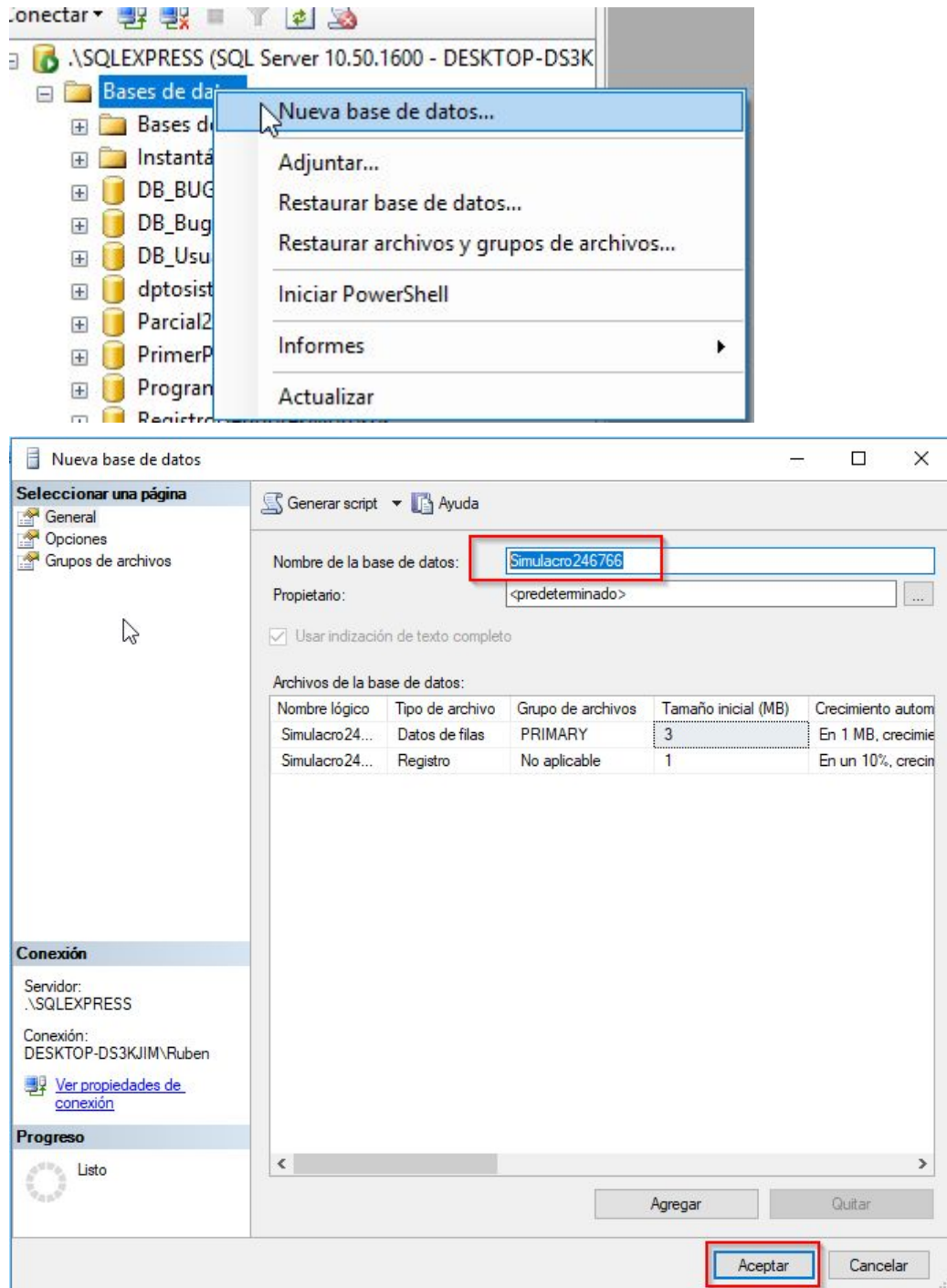
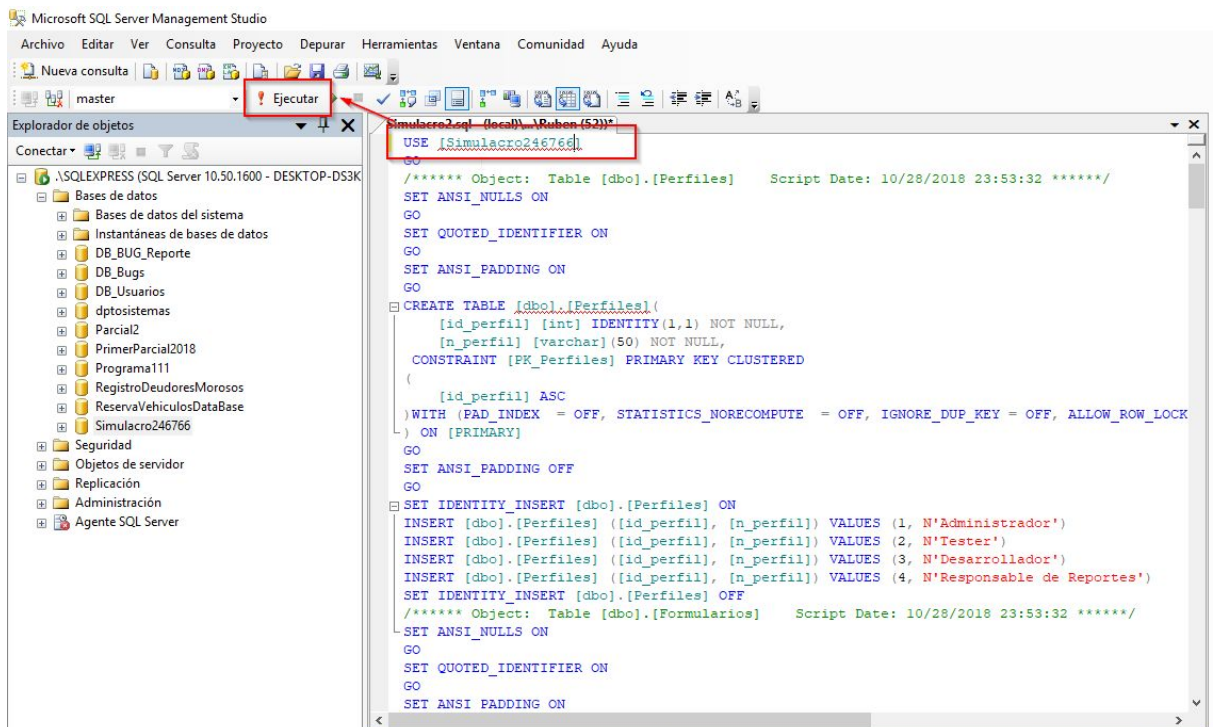
