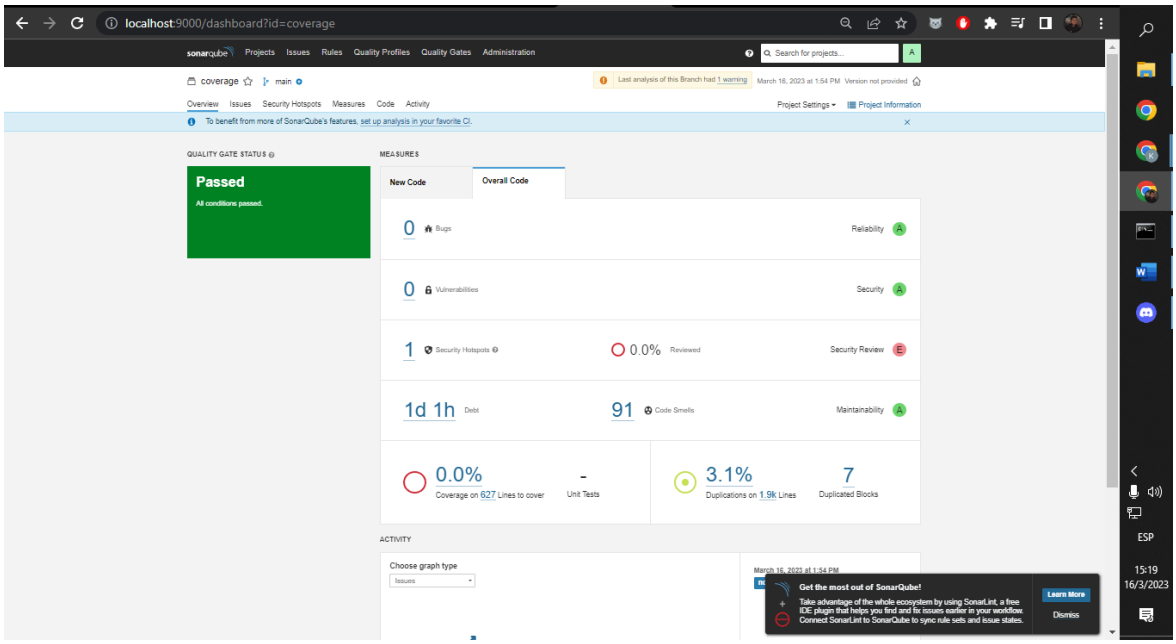
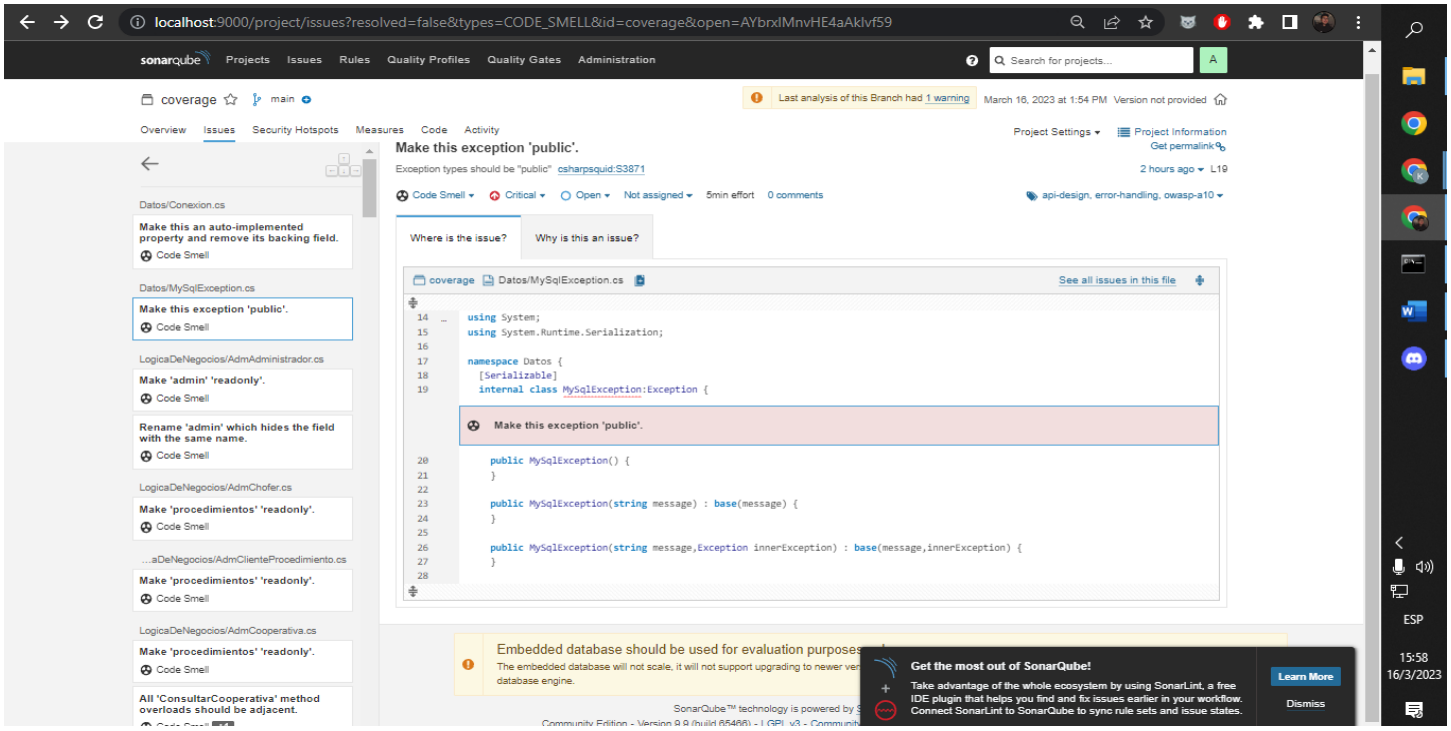
REQUISITOS DEL PROYECTO



