Universidad de Guayaquil
Facultad de Ciencias Matemáticas y Físicas
Carrera de Software

2022

Proyecto

Docente: CUENCA ORTEGA ANGEL EDUARDO

INTEGRANTES

- Barrera Remache Carlos
- Velasquez Choez Kevin
- Beltran Santistevan Carlos

Guayaquil – 15-01-2023

Materia: VERIFICACIÓN Y VALIDACIÓN DE SOFTWARE

Paralelo: SOF-S-MA-8-1

Ciclo: 2022 - 2023 CII

Contenido

Objetivo

Aplicar el análisis estático en el proyecto de software desarrollado, para determinar la complejidad ciclomática, errores de convenciones de programación y aplicación de estilo de código, a través de la generación de grafos y uso de guías.

Especificación de guía o estándar de codificación y herramienta

Estándar de codificación

Se estableció como estándar de codificación (errores de convenciones de programación y aplicación de estilo) para C# el brindado por Microsoft. El estándar de codificación lo puede consultar en el siguiente enlace https://learn.microsoft.com/en-us/dotnet/csharp/fundamentals/coding-style/coding-conventions o verifique en el repositorio el documento “C# Coding Conventions _ Microsoft Learn”.

Herramienta

La herramienta usada es ReSharper la cual permite usar la configuración de la herramienta StyleCop, que ayuda a verificar el cumplimiento del estándar de codificación de C# de Microsoft. StyleCop trae por defecto la configuración para el estándar de codificación de C# de Microsoft.

Casos de uso

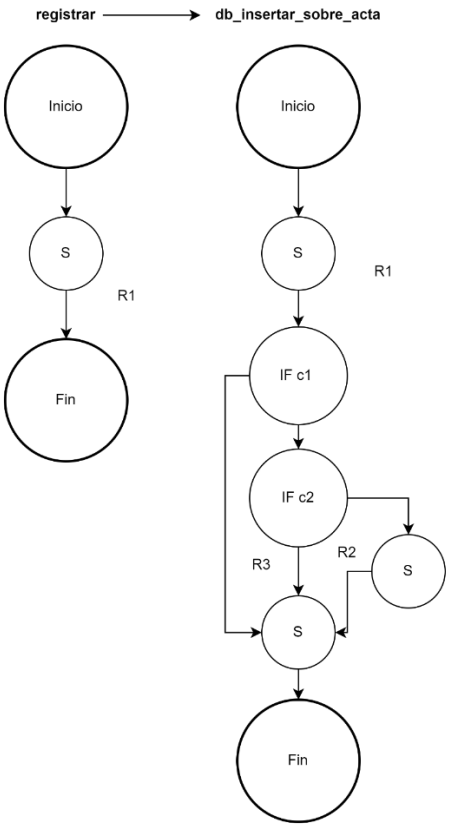
Los siguientes métodos están en la capa de lógica de negocio en sus clases respectivas.

CU1: ClsActa - agregar

Grafo

```
public virtual String registrar() {
    string msj = "";
    //Lista genérica de parámetros
    List<ClsParametros> lst = new List<ClsParametros>();

    try {
        //Pasar los parámetros hacia la capa de acceso a datos
        ClsParametros cp = new ClsParametros();
        lst.Add(cp);
        M.db_insertar_sobre_acta(lst);
        msj = "Insertado correctamente";
    } catch (Exception ex) {
        msj = "Error al insertar los datos";
        return msj;
        throw ex;
    }

    return msj;
}
```

```
public String db_insertar_sobre_acta(List<ClsParametros> lst) {
    String mensaje = "";
    try {
        if (mensaje == "" && lst != null) {
            SqlConnection conexion = abrir_conexion();

            String cadena = "INSERT INTO Acta
              (fecha,partido,marcador_acta,equipoa_acta,equipob_acta) VALUES
              (@fecha,@partido,@marcador_acta,@equipoa_acta,@equipob_acta)";

            SqlCommand comando = new SqlCommand(cadena, conexion);

            comando.ExecuteNonQuery();

        }
    } catch (Exception ex) {
        cerrar_conexion(); //Por si entra aqui con conexion abierta
        throw ex;
    }
    cerrar_conexion();
    return mensaje;
}
```



Complejidad ciclomática

Registrar

Forma 1	Forma 2	Forma 3
M = E – N +2P	M = R	M = NP+1
M = 2 – 3 +2(1) = 1	M =1	M = 0 + 1

db_insertar_sobre_acta

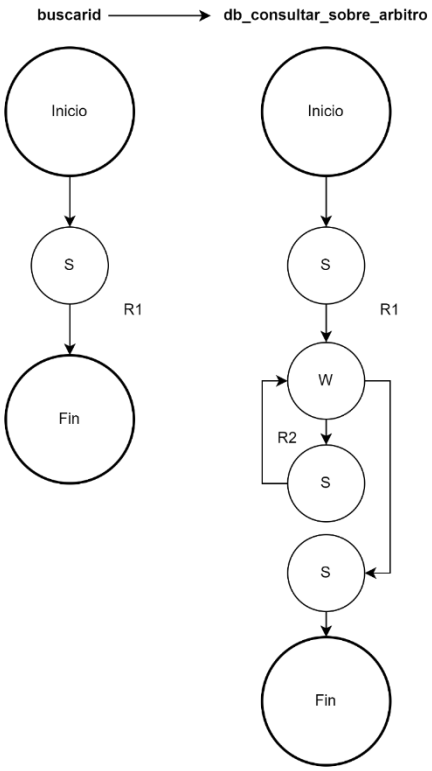
Forma 1	Forma 2	Forma 3
M = E – N +2P	M = R	M = NP+1
M = 8 - 7 + 2(1) = 3	M =3	M = 2 + 1 = 3

CU7: ClsArbitro - consultar

Grafo

```
public override Tuple<List<object>, SqlDataAdapter> Buscarid(int idPersona)
{
    return this.m.db_consultar_sobre_arbitro(
        "SELECT [usuario],[psw],[id_persona],[nombre_persona],[apellido],[cedula],
          [licencia] FROM [dbo].[Arbitro] WHERE id_persona = "
        + idPersona);
}
```

```
public Tuple<List<Object>, SqlDataAdapter> db_consultar_sobre_arbitro(string
  sentenciaSQL) {
    //lreturn.Clear();
    SqlDataReader dr = null;
    List<Object> lreturn = new List<Object>();
    SqlDataAdapter adaptador;
    try {
        SqlConnection conexion = abrir_conexion();
        String cadena = sentenciaSQL;
        SqlCommand comando = new SqlCommand(cadena, conexion);
        //comando.CommandText = "";//Linea opcional para rellenar los datos
        dr = comando.ExecuteReader();
        adaptador = new SqlDataAdapter(cadena, conexion);

        while (dr.Read()) {
            var tmp = new {
                usuario = dr["usuario"].ToString(),
                psw = dr["psw"].ToString(),
                id_persona = Convert.ToInt32(dr["id_persona"]),
                nombre_persona = dr["nombre_persona"].ToString(),
                apellido = dr["apellido"].ToString(),
                cedula = dr["cedula"].ToString(),
                licencia = dr["licencia"].ToString()
            };
            lreturn.Add(tmp);
        }
    } catch (Exception ex) {
        cerrar_conexion(); //Por si entra aqui con conexion abierta
        throw ex;
    }
    cerrar_conexion();
    return Tuple.Create(lreturn, adaptador);
    //return lreturn;
}
```



