APP CONNECT TO PEOPLE

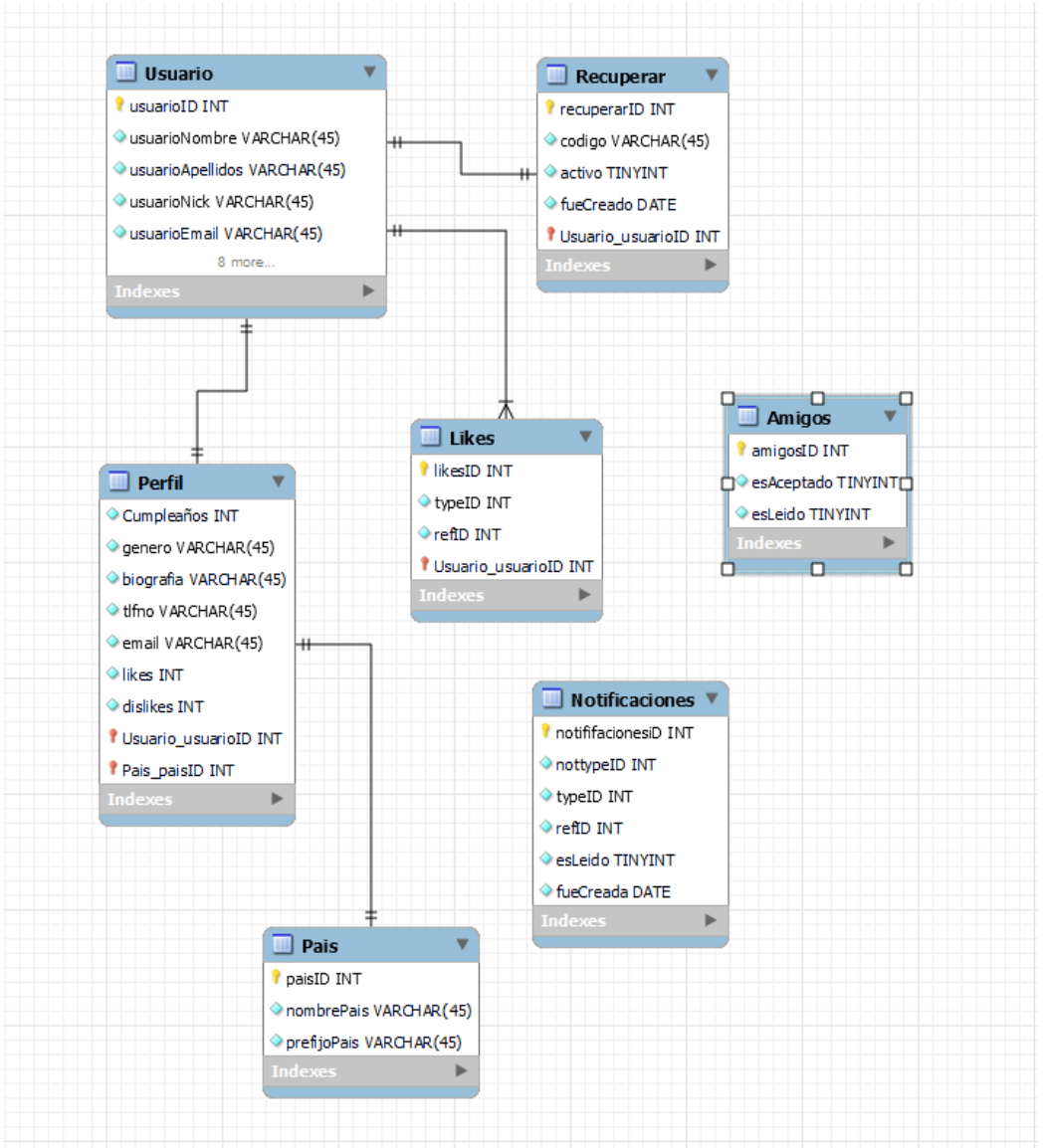
Jim Zúñiga

INTRODUCCION

Una aplicación diseñada para que la gente conecte con todo el mundo. Pero a diferencia de otras, que sean con los filtros que uno quiere elegir, es decir, si un usuario solo quiere interactuar con gente de ideología centro-izquierda, podrá hacerlo, si solo quiere llevarse con ideología derecha también podrá.