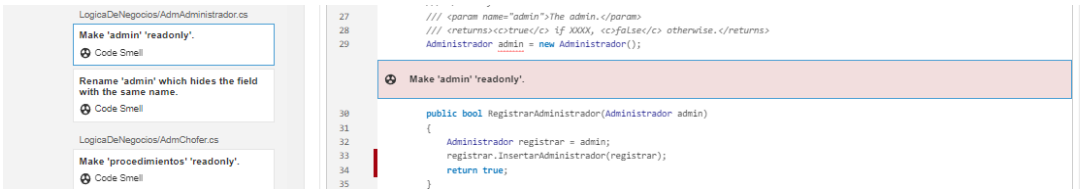
El Sistema esta codificado con C# el cual se debió implementar varios archivos externos debido que es (.net). Se realizo el testing estático sobre código utilizando la herramienta SonarQube y se observo que no existen bugs y 0 vulnerabilidades este análisis, por ahora se puede decir que es un sistema seguro.

Se reviso (**code smell**) y existen una gran cantidad de malos olores.

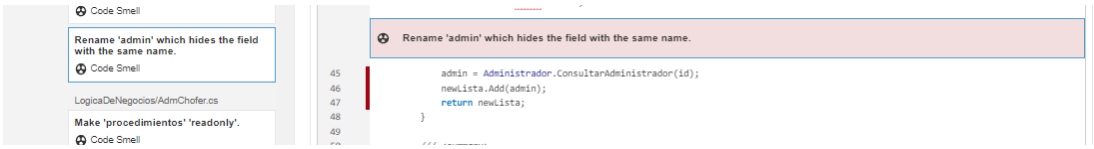


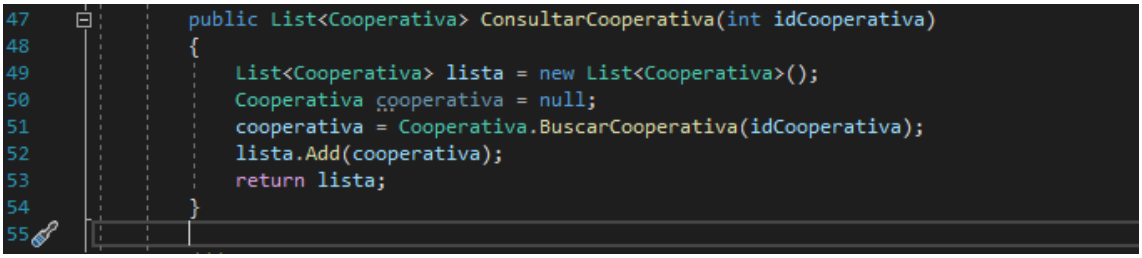
- Clases que deben ser pública.

