ACCESO A DATOS
UD2: MANEJO DE CONECTORES

EXPLICACION ENUNCIADO 7: Insertar

```
private void btnInsertarActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    //PRIMERA OPCION:Comprobamos que existe antes de insertar y sino existe lo insertamos.
    Departamento d;

    if (txtNumero.getText().isEmpty() || txtNombre.getText().isEmpty() || txtLoc.getText().isEmpty()) {
        JOptionPane.showMessageDialog(null, "Faltan Datos");
        return;
    }
    //Desde este try se abre la conexión, por lo que en cualquier via de salida del programa se debe de cerrar
    //Obligamos a que inserte todos los datos
    try {
        d = GestionDepartamento.BuscaDepartamento(Integer.valueOf(txtNumero.getText().trim()));
        if (d == null) {
            JOptionPane.showMessageDialog(null,
                    GestionDepartamento.insertarDepartamento(Integer.valueOf(txtNumero.getText().trim()),
                            txtNombre.getText().trim(), txtLoc.getText().trim())
                    + " registro insertado");
        } else {
            JOptionPane.showMessageDialog(null, "Ya existe el departamento");
            return;
        }
        Pool.getCurrentConexion().commit();//Ha ido todo bien commit
    } catch (SQLException ex) {
        try {
            Pool.getCurrentConexion().rollback();//Se ha producido un error rollback
            Logger.getLogger(Enunciado7_comprobandopreviamente.class.getName()).log(Level.SEVERE, null, ex);
        } catch (SQLException ex1) {
            Logger.getLogger(Enunciado7_comprobandopreviamente.class.getName()).log(Level.SEVERE, null, ex1);
        }
    } catch (NumberFormatException e) {
        JOptionPane.showMessageDialog(null, "Datos incorrectos");
        ponerblancos();
    } finally {
        Pool.Cerrar();
    }
```

Pasos:

1.- Inicialmente asegurarnos que se introducen todos los datos. (1)

- Faltan datos: 1 y sale. No hay conexión

2.- Si se introducen todos los datos, pero se introducen mal (ejemplo un numero de departamento abc y no numérico) va al punto (7)

- 1-2-7-8 (En el método pool.cerrar solo cerramos la conexión si se ha abierto). No hay conexión. Parece que abre la conexión en el punto 2 y la cierra en el 8 pero realmente no llega a hacerlo porque salta la excepción al punto 7.

3.- Si se introducen todos los datos y bien busca el departamento para ver si existe antes de insertarlo (2). En la primera consulta (abre la conexión).Si devuelve null quiere decir que podemos insertarlo (3) ya que NO EXISTE PREVIAMENTE. Si todo va bien IMPORTANTE COMMIT (5). En el caso de que diera un error la consulta lanzaría la excepción de SQLException y haría un ROLLBACK (6). Tanto haga el commit (todo bien) como el rollback (algo mal) siempre cerramos la conexión (8).

- Proceso correcto.Se introducen datos bien y no existe el departamento: 1-2-3-5-8. Abre la conexión en el punto 2 y lo cierra en el 8. Commit en el 5.

- Proceso incorrecto falla la inserccion.Se introducen datos bien y no existe el departamento: 1-2-3-6-8. Abre la conexión en el punto 2 y lo cierra en el 8. Rollback en el 6.

4.- Si se introducen todos los datos y bien busca el departamento para ver si existe antes de insertarlo (2). En la primera consulta (abre la conexión).Si devuelve el departamento quiere decir que no lo insertamos (4) porque ya EXISTE PREVIAMENTE. Por último cerramos la conexión.

- 1-2-4-8

EXPLICACION ENUNCIADO 7: Modificar

```
private void btnModificarActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:

    /**
     * PRIMERA OPCION:Comprobamos si existe antes de mofificarlo
     */
    if (txtNumero.getText().isEmpty() || txtNombre.getText().isEmpty() || txtLoc.getText().isEmpty()) {
        JOptionPane.showMessageDialog(null, "Faltan Datos");
        return;
    }
    Departamento d;
    try {
        //Desde este try se abre la conexión, por lo que en cualquier via de salida del programa se debe de cerrar
        //Obligamos a que inserte todos los datos
        d = GestionDepartamento.BuscaDepartamento(Integer.valueOf(txtNumero.getText().trim()));
        if (d != null) {
            JOptionPane.showMessageDialog(null, GestionDepartamento.ModificarDepartamento(Integer.valueOf(txtNumero.getText().trim()),
                    txtNombre.getText().trim(), txtLoc.getText().trim())
                    + " registro modificado");
        } else {
            JOptionPane.showMessageDialog(null, "No existe el departamento a modificar");
            return;
        }
        Pool.getCurrentConexion().commit();//Ha ido todo bien commit
    } catch (SQLException ex) {
        try {
            Pool.getCurrentConexion().rollback();//Se ha producido un error rollback
            Logger.getLogger(Enunciado7_comprobandopreviamente.class.getName()).log(Level.SEVERE, null, ex);
        } catch (SQLException ex1) {
            Logger.getLogger(Enunciado7_comprobandopreviamente.class.getName()).log(Level.SEVERE, null, ex1);
        }
    } catch (NumberFormatException e) {
        JOptionPane.showMessageDialog(null, "Datos incorrectos");
        ponerblancos();
    } finally {
        Pool.Cerrar();
    }

}
```

Pasos:

1.- Inicialmente asegurarnos que se introducen todos los datos. (1)

- Faltan datos: 1 y sale. No hay conexión

2.- Si se introducen todos los datos, pero se introducen mal (ejemplo un numero de departamento abc y no numérico) va al punto (7)

- 1-2-7-8 (En el método pool.cerrar solo cerramos la conexión si se ha abierto). No hay conexión. Parece que abre la conexión en el punto 2 y la cierra en el 8 pero realmente no llega a hacerlo porque salta la excepción al punto 7.

