# BugTracker - Transacciones

## Manejo de transacciones

Una transacción es un conjunto de operaciones que se debe ejecutar todas juntas o ninguna.

Ejemplo:

1. Crear un bug y su correspondiente registro histórico. Implica insertar un registro en la tabla bugs y un registro en la tabla histobugs. Si alguna de las dos operaciones no se efectúa la base de datos queda inconsistente (no refleja la realidad y/o la implementación esperada).
2. Actualizar un bug y su correspondiente registro histórico. Implica actualizar un registro en la tabla bugs e insertar un registro en la tabla bugshistorico. Si alguna de las dos operaciones no se efectúa la base de datos queda inconsistente (no refleja la realidad y/o la implementación esperada).
3. Un caso típico de transacción es la generación de un pedido, presupuesto o factura.

En estos casos resulta necesario insertar un registro de encabezado, uno o mas registros de detalle y probablemente actualizaciones de otras tablas para reflejar comisiones, stock, etc.

## Datamanager

Agregamos al proyecto bugtrucker una clase nueva “DataManager” que se encarga de manejar las transacciones. Provee los métodos necesarios interactuar con la base de datos:

-Abrir la conexión.

-Cerrar la conexión.

-Iniciar Transacción.

-Ejecutar Sentencias Insert, Delete, Update.

-Ejecutar Consultas.

-Confirmar la transacción (Commit: Hace efectivo el cambio en la base de datos).

-Volver la transacción al punto inicial (Rollback: Vuelve la base de datos al estado anterior al inicio de la transacción).

|  |
| --- |
|  |
| Clase DataManager |

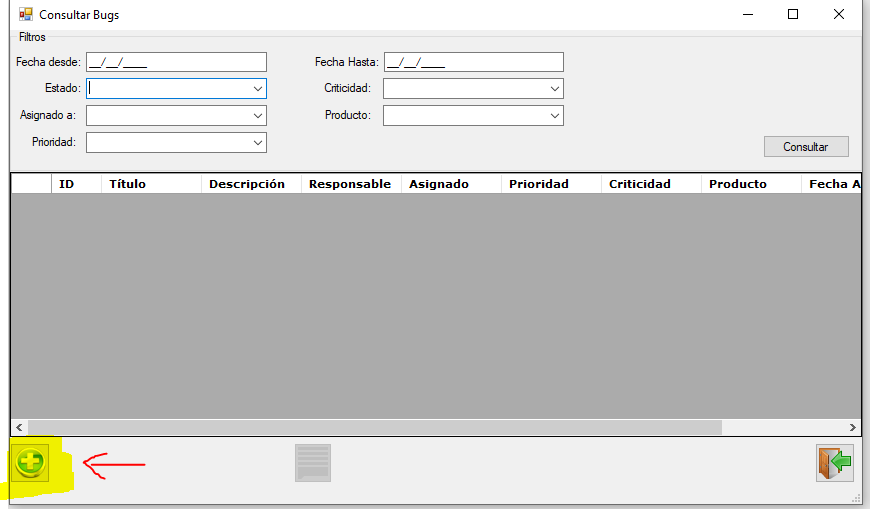
## Ejemplo de transacciones en el proyecto BugTracker

Se crea un bug (Insert en la tabla Bugs), se obtiene el identity insertado y se crea un histórico del bug (Insert en la tabla HistoricoBugs).

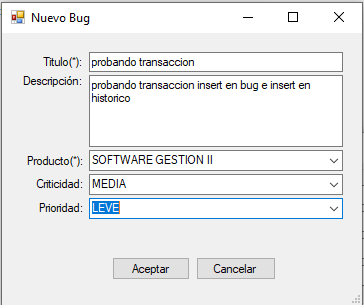
Para correr el ejemplo, actualizar la base de datos con el script de la carpeta “database”.

En la clase dbhelper y en la clase DataManager actualizar el string de conexión.

Ejecutar el proyecto - Seleccionar Consultar Bugs – Presionar +

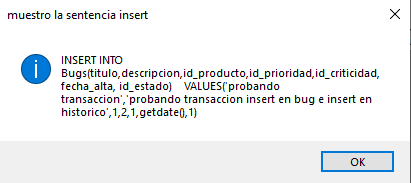


Se levantará el formulario para cargar el bug:

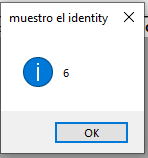


Al presionar Aceptar, se ejecuta la transacción:

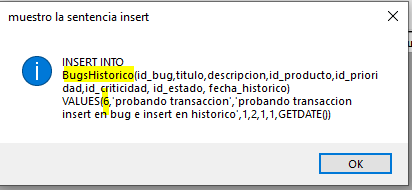
1. Insert en tabla Bugs



1. Obtiene el identity generado en la tabla Bugs (valor del campo id Ultimo insertado).



1. Insert en la tabla HistoricoBugs



Desde el formulario se invoca al servicio y el este ultimo invoca al dao: método createBugConHistorial

public bool createBugConHistorial(Bug oBug)

{

DataManager dm = new DataManager();

try

{

//Select @@identity obtiene el identity insertado

//valor de id estado, id responsable y asignado fijos

string sql = "INSERT INTO Bugs(titulo,descripcion,id\_producto,id\_prioridad,id\_criticidad,fecha\_alta, id\_estado,id\_usuario\_responsable,id\_usuario\_asignado,borrado) " +

" VALUES("+

"'"+oBug.Titulo +"'"+","+

"'"+oBug.Descripcion +"'"+","+

oBug.Producto.IdProducto +","+

oBug.Prioridad.IdPrioridad + "," +

oBug.Criticidad.IdCriticidad + "," +

"getdate(),"+"1,"+"1,"+"1,"+"0"+")";

dm.Open();

dm.BeginTransaction();

//Ejecuto el insert del bug

dm.EjecutarSQL(sql);

MessageBox.Show(sql, "muestro la sentencia insert", MessageBoxButtons.OK, MessageBoxIcon.Information);

var newId = dm.ConsultaSQLScalar(" SELECT @@IDENTITY");

MessageBox.Show(Convert.ToString(newId), "muestro el identity", MessageBoxButtons.OK, MessageBoxIcon.Information);

//Guarda en id\_bug el identity generado

oBug.IdBug = Convert.ToInt32(newId);

string sqlhisto = "INSERT INTO BugsHistorico(id\_bug,titulo,descripcion,id\_producto,id\_prioridad,id\_criticidad, id\_estado, fecha\_historico)" + " VALUES(" + oBug.IdBug + "," + "'" + oBug.Titulo + "'" + "," + "'" + oBug.Descripcion + "'" + "," + oBug.Producto.IdProducto + "," + oBug.Prioridad.IdPrioridad + "," + oBug.Criticidad.IdCriticidad + "," + 1 + "," + "GETDATE()" + ")";

MessageBox.Show(sqlhisto, "muestro la sentencia insert", MessageBoxButtons.OK, MessageBoxIcon.Information);

//Guarda en id\_bug el identity generado

dm.EjecutarSQL(sqlhisto);

dm.Commit();

return true;

}

catch (Exception ex)

{

dm.Rollback();

return false;

}

finally

{

// Cierra la conexión

dm.Close();

}

}

}

Se encierra en un bloque try-catch la transacción.

-Si se ejecuta la línea anterior al catch significa que todas las sentencias se ejecutaron con éxito, por lo tanto se puede hacer commit.

-Si se ejecuta el catch ocurrió algún error.

-Finalmente se ejecuta el bloque finally que cierra y dispone la conexión.

|  |
| --- |
|  |
| Tabla Bugs |
|  |
| Tabla HistoricoBugs |