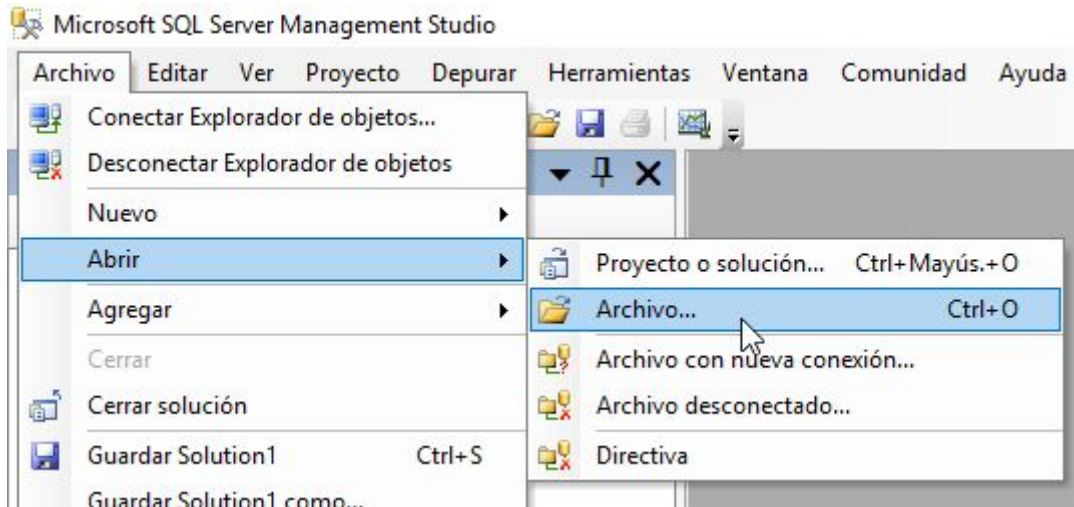