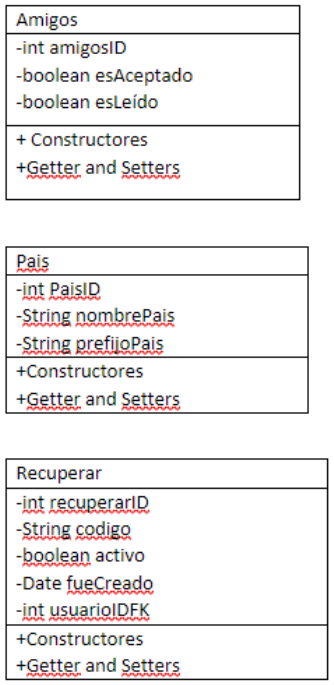
Que alguien se sienta seguro, solo quien deje el usuario tendrá permisos para visualizar el perfil de él, y viceversa, teniendo así una gran privacidad en todos los sentidos.

Modelo BBDD


Usuario
usuarioID INT
usuarioNombre VARCHAR(45)
usuarioApellidos VARCHAR(45)
usuarioNick VARCHAR(45)
usuarioEmail VARCHAR(45)
8 more...
Indexes
Recuperar
recuperarID INT
codigo VARCHAR(45)
activo TINYINT
fueCreado DATE
Usuario_usuarioID INT
Indexes
Perfil
Cumpleaños INT
genero VARCHAR(45)
biografia VARCHAR(45)
tlfno VARCHAR(45)
email VARCHAR(45)
likes INT
dislikes INT
Usuario_usuarioID INT
Pais_paisID INT
Indexes
Likes
likesID INT
typeID INT
refID INT
Usuario_usuarioID INT
Indexes
Amigos
amigosID INT
esAceptado TINYINT
esLeido TINYINT
Indexes
Notificaciones
notififacionesiD INT
nottypeID INT
typeID INT
refID INT
esLeido TINYINT
fueCreada DATE
Indexes
Pais
paisID INT
nombrePais VARCHAR(45)
prefijoPais VARCHAR(45)
Indexes

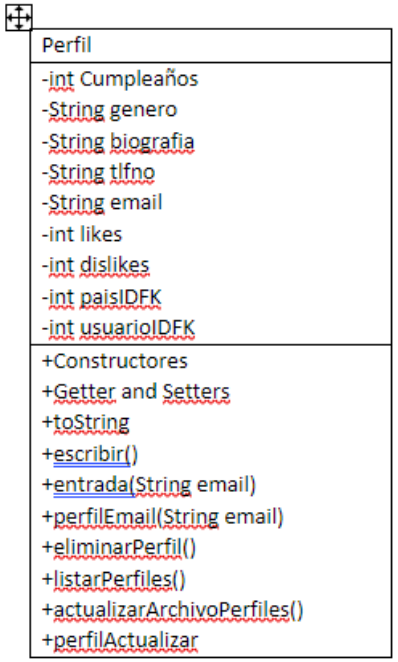
Paquete Dominio.

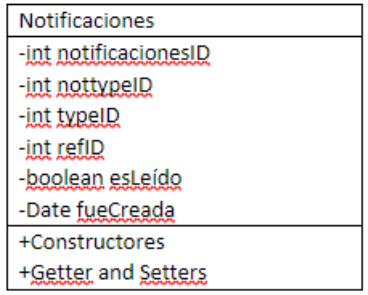
Clases pertenecientes.



Paquete Dominio.

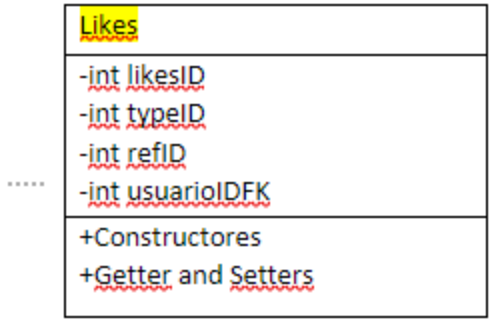
Clases pertenecientes.