- Atributos globales que únicamente se utiliza para leer datos.



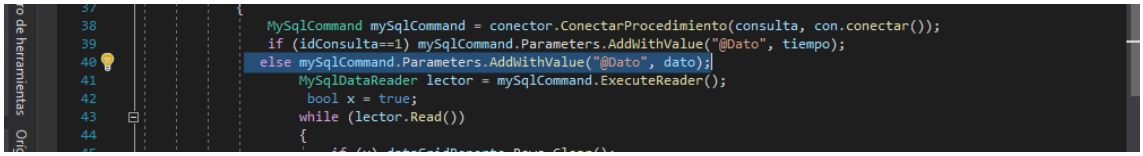
- Asignación innecesaria para un valor null.



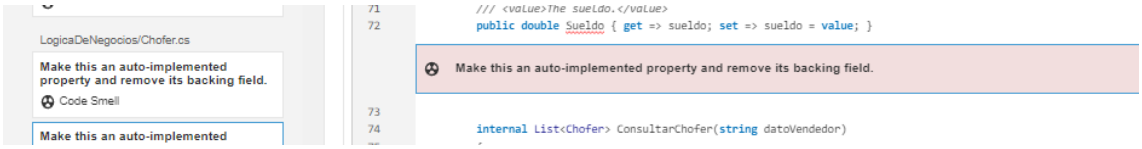


- Condición no completa puede generar deficiencia en el diseño.

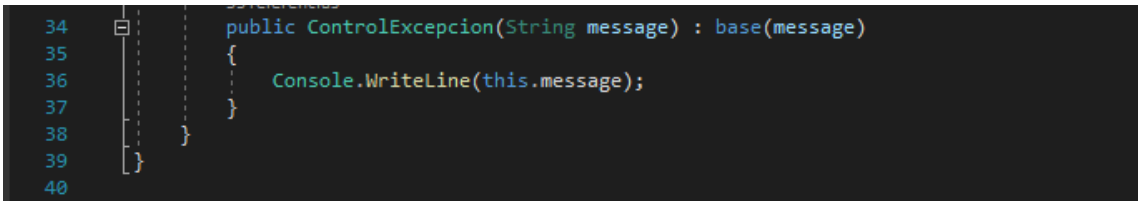
- Esta línea no se ejecutará condicionalmente; solo lo estará la primera línea de este bloque de 2 líneas. El resto se ejecutará incondicionalmente.



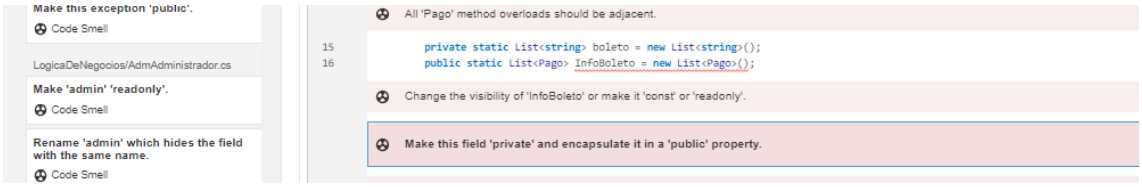
- Implementación de 2 propiedades



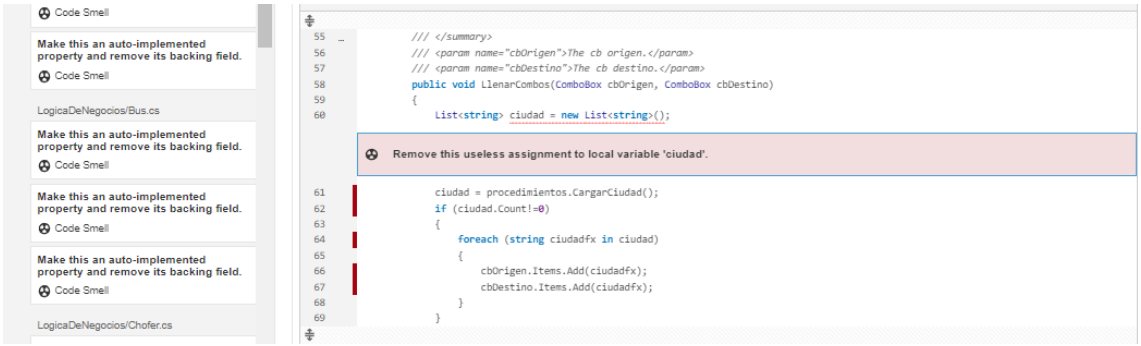
- No se define el tipo de dato para el retorno de valor a la clase