## Paso a Paso - Simulacro Segundo Parcial

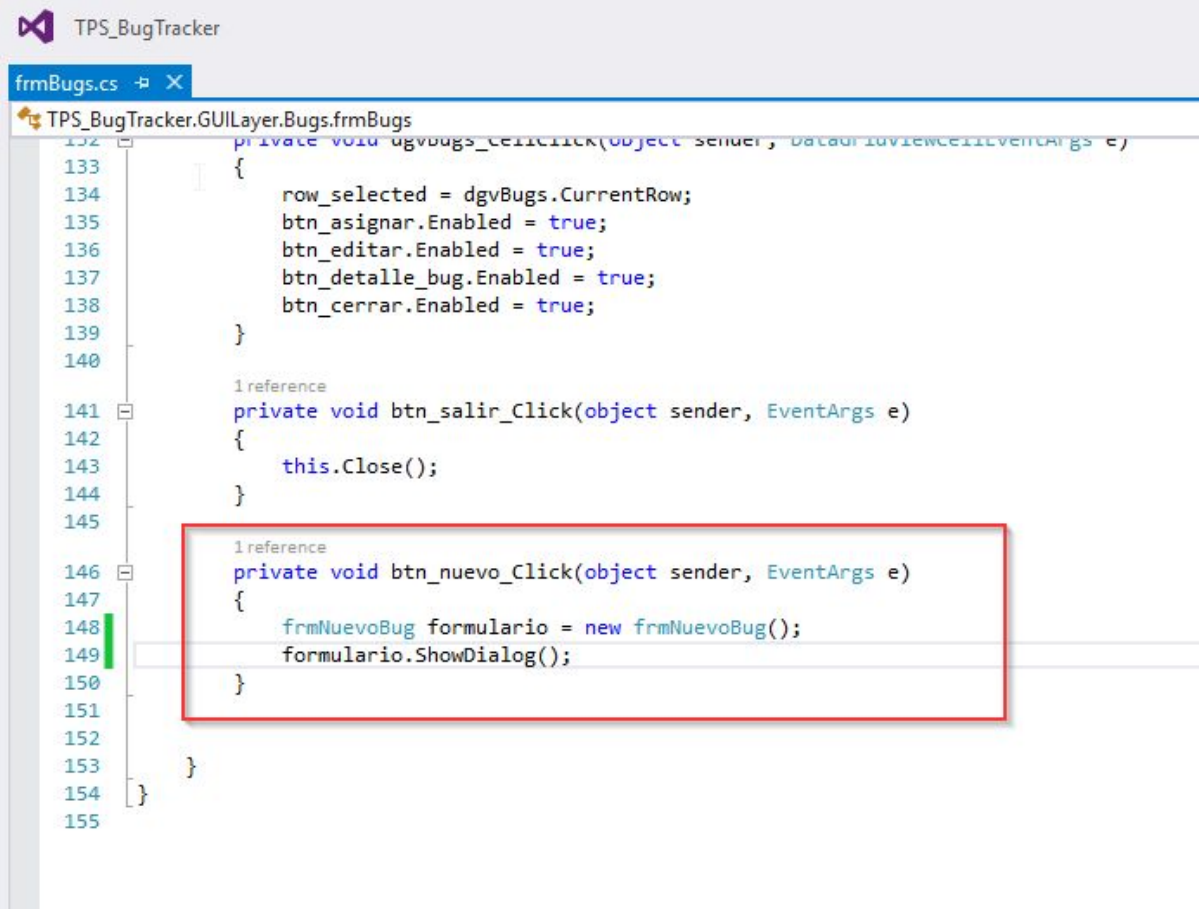
1. Crear base de datos Simulacro2+[legajo] (NO usar guiones).



