

Secciones Críticas en la Aplicación de Gestión de Biblioteca

Niurca Bone Arroyo, Khriz Coronel Gómez, Johan Carvajal Loor, Bryan Figueroa Morales

Introducción

Este documento presenta las secciones críticas identificadas en el código del Sistema de gestión de biblioteca desarrollado en C# en el segundo semestre de la carrera cada sección está marcada con regiones #region y #endregion. Cada sección crítica se acompaña de una explicación detallada sobre por qué es considerada crítica, destacando los riesgos asociados en un entorno concurrente donde múltiples hilos o usuarios podrían acceder a la base de datos simultáneamente.

- 1 **Desarrolle en C#.Net 2022 un programa que acceda a una base de datos de SQL Server 2019 y permita grabar, eliminar, buscar y editar datos en una tabla.**



2 Secciones Críticas y Explicaciones

2.1 Actualizar (CsConexionDataBase)

```
31 referencias
public void Actualizar(string consulta)
{
    #region SeccionCritica_Actualizar
    conexion.Open();
    SqlCommand comando = new SqlCommand(consulta, conexion);
    comando.ExecuteNonQuery();
    conexion.Close();
    #endregion SeccionCritica_Actualizar
}
```

Explicación: Esta sección realiza una actualización en la base de datos (por ejemplo, INSERT, UPDATE o DELETE). Es crítica porque, si varios hilos o usuarios ejecutan esta

operación simultáneamente, podrían surgir condiciones de carrera, donde una actualización sobrescribe a otra, o bloqueos de recursos que afecten la integridad de los datos.

```
1 referencia
public void GuardarEditorial(string codigo, string editorial, string estado, string fecha)
{
    try
    {
        string consulta = $"INSERT INTO EDITORIAL(IdEditorial, Editorial, Estado, FechaCre
        dataBase.Actualizar(consulta);
        MessageBox.Show("Editorial agregada con éxito.", "Éxito", MessageBoxButtons.OK, Me
    }
    catch (ArgumentException ex)
    {
        MessageBox.Show(ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}

1 referencia
public void EditarEditorial(string Editorial, string estado, string id)
{
    dataBase.Actualizar("Update EDITORIAL set Editorial = '" + Editorial + "', Estado = '"
```

2.2 Extraer (CsConexionDataBase)

```
17 referencias
public string Extraer(string consulta, string columna)
{
    #region SeccionCritica_Extraer
    string resultado = "";
    conexion.Open();
    SqlCommand comando = new SqlCommand(consulta, conexion);
    SqlDataReader leer = comando.ExecuteReader();
    while (leer.Read()) { resultado = leer[columna].ToString(); }
    conexion.Close();
    return resultado;
    #endregion SeccionCritica_Extraer
}
```

Explicación: Esta sección extrae un valor único de la base de datos, como un IdUsuario o IdGenero. Es crítica porque, en un entorno concurrente, múltiples hilos podrían leer datos obsoletos si otro hilo actualiza la tabla durante la operación, o generar interferencias en la conexión debido al uso simultaneo de SqlDataReader. Esto afecta la precisión en consultas como las de recuperación de IDs.

```
1 referencia
public void ActualizarContraseña(string correo, string NuevaClave)
{
    string consulta = " select IdUsuario from USUARIO where Correo='" + correo + "'";
    idUsuario = Extraer(consulta, "IdUsuario");
    string consulta01 = "update CREDENCIAL set Contraseña='" + NuevaClave + "' where IdUsuario='" + idUsuario +
    Actualizar(consulta01);
    MessageBox.Show("🔑 Tu contraseña ha sido actualizada exitosamente. Puedes ahora acceder con tu nueva contr
```

2.3 ExtraerImagen (CsConexionDataBase)

```

4 referencias
public PictureBox ExtraerImagen(string consulta, string columna, PictureBox portada)
{
    #region SeccionCritica_ExtraerImagen
    conexion.Open();
    SqlCommand comando = new SqlCommand(consulta, conexion);
    SqlDataReader leer = comando.ExecuteReader();
    try
    {
        if (leer.Read())
        {
            MemoryStream memoria = new MemoryStream((byte[])leer[columna]);
            Bitmap bitmap = new Bitmap(memoria);
            portada.BackgroundImage = bitmap;
        }
    } catch { conexion.Close(); }
    conexion.Close();
    return portada;
    #endregion SeccionCritica_ExtraerImagen
}

