PRÁCTICA EVALUABLE UD4 - REGISTRO/LOGIN USUARIOS JAVA FX

La presente práctica tiene como objetivo realizar un aplicación en Java FX que realice el registro y login de usuarios.

REGISTRO USUARIOS

• Registra usuarios y contraseñas asociadas introducidas a través de una interfaz gráfica. La interfaz debe contener un campo de texto (tipo TextField) para el nombre del usuario (username) y otros dos campos de texto (tipo PasswordField) para la contraseña y la confirmación de ésta. Debajo de cada password debe haber un checkbox para mostrar/ocultar contraseña según se encuentre marcado o no. Asimismo, deberá contener otra etiqueta para mostrar mensajes de error/información (inicialmente con texto vacío) así como un botón de Registro para guardar la información, comprobando previamente que:

1) No contenga ningún campo vacío y tengan una longitud mínima de 5 caracteres y máximo 10. Además, el password debe contener, al menos, una letra mayúscula y un dígito.
2) Los 2 campos para introducir el password deben coincidir.

• Los registros de los usuarios se almacenan en una estructura de datos diseñada al efecto. La estructura de datos se llamará “UsersPasswordsData” y deberá ser una clase .java dentro del proyecto. La estructura puede serializarse y deserializarse.

• Cada vez que registramos un usuario se actualiza la estructura de datos y se almacena en un fichero. A la hora de registrar un nuevo usuario, debemos comprobar que no hemos alcanzado el número máximo de usuarios, en nuestro caso, diez.

• Cada vez que se inicia el componente recupera los datos del fichero, y permite seguir registrando usuarios adicionales.

LOGIN USUARIOS

a) Recopilar por la interfaz gráfica del componente el nombre de usuario (username) y la contraseña (PasswordField) introducidos por el usuario, cuando el usuario pulse un botón “Login” de la interfaz.

b) Cargar la estructura de datos UsersPasswordsData generada previamente en el registro de usuarios, almacenando las contraseñas tras convertirlas a un hash SHA256.

c) Verificar si los datos recopilados mediante el interfaz gráfico, en el apartado a), coinciden con los que hay en la estructura cargada en el apartado b) y mostrar un mensaje en una etiqueta informando del login correcto o fallido.

PANTALLA INICIAL

- En la pantalla inicial deberá mostrarse un logo con un mensaje de bienvenida en el que aparecerán 2 botones (Registro y Login) que abrirán en una nueva ventana la opción seleccionada.

OTRAS CONSIDERACIONES

- LAYOUT: GRIDPANE

La aplicación en JavaFX en cada una de las diferentes ventanas (Pantalla Inicial, Registro y Login), deben disponer de, al menos, un GRIDPANE donde deberá tener una imagen encima de los diferentes campos de texto. Se debe mantener centrado al redimensionar la pantalla, respetando las dimensiones mínimas que garanticen su usabilidad.

- ESTILOS CSS

- Las diferentes interfaces gráficas deberán contener estilos para cambiar el aspecto de:

• Fondo • Etiquetas • Entradas de texto y contraseña
• Botón • Etiqueta
• Utiliza la referencia oficial JavaFX CSS Reference Guide si tienes dudas.

Se deben utilizar:

• Hoja de estilos externa
• Selectores de tipo
• Selectores de clase
• Selectores de ID
• Cada uno de los selectores debe modificar cuatro propiedades (salvo en el caso del fondo del contenedor raíz)

VARIANTE

Aprovechar el enunciado anterior para la realización del registro/login a través de una Base de Datos.

ENTREGA. Al finalizar la práctica, el alumno deberá facilitar al profesor a través del aula virtual los siguientes ficheros:

- Proyecto JavaFx **Sin** BBDD: Estructura del nuevo proyecto javaFX generado en fichero .zip, y el .jar funcionando correctamente.
- Proyecto JavaFx **CON** BBDD: Estructura del nuevo proyecto javaFX generado en fichero .zip, y el .jar funcionando correctamente.

CONSEJOS/TIPS para la realización de la práctica:

- Estructura de Datos

```
private static final int MAX_USERS_QUANTITY = 10;
private String[][] usersPasswordsArray;
private int usersQuantity;
private String dataFilename;
```

- Serialización / Deserialización

- Serialize

```
import java.io.*;

//
FileOutputStream fileOut = new FileOutputStream(filename);
ObjectOutputStream out = new ObjectOutputStream(fileOut);
out.writeObject(this);
out.close();
fileOut.close();
//
```

- Deserialize

```
import java.io.*;

//
FileInputStream fileIn = new FileInputStream(filename);
ObjectInputStream in = new ObjectInputStream(fileIn);
DataClass data = (DataClass) in.readObject();
in.close();
fileIn.close();
//
```

- Generar un hash SHA-256:

```
import java.security.MessageDigest;

//
 try{
        final MessageDigest digest = MessageDigest.getInstance("SHA-256");
        final byte[] hash = digest.digest(newPassword.getBytes("UTF-8"));
        final StringBuilder hexString = new StringBuilder();
        for (int i = 0; i < hash.length; i++) {
           final String hex = Integer.toHexString(0xff & hash[i]);
           if(hex.length() == 1)
              hexString.append('0');
           hexString.append(hex);
        }
        newHashedPassword = hexString.toString();
     } catch(Exception ex){
        newHashedPassword = "FAILED HASH";
     }
//
```

