# BugTracker - Form Consulta Bugs

## 1. Clonar Repositorio (Clone/Checkout)

**1.1. Ejecutar comando clone para descargar repositorio:**

$ git clone https://github.com/utn-frc-pav1-3k5-2019/semana\_4\_bugtracker\_consulta\_bugs.git

\*\*1.2. Ubicarse en la carpeta generada con el nombre del repositorio:

$ cd semana\_4\_bugtracker\_consulta\_bugs

**1.3. Crear un nuevo branch (rama)**

Para crear una nueva rama (branch) y saltar a ella, en un solo paso, puedes utilizar el comando git checkout con la opción -b, indicando el nombre del nuevo branch (reemplazando el nro de legajo) de la siguiente forma branch\_{legajo}, para el legajo 12345:

$ git checkout -b branch\_12345   
Switched to a new branch "12345"

Y para que se vea reflejada en GitHub:

$ git push --set-upstream origin branch\_12345

## 2. Ejecutar Script Base de datos

**2.1. 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:** \*\*\*\*\*\*

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

**2.3. Editar Script** > Antes de ejecutar editar el archivo BugTracker\_DB.sql y reemplazar el nombre de la base de datos BugTracker, 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. BugTracker\_12345   
CREATE DATABASE [BugTracker\_12345]  
GO  
-- IMPORTANTE!!! REEMPLAZAR nombre de base de datos usando el legajo de cada uno: Ej. BugTracker\_12345  
USE [BugTracker\_12345]  
GO  
/\*\*\*\*\*\* Object: Table [dbo].[Bugs] Script Date: 20/8/2019 22:43:37 \*\*\*\*\*\*/  
SET ANSI\_NULLS ON  
GO  
SET QUOTED\_IDENTIFIER ON  
GO

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

## 3. Menú

Para continuar Abrir la solución que se encuentra en la dirección src/BugTracker.sln.

La MenuStrip clase proporciona un sistema de menús para formularios.

La ToolStripMenuItem clase proporciona propiedades que le permiten configurar la apariencia y la funcionalidad de un elemento de menú. Para que se muestre ToolStripMenuItem un objeto, debe agregarlo a un MenuStripobjeto

### 3.1. Como crear un menú estándar

El Diseñador de Windows Forms puede rellenar automáticamente un [MenuStrip](https://docs.microsoft.com/es-es/dotnet/api/system.windows.forms.menustrip) control con elementos de menú estándar.

1. Desde el **cuadro de herramientas**, arrastre un [MenuStrip](https://docs.microsoft.com/es-es/dotnet/api/system.windows.forms.menustrip) control al formulario.
2. Haga clic en el [MenuStrip](https://docs.microsoft.com/es-es/dotnet/api/system.windows.forms.menustrip) glifo de etiqueta inteligente del control (VS_WinFormSmtTagGlyph) y seleccione **insertar elementos estándar**.

* El [MenuStrip](https://docs.microsoft.com/es-es/dotnet/api/system.windows.forms.menustrip) control se rellena con los elementos de menú estándar.

1. Haga clic en el **Archivo** elemento de menú para ver sus elementos de menú predeterminados y los iconos correspondientes. ### 3.3. Elementos de un MenuStrip

* MenuItem ([ToolStripMenuItem](https://docs.microsoft.com/es-es/dotnet/api/system.windows.forms.toolstripmenuitem?view=netframework-4.8))
* ComboBox ([ToolStripComboBox](https://docs.microsoft.com/es-es/dotnet/api/system.windows.forms.toolstripcombobox?view=netframework-4.8))
* TextBox ([ToolStripTextBox](https://docs.microsoft.com/es-es/dotnet/api/system.windows.forms.toolstriptextbox?view=netframework-4.8))

### 3.3. Como crear un control StatusStrip

Use el [StatusStrip](https://docs.microsoft.com/es-es/dotnet/api/system.windows.forms.statusstrip) control para mostrar el estado de las aplicaciones de Windows Forms. En el ejemplo actual, se muestran los elementos de menú seleccionados por el usuario en un [StatusStrip](https://docs.microsoft.com/es-es/dotnet/api/system.windows.forms.statusstrip)control.

1. Desde el **cuadro de herramientas**, arrastre un [StatusStrip](https://docs.microsoft.com/es-es/dotnet/api/system.windows.forms.statusstrip) control al formulario.

* El [StatusStrip](https://docs.microsoft.com/es-es/dotnet/api/system.windows.forms.statusstrip) control se acopla automáticamente a la parte inferior del formulario.

