```java
package opo.informatica.modelos;

import java.io.File;
import java.util.Date;

/**
 * RedSocial
 */
public class RedSocial {

    private String nombre;
    private Date fechaLanzamiento;
    private File logo;

    public RedSocial(){}

    public RedSocial(String nombre, Date fecha){
        this.nombre = nombre;
        this.fechaLanzamiento = fecha;
    }

    public void setNombre(String nom) {
        this.nombre = nom;
    }

    public String getNombre() {
        return this.nombre;
    }

    public void setFechaLanzamiento(Date fecha) {
        this.fechaLanzamiento = fecha;
    }

    public Date getFechaLanzamiento() {
        return this.fechaLanzamiento;
    }

    public void setLogo(File logo) {
        this.logo = logo;
    }

    public File getLogo() {
        return this.logo;
    }
}
```

```java
package opo.informatica.modelos;

import java.util.Date;

public class Usuario {

    private int usrId;
    private String nombre;
    private String apellido1;
    private String apellido2;
    private String pais;
    private int telefono;
    private String email;
    private Date fechaNacimiento;

    public Usuario(){}

    public Usuario(int id, String nombre, String apellido1, String apellido2, String pais, int
     tlf, Date fechaNac, String email){
        this.usrId = id;
        this.nombre = nombre;
        this.apellido1 = apellido1;
        this.apellido2 = apellido2;
        this.pais = pais;
        this.telefono = tlf;
        this.fechaNacimiento = fechaNac;
        this.email = email;
    }

    public void setUsrId(int id){
        this.usrId = id;
    }

    public int getUsrId(){
        return this.usrId;
    }

    public void setNombre(String nom) {
        this.nombre = nom;
    }

    public String getNombre() {
        return this.nombre;
    }

    public void setApellido1(String apellido1) {
        this.apellido1 = apellido1;
    }

    public String getApellido1(){
        return this.apellido1;
    }

    public void setApellido2(String apellido2) {
        this.apellido2 = apellido2;
    }

    public String getApellido2() {
        return this.apellido2;
    }

    public void setPais(String pais) {
        this.pais = pais;
    }

    public String getPais(){
        return this.pais;
    }

```

```java
    public void setTelefono(int tlf) {
        this.telefono = tlf;
    }

    public int getTelefono() {
        return this.telefono;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public String getEmail() {
        return this.email;
    }

    public void setFechaNacimiento(Date fecha) {
        this.fechaNacimiento = fecha;
    }

    public Date getFechaNacimiento() {
        return this.fechaNacimiento;
    }
}

```

```java
package opo.informatica.modelos;

/**
 * https://www.baeldung.com/java-new-custom-exception
 */
public class SeguidorExcepcion extends Exception {

    public SeguidorExcepcion(){
        super();
    }

    public SeguidorExcepcion(String mensaje) {
        super(mensaje);
    }

    public SeguidorExcepcion(String mensaje, Throwable err){
        super(mensaje, err);
    }
}

```

```java
package opo.informatica.accesodatos;

import java.util.List;

import opo.informatica.modelos.RedSocial;
import opo.informatica.modelos.Usuario;

public interface UsuarioDAO {

    public void alta(Usuario usr);
    public void baja(Usuario usr);
    public Usuario buscar(int id);
    public void modificar(Usuario usr);
    public boolean sigueA(Usuario usr1, Usuario usr2, RedSocial redSoc);
    public void empezarASeguir(Usuario usr1, Usuario usr2, RedSocial redSoc);
    public void dejarDeSeguir(Usuario usr1, Usuario usr2, RedSocial redSoc);
    public List<Usuario> obtenerSeguidores(Usuario usr, RedSocial redSoc);
    public List<Usuario> obtenerSeguidosPor(Usuario usr, RedSocial redSoc);
}