RÚBRICA Diseño y lógica

Guía de evaluación / Rúbrica				
Porcentaje de nota	***Criterio***	***Excelente***	***Aceptable***	***Necesita mejorar***
10%	**Creación clase específica para estructura de datos**	• Se ha portado a una clase específica la estructura de datos.	-	• No se ha realizado la tarea requerida.
		1	-	0
20%	**Diseño del componente visual**	• El diseño permite recopilar la información necesaria. • El diseño permite ejecutar las acciones requeridas.	-	• El diseño no permite recopilar la información necesaria y/o ejecutar las acciones requeridas.
		1,5	-	0
50%	**Lógica de control del componente**	• El componente utiliza correctamente las estructuras de datos necesarias. • El componente contempla la serialización y deserialización de datos. • El componente verifica correctamente el usuario y contraseña aportadas en la interfaz gráfica.	• Se utilizan las estructuras de datos esperadas, aunque no correctamente. • Se serializan/deserializan los datos, pero incorrectamente. • La verificación de usuario y contraseña no son completamente correctas.	• No se ha contemplado ninguno de los requisitos para la lógica del componente.
		5	Entre 0,5 y 4,9	0
20%	**Ejecución correcta del componente**	• El componente se ejecuta correctamente. • No surgen errores o excepciones no controladas.	• El componente da fallos o excepciones no controladas durante su ejecución.	• El componente no se ejecuta.
		2	1	0

(*) En el caso de la versión con BBDD, se comprobará que la aplicación realiza el registro/login usuarios mediante el uso de BBDD creadas a tal efecto.

RÚBRICA LAYOUTS

	Item	Peso	Nivel de logro		
			Excelente	**Aceptable**	**Necesita mejorar**
1	Compilación y ejecución	**10**	El programa compila y se entrega correctamente empaquetado	El programa presenta presenta muestra advertencias al compilar	El programa no compila o no se puede ejecutar
			10	5	0
2	Uso de GridPane	**25**	Utiliza el control GridPane correctamente y define las dimensiones mediante porcentajes	Utiliza GridPane pero algunas dimensiones no se definen en porcentaje	No utiliza GridPane o no define ninguna dimensión en porcentaje
			25	12,5	0
3	Adecuación de los componentes	**10**	Utiliza los componentes adecuados sin añadir elementos innecesarios	Utiliza un componente inadecuado o añade un elemento innecesario	Utiliza más de un componente inadecuado o añade más de un elemento innecesario
			10	5	0
4	Uso de dimensiones mínimas	**15**	Respeta las dimensiones mínimas de los componentes garantizando la usabilidad	Un elemento no respeta las dimensiones mínimas	Más de un elemento no respeta las dimensiones mínimas
			15	7,5	0
5	Diseño responsivo	**20**	El diseño se mantiene centrado al redimensionar la ventana	Un componente no se mantiene centrado al redimensionar la ventana	Más de un componente no se mantiene centrado al redimensionar la ventana
			20	10	0
6	Interaccion	**10**	El programa muestra el mensaje de error de forma legible y centrada al accionar el botón	El programa muestra el mensaje de error de forma poco legible o no centrada al accionar el botón	El programa no muestra el mensaje de error al accionar el botón
			10	5	0
7	Calidad de código	**10**	El código está correctamente formateado sin líneas extras ni comentarios innecesarios. No hay errores tipográficos. Se usan paquetes personalizados.	El código contiene líneas extras o comentarios innecesarios o hay un error tipográfico	El código contiene líneas extras o comentarios innecesarios o hay más de un error tipográfico o no se usan paquetes personalizados.
			10	5	0
	Total	100	100	50	0

RÚBRICA ESTILOS CSS

Item		Peso	Nivel de logro		
			Excelente	**Aceptable**	**Necesita mejorar**
1	Utiliza estilos definidos en un fichero CSS externo	**5**	Los estilos están un fichero CSS externo, en la carpeta de recursos, dentro del paquete de la aplicación	Una de las condiciones no se cumple	No utiliza hoja de estilos externa
			5	2,5	0
2	Utiliza selectores de tipo	**25**	Utiliza selectores de tipo para modificar cuatro propiedades	Utiliza selectores de tipo para modificar menos de tres propiedades	Utiliza selectores de tipo para modificar menos de dos propiedades
			25	12,5	0
3	Utiliza selectores de clase	**25**	Utiliza selectores de clase para modificar cuatro propiedades	Utiliza selectores de clase para modificar menos de tres propiedades	Utiliza selectores de clase para modificar menos de dos propiedades
			25	12,5	0
4	Utiliza selectores de ID	**25**	Utiliza selectores de ID para modificar cuatro propiedades	Utiliza selectores de ID para modificar menos de tres propiedades	Utiliza selectores de ID para modificar menos de dos propiedades
			25	12,5	0
5	Consistencia	**20**	Los estilos aplicados son consistentes, cohesivos y sin estridencias	Una de las condiciones no se cumple	Dos de las condiciones no se cumplen
			20	10	0
	Total	100	100	50	0