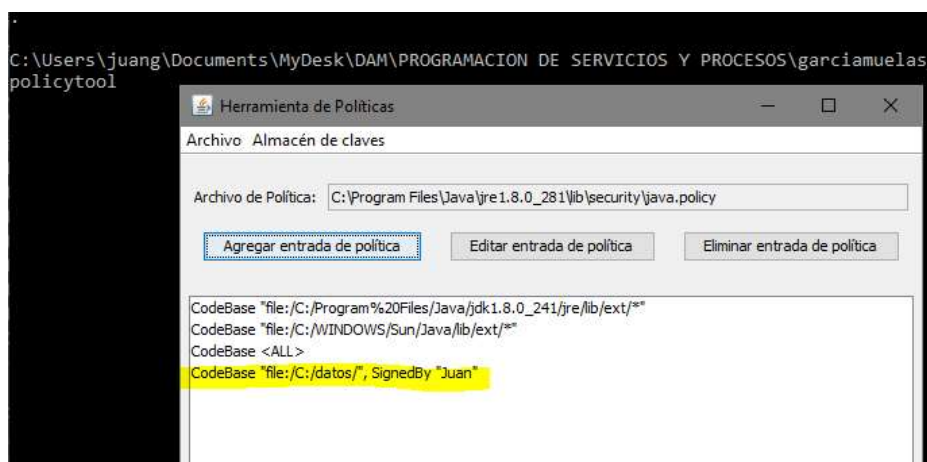




Vemos los permisos incluidos:



Al aceptar, comprobamos la correcta inserción de los permisos con nuestra firma:



Tras esto, si intentamos acceder desde un directorio que no sea el indicado, nos denegará el acceso

```
Administrador: Símbolo del sistema
C:\Users\juang\Documents\MyDesk\DAM\PROGRAMACION DE SERVICIOS Y PROCESOS\garciamuelas_juanantonio_PSP06_Tarea>java -D
java.security.manager -cp sValidacionRegistro.jar ValidacionRegistro
java.security.AccessControlException: access denied ("java.util.logging.LoggingPermission" "control")
    at java.security.AccessControlContext.checkPermission(Unknown Source)
    at java.security.AccessController.checkPermission(Unknown Source)
    at java.lang.SecurityManager.checkPermission(Unknown Source)
    at java.util.logging.LogManager.checkPermission(Unknown Source)
    at java.util.logging.Handler.checkPermission(Unknown Source)
    at java.util.logging.FileHandler.<init>(Unknown Source)
    at ValidacionRegistro.<init>(ValidacionRegistro.java:75)
    at ValidacionRegistro.main(ValidacionRegistro.java:185)
*****
(solo con formato de 8 letras minusculas)
LOGIN:
hanglong
Bienvenido, hanglong el LOGIN es correcto.
*****
(Formato de 8 letras+extension de 3 (Ejemplo: Ficheros.docx)
INDIQUE EL ARCHIVO A OBTENER:
validado.txt
Obteniendo validado.txt.
access denied ("java.io.FilePermission" "validado.txt" "read")
```