Complejidad ciclomática

buscarid()

Forma 1	**Forma 2**	**Forma 3**
M = E - N +2P	M = R	M = NP+1
M = 2 - 3 + 2 = 1	M = 1	M = 0 + 1 = 1

db_consultar_sobre_arbitro ()

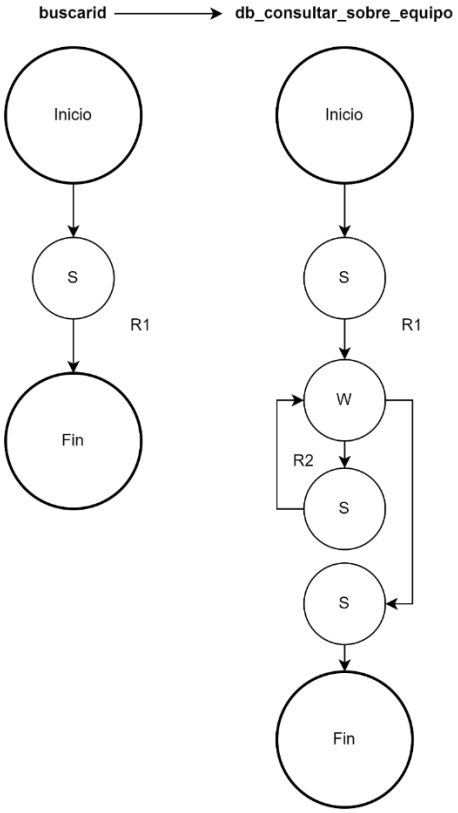
Forma 1	**Forma 2**	**Forma 3**
M = E - N + 2P	M = R	M = NP+1
M = 6 – 6 + 2 = 2	M = 2	M= 1 + 1 = 2

CU8: ClsEquipo - consultar

Grafo

```
public virtual Tuple<List<object>, SqlDataAdapter> Buscarid(int idEquipo)
{
    return this.m.db_consultar_sobre_equipo(
        "SELECT [id_equipo],[jugadores] FROM [dbo].[Equipo] WHERE id_equipo = " +
          idEquipo);
}
```

```
public Tuple<List<Object>, SqlDataAdapter> db_consultar_sobre_equipo(string
  sentenciaSQL) {
    //lreturn.Clear();
    SqlDataReader dr = null;
    List<Object> lreturn = new List<Object>();
    SqlDataAdapter adaptador;
    try {
        SqlConnection conexion = abrir_conexion();
        String cadena = sentenciaSQL;
        SqlCommand comando = new SqlCommand(cadena, conexion);
        //comando.CommandText = "";//Linea opcional para rellenar los datos
        dr = comando.ExecuteReader();
        adaptador = new SqlDataAdapter(cadena, conexion);

        while (dr.Read()) {
            var tmp = new {
                id_equipo = Convert.ToInt32(dr["id_equipo"]),
                jugadores = Convert.ToInt32(dr["jugadores"])
            };
            lreturn.Add(tmp);
        }
    } catch (Exception ex) {
        cerrar_conexion(); //Por si entra aqui con conexion abierta
        throw ex;
    }
    cerrar_conexion();
    return Tuple.Create(lreturn, adaptador);
    //return lreturn;
}
```



Complejidad ciclomática

buscarid()

Forma 1	**Forma 2**	**Forma 3**
M = E - N +2P	M = R	M = NP+1
M = 2 - 3 + 2 = 1	M = 1	M = 0 + 1 = 1

db_consultar_sobre_equipo()

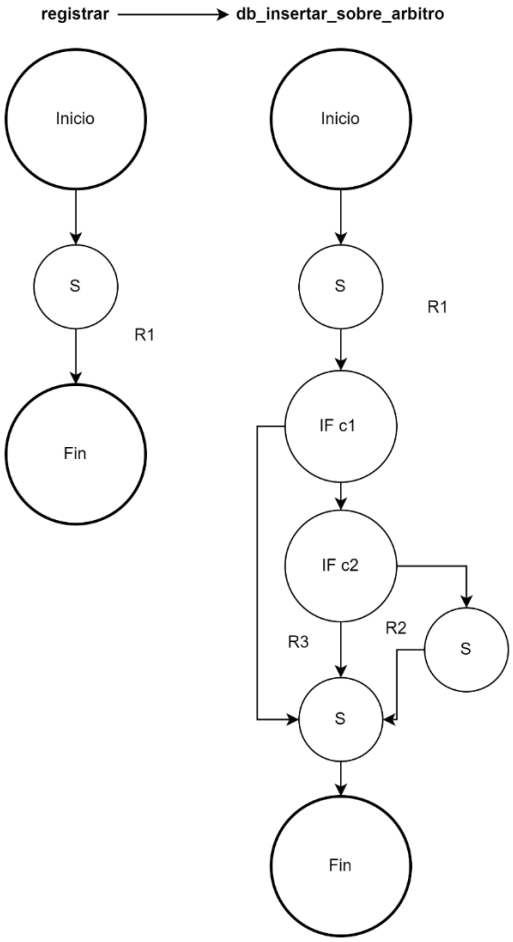
Forma 1	**Forma 2**	**Forma 3**
M = E - N + 2P	M = R	M = NP+1
M = 6 – 6 + 2 = 2	M = 2	M= 1 + 1 = 2

CU9: ClsArbitro – agregar

Grafo

```
public override string Registrar()
{
    string msj = string.Empty;

    // Lista genérica de parámetros
    List<ClsParametros> lst = new List<ClsParametros>();

    try
    {
        // Pasar los parámetros hacia la capa de acceso a datos
        ClsParametros cp = new ClsParametros();
        cp.setArbitro(
            this.Usuario,
            this.Psw,
            this.Id_persona,
            this.Nombres,
            this.Apellidos,
            this.Cedula,
            this.Licencia);
        lst.Add(cp);
        this.m.db_insertar_sobre_arbitro(lst);
        msj = "Insertado correctamente";
    }
    catch (Exception ex)
    {
        msj = "Error al insertar los datos";
        return msj;
        throw ex;
    }

    return msj;
}
```

```
public String db_insertar_sobre_arbitro(List<ClsParametros> lst) {
    String mensaje = "";
    try {
        if (mensaje == "" && lst != null) {
            SqlConnection conexion = abrir_conexion();

            String cadena = "INSERT INTO Arbitro
              (usuario,psw,nombre_persona,apellido,cedula,licencia) VALUES
              (@usuario,@psw,@nombre_persona,@apellido,@cedula,@licencia)";

            SqlCommand comando = new SqlCommand(cadena, conexion);

            comando.Parameters.AddWithValue("@usuario", lst[0].Usuario);
            comando.Parameters.AddWithValue("@psw", lst[0].Psw);
            comando.Parameters.AddWithValue("@nombre_persona", lst[0].Nombres);
            comando.Parameters.AddWithValue("@apellido", lst[0].Apellidos);
            comando.Parameters.AddWithValue("@cedula", lst[0].Cedula);
            comando.Parameters.AddWithValue("@licencia", lst[0].Licencia);

            comando.ExecuteNonQuery();

        }
    } catch (Exception ex) {
        cerrar_conexion(); //Por si entra aqui con conexion abierta
        throw ex;
    }
    cerrar_conexion();
    return mensaje;
}
```