- Clases que deben ser privadas por temas de seguridad

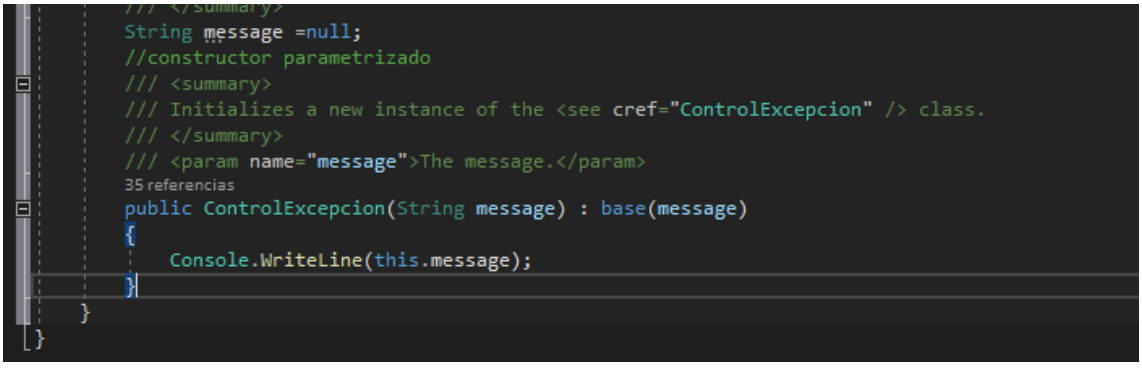


- Crear variables que ya reciben datos



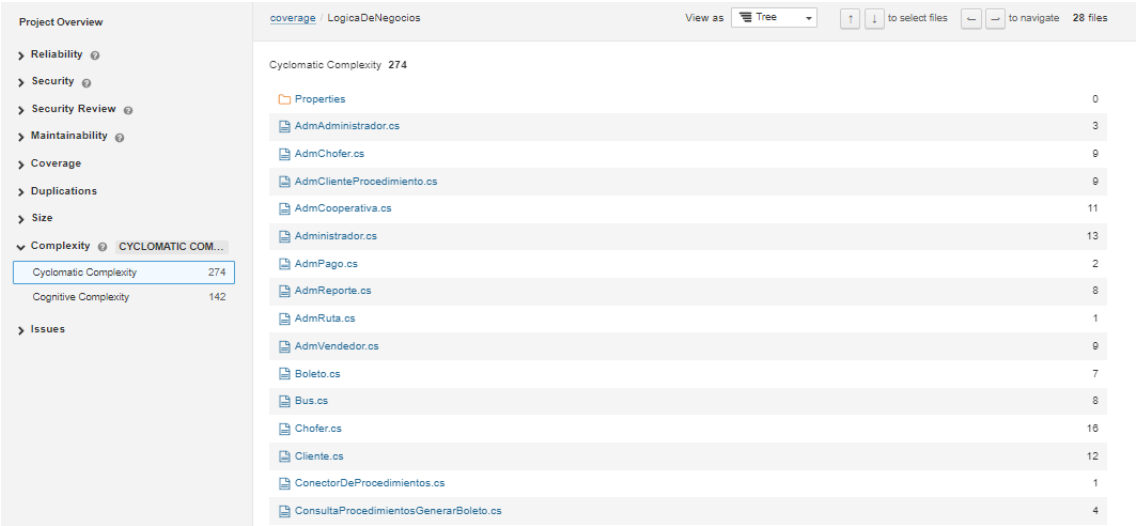
- Fallas de sintaxis

- Variables que no se utilizan (**message**)



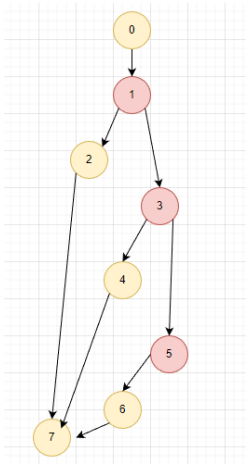
Complejidad Ciclomatica

Complejidad Ciclomatica (274)