2. Ejecutar el script **Simulacro2.sql** para crear las tablas de la base de datos creada anteriormente (se encuentra en la carpeta dp), usar el nombre de la base de datos creada anteriormente:

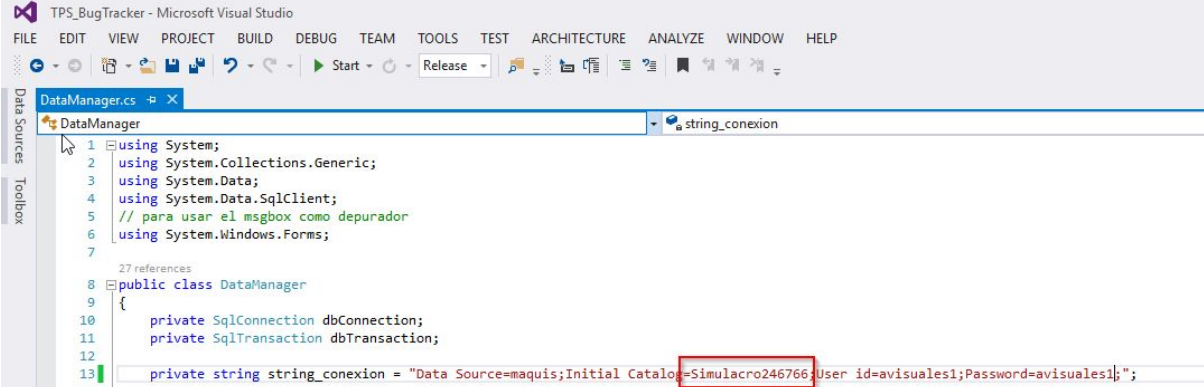


3. En el formulario **frmBugs** agregamos el evento click al botón Editar para que abra el formulario **frmNuevoBug**:



```
TPS_BugTracker
frmBugs.cs
TPS_BugTracker.GUILayer.Bugs.frmBugs
132 private void dgvBugs_CellClick(object sender, DataGridViewCellEventArgs e)
133 {
134     row_selected = dgvBugs.CurrentRow;
135     btn_asignar.Enabled = true;
136     btn_editar.Enabled = true;
137     btn_detalle_bug.Enabled = true;
138     btn_cerrar.Enabled = true;
139 }
140
141 1 reference
142 private void btn_salir_Click(object sender, EventArgs e)
143 {
144     this.Close();
145 }
146 1 reference
147 private void btn_nuevo_Click(object sender, EventArgs e)
148 {
149     frmNuevoBug formulario = new frmNuevoBug();
150     formulario.ShowDialog();
151 }
152
153 }
154 }
155
```

4. En clase **DataManager** modificar la variable **string\_conexion** con el nombre de la base de datos creada en el punto 1:



```
1 using System;
2 using System.Collections.Generic;
3 using System.Data;
4 using System.Data.SqlClient;
5 // para usar el msgbox como depurador
6 using System.Windows.Forms;
7
8 public class DataManager
9 {
10     private SqlConnection dbConnection;
11     private SqlTransaction dbTransaction;
12
13     private string string_conexion = "Data Source=maquis;Initial Catalog=Simulacro246766;User id=avisuales1;Password=avisuales1;";
14 }
```