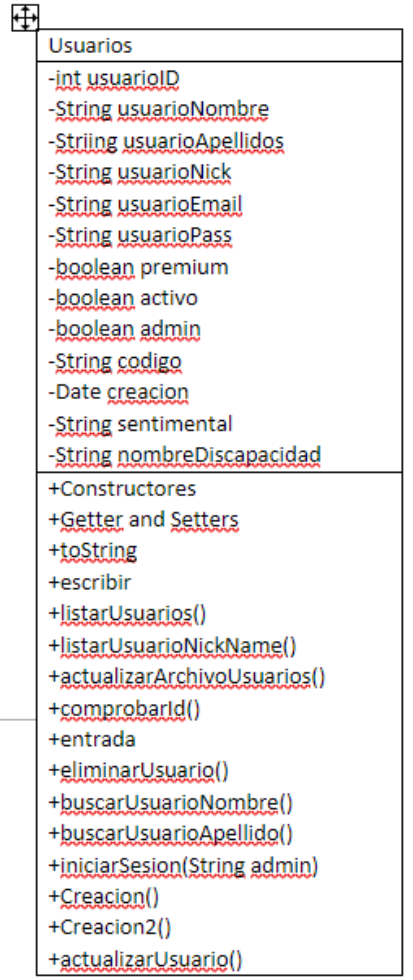




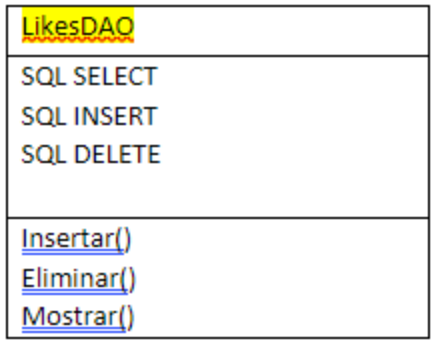
Paquete Dominio.

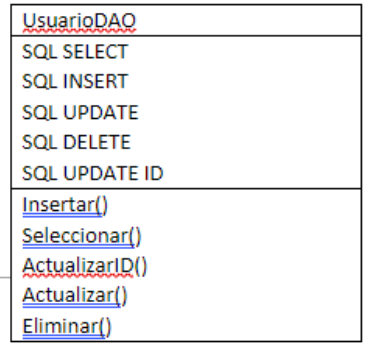
Clases pertenecientes.