```

```java
package opo.informatica.accesodatos;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;

import opo.informatica.modelos.RedSocial;
import opo.informatica.modelos.SeguidorExcepcion;
import opo.informatica.modelos.Usuario;

public class JDBCUsuarioDAO implements UsuarioDAO {

    private static final String jdbcURL = "jdbc:mysql://localhost:3307/bdRedesSociales";
    private static final String jdbcUsername = "oposicion";
    private static final String jdbcPassword = "1234";

    private static final String SQL_Obtener_Seguidores = "  SELECT u.* \r\n" + //
                                                         "    FROM Usuario u \r\n" + //
                                                         "   WHERE u.usrId IN (  SELECT s.seguidorId \r\n" + //
                                                         "                         FROM Seguidores s \r\n" + //
                                                         "                        WHERE s.seguidoId = ? \r\n" + //
                                                         "                          AND s.nomRS = ?)";

    private static final String SQL_Esta_Suscrito = "  SELECT count(*) \r\n" + //
                                                    "    FROM Suscripcion sus \r\n" + //
                                                    "   WHERE sus.usrId = ? \r\n" + //
                                                    "     AND sus.nomRs = ?";

    private static final String SQL_Obtener_Usuarios = "  SELECT * \r\n" +
                                                       "    FROM Usuario ";

    private static final String SQL_Obtener_Redes_Sociales = "  SELECT * \r\n" +
                                                             "    FROM RedSocial ";

    @Override
    public void alta(Usuario usr) {
        // TODO Auto-generated method stub
        throw new UnsupportedOperationException("Unimplemented method 'alta'");
    }

    @Override
    public void baja(Usuario usr) {
        // TODO Auto-generated method stub
        throw new UnsupportedOperationException("Unimplemented method 'baja'");
    }

    @Override
    public Usuario buscar(int id) {
        // TODO Auto-generated method stub
        throw new UnsupportedOperationException("Unimplemented method 'buscar'");
    }

    @Override
    public void modificar(Usuario usr) {
        // TODO Auto-generated method stub
        throw new UnsupportedOperationException("Unimplemented method 'modificar'");
    }

    @Override
```

```
66      public boolean sigueA(Usuario usr1, Usuario usr2, RedSocial redSoc) {
67          // TODO Auto-generated method stub
68          throw new UnsupportedOperationException("Unimplemented method 'sigueA'");
69      }
70
71      @Override
72      public void empezarASeguir(Usuario usr1, Usuario usr2, RedSocial redSoc) {
73          // TODO Auto-generated method stub
74          throw new UnsupportedOperationException("Unimplemented method 'empezarASeguir'");
75      }
76
77      @Override
78      public void dejarDeSeguir(Usuario usr1, Usuario usr2, RedSocial redSoc) {
79          // TODO Auto-generated method stub
80          throw new UnsupportedOperationException("Unimplemented method 'dejarDeSeguir'");
81      }
82
83      /**
84       * Obtener, mediante unconector ODBC (JDBC o similar), una lista de seguidores de un
         usuario
85       * dado en una red social dada.
86       * En particular, si un usuario no está suscrito a una redsocial y tratamos de obtener
         sus
87       * seguidores en dicha red social, deberá lanzarse la excepción SeguidorExcepcion.
88       */
89      @Override
90      public List<Usuario> obtenerSeguidores(Usuario usr, RedSocial redSoc) {
91          List<Usuario> seguidores = new ArrayList<>();
92
93          // Obtenemos la conexión JDBC
94          // Utilizammos el try-with-resources para evitar cerrar nosotros los recursos en caso
            de Excepción
95          try (Connection connection = this.getConnection();
96              // Creamos la sentencia a ejecutar por la BBDD