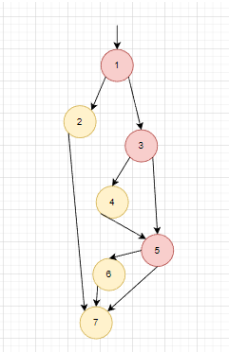
- AdmVendedor **Consultar Vendedor ()**

```
        List<Vendedor> vendedor = null;
        if (String.IsNullOrEmpty(datoVendedor))
        {
            throw new ControlExcepcion("Campo vacio por favor rellenar");
        }
        else
        {
            if (datoVendedor == "Iniciar_data_grid_datos")
            {
                datoVendedor = "";
            }
            vendedor = procedimientos.BuscarVendedor(datoVendedor);

            if (vendedor.Count == 0)
            {
                throw new ControlExcepcion("Vendedor no encontrado");
            }
        }

        return vendedor;
    }
```



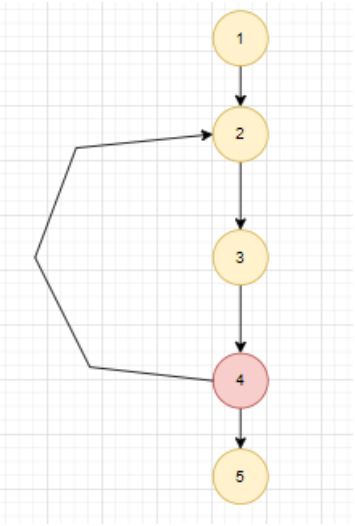
- AdmCliente **Consultar Cliente ()**

```
public List<Cliente> ConsultarCliente(string Dato )
{
    List<Cliente> client = null;
    if (String.IsNullOrEmpty(Dato))
    {
        throw new ControlExcepcion("Campo vacio por favor rellenar");
    }
    else
    {
        if (Dato == "Iniciar_data_grid_datos")
        {
            Dato = "";
        }
        client = procedimientos.BuscarCliente(Dato);

        if (client.Count == 0)
        {
            throw new ControlExcepcion("Cliente no encontrado");
        }
    }

    return client;
}
```



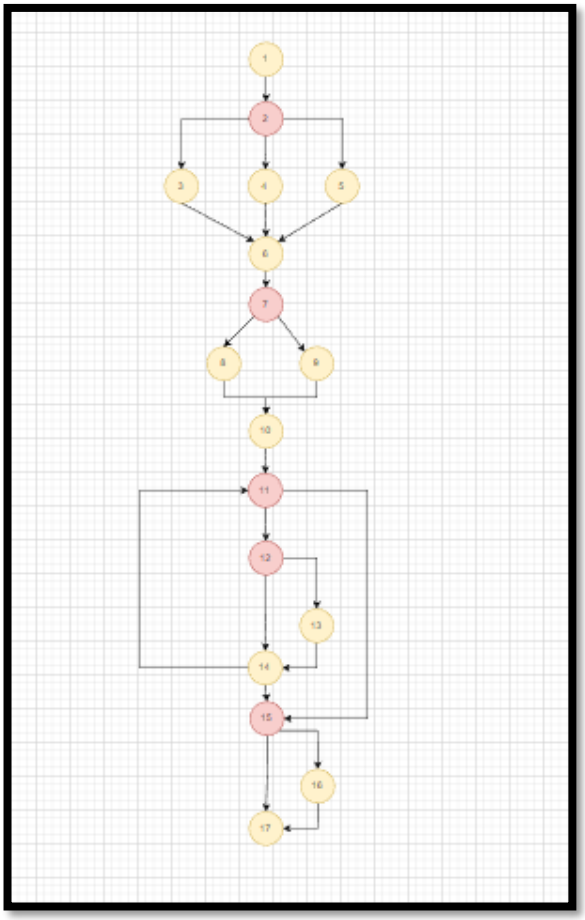
- CalcularTotalPagar**()**

```
/// <summary>
/// Se realiza el medodo para obtener el total a pagar
/// </summary>
/// <param name="precio"></param>
/// <returns></returns>
public double calcularTotalPagar(List<double> precio)
{
    double totalPagar = 0;
    foreach(double p in precio)
    {
        totalPagar += p;
    }
    return totalPagar;
}
}
```