Complejidad ciclomática

registrar()

Forma 1	**Forma 2**	**Forma 3**
M = E - N +2P	M = R	M = NP+1
M = 2 - 3 + 2 = 1	M = 1	M = 0 + 1 = 1

db_insertar_sobre_arbitro()

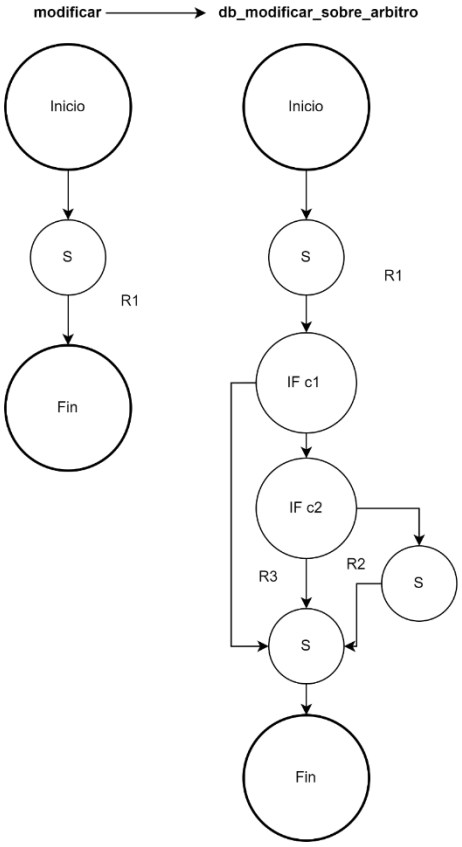
Forma 1	**Forma 2**	**Forma 3**
M = E - N + 2P	M = R	M = NP+1
M = 8 – 7 + 2 = 3	M = 3	M= 2 + 1 = 3

CU20: ClsArbitro - editar

Grafo

```
public override string Modificar()
{
    string msj = string.Empty;

    // Lista genérica de parámetros
    List<ClsParametros> lst = new List<ClsParametros>();

    try
    {
        // Pasar los parámetros hacia la capa de acceso a datos
        ClsParametros cp = new ClsParametros();
        cp.setArbitro(
            this.Usuario,
            this.Psw,
            this.Id_persona,
            this.Nombres,
            this.Apellidos,
            this.Cedula,
            this.Licencia);
        lst.Add(cp);
        this.m.db_modificar_sobre_arbitro(lst);
        msj = "Insertado correctamente";
    }
    catch (Exception ex)
    {
        msj = "Error al insertar los datos";
        return msj;
        throw ex;
    }

    return msj;
}
```

```
public String db_modificar_sobre_arbitro(List<ClsParametros> lst) {
    String mensaje = "";
    try {
        if (mensaje == "" && lst != null) {
            SqlConnection conexion = abrir_conexion();

            String cadena = "UPDATE Arbitro SET usuario = @usuario, psw =
              @pswnombre_persona = @Nombres, Apellidos = @apellido, cedula = @cedula,
               licencia = @licencia WHERE id_persona = @id_persona";

            SqlCommand comando = new SqlCommand(cadena, conexion);

            comando.Parameters.AddWithValue("@usuario", lst[0].Usuario);
            comando.Parameters.AddWithValue("@psw", lst[0].Psw);
            comando.Parameters.AddWithValue("@id_persona", lst[0].Id_persona);
            comando.Parameters.AddWithValue("@nombre_persona", lst[0].Nombres);
            comando.Parameters.AddWithValue("@apellido", lst[0].Apellidos);
            comando.Parameters.AddWithValue("@cedula", lst[0].Cedula);
            comando.Parameters.AddWithValue("@licencia", lst[0].Licencia);

            comando.ExecuteNonQuery();

        }
    } catch (Exception ex) {
        cerrar_conexion(); //Por si entra aqui con conexion abierta
        throw ex;
    }
    cerrar_conexion();
    return mensaje;
}
```



Complejidad ciclomática

modificar()

Forma 1	Forma 2	Forma 3
M = E - N +2P	M = R	M = NP+1
M = 2 - 3 + 2 = 1	M = 1	M = 0 + 1 = 1

db_modificar_sobre_arbitro()

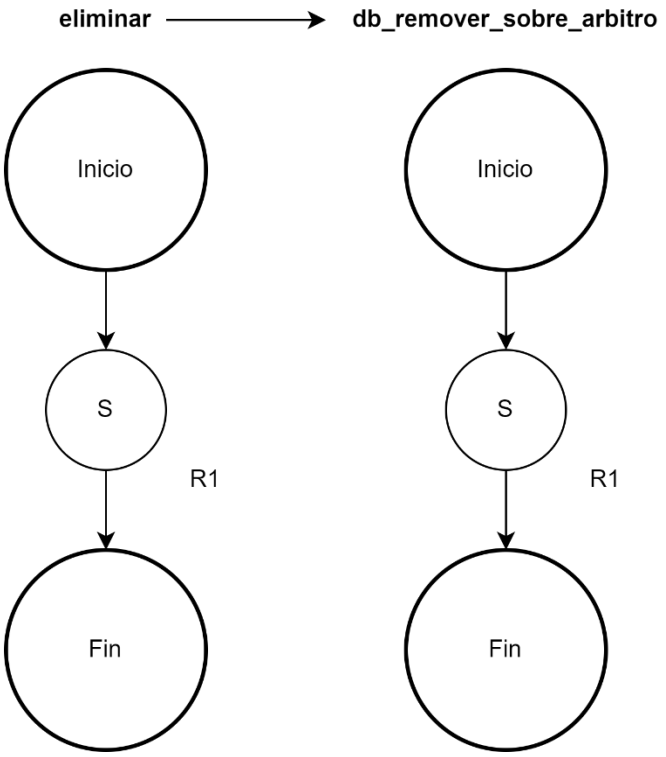
Forma 1	Forma 2	Forma 3
M = E - N + 2P	M = R	M = NP+1
M = 8 – 7 + 2 = 3	M = 3	M= 2 + 1 = 3

CU21: ClsArbitro - eliminar

Grafo

```
public override string Eliminar(int idPersona)
{
    return this.m.db_remover_sobre_arbitro(idPersona);
}
```

```
public String db_remover_sobre_arbitro(int Id_persona) {
    String mensaje = "";
    try {
        SqlConnection conexion = abrir_conexion();
        //Alerta cambiar el idCambiar por el de la base de datos
        String cadena = "delete from Arbitro where Id_persona = @Id_persona";
        SqlCommand comando = new SqlCommand(cadena, conexion);
        comando.Parameters.AddWithValue("@Id_persona", Id_persona);
        comando.ExecuteNonQuery();
        mensaje = "Se realizo la consulta exitosamente de eliminacion de registro";
    } catch (Exception ex) {
        cerrar_conexion(); //Por si entra aqui con conexion abierta
        throw ex;
    }
    cerrar_conexion();
    return mensaje;
}
```



