1- Desarrolla una aplicación gráfica en Java (Swing) que nos permita administrar los usuarios de una aplicación. Los usuarios deberán almacenarse en una base de datos MySQL. La base de datos se denominará **gestionusuarios** y constará de la tabla **usuario**.

usuario		
PK	nombre	VARCHAR(25)
	password	VARCHAR(64)
	email	VARCHAR(50)
	activo	BIT
	rol	VARCHAR(25)
	fecha_alta	DATE
	ultimo_acceso	TIMESTAMP

Diagrama de casos de uso

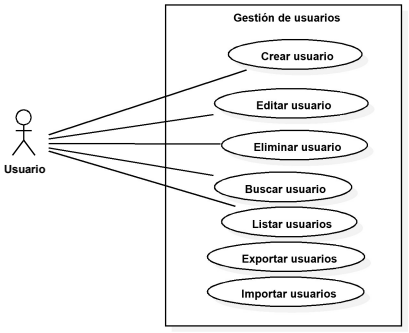
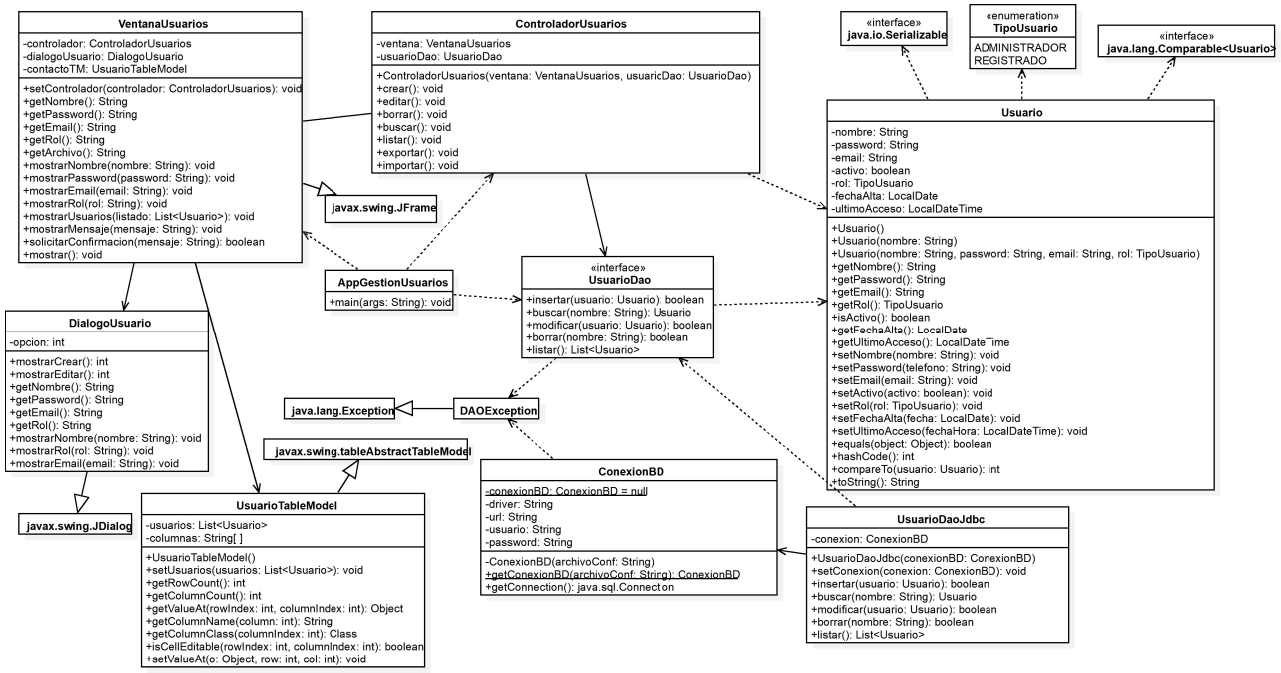


Diagrama de clases



La exportación de los usuarios se deberá hacer a formato CSV.
La importación de los usuarios se deberá hacer desde CSV.