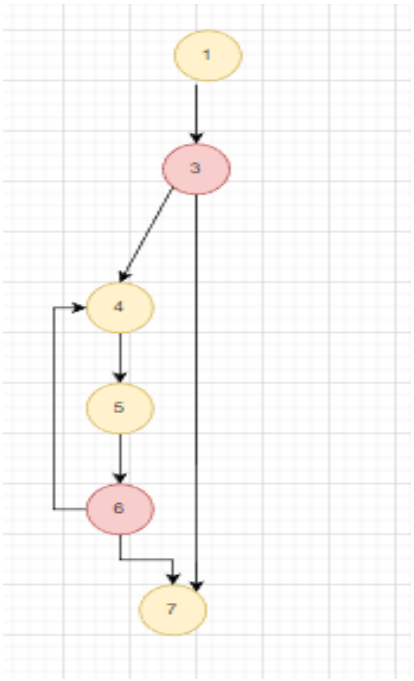
- LLenarDataGrid()

```
Conexion con = new Conexion();
ConectorDeProcedimientos conector = new ConectorDeProcedimientos();
string consulta = "";
switch (idConsulta)
{
    case 1:  consulta = "BuscarReporteFecha";
        tiempo = Convert.ToDateTime(dato);
        break;
    case 2:  consulta = "BuscarReporteCooperativa";
        break;
    case 3: consulta = "BuscarReporteCedula"; break;
    default: consulta = "BuscarReporte"; break;
}
try
{
    MySqlCommand mySqlCommand = conector.ConectarProcedimiento(consulta, con.conectar());
    if (idConsulta==1) mySqlCommand.Parameters.AddWithValue("@Dato", tiempo);
   else mySqlCommand.Parameters.AddWithValue("@Dato", dato);
        MySqlDataReader lector = mySqlCommand.ExecuteReader();
         bool x = true;
        while (lector.Read())
        {
            if (x) dataGridReporte.Rows.Clear();
            int numerofila = dataGridReporte.Rows.Count;
            dataGridReporte.Rows.Add(1);
            dataGridReporte.Rows[numerofila].Cells[0].Value = lector["Id_compra"].ToString();
            dataGridReporte.Rows[numerofila].Cells[1].Value = lector["Fecha_Compra"].ToString();
            dataGridReporte.Rows[numerofila].Cells[2].Value = lector["Fecha_Salida"].ToString();
            dataGridReporte.Rows[numerofila].Cells[3].Value = lector["Cedula_Comprador"].ToString();
            dataGridReporte.Rows[numerofila].Cells[4].Value = lector["Cedula_cliente"].ToString();
            dataGridReporte.Rows[numerofila].Cells[5].Value = lector["cooperativa"].ToString();
            dataGridReporte.Rows[numerofila].Cells[6].Value = lector["HoraSalida"].ToString();
            dataGridReporte.Rows[numerofila].Cells[7].Value = lector["Placa"].ToString();
            x = false;
    }
    con.cerrar();
    if (x)
    {
        throw new ControlExcepcion("No existe reporte");
    }
    return x;
}
catch
{
    throw new ControlExcepcion("No existe reporte");
}
```



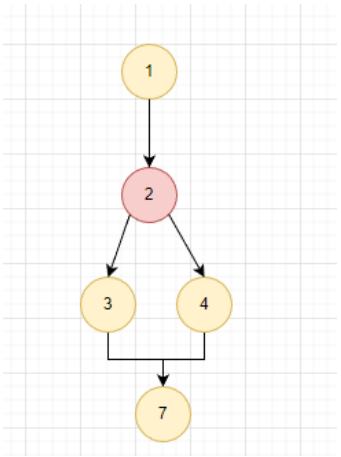
- LLenarComboAsiento ()

```
        public void LlenarComboAsientos(int busId, ComboBox cbNumeroAsientos,int BoletoID)
        {
            // Se crea una lista string llamada  NumeroAsiento en la que se almacenara los numeros de asientos de un bus en
particular.
            List<string> NumeroAsiento = procedimientos.BuscarNumerosAsientos(busId, BoletoID);
            if (NumeroAsiento.Count != 0)
            {
                foreach (string asiento in NumeroAsiento)
                {
                    cbNumeroAsientos.Items.Add(asiento);
                }
            }

        }
    }
}
```



