# BugTracker - Form Consulta Bugs

## 1. Ejecutar Script Base de datos

**1.1. Iniciar la aplicación Sql Server Management Studio**

Solicitará ingresar los datos de la base de datos para generar una conexión, completar los datos y hacer click en **Connect**. Los datos del servidor del labsis son:

* **Tipo Servidor:** Database Engine
* **Nombre Servidor:** maquis
* **Autenticación:** SQL Server Authentication.
* **Nombre de Usuario:** avisuales1
* **Contraseña:** \*\*\*\*\*\*

**1.2. Abrir archivo BugTracker\_Crear\_BaseDatos.sql** Ir a la opción Archivo -> Abrir -> Archivo (o combinación de teclas Ctrl + O) y buscar el archivo BugTracker\_Crear\_BaseDatos.sql en el disco local.

**1.3. Editar Script** > Antes de ejecutar editar el archivo BugTracker\_Crear\_BaseDatos.sql y reemplazar el nombre de la base de datos, agregando el número de legajo de la siguiente forma BugTracker\_12345, reemplazando 12345 por el número de legajo:

USE [master]  
  
-------------------------------------------------------------------------------------------  
/\*-------------------------------------------------------------------------------------------  
IMPORTANTE!!! REEMPLAZAR nombre de base de datos usando el legajo de cada uno: Ej. BugTracker12345 \*/  
   
CREATE DATABASE [BugTracker12345]  
GO  
  
USE [BugTracker12345]  
GO  
-------------------------------------------------------------------------------------------  
-------------------------------------------------------------------------------------------  
  
SET ANSI\_NULLS ON  
GO  
SET QUOTED\_IDENTIFIER ON  
GO

**1.4. Ejecutar Script** Para ejecutar el script hacer click sobre el botón Ejecutar (o usar la tecla F5)

## 2. Temas

IMPORTANTE: CORRER EL SCRIPT PARA ACTUALIZAR LA TABLA HISTORICO DE BUGS

### 2.1. Control Grilla (DataGridView)

El control [DataGridView](https://docs.microsoft.com/es-es/dotnet/api/system.windows.forms.datagridview?view=netframework-4.8) proporciona una tabla personalizable para mostrar los datos. La clase [DataGridView](https://docs.microsoft.com/es-es/dotnet/api/system.windows.forms.datagridview?view=netframework-4.8) permite la personalización de celdas, filas, columnas y bordes mediante el uso de propiedades [DefaultCellStyle](https://docs.microsoft.com/es-es/dotnet/api/system.windows.forms.datagridview.defaultcellstyle?view=netframework-4.8), [ColumnHeadersDefaultCellStyle](https://docs.microsoft.com/es-es/dotnet/api/system.windows.forms.datagridview.columnheadersdefaultcellstyle?view=netframework-4.8), [CellBorderStyle](https://docs.microsoft.com/es-es/dotnet/api/system.windows.forms.datagridview.cellborderstyle?view=netframework-4.8) y [GridColor](https://docs.microsoft.com/es-es/dotnet/api/system.windows.forms.datagridview.gridcolor?view=netframework-4.8). Para obtener más información, lea [formato y estilo básicos en el control DataGridView Windows Forms](https://docs.microsoft.com/es-es/dotnet/framework/winforms/controls/basic-formatting-and-styling-in-the-windows-forms-datagridview-control?view=netframework-4.8).

Puede utilizar un control [DataGridView](https://docs.microsoft.com/es-es/dotnet/api/system.windows.forms.datagridview?view=netframework-4.8) para Mostrar datos con o sin un origen de datos subyacente.