Complejidad ciclomática

eliminar()

Forma 1	Forma 2	Forma 3
M = E - N +2P	M = R	M = NP+1
M = 2 - 3 + 2 = 1	M = 1	M = 0 + 1 = 1

db_remover_sobre_arbitro()

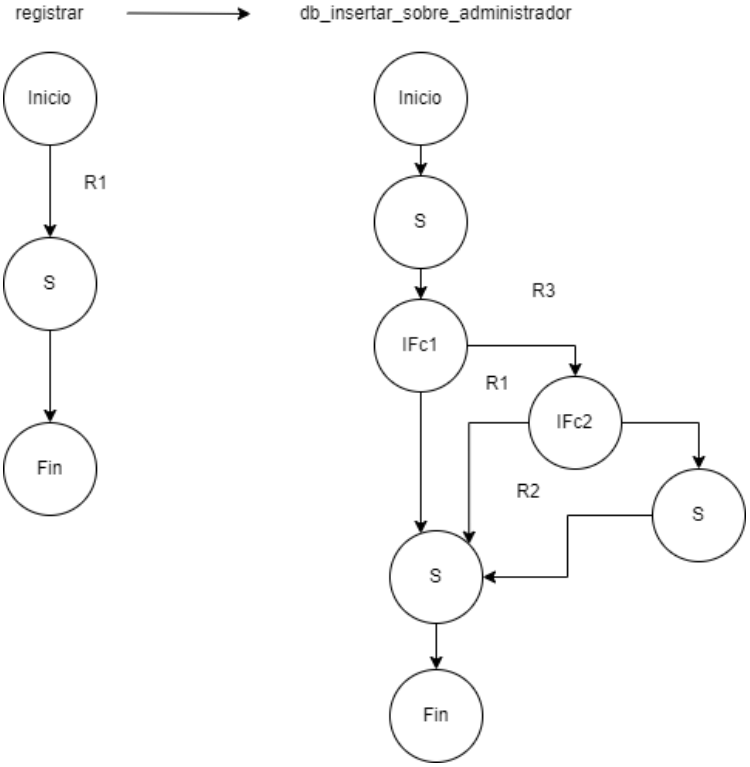
Forma 1	Forma 2	Forma 3
M = E - N +2P	M = R	M = NP+1
M = 2 - 3 + 2 = 1	M = 1	M = 0 + 1 = 1

CU24: ClsAdministrador - agregar

Grafo

```
//Registrar administrador
4 referencias
public override string Registrar()
{
    string msj = "";

    //Lista genérica de parámetros
    List<ClsParametros> lst = new List<ClsParametros>();

    try
    {
        //Pasar los parámetros hacia la capa de acceso a datos
        ClsParametros cp = new ClsParametros();
        cp.setAdministrador(Id_persona, Nombres, Apellidos, Cedula, Usuario, Psw);
        lst.Add(cp);
        M.db_insertar_sobre_administrador(lst);
        msj = "Insertado correctamente";
    }
    catch (Exception ex)
    {
        msj = "Error al insertar los datos";
        return msj;
        throw ex;
    }

    return msj;
}
```

```
1 referencia
public String db_insertar_sobre_administrador(List<ClsParametros> lst)
{
    String mensaje = "";
    try {
        if (mensaje == "" && lst != null) {
            SqlConnection conexion = abrir_conexion();

            String cadena = "INSERT INTO Administrador (nombre_persona,apellido,cedula,usuario,psw) "
                          + "VALUES(@nombre_persona,@apellido,@cedula,@usuario,@psw)";

            SqlCommand comando = new SqlCommand(cadena, conexion);

            comando.Parameters.AddWithValue(parameterName: "@nombre_persona", lst[0].Nombres);
            comando.Parameters.AddWithValue(parameterName: "@apellido", lst[0].Apellidos);
            comando.Parameters.AddWithValue(parameterName: "@cedula", lst[0].Cedula);
            comando.Parameters.AddWithValue(parameterName: "@usuario", lst[0].Usuario);
            comando.Parameters.AddWithValue(parameterName: "@psw", lst[0].Psw);

            comando.ExecuteNonQuery();

        }
    } catch (Exception ex) {
        cerrar_conexion(); //Por si entra aqui con conexion abierta
        throw ex;
    }
    cerrar_conexion();
    return mensaje;
}
```



Complejidad ciclomática

registrar

Forma 1	Forma 2	Forma 3
M = E-N+2P	M = NP+1	M=R
M=2-3+2(1)=1	M=0+1=1	M=1

db_insertar_sobre_administrador

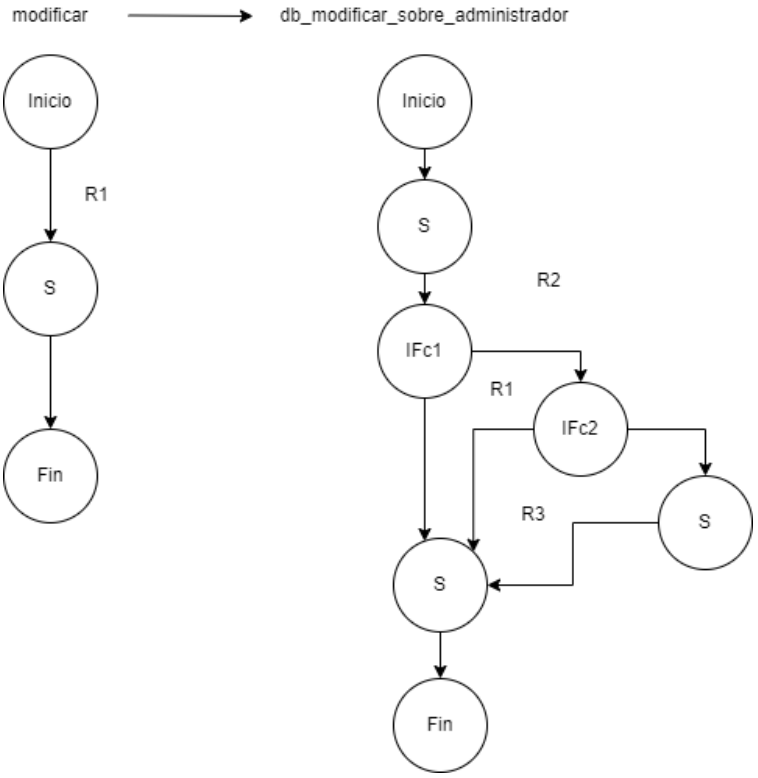
Forma 1	Forma 2	Forma 3
M = E-N+2P	M = NP+1	M=R
M= 8-7+2(1)=3	M= 2+1=3	M=3