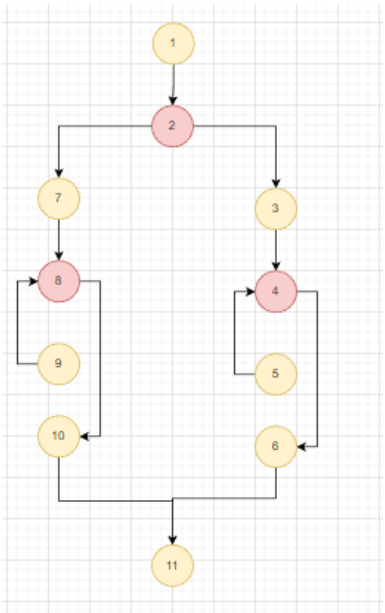
- Login()

```
{
    List<int> IdPeronsaAndRol = procedimientos.IniciasSeccion(usuario, password);
    if(IdPeronsaAndRol.Count > 0)
    {
        MessageBox.Show("Inicio de sesión realizado con éxito.");
    }
    else
    {
        throw new ControlExcepcion("Usuario y/o contrasena incorrecta");

    }
    return IdPeronsaAndRol;
}
```



- LlenarCombo()

```
internal void LLenarCombos(int idCooperativa, Guna2ComboBox combo,int x)
{
   Conexion con = new Conexion();
    ConectorDeProcedimientos conector = new ConectorDeProcedimientos();
    if (x != 0)
    {
        Console.WriteLine("KKKKK"+x);
        try
        {
            Console.WriteLine(idCooperativa + "" + x);
            MySqlCommand mySqlCommand = conector.ConectarProcedimiento("spl_Buscarbus", con.conectar());
            mySqlCommand.Parameters.AddWithValue("@idCooperativa", idCooperativa);
            mySqlCommand.Parameters.AddWithValue("@idRuta", x);
            MySqlDataReader lector = mySqlCommand.ExecuteReader();
            while (lector.Read())
            {
                Console.WriteLine("jjjjjjjjjjjjjjjjjjjjjjj");
                string anexar = lector["Placa"].ToString();
                combo.Items.Add(anexar);
                ;
                ;
            }
            con.cerrar();
        }
        catch (MySqlException ex)
        {
            Console.WriteLine("Error emitido por: " + ex);
        }
    }
    else
    {
        try
        {
            MySqlCommand mySqlCommand = conector.ConectarProcedimiento("spl_BuscarRutA", con.conectar());
            mySqlCommand.Parameters.AddWithValue("@idCooperativa", idCooperativa);
            MySqlDataReader lector = mySqlCommand.ExecuteReader();
            while (lector.Read())
            {
                string anexar = lector["idRuta"].ToString() + ")  De: " + lector["Salida"].ToString() + "
lector["Destino"].ToString(); ;
                combo.Items.Add(anexar);
            }
            con.cerrar();
        }
        catch (MySqlException ex)
        {
```



- GuardarPdf()

```
public static List<string> GuardarPdf()
{
    List<string> list = new List<string>();
    string tablas = string.Empty;
    double precio=0;
    string cliente= string.Empty;
    Conexion con = new Conexion();
    ConectorDeProcedimientos conector = new ConectorDeProcedimientos();
    try
    {
        MySqlCommand mySqlCommand = conector.ConectarProcedimiento("ImprimirBoleto", con.conectar());
        mySqlCommand.Parameters.AddWithValue("@cantboleto", MNumeroboleto);
        MySqlDataReader lector = mySqlCommand.ExecuteReader();
        while (lector.Read())
        {
            tablas += "<tr>";
            tablas += "<td>" + lector["Id_compra"].ToString() + "</td>";
            tablas += "<td>" + lector["cooperativa"].ToString() + "</td>";
            tablas += "<td>" + lector["Placa"].ToString() + "</td>";
            tablas += "<td>" + lector["Cedula_Comprador"].ToString() + "</td>";
            tablas += "<td>" + lector["Fecha_Salida"].ToString() + "</td>";
            tablas += "<td>" + lector["HoraSalida"].ToString() + "</td>";
            tablas += "<td>" + lector["Precio"].ToString() + "</td>";
            tablas += "</tr>";
            precio = precio + Convert.ToDouble(lector["Precio"]);
            MNumeroboleto--;
            cliente = lector["Nombre_cliente"].ToString();
        }
        list.Add(tablas);
        list.Add(Convert.ToString(precio));
        list.Add(cliente);
        con.cerrar();
    }
    catch (MySqlException ex)
    {
        Console.WriteLine("Error emitido por: " + ex);
    }
    return list;
}
}
```