* **Sin especificar un origen de datos**, puede crear columnas y filas que contengan datos y agregarlas directamente a [DataGridView](https://docs.microsoft.com/es-es/dotnet/api/system.windows.forms.datagridview?view=netframework-4.8) mediante las propiedades [Rows](https://docs.microsoft.com/es-es/dotnet/api/system.windows.forms.datagridview.rows?view=netframework-4.8) y [Columns](https://docs.microsoft.com/es-es/dotnet/api/system.windows.forms.datagridview.columns?view=netframework-4.8) . También puede utilizar la colección [Rows](https://docs.microsoft.com/es-es/dotnet/api/system.windows.forms.datagridview.rows?view=netframework-4.8) para tener acceso a los objetos [DataGridViewRow](https://docs.microsoft.com/es-es/dotnet/api/system.windows.forms.datagridviewrow?view=netframework-4.8) y la propiedad [DataGridViewRow.Cells](https://docs.microsoft.com/es-es/dotnet/api/system.windows.forms.datagridviewrow.cells?view=netframework-4.8) para leer o escribir los valores de las celdas directamente. El indexador [Item[String, Int32]](https://docs.microsoft.com/es-es/dotnet/api/system.windows.forms.datagridview.item?view=netframework-4.8) también proporciona acceso directo a las celdas.
* **Con un origen de datos**, puede establecer las propiedades [DataSource](https://docs.microsoft.com/es-es/dotnet/api/system.windows.forms.datagridview.datasource?view=netframework-4.8), [DataMember](https://docs.microsoft.com/es-es/dotnet/api/system.windows.forms.datagridview.datamember?view=netframework-4.8) y para enlazar el [DataGridView](https://docs.microsoft.com/es-es/dotnet/api/system.windows.forms.datagridview?view=netframework-4.8) a un origen de datos y rellenarlo automáticamente con datos. Para obtener más información, lea [Mostrar datos en el control DataGridView de Windows Forms](https://docs.microsoft.com/es-es/dotnet/framework/winforms/controls/displaying-data-in-the-windows-forms-datagridview-control?view=netframework-4.8).

Al trabajar con grandes cantidades de datos, puede establecer la propiedad [VirtualMode](https://docs.microsoft.com/es-es/dotnet/api/system.windows.forms.datagridview.virtualmode?view=netframework-4.8) en true para mostrar un subconjunto de los datos disponibles. El modo virtual requiere la implementación de una memoria caché de datos [DataGridView](https://docs.microsoft.com/es-es/dotnet/api/system.windows.forms.datagridview?view=netframework-4.8) desde la que se rellena el control. Para obtener más información, lea [modos de presentación de datos en el control DataGridView Windows Forms](https://docs.microsoft.com/es-es/dotnet/framework/winforms/controls/data-display-modes-in-the-windows-forms-datagridview-control?view=netframework-4.8).

### 2.2. Consulta Multitabla

#### Composición Interna "Producto cartesiano"

El producto cartesiano es un tipo de composición de tablas, aplicando el producto cartesiano a dos tablas se obtiene una tabla con las **columnas** de la **primera tabla** **unidas** a las **columnas** de la **segunda tabla**, y las filas de la tabla resultante son **todas las posibles concatenaciones** de **filas** de la **primera tabla** con filas de la **segunda tabla**.

Se indica **poniendo** en la **FROM** las **tablas** que queremos componer **separadas por comas**, podemos obtener así el producto cartesiano de dos, tres, o más tablas.

#### Composición Interna "InnerJoin".

El **INNER JOIN** es otro tipo de composición de tablas, permite **emparejar filas** de distintas tablas de forma **más eficiente** que con el producto cartesiano **cuando** una de las **columnas de emparejamiento** está **indexada**. Ya que en vez de hacer el producto cartesiano completo y luego seleccionar la filas que cumplen la condición de emparejamiento, para cada fila de una de las tablas **busca directamente** en la otra tabla **las filas que** cumplen la condición, con lo cual **se emparejan** sólo las filas que luego aparecen en el resultado. Ejemplo:

SELECT \*   
 FROM pedidos   
 INNER JOIN clientes ON pedidos.clie = clientes.numclie

Con alias

SELECT \*   
 FROM pedidos P  
 INNER JOIN clientes C ON P.clie = C.numclie

También puede escribirse como

SELECT \*   
 FROM pedidos, clientes

WHERE pedidos.clie = clientes.numclie

SELECT \*   
 FROM pedidos P, clientes C

WHERE P.clie = C.numclie

#### Composición externa "Left"

El **LEFT JOIN** y **RIGHT JOIN** son otro tipo de composición de tablas, también denominada **composición externa**. Son una extensión del **INNER JOIN**.

La descripción de la sintaxis es la **misma** que la del **INNER JOIN** , lo único que cambia es la palabra **INNER** por **LEFT** (**izquierda** en inglés) .

Esta operación consiste en **añadir al resultado** del **INNER JOIN** las **filas** de la **tabla** de la **izquierda** que **no tienen correspondencia** en la otra tabla, y **rellenar** en esas filas los **campos** de la **tabla** de la **derecha** con **valores nulos**.

#### Composición externa "Right"

La sintaxis es la misma que la del **INNER JOIN**, lo único que cambia es la palabra **INNER** por **RIGHT** (**derecha** en inglés).