97              PreparedStatement ps = connection.prepareStatement(SQL_Obtener_Seguidores);
98          ) {
99
100             // En particular, si un usuario no está suscrito a una redsocial y tratamos de
                obtener sus
101             // seguidores en dicha red social, deberá lanzarse la excepción SeguidorExcepcion.
102             if (!this.estaSuscrito(usr, redSoc)) {
103                 String msgError = "El usuario "+usr.getNombre()+ " "+usr.getApellido1() + " "
                    +usr.getApellido2() + " no está suscrito a " + redSoc.getNombre() + ". Por lo
                    que no puede tener seguidores en esta Red Social.";
104                 throw new SeguidorExcepcion(msgError);
105             }
106
107             // Pasamos los parámetros para filtrar la sentencia a la BBDD
108             ps.setInt(1, usr.getUsrId());
109             ps.setString(2, redSoc.getNombre());
110
111             // Pedimos a la BBDD que ejecute la consulta
112             // y nos devuelva un objeto con el conjunto de resultados
113             ResultSet rs = ps.executeQuery();
114
115             // Procesamos el conjunto de resultados para devolverlos en la
116             // estructura de salida de este método
117             while(rs.next()) {
118                 int id = rs.getInt("usrId");
119                 String nom = rs.getString("nombre");
120                 String apellido1 = rs.getString("apellido1");
121                 String apellido2 = rs.getString("apellido1");
122                 String pais = rs.getString("pais");
123                 int tlf = rs.getInt("tel");
124                 Date fechaNac = new Date(rs.getDate("fechaNac").getTime());
125                 String email = rs.getString("email");
126
127                 seguidores.add(new Usuario(id, nom, apellido1, apellido2, pais, tlf, fechaNac,
                     email));
```

```
            }

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

    @Override
    public List<Usuario> obtenerSeguidosPor(Usuario usr, RedSocial redSoc) {
        // TODO Auto-generated method stub
        throw new UnsupportedOperationException("Unimplemented method 'obtenerSeguidosPor'");
    }

    /**
     * Obtener un conector ODBC (JDBC o similar)
     * @return
     */
    private Connection getConnection(){
        Connection connection = null;
        try {
            Class.forName("com.mysql.jdbc.Driver");
            connection = DriverManager.getConnection(jdbcURL,jdbcUsername,jdbcPassword);
        } catch (SQLException | ClassNotFoundException e) {
            e.printStackTrace();
        }
        return connection;
    }

    /**
     * Comprueba si un usuario está suscrito a una redsocial
     * @param usr
     * @param redSoc
     * @return
     */
    private boolean estaSuscrito(Usuario usr, RedSocial redSoc) {
        boolean result = false;

        // Obtenemos la conexión JDBC
        // Utilizamos el try-with-resources para evitar cerrar nosotros los rescursos en caso
        de Excepción
        try (Connection connection = this.getConnection();
            // Creamos la sentencia a ejecutar por la BBDD
            PreparedStatement ps = connection.prepareStatement(SQL_Esta_Suscrito);
        ) {
            // Pasamos los parámetros para filtrar la consulta a la BBDD
            ps.setInt(1, usr.getUsrId());
            ps.setString(2, redSoc.getNombre());

            // Pedimos a la BBDD que ejecute la consulta
            // y nos devuelva un objeto con el conjunto de resultados
            ResultSet rs = ps.executeQuery();

            // Procesamos el conjunto de resultados para devolverlos en la
            // estructura de salida de este método
            while(rs.next()){
                // https://www.tutorialspoint.com/how-to-get-the-row-count-in-jdbc
                // Al ser un SELECT COUNT(*) pedimos el resultado de la 1ª columna
                int respuesta = rs.getInt(1);
                if (respuesta > 0){
                    result = true;
                }
            }

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