CU25: ClsAdministrado - editar

Grafo

```
//Modificar administrador
3 referencias
public override String Modificar()
{
    string msj = "";

    //Lista genérica de parámetros
    List<ClsParametros> lst = new List<ClsParametros>();

    try
    {
        //Pasar los parámetros hacia la capa de acceso a datos
        ClsParametros cp = new ClsParametros();
        cp.setAdministrador(Id_persona, Nombres, Apellidos, Cedula, Usuario, Psw);
        lst.Add(cp);
        M.db_modificar_sobre_administrador(lst);
        msj = "Insertado correctamente";
    }
    catch (Exception ex)
    {
        msj = "Error al insertar los datos";
        return msj;
        throw ex;
    }

    return msj;
}
```

```
1 referencia
public String db_modificar_sobre_administrador(List<ClsParametros> lst)
{
    String mensaje = "";
    try {
        if (mensaje == "" && lst != null) {
            SqlConnection conexion = abrir_conexion();

            String cadena = "UPDATE Administrador SET nombre_persona = @Nombres, Apellidos = @apellido,"
                          + " cedula = @cedula, usuario = @usuario, psw = @psw WHERE id_persona = "
                          + "@id_persona";

            SqlCommand comando = new SqlCommand(cadena, conexion);

            comando.Parameters.AddWithValue(parameterName: "@id_persona", lst[0].Id_persona);
            comando.Parameters.AddWithValue(parameterName: "@nombre_persona", lst[0].Nombres);
            comando.Parameters.AddWithValue(parameterName: "@apellido", lst[0].Apellidos);
            comando.Parameters.AddWithValue(parameterName: "@cedula", lst[0].Cedula);
            comando.Parameters.AddWithValue(parameterName: "@usuario", lst[0].Usuario);
            comando.Parameters.AddWithValue(parameterName: "@psw", lst[0].Psw);

            comando.ExecuteNonQuery();

        }
    } catch (Exception ex) {
        cerrar_conexion(); //Por si entra aqui con conexion abierta
        throw ex;
    }
    cerrar_conexion();
    return mensaje;
}
```



Complejidad ciclomática

modificar

Forma 1	**Forma 2**	**Forma 3**
M = E-N+2P	M = NP+1	M=R
M=2-3+2(1)=1	M=0+1=1	M=1

db_modificar_sobre_administrador

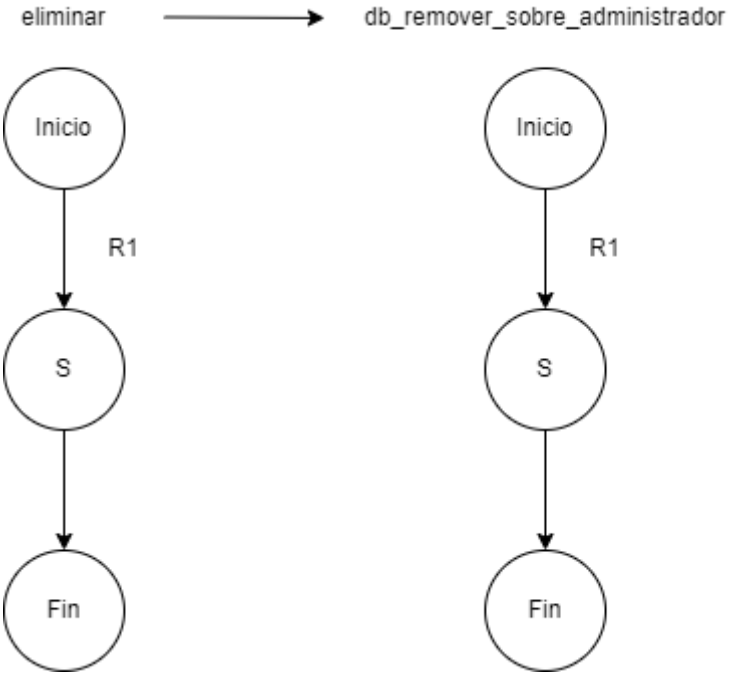
Forma 1	**Forma 2**	**Forma 3**
M = E-N+2P	M = NP+1	M=R
M= 8-7+2(1)=3	M= 2+1=3	M=3

CU26: ClsAdministrado - eliminar

Grafo

```
4 referencias
public override String Eliminar(int Id_persona)
{
    return M.db_remover_sobre_administrador(Id_persona);
}
```

```
public String db_remover_sobre_administrador(int Id_persona)
{
    String mensaje = "";
    try {
        SqlConnection conexion = abrir_conexion();
        //Alerta cambiar el idCambiar por el de la base de datos
        String cadena = "delete from Administrador where Id_persona = @Id_persona";
        SqlCommand comando = new SqlCommand(cadena, conexion);
        comando.Parameters.AddWithValue(parameterName: "@Id_persona", Id_persona);
        comando.ExecuteNonQuery();
        mensaje = "Se realizo la consulta exitosamente de eliminacion de registro";
    } catch (Exception ex) {
        cerrar_conexion(); //Por si entra aqui con conexion abierta
        throw ex;
    }
    cerrar_conexion();
    return mensaje;
}
```



Complejidad ciclomática

eliminar

Forma 1	Forma 2	Forma 3
M = E-N+2P	M = NP+1	M=R
M=2-3+2(1)=1	M=0+1=1	M=1

db_remover_sobre_administrador

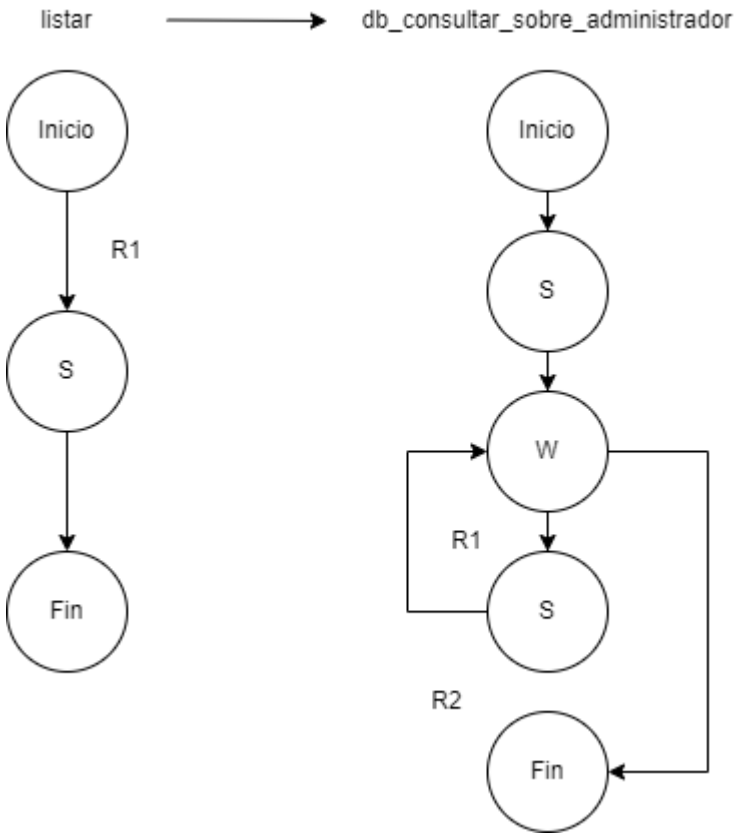
Forma 1	Forma 2	Forma 3
M = E-N+2P	M = NP+1	M=R
M=2-3+2(1)=1	M= 0+1=1	M=1