3.- Si se introducen todos los datos y bien busca el departamento para ver si existe antes de modificarlo (2). En la primera consulta (abre la conexión).Si devuelve distinto de null quiere decir que podemos modificarlo (3) ya que EXISTE PREVIAMENTE. Si todo va bien IMPORTANTE COMMIT (5). En el caso de que diera un error la consulta lanzaría la excepción de SQLException y haría un ROLLBACK (6). Tanto haga el commit (todo bien) como el rollback (algo mal) siempre cerramos la conexión (8).

- Proceso correcto.Se introducen datos bien y si existe el departamento: 1-2-3-5-8. Abre la conexión en el punto 2 y lo cierra en el 8. Commit en el 5.
- Proceso incorrecto falla la modificación.Se introducen datos bien y existe el departamento: 1-2-3-6-8. Abre la conexión en el punto 2 y lo cierra en el 8. Rollback en el 6.

4.- Si se introducen todos los datos y bien busca el departamento para ver si existe antes de modificarlo (2) pero no existe. En la primera consulta (abre la conexión).Si no existe el departamento no podemos modificarlo (4). Por último cerramos la conexión.

- 1-2-4-8

EXPLICACION ENUNCIADO 7: Borrar restringido. Si tiene empleados el departamento no podemos borrarlo

```
private void btnBorrarActionPerformed(java.awt.event.ActionEvent evt) {

    //En este caso vamos a hacer un borrado restringido. De tal manera que si existen empleados
    //No podemos realizar el borrado.
    //PRIMERA OPCION:Comprobamos que existe antes de borrar. Si no existe no borramos
    //Además comprobamos que no tiene empleados antes de borrarlos
    if (txtNumero.getText().isEmpty() || txtNombre.getText().isEmpty() || txtLoc.getText().isEmpty()) {
        JOptionPane.showMessageDialog(null, "Faltan Datos");
        return;
    }
    Departamento d;
    try {
        //Desde este try se abre la conexión, por lo que en cualquier via de salida del programa se debe de cerrar
        //Obligamos a que inserte todos los datos
        d = GestionDepartamento.BuscaDepartamento(Integer.valueOf(txtNumero.getText().trim()));
        if (d != null) {
            if (GestionDepartamento.cuentaEmpleados(d.getNum()) != 0) {
                JOptionPane.showMessageDialog(null, "Tiene empleados");
                return;
            } else {
                JOptionPane.showMessageDialog(null, GestionDepartamento.borrarDepartamento(Integer.valueOf(txtNumero.getText().trim()))
                        + " registro borrado");
            }
        } else {
            JOptionPane.showMessageDialog(null, "No existe el departamento a borrar");
            return;
        }
        Pool.getCurrentConexion().commit();//Ha ido todo bien commit
    } catch (SQLException ex) {
        try {
            Pool.getCurrentConexion().rollback();//Se ha producido un error rollback
            Logger.getLogger(Enunciado7_comprobandopreviamente.class.getName()).log(Level.SEVERE, null, ex);
        } catch (SQLException ex1) {
            Logger.getLogger(Enunciado7_comprobandopreviamente.class.getName()).log(Level.SEVERE, null, ex1);
        }
    } catch (NumberFormatException e) {
        JOptionPane.showMessageDialog(null, "Datos incorrectos");
        ponerblancos();
    } finally {
        Pool.Cerrar();
    }
}
```

Pasos:

1.- Inicialmente asegurarnos que se introducen todos los datos. (1)

- Faltan datos: 1 y sale. No hay conexión

2.- Si se introducen todos los datos, pero se introducen mal (ejemplo un numero de departamento abc y no numérico) va al punto (7)

- 1-2-8-9 (En el método pool.cerrar solo cerramos la conexión si se ha abierto). No hay conexión. Parece que abre la conexión en el punto 2 y la cierra en el 9 pero realmente no llega a hacerlo porque salta la excepción al punto 8.

3.- Si se introducen todos los datos y bien busca el departamento para ver si existe antes de borrarlo (2). En la primera consulta (abre la conexión).Si devuelve distinto de null quiere decir que podemos borrarlo (4) ya que EXISTE PREVIAMENTE, pero antes de pasar a borrarlo comprobamos si ese departamento tiene empleados (3) o no (4) ya que si tiene empleados no podremos borrarlo. Si todo va bien IMPORTANTE COMMIT (6). En el caso de que diera un error la consulta lanzaría la excepción de SQLException y haría un ROLLBACK (7). Tanto haga el commit (todo bien) como el rollback (algo mal) siempre cerramos la conexión (9).

- Proceso correcto.Se introducen datos bien y si existe el departamento comprobamos que no tiene empleados. Entonces borramos: 1-2-4-6-9. Abre la conexión en el punto 2 y lo cierra en el 9. Commit en el 6.
- No se puede borrar porque tiene empleados.Se introducen datos bien y si existe el departamento comprobamos que tiene empleados y no lo podemos borrar. 1-2-3-9. Abre la conexión en el punto 2 y lo cierra en el 9.
- Proceso incorrecto el borrado.Se introducen datos bien y existe el departamento y no tiene empleados, pero falla el borrado: 1-2-4-7-9. Abre la conexión en el punto 2 y lo cierra en el 8. Rollback en el 6.

4.- Si se introducen todos los datos y bien busca el departamento para ver si existe antes de borrarlo (2) pero no existe. En la primera consulta (abre la conexión).Si no existe el departamento no podemos borrarlo (5). Por último cerramos la conexión.

- 1-2-5-9