```java

    public List<Usuario> obtenerUsuarios(){
        List<Usuario> usuarios = new ArrayList<>();

        // Obtenemos la conexión JDBC
        // Utilizamos el try-with-resources para evitar cerrar nosotros los recursos en caso de Excepción.
        try (Connection connection = this.getConnection();
            // Creamos la sentencia a ejecutar en la BBDD
            Statement statement = connection.createStatement();
        ) {
            // Pedimos a la BBDD que ejecute la consulta que le pasamos en la sentencia
            // y nos devuelva un objeto con el conjunto de resultados
            ResultSet rs = statement.executeQuery(SQL_Obtener_Usuarios);

            // Procesamos el conjunto de resultados para devolverlos en la
            // estructura de salida de este método
            while(rs.next()){
                int id = rs.getInt("usrId");
                String nom = rs.getString("nombre");
                String apellido1 = rs.getString("apellido1");
                String apellido2 = rs.getString("apellido1");
                String pais = rs.getString("pais");
                int tlf = rs.getInt("tel");
                Date fechaNac = new Date(rs.getDate("fechaNac").getTime());
                String email = rs.getString("email");

                usuarios.add(new Usuario(id, nom, apellido1, apellido2, pais, tlf, fechaNac, email));
            }

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

        return usuarios;
    }

    public List<RedSocial> obtenerRedesSociales() {
        List<RedSocial> redes = new ArrayList<>();

        // Obtenemos la conexión JDBC
        // Utilizamos el try-with-resources para evitar cerrar nosotros los recursos en caso de Excepción.
        try(Connection connection = this.getConnection();
            // Creamos la sentencia a ejecutar en la BBDD
            Statement statement = connection.createStatement();
        ){
            // Pedimos a la BBDD que ejecute la consulta que le pasamos en la sentencia
            // y nos devuelva un objeto con el conjunto de resultados
            ResultSet rs = statement.executeQuery(SQL_Obtener_Redes_Sociales);

            // Procesamos el conjunto de resultados para devolverlos en la
            // estrucutura de salida de este método
            while(rs.next()){
                String nombre = rs.getString("nombre");
                //String url = rs.getString("url");
                Date fechaLanzamiento = new Date(rs.getDate("fechaLanzamiento").getTime());
                // https://www.mysqltutorial.org/mysql-jdbc-blob
                //File logo = new File

                redes.add(new RedSocial(nombre, fechaLanzamiento));
            }

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

```

```
262             return redes;
263         }
264     }
265
```

```java
package opo.informatica;

import java.util.List;

import opo.informatica.accesodatos.JDBCUsuarioDAO;
import opo.informatica.modelos.RedSocial;
import opo.informatica.modelos.Usuario;

public class Prueba {

    public static void main(String[] args) {
        JDBCUsuarioDAO dao = new JDBCUsuarioDAO();

        List<Usuario> usuarios = dao.obtenerUsuarios();
        List<RedSocial> redeRedSociales = dao.obtenerRedesSociales();

        System.out.println("Mostramos la lista de seguidores para cada usuario y red Social");
        for (Usuario usuario : usuarios) {
        int orden = 1;
          for (RedSocial redSocial : redeRedSociales) {
            System.out.println("## ******************** ##");
            System.out.println("Seguidores de "+usuario.getNombre()+" "+usuario.getApellido1
            ()+" en "+redSocial.getNombre());

            List<Usuario> seguidores = dao.obtenerSeguidores(usuario, redSocial);
            for (Usuario seguidor : seguidores) {
                System.out.println(orden++ +" - " + seguidor.getNombre() + " " + seguidor.
                getApellido1() +" - " +seguidor.getPais() );
            }
          }
        }
    }
}

```

```
SALIDA:

PS C:\Proyectos\Java\Servlets\ExamenInformatica2021Ej3>  c:; cd 'c:\Proyectos\Java\Servlets\ExamenInformatica2021Ej3'; & 'C:\Program Files\Java\jdk1.8.0_361\bin\java.exe' '-cp' 'C:\Users\toppe\AppData\Local\Temp\cp_aw6cmul7wod0w16r9icy51fx0.jar' 'opo.informatica.Prueba'
Loading class `com.mysql.jdbc.Driver'. This is deprecated. The new driver class is `com.mysql.cj.jdbc.Driver'. The driver is automatically registered via the SPI and manual loading of the driver class is generally unnecessary.
Mostramos la lista de seguidores para cada usuario y red Social
## ******************** ##
Seguidores de Alberto MartÃ-nez en facebook
1 - JosÃ© MarÃ-a ViÃ±uelas - esp
2 - Cristina Lorente - esp
## ******************** ##
Seguidores de Alberto MartÃ-nez en instagram
opo.informatica.modelos.SeguidorExcepcion: El usuario Alberto MartÃ-nez MartÃ-nez no está suscrito a instagram. Por lo que no puede tener seguidores en esta Red Social.
        at opo.informatica.accesodatos.JDBCUsuarioDAO.obtenerSeguidores(JDBCUsuarioDAO.java:104)
        at opo.informatica.Prueba.main(Prueba.java:24)
        Suppressed: java.lang.IllegalArgumentException: Self-suppression not permitted
                at java.lang.Throwable.addSuppressed(Throwable.java:1082)
                at opo.informatica.accesodatos.JDBCUsuarioDAO.obtenerSeguidores(JDBCUsuarioDAO.java:130)
                ... 1 more
        [CIRCULAR REFERENCE:opo.informatica.modelos.SeguidorExcepcion: El usuario Alberto MartÃ-nez MartÃ-nez no está suscrito a instagram. Por lo que no puede tener seguidores en esta Red Social.]
## ******************** ##
Seguidores de Alberto MartÃ-nez en linkedIn
opo.informatica.modelos.SeguidorExcepcion: El usuario Alberto MartÃ-nez MartÃ-nez no está suscrito a linkedIn. Por lo que no puede tener seguidores en esta Red Social.
        at opo.informatica.accesodatos.JDBCUsuarioDAO.obtenerSeguidores(JDBCUsuarioDAO.java:104)
        at opo.informatica.Prueba.main(Prueba.java:24)
        Suppressed: java.lang.IllegalArgumentException: Self-suppression not permitted
                at java.lang.Throwable.addSuppressed(Throwable.java:1082)
                at opo.informatica.accesodatos.JDBCUsuarioDAO.obtenerSeguidores(JDBCUsuarioDAO.java:130)
                ... 1 more
        [CIRCULAR REFERENCE:opo.informatica.modelos.SeguidorExcepcion: El usuario Alberto MartÃ-nez MartÃ-nez no está suscrito a linkedIn. Por lo que no puede tener seguidores en esta Red Social.]
## ******************** ##
Seguidores de MarÃ-a Ã?ngeles Robledo en facebook
opo.informatica.modelos.SeguidorExcepcion: El usuario MarÃ-a Ã?ngeles Robledo Robledo no está suscrito a facebook. Por lo que no puede tener seguidores en esta Red Social.
        at opo.informatica.accesodatos.JDBCUsuarioDAO.obtenerSeguidores(JDBCUsuarioDAO.java:104)
        at opo.informatica.Prueba.main(Prueba.java:24)
        Suppressed: java.lang.IllegalArgumentException: Self-suppression not permitted
                at java.lang.Throwable.addSuppressed(Throwable.java:1082)
                at opo.informatica.accesodatos.JDBCUsuarioDAO.obtenerSeguidores(JDBCUsuarioDAO.java:130)
                ... 1 more
        [CIRCULAR REFERENCE:opo.informatica.modelos.SeguidorExcepcion: El usuario MarÃ-a Ã?ngeles Robledo Robledo no está suscrito a facebook. Por lo que no puede tener seguidores en esta Red Social.]
## ******************** ##
Seguidores de MarÃ-a Ã?ngeles Robledo en instagram
1 - Cristina Lorente - esp
## ******************** ##
Seguidores de MarÃ-a Ã?ngeles Robledo en linkedIn
opo.informatica.modelos.SeguidorExcepcion: El usuario MarÃ-a Ã?ngeles Robledo Robledo no está suscrito a linkedIn. Por lo que no puede tener seguidores en esta Red Social.
        at opo.informatica.accesodatos.JDBCUsuarioDAO.obtenerSeguidores(JDBCUsuarioDAO.java:104)
        at opo.informatica.Prueba.main(Prueba.java:24)
```