CU27: ClsAdministrado - consultar

Grafo

```
//Lista administrador
8 referencias
public override Tuple<List<Object>, SqlDataAdapter> Listar()
{
    return M.db_consultar_sobre_administrador( sentenciaSQL: "SELECT [id_persona],[nombre_persona],[apellido],"
                                             + "[cedula],[usuario],[psw] FROM [dbo].[Administrador]"); // Tuple<L
}
```

```
2 referencias
public Tuple<List<Object>, SqlDataAdapter> db_consultar_sobre_administrador(string sentenciaSQL)
{
    //lreturn.Clear();
    SqlDataReader dr = null;
    List<Object> lreturn = new List<Object>();
    SqlDataAdapter adaptador;
    try {
        SqlConnection conexion = abrir_conexion();
        String cadena = sentenciaSQL;
        SqlCommand comando = new SqlCommand(cadena, conexion);
        //comando.CommandText = "";//Linea opcional para rellenar los datos
        dr = comando.ExecuteReader();
        adaptador = new SqlDataAdapter(cadena, conexion);

        while (dr.Read()) {
            var tmp {id_persona,nombre_persona,} = new {
                id_persona = Convert.ToInt32(dr["id_persona"]),
                nombre_persona = dr["nombre_persona"].ToString(),
                apellido = dr["apellido"].ToString(),
                cedula = dr["cedula"].ToString(),
                usuario = dr["usuario"].ToString(),
                psw = dr["psw"].ToString()
            };
            lreturn.Add(tmp);
        }
    } catch (Exception ex) {
        cerrar_conexion(); //Por si entra aqui con conexion abierta
        throw ex;
    }
    cerrar_conexion();
    return Tuple.Create(lreturn, adaptador);//return ;
}
```



Complejidad ciclomática

listar

Forma 1	**Forma 2**	**Forma 3**
M = E-N+2P	M = NP+1	M=R
M=2-3+2(1)=1	M=0+1=1	M=1

db_conultar_sobre_administrador

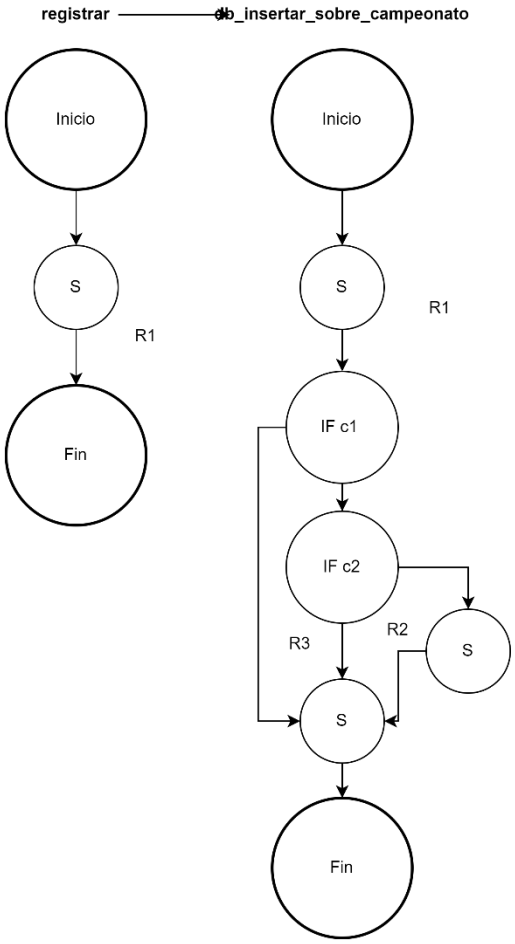
Forma 1	**Forma 2**	**Forma 3**
M = E-N+2P	M = NP+1	M=R
M=5-5+2(1)=2	M=1+1=2	M=2

CU28: ClsCampeonato - agregar

Grafo

```
public virtual String registrar() {
    string msj = "";

    //Lista genérica de parámetros
    List<ClsParametros> lst = new List<ClsParametros>();

    try {
        //Pasar los parámetros hacia la capa de acceso a datos
        ClsParametros cp = new ClsParametros();
        lst.Add(cp);
        M.db_insertar_sobre_campeonato(lst);
        msj = "Insertado correctamente";
    } catch (Exception ex) {
        msj = "Error al insertar los datos";
        return msj;
        throw ex;
    }

    return msj;
}
```

```
public String db_insertar_sobre_campeonato(List<ClsParametros> lst) {
    String mensaje = "";
    try {
        if (mensaje == "" && lst != null) {
            SqlConnection conexion = abrir_conexion();

            String cadena = "INSERT INTO Campeonato (nombre_campeonado,fechas) VALUES
              (@nombre_campeonado,@fechas)";

            SqlCommand comando = new SqlCommand(cadena, conexion);

            comando.Parameters.AddWithValue("@nombre_campeonado", lst
              [0].Nombre_campeonado);

            comando.ExecuteNonQuery();

        }
    } catch (Exception ex) {
        cerrar_conexion(); //Por si entra aqui con conexion abierta
        throw ex;
    }
    cerrar_conexion();
    return mensaje;
}
```



Complejidad ciclomática

Registrar

Forma 1	Forma 2	Forma 3
M = E - N +2P	M = R	M = NP+1
M = 2-3+2(1) = 1	M = 1	M = 0+1=1

db_insertar_sobre_campeonato

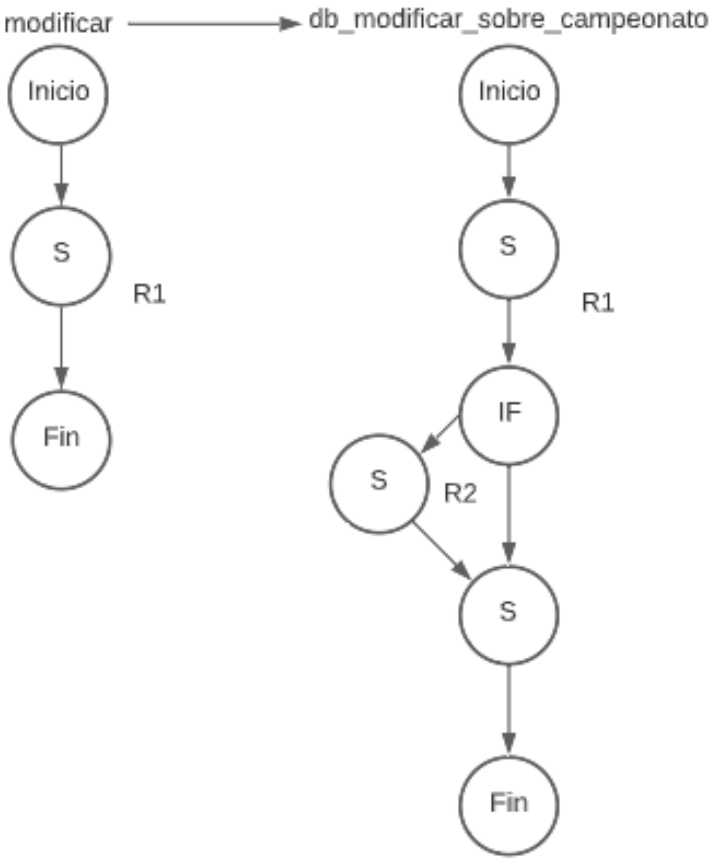
Forma 1	Forma 2	Forma 3
M = E - N + 2P	M = R	M = NP+1
M = 8 - 7 + 2(1) = 3	M = 3	M = 1+1=3