5. En la clase **BugDao** agregamos el método **crearBug** que ejecute el INSERT para dar de alta un Bug y un BugsHistorico a la base de datos:



```
1 reference
public bool crearBug(Bug bug)
{
    DataManager dm = new DataManager(); // Creamos una instancia de la clase DataManager
    try
    {
        //Abrimos una Conexión
        dm.Open();

        //INICIAMOS LA TRANSACCION // Iniciamos la transacción
        dm.BeginTransaction();

        string insertBug = "INSERT INTO Bugs (titulo, descripcion, id_producto, id_prioridad, id_criticidad, id_estado, fecha_alta) +
            "VALUES (@titulo, @descripcion, @id_producto, @id_prioridad, @id_criticidad, @id_estado, GETDATE())";

        List<SqlParameter> paramBug = new List<SqlParameter>();
        paramBug.Add(new SqlParameter("id_bug", bug.id_bug));
        paramBug.Add(new SqlParameter("titulo", bug.titulo));
        paramBug.Add(new SqlParameter("descripcion", bug.descripcion));
        paramBug.Add(new SqlParameter("id_producto", bug.id_producto));
        paramBug.Add(new SqlParameter("id_prioridad", bug.id_prioridad));
        paramBug.Add(new SqlParameter("id_criticidad", bug.id_criticidad));
        paramBug.Add(new SqlParameter("id_estado", bug.id_estado));

        dm.EjecutarSQL(insertBug, paramBug);

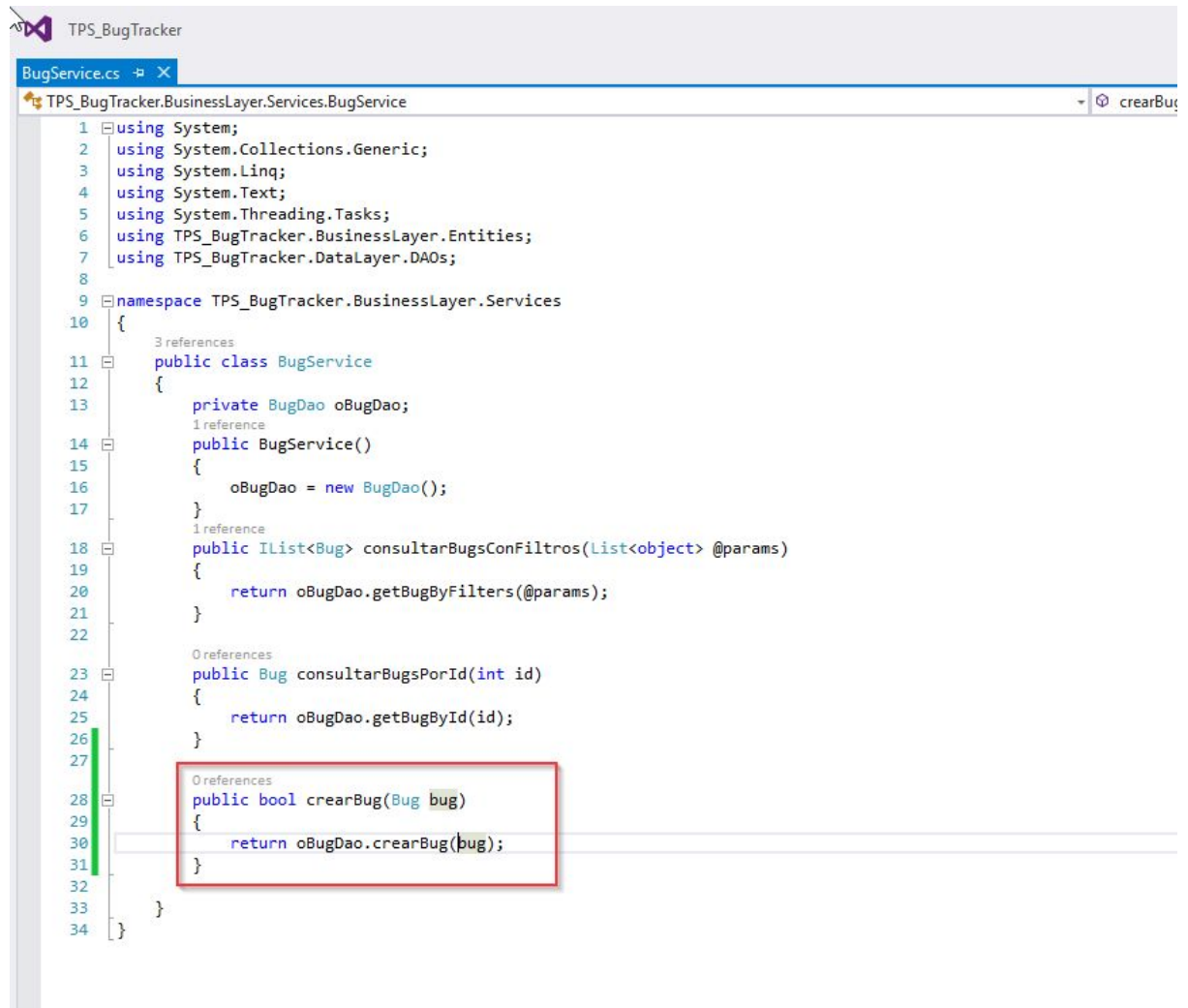