```
        Suppressed: java.lang.IllegalArgumentException: Self-suppression not permitted
                at java.lang.Throwable.addSuppressed(Throwable.java:1082)
                at opo.informatica.accesodatos.JDBCUsuarioDAO.obtenerSeguidores(JDBCUsuarioDAO
                .java:130)
                ... 1 more
        [CIRCULAR REFERENCE:opo.informatica.modelos.SeguidorExcepcion: El usuario MarÃ-a Ã?
        ngeles Robledo Robledo no está suscrito a linkedIn. Por lo que no puede tener
        seguidores en esta Red Social.]
## ******************** ##
Seguidores de JosÃ© MarÃ-a ViÃ±uelas en facebook
1 - Alberto MartÃ-nez - esp
## ******************** ##
Seguidores de JosÃ© MarÃ-a ViÃ±uelas en instagram
opo.informatica.modelos.SeguidorExcepcion: El usuario JosÃ© MarÃ-a ViÃ±uelas ViÃ±uelas no est
á suscrito a instagram. Por lo que no puede tener seguidores en esta Red Social.
        at opo.informatica.accesodatos.JDBCUsuarioDAO.obtenerSeguidores(JDBCUsuarioDAO.java:
        104)
        at opo.informatica.Prueba.main(Prueba.java:24)
        Suppressed: java.lang.IllegalArgumentException: Self-suppression not permitted
                at java.lang.Throwable.addSuppressed(Throwable.java:1082)
                at opo.informatica.accesodatos.JDBCUsuarioDAO.obtenerSeguidores(JDBCUsuarioDAO
                .java:130)
                ... 1 more
        [CIRCULAR REFERENCE:opo.informatica.modelos.SeguidorExcepcion: El usuario JosÃ© MarÃ-a
         ViÃ±uelas ViÃ±uelas no está suscrito a instagram. Por lo que no puede tener
        seguidores en esta Red Social.]
## ******************** ##
Seguidores de JosÃ© MarÃ-a ViÃ±uelas en linkedIn
opo.informatica.modelos.SeguidorExcepcion: El usuario JosÃ© MarÃ-a ViÃ±uelas ViÃ±uelas no est
á suscrito a linkedIn. Por lo que no puede tener seguidores en esta Red Social.
        at opo.informatica.accesodatos.JDBCUsuarioDAO.obtenerSeguidores(JDBCUsuarioDAO.java:
        104)
        at opo.informatica.Prueba.main(Prueba.java:24)
        Suppressed: java.lang.IllegalArgumentException: Self-suppression not permitted
                at java.lang.Throwable.addSuppressed(Throwable.java:1082)
                at opo.informatica.accesodatos.JDBCUsuarioDAO.obtenerSeguidores(JDBCUsuarioDAO
                .java:130)
                ... 1 more
        [CIRCULAR REFERENCE:opo.informatica.modelos.SeguidorExcepcion: El usuario JosÃ© MarÃ-a
         ViÃ±uelas ViÃ±uelas no está suscrito a linkedIn. Por lo que no puede tener seguidores
         en esta Red Social.]
## ******************** ##
Seguidores de Cristina Lorente en facebook
## ******************** ##
Seguidores de Cristina Lorente en instagram
1 - MarÃ-a Ã?ngeles Robledo - esp
## ******************** ##
Seguidores de Cristina Lorente en linkedIn
opo.informatica.modelos.SeguidorExcepcion: El usuario Cristina Lorente Lorente no está
suscrito a linkedIn. Por lo que no puede tener seguidores en esta Red Social.
        at opo.informatica.accesodatos.JDBCUsuarioDAO.obtenerSeguidores(JDBCUsuarioDAO.java:
        104)
        at opo.informatica.Prueba.main(Prueba.java:24)
        Suppressed: java.lang.IllegalArgumentException: Self-suppression not permitted
                at java.lang.Throwable.addSuppressed(Throwable.java:1082)
                at opo.informatica.accesodatos.JDBCUsuarioDAO.obtenerSeguidores(JDBCUsuarioDAO
                .java:130)
                ... 1 more
        [CIRCULAR REFERENCE:opo.informatica.modelos.SeguidorExcepcion: El usuario Cristina
        Lorente Lorente no está suscrito a linkedIn. Por lo que no puede tener seguidores en
        esta Red Social.]
PS C:\Proyectos\Java\Servlets\ExamenInformatica2021Ej3>
```

```java
package net.javaguides.usermanagement.model;

public class User {
    protected int id;
    protected String name;
    protected String email;
    protected String country;

    public User() {
    }

    public User(String name, String email, String country) {
        this.name = name;
        this.email = email;
        this.country = country;
    }

    public User(int id, String name, String email, String country) {
        this.id = id;
        this.name = name;
        this.email = email;
        this.country = country;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public String getCountry() {
        return country;
    }

    public void setCountry(String country) {
        this.country = country;
    }

}
```

```java
package net.javaguides.usermanagement.dao;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

import net.javaguides.usermanagement.model.User;