CU29: ClsCampeonato - editar

Grafo

```
public virtual String modificar() {
    string msj = "";

    //Lista genérica de parámetros
    List<ClsParametros> lst = new List<ClsParametros>();

    try {
        //Pasar los parámetros hacia la capa de acceso a datos
        ClsParametros cp = new ClsParametros();
        lst.Add(cp);
        M.db_modificar_sobre_campeonato(lst);
        msj = "Insertado correctamente";
    } catch (Exception ex) {
        msj = "Error al insertar los datos";
        return msj;
        throw ex;
    }

    return msj;
}
```

```
public String db_modificar_sobre_campeonato(List<ClsParametros> lst) {
    String mensaje = "";
    try {
        if (mensaje == "" && lst != null) {
            SqlConnection conexion = abrir_conexion();

            String cadena = "UPDATE Campeonato SET nombre_campeonado =
              @nombre_campeonadofechas_partidos_equipoa_partido_jugadores_nombre_persona =
              @fechas_partidos_equipoa_partido_jugadores_nombre_persona, fechas_partidos_equipoa_partido_jugadores_apellido =
              @fechas_partidos_equipoa_partido_jugadores_apellido, fechas_partidos_equipoa_partido_jugadores_cedula =
              @fechas_partidos_equipoa_partido_jugadores_cedula, fechas_partidos_equipoa_partido_jugadores_numero =
              @fechas_partidos_equipoa_partido_jugadores_numerofechas_partidos_equipob_partido_jugadores_nombre_persona =
              @fechas_partidos_equipob_partido_jugadores_nombre_persona, fechas_partidos_equipob_partido_jugadores_apellido =
              @fechas_partidos_equipob_partido_jugadores_apellido, fechas_partidos_equipob_partido_jugadores_cedula =
              @fechas_partidos_equipob_partido_jugadores_cedula, fechas_partidos_equipob_partido_jugadores_numero =
              @fechas_partidos_equipob_partido_jugadores_numerofechas_partidos_marcador_partido_goleaequipoa =
              @fechas_partidos_marcador_partido_goleaequipoa, fechas_partidos_marcador_partido_golesequipob =
              @fechas_partidos_marcador_partido_golesequipob, fechas_partidos_arbitroprincipal_usuario =
              @fechas_partidos_arbitroprincipal_usuario, fechas_partidos_arbitroprincipal_psw =
              @fechas_partidos_arbitroprincipal_pswfechas_partidos_arbitroprincipal_nombre_persona =
              @fechas_partidos_arbitroprincipal_nombre_persona, fechas_partidos_arbitroprincipal_apellido =
              @fechas_partidos_arbitroprincipal_apellido, fechas_partidos_arbitroprincipal_cedula =
              @fechas_partidos_arbitroprincipal_cedula, fechas_partidos_arbitroprincipal_licencia =
              @fechas_partidos_arbitroprincipal_licencia, fechas_numero_fecha = @fechas_numero_fecha, fechas_fechainicio =
              @fechas_fechainicio, fechas_fechafin = @fechas_fechafin WHERE id_campeonato = @id_campeonato";

            SqlCommand comando = new SqlCommand(cadena, conexion);

            comando.ExecuteNonQuery();

        }
    } catch (Exception ex) {
        cerrar_conexion(); //Por si entra aqui con conexion abierta
        throw ex;
    }
    cerrar_conexion();
    return mensaje;
}
```



Complejidad ciclomática

Modificar

Forma 1	**Forma 2**	**Forma 3**
M = E - N +2P	M = R	M = NP+1
M = 2-3+2(1) = 1	M = 1	M = 0+1=1

db_modificar_sobre_campeonato

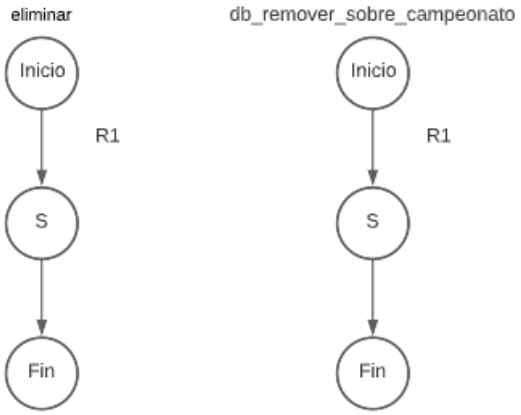
Forma 1	**Forma 2**	**Forma 3**
M = E - N + 2P	M = R	M = NP+1
M = 6 - 6 + 2(1) = 2	M = 2	M = 1+1=2

CU30: ClsCampeonato - eliminar

Grafo

```
public virtual String eliminar(int Id_campeonato)
{
    return M.db_remover_sobre_campeonato(Id_campeonato);
}
```

```
public String db_remover_sobre_campeonato(int Id_campeonato) {
    String mensaje = "";
    try {
        SqlConnection conexion = abrir_conexion();
        //Alerta cambiar el idCambiar por el de la base de datos
        String cadena = "delete from Campeonato where Id_campeonato =
          @Id_campeonato";
        SqlCommand comando = new SqlCommand(cadena, conexion);
        comando.Parameters.AddWithValue("@Id_campeonato", Id_campeonato);
        comando.ExecuteNonQuery();
        mensaje = "Se realizo la consulta exitosamente de eliminacion de
          registro";
    } catch (Exception ex) {
        cerrar_conexion(); //Por si entra aqui con conexion abierta
        throw ex;
    }
    cerrar_conexion();
    return mensaje;
}
```



Complejidad ciclomática

Eliminar

Forma 1	Forma 2	Forma 3
M = E - N +2P	M = R	M = NP+1
M = 2-3+2(1) = 1	M = 1	M = 0+1=1

db_remover_sobre_campeonato

Forma 1	Forma 2	Forma 3
M = E - N +2P	M = R	M = NP+1
M = 2-3+2(1) = 1	M = 1	M = 0+1=1