        string insertHistorico = "INSERT INTO BugsHistorico (id_bug, titulo, descripcion, id_producto, id_prioridad, id_criticidad, id_responsable, id_estado, fecha_historico) +
            "VALUES (@IDENTITY, @titulo, @descripcion, @id_producto, @id_prioridad, @id_criticidad, @id_responsable, @id_estado, GETDATE())";

        //Obtenemos el primer registro del historial para insertar en la tabla de historial
        HistorialBug historial = bug.historial.First();
        List<SqlParameter> paramHistorico = new List<SqlParameter>();
        paramHistorico.Add(new SqlParameter("titulo", bug.titulo));
        paramHistorico.Add(new SqlParameter("descripcion", bug.descripcion));
        paramHistorico.Add(new SqlParameter("id_producto", bug.id_producto));
        paramHistorico.Add(new SqlParameter("id_prioridad", bug.id_prioridad));
        paramHistorico.Add(new SqlParameter("id_criticidad", bug.id_criticidad));
        paramHistorico.Add(new SqlParameter("id_responsable", historial.responsable));
        paramHistorico.Add(new SqlParameter("id_estado", bug.id_estado));

        dm.EjecutarSQL(insertHistorico, paramHistorico);

        dm.Commit(); // Si todo esta bien confirmamos la transacción.
        return true;
    }
    catch (Exception ex)
    {
        dm.Rollback(); // Si algo salió mal volvemos atrás todo.
        return false;
    }
    finally
    {
        // Cierra la conexión
        dm.Close();
    }
}
```