public class UserDAO {
    private static final String jdbcURL = "jdbc:mysql://localhost:3307/demo";
    private static final String jdbcUsername = "javaguides";
    private static final String jdbcPassword = "1234";

    private static final String INSERT_USERS_SQL = "INSERT INTO usuarios (name, email,
    country) "
    + "VALUES (?, ?, ?);";
    private static final String SELECT_USER_BY_ID = "SELECT * FROM usuarios WHERE id = ?";
    private static final String SELECT_ALL_USERS = "SELECT * FROM usuarios";
    private static final String DELETE_USERS_SQL = "DELETE FROM usuarios WHERE id = ?;";
    private static final String UPDATE_USERS_SQL= "UPDATE usuarios SET name = ?, email = ?,
    country = ? WHERE id = ?;";

    public UserDAO() {}

    protected Connection getConnection() {
        Connection connection = null;
        try {
            Class.forName("com.mysql.jdbc.Driver");
            connection = DriverManager.getConnection(jdbcURL, jdbcUsername, jdbcPassword);
        } catch (SQLException | ClassNotFoundException e) {
            e.printStackTrace();
        }

        return connection;
    }

    public void insertUSer(User user) throws SQLException {
        System.out.println(INSERT_USERS_SQL);
        // try-with-resource statement will auto close the connection.
        try (Connection connection = getConnection(); PreparedStatement preparedStatement =
        connection.prepareStatement(INSERT_USERS_SQL)) {
            preparedStatement.setString(1, user.getName());
            preparedStatement.setString(2,user.getEmail());
            preparedStatement.setString(3,user.getCountry());
            System.out.println(preparedStatement);
            preparedStatement.executeUpdate();
        } catch (SQLException e) {
            printSQLException(e);
        }
    }

    public User selecUser(int id) {
        User user = null;
        // Step 1: Establishing a Connection
        try (Connection connection = getConnection();
             PreparedStatement preparedStatement = connection.prepareStatement(
             SELECT_USER_BY_ID)
        ) {
            // Step 2: Create a statement using connection object
            preparedStatement.setInt(1, id);
            System.out.println(preparedStatement);
            // Step 3: Execute the query or update query
            ResultSet rs = preparedStatement.executeQuery();

            // Step 4: Process the ResultSet object.
```

```
66              while (rs.next()) {
67                  String name = rs.getString("name");
68                  String email = rs.getString("email");
69                  String country = rs.getString("country");
70                  user = new User(id, name, email, country);
71              }
72          } catch (SQLException e) {
73              printSQLException(e);
74          }
75
76          return user;
77      }
78
79      public List<User> selectAllUsers() {
80
81          // using try-with-resources to avoid closing resources (boiler plate code)
82          List<User> users = new ArrayList<>();
83          // Step 1: Establishing a Connection
84          try (Connection connection = getConnection();
85               // Step 2: Create a statement using connection object
86               PreparedStatement preparedStatement = connection.prepareStatement(
                 SELECT_ALL_USERS)
87          ) {
88              System.out.println(preparedStatement);
89              // Step 3: Execute the query or update query
90              ResultSet rs = preparedStatement.executeQuery();
91
92              // Step 4: Process the ResultSet object.
93              while (rs.next()) {
94                  int id = rs.getInt("id");
95                  String name = rs.getString("name");
96                  String email = rs.getString("email");
97                  String country = rs.getString("country");
98                  users.add(new User(id, name, email, country));
99              }
100
101         } catch (SQLException e) {
102             printSQLException(e);
103         }
104
105         return users;
106     }
107
108     public boolean deleteUser(int id) throws SQLException {
109         boolean rowDeleted = false;
110         try (Connection connection = getConnection(); PreparedStatement statement = connection
            .prepareStatement(DELETE_USERS_SQL)) {
111             statement.setInt(1, id);
112             rowDeleted = statement.executeUpdate() > 0;
113         } catch (SQLException e) {
114             printSQLException(e);
115         }
116
117         return rowDeleted;
118     }
119
120     public boolean updateUser(User user) throws SQLException {
121         boolean rowUpdated = false;
122         try (Connection connection = getConnection(); PreparedStatement statement = connection
            .prepareStatement(UPDATE_USERS_SQL)) {
123             statement.setString(1, user.getName());
124             statement.setString(2, user.getEmail());
125             statement.setString(3, user.getCountry());
126             statement.setInt(4, user.getId());
127
128             rowUpdated = statement.executeUpdate() > 0;
129         } catch (SQLException e) {
130             printSQLException(e);
131         }
```