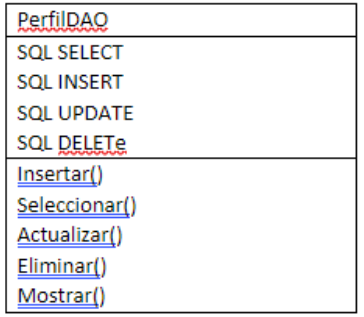


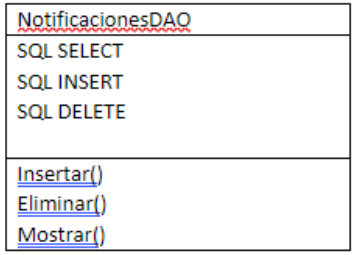


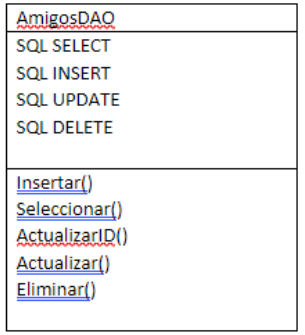
Paquete Datos
Clases pertenecientes

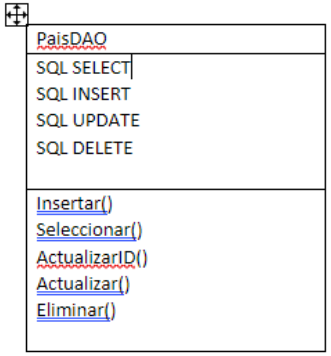


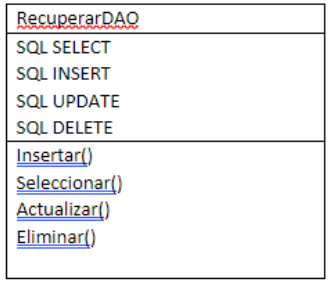




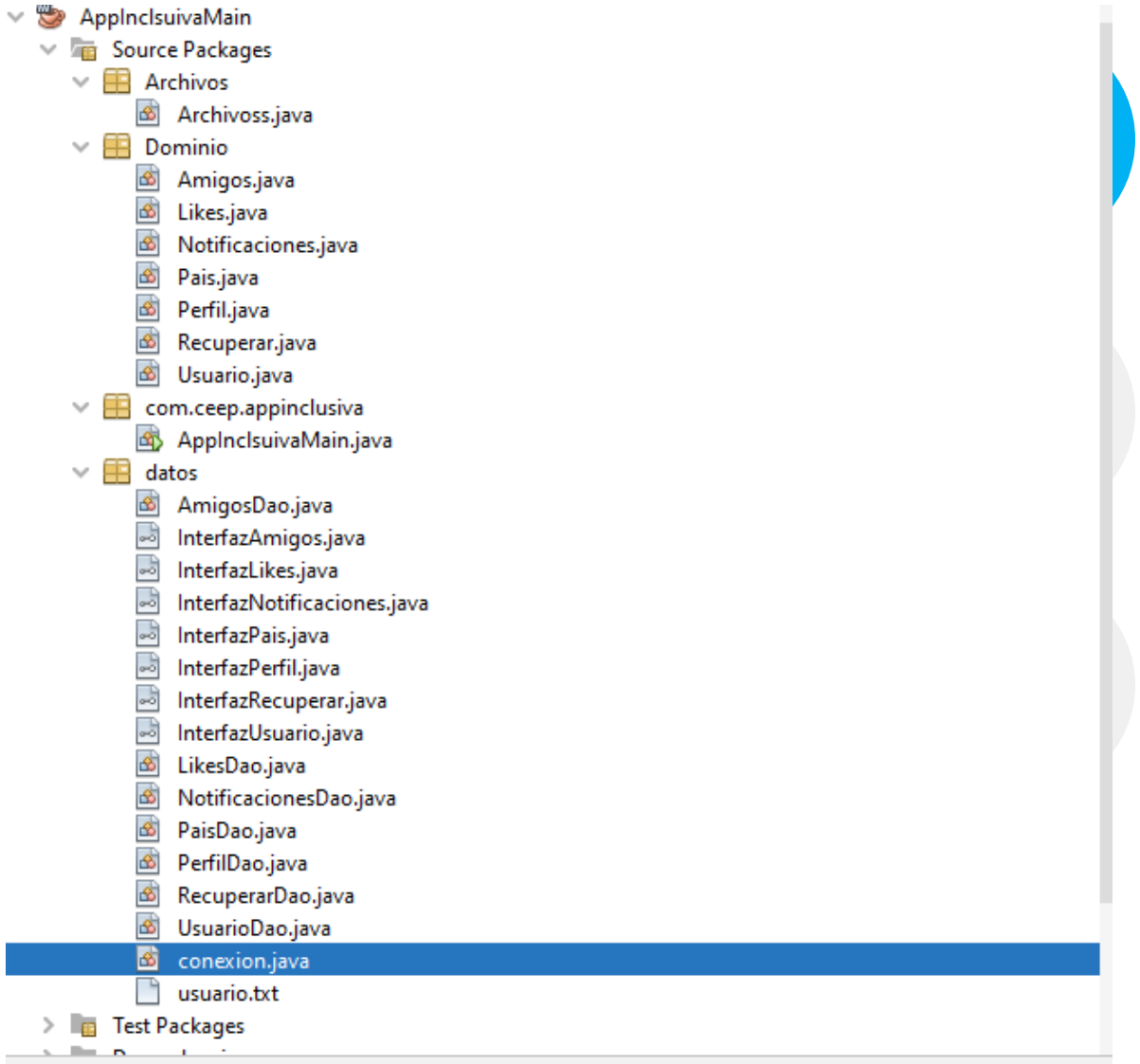








ARQUITECTURA DEL PROYECTO



Arquitectura Conexion

```java
public class conexion {
    private static final String JDBC_URL = "jdbc:mysql://localhost:3306/AppInclsuiva?useSSL=false"
                    + "&useTimezone=true&serverTimezone=UTC"
                    + "&allowPublickeyRetrival=true";
    private static final String JDBC_USER = "root";
    private static final String JDBC_PASSWORD = "1234";

    public static  Connection getConnection() throws SQLException{
        return DriverManager.getConnection( url: JDBC_URL,  user: JDBC_USER, password: JDBC_PASSWORD);
    }

    public static void close (ResultSet rs) throws SQLException{
        rs.close();
    }

    public static void close (Statement stm) throws SQLException{
        stm.close();
    }

    public static void close (PreparedStatement stm) throws SQLException{
        stm.close();
    }

    public static void close (Connection conn) throws SQLException{
        conn.close();
    }
```

Metodo CRUD y arquitectura Datos

UsuarioDao.java | InterfazUsuario.java | TestMysql.java | Perfil.java | Usuario.java