Esta operación consiste en **añadir al resultado** del **INNER JOIN** las **filas** de la **tabla** de la **derecha** que **no tienen correspondencia** en la otra tabla, y **rellenar** en esas filas los **campos** de la **tabla** de la **izquierda** con **valores nulos**.

### 2.2. Un ejemplo de consulta Multitabla

Cuando debemos listar los **Bugs** en una grilla con solo una consulta simple (con lo veníamos haciendo) nos encontramos en la respuesta de la consulta con columnas que tienen un valor no representa nada más que una FK (clave foranea) y que no es de interés para el usuario, este es el caso de la columna **id\_producto** (FK a **Productos**). Por lo tanto, para poder obtener el **nombre** de un producto asociado a cada **Bug** es necesario hacer una consulta multitabla como se ve a continuación, sumando a la consulta la tabla **Productos** (situación que se repite para las tablas Prioridades, Criticidades y Estados):

SELECT B.id\_bug,

B.titulo,

B.descripcion,

B.fecha\_alta,

U.usuario as responsable,

U.usuario as asignado,

Pro.nombre as producto,

Pri.nombre as prioridad

From Bugs B

LEFT JOIN Usuarios U ON U.id\_usuario = B.id\_usuario\_responsable

LEFT JOIN Usuarios ON U.id\_usuario = B.id\_usuario\_asignado

INNER JOIN Productos Pro ON Pro.id\_producto = B.id\_producto

INNER JOIN Prioridades Pri ON Pri.id\_prioridad = B.id\_prioridad

INNER JOIN Criticidades Cri ON Cri.id\_criticidad = B.id\_criticidad

INNER JOIN Estados E ON E.id\_estado = B.id\_estadoWHERE 1=1

## 3. Actividades

Como primer paso abrir la solución que se encuentra en la carpeta src/BugTracker.sln.

3.1. Inicializar Grilla ConsultaBugs.

**a. En el frmConsultaBugs agregar el método InitializeDataGridView():** Abrir el formulario frmConsultaBugs y agregar el siguiente método para inicializar la grilla de Bugs (dgvBugs):

private void InitializeDataGridView()  
 {  
 // Cree un DataGridView no vinculado declarando un recuento de columnas.  
 dgvBugs.ColumnCount = 4;  
 dgvBugs.ColumnHeadersVisible = true;  
  
 // Configuramos la AutoGenerateColumns en false para que no se autogeneren las columnas  
 dgvBugs.AutoGenerateColumns = false;  
  
 // Cambia el estilo de la cabecera de la grilla.  
 DataGridViewCellStyle columnHeaderStyle = new DataGridViewCellStyle();  
  
 columnHeaderStyle.BackColor = Color.Beige;  
 columnHeaderStyle.Font = new Font("Verdana", 8, FontStyle.Bold);  
 dgvBugs.ColumnHeadersDefaultCellStyle = columnHeaderStyle;  
  
 // Definimos el nombre de la columnas y el DataPropertyName que se asocia a DataSource  
 dgvBugs.Columns[0].Name = "ID";  
 dgvBugs.Columns[0].DataPropertyName = "id\_bug";  
 dgvBugs.Columns[0].Width = 50;  
  
 dgvBugs.Columns[1].Name = "Título";  
 dgvBugs.Columns[1].DataPropertyName = "titulo";  
   
 dgvBugs.Columns[2].Name = "Descripción";  
 dgvBugs.Columns[2].DataPropertyName = "descripcion";  
   
 dgvBugs.Columns[3].Name = "Fecha Alta";  
 dgvBugs.Columns[3].DataPropertyName = "fecha\_alta";  
  
 // Cambia el tamaño de la altura de los encabezados de columna.  
 dgvBugs.AutoResizeColumnHeadersHeight();  
  
 // Cambia el tamaño de todas las alturas de fila para ajustar el contenido de todas las celdas que no sean de encabezado.  
 dgvBugs.AutoResizeRows(  
 DataGridViewAutoSizeRowsMode.AllCellsExceptHeaders);  
 }

**b. Invocar a InitializeDataGridView() desde constructor del formulario:** En el constructor del formulario frmConsultaBugs vamos a invocar al método InitializeDataGridView() de la siguiente forma:

public frmConsultaBugs()  
 {  
 InitializeComponent();  
 // Inicializamos la grilla de bugs   
 InitializeDataGridView();  
 }

**c. Agregar columnas a la grilla de bugs:** En el método InitializeDataGridView() agregar las siguientes columnas y completar con información del Bug:

* Responsable
* Asignado
* Producto
* Prioridad
* Criticidad
* Estado

**d. Habilitar el Button btnDetalleBug cuando se selecciona un Bug en la grilla dgvBugs:** Para ello vamos agregar el evento CellClick() a la grilla dgvBugs que habilite el Button btnDetalleBug de la siguiente forma:

private void dgvBugs\_CellClick(object sender, DataGridViewCellEventArgs e)  
 {  
 // Cuando seleccionamos una fila de la grilla habilitamos el boton btnDetalleBug.  
 btnDetalleBug.Enabled = true;  
 }

### 3.2. Detalle de Bug

**a. En frmDetalleBug agregar el método InicializarDetalleBug()** En base a un idBug vamos a consultar los datos a la base de datos y cargar los controles TextBox con el valor de cada columna:

internal void InicializarDetalleBug(string idBug)  
 {  
internal void InicializarDetalleBug(string idBug)

{

var strSql = String.Concat("SELECT B.id\_bug, ",

" B.titulo,",

" B.descripcion,",

" B.fecha\_alta,",

" U.usuario as responsable, ",

" U.usuario as asignado, ",

" Pro.nombre as producto, ",

" Pri.nombre as prioridad, ",

" Cri.nombre as criticidad, ",

" E.nombre as estado",

" FROM Bugs B",

" LEFT JOIN Usuarios U ON U.id\_usuario = B.id\_usuario\_responsable",

" LEFT JOIN Usuarios ON U.id\_usuario = B.id\_usuario\_asignado",

" INNER JOIN Productos Pro ON Pro.id\_producto = B.id\_producto",

" INNER JOIN Prioridades Pri ON Pri.id\_prioridad = B.id\_prioridad",

" INNER JOIN Criticidades Cri ON Cri.id\_criticidad = B.id\_criticidad",

" INNER JOIN Estados E ON E.id\_estado = B.id\_estado",

" WHERE id\_bug = " + idBug.ToString());

var resultado = DBHelper.GetDBHelper().ConsultaSQL(strSql);

if (resultado.Rows.Count > 0)

{

var oBugSelected = resultado.Rows[0];

txtNroBug.Text = oBugSelected["id\_bug"].ToString();

txtTitulo.Text = oBugSelected["titulo"].ToString();

txtDescripcion.Text = oBugSelected["descripcion"].ToString();

txtFechaAlta.Text = oBugSelected["fecha\_alta"].ToString();

txtProducto.Text = oBugSelected["producto"].ToString();

txtPrioridad.Text = oBugSelected["prioridad"].ToString();

txtCriticidad.Text = oBugSelected["criticidad"].ToString();

txtEstado.Text = oBugSelected["estado"].ToString();

}

InicializarHistoricoBug(idBug);

}

**b. Volvemos alfrmConsultaBugs y le agregamos al Button btnDetalleBug el evento Click():** En la definición del evento Click() se debe generar una instancia del formulario frmDetalleBug, invocar método InicializarDetalleBug con el id\_bug seleccionado y finalmente mostrar en pantalla el formulario:

private void btnDetalleBug\_Click(object sender, EventArgs e)  
 {  
 if (dgvBugs.CurrentRow != null)  
 {  
 frmDetalleBug frmDetalle = new frmDetalleBug();  
 var selectedItem = (DataRowView)dgvBugs.CurrentRow.DataBoundItem;  
 frmDetalle.InicializarDetalleBug(selectedItem["id\_bug"].ToString());  
 frmDetalle.ShowDialog();  
 }  
 }

**c. En frmDetalleBug agregar al Button Aceptar el evento Click() con lógica para cerrar el formulario.**

### 3.3. Listado Histórico Bugs

**a. En frmDetalleBug cargar en la grilla dgvHistoricoBug los datos históricos del bug seleccionado.** Esta actividad es similar a la 3.1, solo que en este caso la grilla debe contener datos de la tabla BugsHistorico:

Tabla **BugsHistorico**:

* [PK] id\_bug\_historico (int)
* fecha\_hitorico (date)
* titulo (varchar(100))
* descripcion (varchar(1000))
* [FK] id\_bug (int)
* [FK] id\_usuario\_responsable (int)
* [FK] id\_usuario\_asignado (int)
* [FK] id\_producto (int)
* [FK] id\_prioridad (int)
* [FK] id\_criticidad (int)
* [FK] id\_estado (int)