```
132
133             return rowUpdated;
134         }
135
136         private void printSQLException(SQLException ex) {
137             for (Throwable e: ex) {
138                 if (e instanceof SQLException) {
139                     e.printStackTrace(System.err);
140                     System.err.println("SQLState: " + ((SQLException)e).getSQLState());
141                     System.err.println("Error Code: " + ((SQLException)e).getErrorCode());
142                     System.err.println("Message: " + e.getMessage());
143                     Throwable t = ex.getCause();
144                     while (t != null) {
145                         System.out.println("Cause: " + t);
146                         t = t.getCause();
147                     }
148                 }
149             }
150         }
151     }
152
```

```java
package net.javaguides.usermanagement.web;

import java.io.IOException;
import java.sql.SQLException;
import java.util.List;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import net.javaguides.usermanagement.dao.UserDAO;
import net.javaguides.usermanagement.model.User;

@WebServlet("/")
public class UserServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
    private UserDAO userDAO;

    public void init() {
        userDAO = new UserDAO();
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
    ServletException, IOException {
        doGet(request, response);
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
    ServletException, IOException {
        String action = request.getServletPath();

        try {
            switch (action) {
                case "/new":
                    showNewForm(request, response);
                    break;
                case "/insert":
                    insertUser(request, response);
                    break;
                case "/delete":
                    deleteUser(request, response);
                    break;
                case "/edit":
                    showEditForm(request, response);
                    break;
                case "/update":
                    updateUser(request, response);
                    break;
                default:
                    listUser(request, response);
                    break;
            }
        } catch (SQLException ex) {
            throw new ServletException(ex);
        }
    }

    private void listUser(HttpServletRequest request, HttpServletResponse response) throws
    SQLException, ServletException, IOException {
        List<User> listUser = userDAO.selectAllUsers();
        request.setAttribute("listUser", listUser);
        RequestDispatcher dispatcher = request.getRequestDispatcher("user-list.jsp");
        dispatcher.forward(request, response);
    }

    private void showNewForm(HttpServletRequest request, HttpServletResponse response) throws
```

```
        ServletException, IOException {
67          RequestDispatcher dispatcher = request.getRequestDispatcher("user-form.jsp");
68          dispatcher.forward(request, response);
69      }
70
71      private void showEditForm(HttpServletRequest request, HttpServletResponse response) throws
         SQLException, ServletException, IOException {
72          int id = Integer.parseInt(request.getParameter("id"));
73          User existingUser = userDAO.selecUser(id);
74          RequestDispatcher dispatcher = request.getRequestDispatcher("user-form.jsp");
75          request.setAttribute("user", existingUser);
76          dispatcher.forward(request, response);
77      }
78
79      private void insertUser(HttpServletRequest request, HttpServletResponse response) throws
        SQLException, IOException {
80          String name = request.getParameter("name");
81          String email = request.getParameter("email");
82          String country = request.getParameter("country");
83          User newUSer = new User(name, email, country);
84          userDAO.insertUSer(newUSer);
85          response.sendRedirect("list");
86      }
87
88      private void updateUser(HttpServletRequest request, HttpServletResponse response) throws
        SQLException, IOException {
89          int id = Integer.parseInt(request.getParameter("id"));
90          String name = request.getParameter("name");
91          String email = request.getParameter("email");
92          String country = request.getParameter("country");
93
94          User user = new User(id, name, email, country);
95          userDAO.updateUser(user);
96          response.sendRedirect("list");
97      }
98
99      private void deleteUser(HttpServletRequest request, HttpServletResponse response) throws
        SQLException, IOException {
100         int id = Integer.parseInt(request.getParameter("id"));
101         userDAO.deleteUser(id);
102         response.sendRedirect("list");
103     }
104 }
105
```