Source | History

```java
import java.util.List;

/**
 *
 * @author Alumno Mañana
 */
public  class UsuarioDao implements InterfazUsuario{
    private static final String SQL_SELECT = "SELECT * FROM Usuario";
    private static final String SQL_INSERT = "INSERT INTO Usuario (usuarioID, "
            + "usuarioNombre,usuarioApellidos,usuarioNick,usuarioEmail,usuarioPass, "
            + "codigo,creacion,sentimental,nomreDiscapacidad)"
            + "VALUE(?,?,?,?,?,?,?,?,?,?";
    private static final String SQL_UPDATE = "UPDATE Usuario SET usuarioID=?, "
            + "usuarioNombre=?,usuarioApellido=?,usuarioNick=?,usuarioEmail=? "
            + "usuarioPass=?,premium=?,activo=?,admin=?,codigo=?,creacion=?, "
            + "sentimental=?,nombreDiscapacidad=?";
    private static final String SQL_DELETE = "DELETE Usuario WHERE usuarioID=?";

     private static final String SQL_UPDATE_ID = "UPDATE usuario SET "
            + "usuarioNick = ? WHERE usuarioNick = ?;";

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

    @Override
       public int insertar(Usuario usuario){...36 lines }
    @Override
     public List<Usuario> seleccionar () throws SQLException {...36 lines }
    @Override
      public int actualizarID (Usuario u, String prevUsuario){...34 lines }
    @Override
    public int actualizar(Usuario usuario){...36 lines }
    @Override
        public int eliminar(Usuario usuario){...25 lines }


}
```

```java
public class Usuario implements Serializable {

    private int usuarioID;
    private String usuarioNombre;
    private String usuarioApellidos;
    private String usuarioNick;
    private String usuarioEmail;
    private String usuarioPass;
    private boolean premium;
    private boolean activo;
    private boolean admin;
    private String codigo;
    private Date creacion;
    private String sentimental;
    private String nombreDiscapacidad;

    public Usuario() {...2 lines }
    public Usuario(String usuarioNick, String usuarioPass) {...5 lines }
    public Usuario(String usuarioNick, String usuarioPass, Date creacion) {...5 lines }
    public Usuario(String usuarioNombre, String usuarioApellidos, String usuarioNick, String usuarioEmail, String usuarioPass, String codigo, String sentimental, String nombreDiscapa
    public Usuario(int usuarioID, String usuarioNombre, String usuarioApellidos, String usuarioNick, String usuarioEmail, String usuarioPass, String codigo, Date creacion, String ser
    public Usuario(int usuarioID, String usuarioNombre, String usuarioApellidos, String usuarioNick, String usuarioEmail, String usuarioPass, String codigo, String sentimental, Strir

    public int getUsuarioID() {...3 lines }
    public void setUsuarioID(int usuarioID) {...3 lines }
    public String getUsuarioNombre() {...3 lines }
    public void setUsuarioNombre(String usuarioNombre) {...3 lines }
    public String getUsuarioApellidos() {...3 lines }
    public void setUsuarioApellidos(String usuarioApellidos) {...3 lines }
    public String getUsuarioNick() {...3 lines }
    public void setUsuarioNick(String usuarioNick) {...3 lines }
    public String getUsuarioEmail() {...3 lines }
    public void setUsuarioEmail(String usuarioEmail) {...3 lines }
    public String getUsuarioPass() {...3 lines }
    public void setUsuarioPass(String usuarioPass) {...3 lines }
    public boolean isPremium() {...3 lines }
    public void setPremium(boolean premium) {...3 lines }
    public boolean isActivo() {...3 lines }
    public void setActivo(boolean activo) {...3 lines }
    public boolean isAdmin() {...3 lines }
    public void setAdmin(boolean admin) {...3 lines }
    public String getCodigo() {...3 lines }
    public void setCodigo(String codigo) {...3 lines }
    public Date getCreacion() {...3 lines }
```