6. En la clase **BugService** agregamos el método **crearBug** que a su vez invoque al método **BugDao.crearBug()**:



```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6  using TPS_BugTracker.BusinessLayer.Entities;
7  using TPS_BugTracker.DataLayer.DAOs;
8
9  namespace TPS_BugTracker.BusinessLayer.Services
10 {
11     public class BugService
12     {
13         private BugDao oBugDao;
14
15         public BugService()
16         {
17             oBugDao = new BugDao();
18         }
19
20         public IList<Bug> consultarBugsConFiltros(List<object> @params)
21         {
22             return oBugDao.getBugByFilters(@params);
23         }
24
25         public Bug consultarBugsPorId(int id)
26         {
27             return oBugDao.getBugById(id);
28         }
29
30         public bool crearBug(Bug bug)
31         {
32             return oBugDao.crearBug(bug);
33         }
34     }
35 }
```

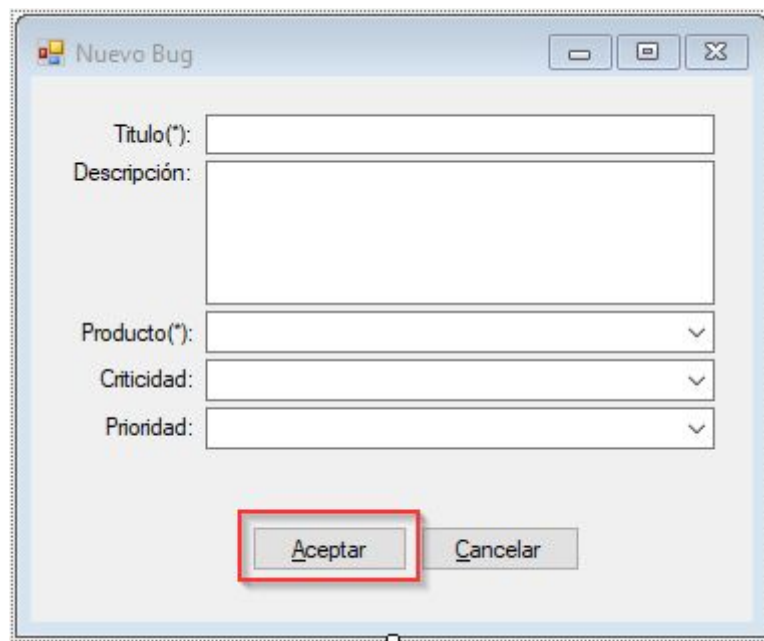
7. En el formulario **frmNuevoBug** programamos un método **validarCampos()** para validar los campos obligatorios del formulario:

```
1 reference
private bool validarCampos()
{
    if (string.IsNullOrEmpty(txtTitulo.Text))
    {
        MessageBox.Show("El campo Título es obligatorio.", "Validaciones", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
        return false;
    }

    if (cboProducto.SelectedIndex <= 0)
    {
        MessageBox.Show("El campo Producto es obligatorio.", "Validaciones", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
        return false;
    }

    return true;
}
```

8. En el formulario **frmNuevoBug** programamos el evento click del botón **Aceptar**, donde validamos los campos obligatorios, creamos una instancia de la clase **Bug** e invocamos el método **BugService.crearBug()**:





```

1 reference
private void btn_aceptar_Click(object sender, EventArgs e)
{
    try
    {
        if (validarCampos())
        {
            //Creamos una instancia de Bug

            Bug nuevoBug = new Bug();
            nuevoBug.titulo = txtTitulo.Text;
            nuevoBug.descripcion = txtDescripcion.Text;

            nuevoBug.id_estado = 1; // Estado "Nuevo"
            nuevoBug.id_prioridad = (int)cboPrioridad.SelectedValue;
            nuevoBug.id_criticidad = (int)cboCriticidad.SelectedValue;
            nuevoBug.id_producto = (int)cboProducto.SelectedValue;

            HistorialBug historial = new HistorialBug();
            historial.responsable = frmPrincipal.obtenerUsuarioLogin().id_usuario;
            historial.fecha = DateTime.Now;

            nuevoBug.addHistorial(historial);

            if (bugService.crearBug(nuevoBug))
            {
                MessageBox.Show("El bug se creó correctamente", "Nuevo Bug", MessageBoxButtons.OK, MessageBoxIcon.Information);
                this.Close();
            }
            else
            {
                MessageBox.Show("No pudo crear el Bug", "Nuevo Bug", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
            }
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show("Se produjo el siguiente error al crear un bug: " + ex.Message, "Error creando Bug", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}

```

9. En el formulario **frmNuevoBug** programamos el evento click del botón **Cancelar**, dónde debemos cerrar el formulario:

```

1 reference
private void btn_cancelar_Click(object sender, EventArgs e)
{
    this.Close();
}

```