1. Haga clic en el [StatusStrip](https://docs.microsoft.com/es-es/dotnet/api/system.windows.forms.statusstrip) del control de botón de lista desplegable y seleccione **StatusLabel como** para agregar un [ToolStripStatusLabel](https://docs.microsoft.com/es-es/dotnet/api/system.windows.forms.toolstripstatuslabel) el control a la [StatusStrip](https://docs.microsoft.com/es-es/dotnet/api/system.windows.forms.statusstrip) control.

#

## 4. Actividad sobre el Menú

En la clase DBHelper.cs cambiar la cadena de conexión según corresponda: - Conexion con usuario/password: - Data Source=maquis;Initial Catalog=BugTracker;User ID=avisuales1;Password=\*\*\*\*\*\*\* - Conexión a través de windows: - Data Source=maquis;Initial Catalog=BugTracker;Integrated Security=true;

4.1. Agregar al menú **Consultar Bugs** de **frmPrincipal**, el evento **Click()** para abrir el formulario **frmConsultaBugs**

private void consultarBugsToolStripMenuItem\_Click(object sender, EventArgs e)  
 {  
 frmConsultaBugs frmDetalle = new frmConsultaBugs();  
 frmDetalle.ShowDialog();  
 }

4.2. Agregar al menú **Salir** de **frmPrincipal**, el evento **Click()** que ejecute el cierre de la aplicación.

private void salirToolStripMenuItem\_Click(object sender, EventArgs e)  
 {  
 this.Close();  
 }

## 5. Completar ComboBox (Filtros Busqueda)

En esta actividad vamos a trabajar con el formulario **frmConsultaBugs**, vamos a cargar los diferentes combos (que van a ser el filtro de búsqueda de bugs) que hay en el formulario con datos de la base de datos:

### Propiedades ComboBox

* **DataSource (Object):** Obtiene o establece el origen de datos de este objeto. Entre los orígenes de datos [BindingSource](https://docs.microsoft.com/es-es/dotnet/api/system.windows.forms.bindingsource) posibles se incluyen un enlazado a datos, una tabla de datos, una vista de datos, un conjunto de datos, un administrador de vistas de datos [IList](https://docs.microsoft.com/es-es/dotnet/api/system.collections.ilist) , una matriz o cualquier clase que implemente la interfaz. Para obtener más información, vea [orígenes de datos compatibles con Windows Forms](https://docs.microsoft.com/es-es/dotnet/framework/winforms/data-sources-supported-by-windows-forms).
  + Si va a enlazar a una tabla (DataTable), establezca DisplayMember la propiedad en el nombre de una columna del origen de datos.
  + - o -
  + Si va a enlazar a [IList](https://docs.microsoft.com/es-es/dotnet/api/system.collections.ilist), establezca el miembro de presentación en una propiedad pública del tipo en la lista.
* **DisplayMember (String):** String que especifica el nombre de una propiedad de objeto que se incluye en la colección especificada por la propiedad DataSource. El valor predeterminado es una cadena vacía ("").
* **ValueMember (String):** String que representa un único nombre de la propiedad del valor de propiedad DataSource, o una jerarquía de nombres de propiedad delimitados por puntos que resuelve el nombre de una propiedad del objeto final enlazado a datos. El valor predeterminado es una cadena vacía ("").
* **SelectedIndex (Int32):** Obtiene o establece el índice que especifica el elemento seleccionado actualmente. Índice de base cero del elemento actualmente seleccionado. Si no hay ningún elemento seleccionado, se devuelve el valor uno negativo (-1).
* **SelectedItem (Object):** Obtiene o establece el elemento seleccionado actualmente en el elemento ComboBox.
* **SelectedText (String):** Obtiene o establece el texto que se selecciona en la parte de un ComboBox que se puede editar.
* **SelectedValue (Object):** Obtiene o establece el valor de la propiedad miembro especificada por la propiedad ValueMember.

En el siguiente link podemos encontrar la definición de este control: <https://docs.microsoft.com/es-es/dotnet/api/system.windows.forms.combobox?view=netframework-4.8#propiedades>

5.1. En la clase **frmConsultaBugs** agregar el método LlenarCombo, para generalizar lógica que completa los ComboBox con datos para que luego pueden ser seleccionados por el usuario:

private void LlenarCombo(ComboBox cbo, Object source, string display, String value)  
 {  
 // Datasource: establece el origen de datos de este objeto.  
 cbo.DataSource = source;  
 // DisplayMember: establece la propiedad que se va a mostrar para este ListControl.  
 cbo.DisplayMember = display;  
 // ValueMember: establece la ruta de acceso de la propiedad que se utilizará como valor real para los elementos de ListControl.  
 cbo.ValueMember = value;  
 //SelectedIndex: establece el índice que especifica el elemento seleccionado actualmente.  
 cbo.SelectedIndex = -1;  
 }