METODOS CRUD 1/2

```java
public int eliminar(Usuario usuario){
    Connection conn = null;
    PreparedStatement stmt = null;
    int registros = 0;

    try {
        conn = getConnection();
        stmt =conn.prepareStatement( sql: SQL_DELETE);

        stmt.setInt( parameterIndex: 1,  x: usuario.getUsuarioID());

        registros = stmt.executeUpdate();

    } catch (SQLException ex) {
        ex.printStackTrace( s: System.out);
    }finally{
        try {
            close( stm: stmt);
            close(conn);
        } catch (SQLException ex) {
            ex.printStackTrace( s: System.out);
        }
    }
    return registros;
}
```

```java
public int insertar(Usuario usuario){
    Connection conn = null;
    PreparedStatement stmt = null;
    int registros = 0;

    try {
        conn = getConnection();
        stmt =conn.prepareStatement( sql: SQL_INSERT);

        stmt.setInt( parameterIndex: 1,  x: usuario.getUsuarioID());
        stmt.setString( parameterIndex: 2,  x: usuario.getUsuarioNombre());
        stmt.setString( parameterIndex: 3,  x: usuario.getUsuarioApellidos());
        stmt.setString( parameterIndex: 4,  x: usuario.getUsuarioNick());
        stmt.setString( parameterIndex: 5,  x: usuario.getUsuarioEmail());
        stmt.setString( parameterIndex: 6,  x: usuario.getUsuarioPass());
        stmt.setString( parameterIndex: 7,  x: usuario.getCodigo());
        stmt.setDate( parameterIndex: 8,  x: usuario.getCreacion());
        stmt.setString( parameterIndex: 9,  x: usuario.getSentimental());
        stmt.setString( parameterIndex: 10,  x: usuario.getNombreDiscapacidad());



        registros = stmt.executeUpdate();

    } catch (SQLException ex) {
        ex.printStackTrace( s: System.out);
    }finally{
        try {
            close( stm: stmt);
            close(conn);
        } catch (SQLException ex) {
            ex.printStackTrace( s: System.out);
        }
    }
    return registros;
}
```

METODOS CRUD 2/2

```java
public List<Usuario> seleccionar () throws SQLException {
    //INICIAR VARIABLES
    Connection conn = null;
    PreparedStatement stmt = null;
    ResultSet rs = null;

    List<Usuario> usuarios = new ArrayList<>();
    conn = getConnection();
    stmt = conn.prepareStatement( sql: SQL_SELECT);
    rs = stmt.executeQuery();


    while(rs.next()){

        String usuarioNombre = rs.getString( columnLabel: "usuarioNombre");
        String usuarioApellidos = rs.getString( columnLabel: "usuarioApellidos");
        String usuarioNick = rs.getString( columnLabel: "usuarioNick");
        String usuarioEmail = rs.getString( columnLabel: "usuarioEmail");
        String usuarioPass = rs.getString( columnLabel: "usuarioPass");
        String codigo = rs.getString( columnLabel: "codigo");
        String sentimental = rs.getString( columnLabel: "sentimental");
        String nombreDiscapacidad = rs.getString( columnLabel: "nombreDiscapacidad");




        //Instancio nuevos objetos
        usuarios.add(new Usuario(usuarioNombre,usuarioApellidos,
        usuarioNick,usuarioEmail,usuarioPass,codigo,
        sentimental,nombreDiscapacidad));
    }
    close(rs);
    close( stm: stmt);
    close(conn);
    return usuarios;
}
```

```java
public int actualizar(Usuario usuario){
        Connection conn = null;
        PreparedStatement stmt = null;
        int registros = 0;

        try {
            conn = getConnection();
            stmt =conn.prepareStatement( sql: SQL_UPDATE);

           stmt.setString( parameterIndex: 1,  x: usuario.getUsuarioNombre());
            stmt.setString( parameterIndex: 2,  x: usuario.getUsuarioApellidos());
            stmt.setString( parameterIndex: 3,  x: usuario.getUsuarioNick());
            stmt.setString( parameterIndex: 4,  x: usuario.getUsuarioEmail());
            stmt.setString( parameterIndex: 5,  x: usuario.getUsuarioPass());
            stmt.setBoolean( parameterIndex: 6,  x: usuario.isPremium());
            stmt.setBoolean( parameterIndex: 7,  x: usuario.isActivo());
            stmt.setBoolean( parameterIndex: 8,  x: usuario.isAdmin());
            stmt.setString( parameterIndex: 9,  x: usuario.getCodigo());
            stmt.setDate( parameterIndex: 10, x: usuario.getCreacion());
            stmt.setString( parameterIndex: 11,  x: usuario.getSentimental());
            stmt.setString( parameterIndex: 12,  x: usuario.getNombreDiscapacidad());

            registros = stmt.executeUpdate();

        } catch (SQLException ex) {
            ex.printStackTrace( s: System.out);
        }finally{
            try {
                close( stm: stmt);
                close(conn);
            } catch (SQLException ex) {
                ex.printStackTrace( s: System.out);
            }
        }
        return registros;
    }
@Override
```