Desarrolla las interfaces, la enumeración (**TipoUsuario**) y las clases que aparecen en el diagrama de clases, implementando los métodos que aparecen y los getter y setter correspondientes.
En la clase **Usuario** sobreescribe los métodos **toString**, **equals** y **hashcode**.
Cualquier excepción en **ConexionBD** y **UsuarioDaoJdbc** deberá capturarse y lanzar una **DaoException**.
Las excepciones llegarán hasta el controlador y serán mostradas en la vista mediante el método **mostrarMensaje(String mensaje)**.
Los datos de la conexión a la base de datos se obtendrán del fichero **conexion.properties**.

Usuario Clase que representa un usuario.

Usuario(String nombre, String password, String email, TipoUsuario rol)
Crea un usuario con los datos recibidos. El atributo **activo** lo inicializa a ***false***, el atributo **fechaAlta** lo inicializa con la fecha del sistema y el atributo **ultimoAcceso** lo inicializa a ***null***.

boolean equals(Object o)
Compara el objeto recibido con este usuario. Devuelve true si el objeto recibido es una instancia de la clase **Usuario** y su atributo nombre coincide con el atributo nombre de este usuario. En caso contrario devuelve ***false***.

int compareTo(Usuario usuario)
Compara dos usuarios. La comparación se basa en el atributo nombre.

String toString()
Devuelve una cadena compuesta por el valor de los atributos separados por comas.

UsuarioDaoJdbc

Boolean**void insertar(Usuario usuario)**
Inserta el usuario recibido en la tabla usuario.

Usuario buscar(String nombre)
Busca en la tabla usuario el registro que coincida con el nombre recibido como parámetro.

Boolean **int modificar(Usuario usuario)**
Modifica el usuario recibido en la tabla usuario. Devuelve el número de registros afectados por la operación de actualización.

Boolean **int borrar(String nombre)**
Elimina de la tabla usuario el usuario cuyo nombre sea igual al recibido como parámetro. Devuelve el número de registros afectados por la operación de actualización.

List<Usuario> listar()
Crea un ArrayList con los usuarios de la tabla usuario, ordenados por nombre, y lo devuelve.

Métodos que deberemos utilizar para dar valor a los parámetros del objeto **PreparedStatement**:

nombre	setString(1,usuario.getNombre())
password	setString(2,usuario.getPassword())
email	setString(3,usuario.getEmail())
activo	setBoolean(4,usuario.isActivo())
rol	setString(5,usuario.getRol().toString())
fechaAlta	setDate(6,Date.valueOf(usuario.getFechaAlta()))
ultimoAcceso	setTimestamp(7,Timestamp.valueOf(usuario.getUltimoAcceso())

En el caso de **ultimoAcceso** deberemos tener en cuenta que puede valer ***null***, en ese caso deberemos asignar ***null***.

Métodos que deberemos utilizar para obtener los datos del objeto **ResultSet**:

nombre	resultSet.getString("nombre")
password	resultSet.getString("password")
email	resultSet.getString("email")
activo	resultSet.getBoolean("activo")
rol	resultSet.getString("rol")
fechaAlta	resultSet.getDate("fecha_alta").toLocalDate()
ultimoAcceso	resultSet.getTimestamp("ultimo_acceso").toLocalDateTime()

En el caso de **ultimoAcceso** deberemos tener en cuenta que puede valer ***null***, en ese caso deberemos asignar ***null***.

En la clase **Controlador**, en el método **crear()**, la **password** solicitada a la ventana deberemos cifrarla antes de crear el objeto de la clase Usuario. La cifraremos en SHA-256.
En la clase **Controlador**, en el método **editar()**, la **password** solicitada a la ventana deberemos cifrarla antes de modificar el objeto de la clase Usuario. La cifraremos en SHA-256.
Para difrar la password en SHA-256 utilizaremos el método estático ***String sha256Hex(String data)*** de la clase **org.apache.commons.codec.digest.DigestUtils**.

Dependencia Maven:

```
<dependency>
    <groupId>commons-codec</groupId>
    <artifactId>commons-codec</artifactId>
    <version>1.14</version>
    <type>jar</type>
</dependency>
```

VentanaUsuarios

Diseño de **VentanaUsuarios**. En ejecución la etiqueta y el campo de texto de **password** no serán visibles.

Aspecto de las ventanas