5.2. Agregar al formulario el evento **Load()** (que se dispara al abrir el formulario), y en dicho evento se procederá a cargar los combos de la siguiente forma:

private void frmBugs\_Load(object sender, EventArgs e)  
 {  
  
 //LLenar combos  
 LlenarCombo(cboEstados, DBHelper.GetDBHelper().ConsultaSQL("Select \* from Estados"), "nombre", "id\_estado");  
  
 LlenarCombo(cboPrioridades, DBHelper.GetDBHelper().ConsultaSQL("Select \* from Prioridades"), "nombre", "id\_prioridad");  
  
 LlenarCombo(cboCriticidades, DBHelper.GetDBHelper().ConsultaSQL("Select \* from Criticidades"), "nombre", "id\_criticidad");  
  
 LlenarCombo(cboAsignadoA, DBHelper.GetDBHelper().ConsultaSQL("Select \* from Usuarios"), "usuario", "id\_usuario");  
  
 LlenarCombo(cboProductos, DBHelper.GetDBHelper().ConsultaSQL("Select \* from Productos"), "nombre", "id\_producto");  
  
 }

## 6. Consultar Bugs

6.1. Agregar al botón **Consultar** el evento **Click()**, para que construya la consulta sql que busque los bugs registrados en la base de datos, con los filtros cargados en el formulario:

Clase **Dictionary**: Representa una colección de claves (TKey) y valores (TValue).

private void btnConsultar\_Click(object sender, EventArgs e)  
 {  
 string strSql = "SELECT TOP 20 \* FROM bugs WHERE 1=1 ";  
 Dictionary<string, object> parametros = new Dictionary<string, object>();  
  
 DateTime fechaDesde;  
 DateTime fechaHasta;  
 if (DateTime.TryParse(txtFechaDesde.Text, out fechaDesde) &&  
 DateTime.TryParse(txtFechaHasta.Text, out fechaHasta))  
 {  
 strSql += " AND (fecha\_alta>=@fechaDesde AND fecha\_alta<=@fechaHasta) ";  
 parametros.Add("fechaDesde", txtFechaDesde.Text);  
 parametros.Add("fechaHasta", txtFechaHasta.Text);  
 }  
  
  
 if (!string.IsNullOrEmpty(cboEstados.Text))  
 {  
 var idEstado = cboEstados.SelectedValue.ToString();  
 strSql += "AND (id\_estado=@idEstado) ";  
 parametros.Add("idEstado", idEstado);  
 }  
  
 if (!string.IsNullOrEmpty(cboAsignadoA.Text))  
 {  
 var asignadoA = cboAsignadoA.SelectedValue.ToString();  
 strSql += "AND (id\_usuario\_asignado=@idUsuarioAsignado) ";  
 parametros.Add("idUsuarioAsignado", asignadoA);  
 }  
  
 if (!string.IsNullOrEmpty(cboPrioridades.Text))  
 {  
 var prioridad = cboPrioridades.SelectedValue.ToString();  
 strSql += "AND (id\_prioridad=@idPrioridad) ";  
 parametros.Add("idPrioridad", prioridad);  
 }  
  
 if (!string.IsNullOrEmpty(cboCriticidades.Text))  
 {  
 var criticidad = cboCriticidades.SelectedValue.ToString();  
 strSql += "AND (id\_criticidad=@idCriticidad) ";  
 parametros.Add("idCriticidad", criticidad);  
 }  
  
 if (!string.IsNullOrEmpty(cboProductos.Text))  
 {  
 var producto = cboProductos.SelectedValue.ToString();  
 strSql += "AND (id\_producto=@idProducto) ";  
 parametros.Add("idProducto", producto);  
 }  
  
 strSql += " ORDER BY fecha\_alta DESC";  
 dgvBugs.DataSource = DBHelper.GetDBHelper().ConsultaSQLConParametros(strSql, parametros);  
 if (dgvBugs.Rows.Count == 0)  
 {  
 MessageBox.Show("No se encontraron coincidencias para el/los filtros ingresados", "Aviso", MessageBoxButtons.OK, MessageBoxIcon.Information);  
 }  
  
 }

## 7. Versionar los cambios locales (add / commit / push)

A continuación, vamos a crear el **commit** y subir los cambios al servidor GitHub.

7.1. **Status**. Verificamos los cambios pendientes de versionar.

$ git status

7.2. **Add** Agregamos todos los archivos nuevos no versionados.

$ git add -A

7.3. **Commit** Generamos un commit con todos los cambios y agregamos un comentario.

$ git commit -a -m "Comentario"

7.4. **Push** Enviamos todos los commits locales a GitHub

$ git push

7.5. **Status** Verificar que no quedaron cambios pendientes de versionar

$ git status