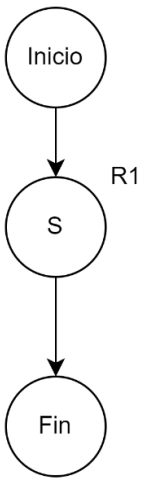
CU31: ClsCampeonato - consultar

Grafo

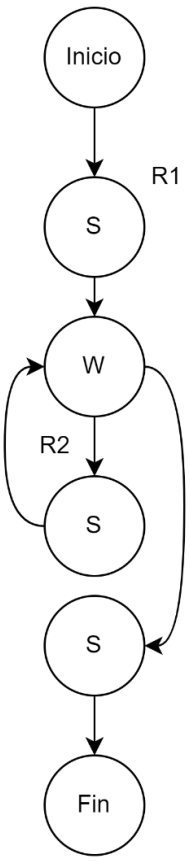
```
public virtual Tuple<List<Object>, SqlDataAdapter> listar() {
    return M.db_consultar_sobre_campeonato("SELECT [id_campeonato],
      [nombre_campeonado],[fechas] FROM [dbo].[Campeonato]");
}
```

```
public Tuple<List<Object>, SqlDataAdapter> db_consultar_sobre_campeonato(string
  sentenciaSQL) {
    //lreturn.Clear();
    SqlDataReader dr = null;
    List<Object> lreturn = new List<Object>();
    SqlDataAdapter adaptador;
    try {
        SqlConnection conexion = abrir_conexion();
        String cadena = sentenciaSQL;
        SqlCommand comando = new SqlCommand(cadena, conexion);
        //comando.CommandText = "";//Linea opcional para rellenar los datos
        dr = comando.ExecuteReader();
        adaptador = new SqlDataAdapter(cadena, conexion);

        while (dr.Read()) {
            var tmp = new {
                id_campeonato = Convert.ToInt32(dr["id_campeonato"]),
                nombre_campeonado = dr["nombre_campeonado"].ToString(),
                fechas = Convert.ToInt32(dr["fechas"])
            };
            lreturn.Add(tmp);
        }
    } catch (Exception ex) {
        cerrar_conexion(); //Por si entra aqui con conexion abierta
        throw ex;
    }
    cerrar_conexion();
    return Tuple.Create(lreturn, adaptador);
    //return lreturn;
}
```

Listar



db_consultar_sobre_campeonato



Complejidad ciclomática

Listar

Forma 1	**Forma 2**	**Forma 3**
M = E - N +2P	M = R	M = NP+1
M = 2-3+2(1) = 1	M = 1	M = 0+1=1

db_consultar_sobre_campeonato

Forma 1	**Forma 2**	**Forma 3**
M = E - N +2P	M = R	M = NP+1
M = 6 – 6 + 2(1) = 2	M = 2	M = 1+1=2

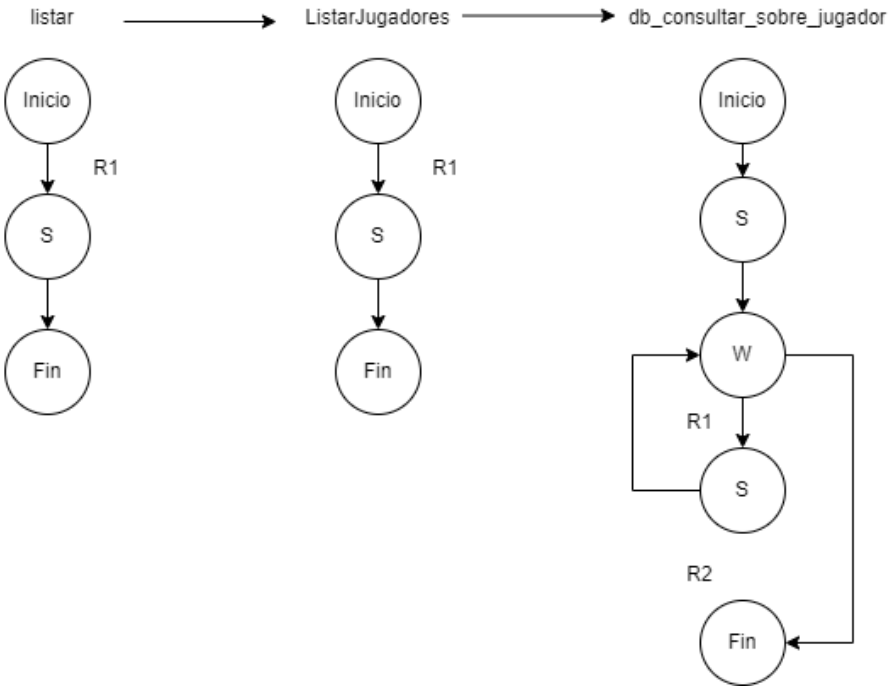
CU33: ClsJugadores - consultar

Grafo

```
3 referencias
public override Tuple<List<Object>, SqlDataAdapter> Listar() {
    return M.ListarJugadores();
}
```

```
1 referencia
public Tuple<List<Object>, SqlDataAdapter> ListarJugadores() {
    return db_consultar_sobre_jugador(sentenciaSQL: "SELECT [id_persona],[nombres],[apellidos],[cedula],"
                                      + "[fechanacimiento],[telefono],[nacionalidad],[numero] "
                                      + "FROM [dbo].[Jugador]");
}
```

```
private Tuple<List<Object>, SqlDataAdapter> db_consultar_sobre_jugador(string sentenciaSQL)
{
    SqlDataReader dr = null;
    List<Object> lreturn = new List<Object>();
    SqlDataAdapter adaptador;
    try {
        SqlConnection conexion = abrir_conexion();
        String cadena = sentenciaSQL;
        SqlCommand comando = new SqlCommand(cadena, conexion);
        //comando.CommandText = "";//Linea opcional para rellenar los datos
        dr = comando.ExecuteReader();
        adaptador = new SqlDataAdapter(cadena, conexion);

        while (dr.Read()) {
            var tmp {id_persona,nombres,...} = new {
                id_persona = Convert.ToInt32(dr["id_persona"]),
                nombres = dr["nombres"].ToString(),
                apellidos = dr["apellidos"].ToString(),
                cedula = dr["cedula"].ToString(),
                fechanacimiento = DateTime.Parse(dr["fechanacimiento"].ToString()),
                telefono = dr["telefono"].ToString(),
                nacionalidad = dr["nacionalidad"].ToString(),
                numero = Convert.ToUInt16(dr["numero"])
            };
            lreturn.Add(tmp);
        }
    } catch (Exception ex) {
        cerrar_conexion(); //Por si entra aqui con conexion abierta
        throw ex;
    }
    cerrar_conexion();
    return Tuple.Create(lreturn, adaptador);//return lreturn;
}
```



Complejidad ciclomática

listar

Forma 1	Forma 2	Forma 3
M = E-N+2P	M = NP+1	M=R
M=2-3+2(1)=1	M=0+1=1	M=1

listarJugadores

Forma 1	Forma 2	Forma 3
M = E-N+2P	M = NP+1	M=R
M=2-3+2(1)=1	M=0+1=1	M=1

db_consultar_sobre_jugador

Forma 1	Forma 2	Forma 3
M = E-N+2P	M = NP+1	M=R
M=5-5+2(1)=2	M=1+1=2	M=2

Enlaces

Github

https://github.com/cbr970801/Proyecto-VVS.git

YouTube

https://youtu.be/oNmZ12shDYk