```

Explicación: Esta sección extrae una imagen (como la columna Foto de LIBRO). Es crítica porque, si otro hilo modifica la imagen durante la lectura, el SqlDataReader podría fallar o devolver datos parciales, resultando en excepciones o imágenes corruptas. Además, la lectura de datos binarios grandes puede bloquear la conexión para otros hilos.

```

1 referencia
public void MostrarPortadaLibro(frmAgregarOEditarLibro formulario)
{
    string consulta = $"Select * from LIBRO where IdLibro = '{codigo}'";
    formulario.ImgLibro = dataBase.ExtraerImagen(consulta, "Foto", formulario.ImgLibro);
}

```

2.4 LlenarLista (CsConexionDataBase)

```

2 referencias
public ComboBox LlenarLista(ComboBox lista, string consulta, string columna)
{
    #region SeccionCritica_LlenarLista
    conexion.Open();
    SqlCommand comando = new SqlCommand(consulta, conexion);
    SqlDataReader leer = comando.ExecuteReader();
    while (leer.Read()) { lista.Items.Add(leer[columna].ToString()); }
    conexion.Close();
    return lista;
    #endregion SeccionCritica_LlenarLista
}

```

Explicación: Esta sección llena un ComboBox con datos de la base de datos, como géneros o editoriales. Es crítica porque, si los datos cambian durante la lectura, el ComboBox podría mostrar información obsoleta o incompleta. Además, lecturas concurrentes masivas pueden afectar el rendimiento de la base de datos.

```

public void MostrarListas(frmAgregarOEditarLibro formulario)
{
    formulario.cbCategoria = dataBase.LlenarLista(formulario.cbCategoria, "Select Genero from GENERO where Estado = 1");
    formulario.cbEditorial = dataBase.LlenarLista(formulario.cbEditorial, "Select Editorial from EDITORIAL where Estado = 1");
}

```

2.5 GuardarImagen (CsConexionDataBase)

```

3 referencias
public void GuardarImagen(PictureBox Imagen, string consulta)
{
    MemoryStream espacio = new MemoryStream();
    Imagen.Image.Save(espacio, ImageFormat.Png);
    byte[] Convertir = espacio.ToArray();
    #region SeccionCritica_GuardarImagen
    conexion.Open();
    SqlCommand Comando = new SqlCommand(consulta, conexion);
    Comando.Parameters.AddWithValue("imagen", Convertir);
    Comando.ExecuteNonQuery();
    conexion.Close();
    #endregion SeccionCritica_GuardarImagen
}

```

Explicación: Esta sección guarda una imagen en la base de datos. Es crítica porque, si dos hilos intentan guardar imágenes en el mismo registro simultáneamente, una podría sobrescribir a la otra, o la operación de escritura podría bloquear la tabla, afectando a otros hilos.

```

private void btnCambiarLogo_Click(object sender, EventArgs e)
{
    OpenFileDialog Imagen = new OpenFileDialog();
    Imagen.Filter = "archivos de imagen (*png)|*png;";
    if (Imagen.ShowDialog() == DialogResult.OK)
    {
        ptbxLogoGeneral.BackgroundImage = null;
        ptbxLogoGeneral.Image = Image.FromFile(Imagen.FileName);
        conexion.GuardarImagen(ptbxLogoGeneral, "Insert into CONFIGURACION(Imagen) Values(@imagen)");
    }
}

```

2.6 Extraer2Parametros(CsConexionDataBase)

```

1 referencia
public List<string> Extraer2Parametros(List<string> datosLibro, string consulta)
{
    #region SeccionCritica_Lista
    conexion.Open();
    SqlCommand comando = new SqlCommand(consulta, conexion);
    try
    {
        SqlDataReader read = comando.ExecuteReader();
        while (read.Read())
        {
            datosLibro.Add(read[0].ToString().Trim());
            datosLibro.Add(read[1].ToString().Trim());
        }
        conexion.Close();
    }
    catch (Exception ex)
    {
        MessageBox.Show("Error " + ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    #endregion SeccionCritica_Lista
    return datosLibro;
}

```

Explicación: Esta sección llena una lista con datos de una columna, como lectores. Es crítica porque cambios en la tabla durante la lectura podrían resultar en una lista incompleta o desactualizada, afectando la funcionalidad de la aplicación.

2.7 VerificarCorreoSQL (CsConexionDataBase)

```

5 referencias
public bool VerificarCorreoSQL(string correo, string consulta)
{
    bool ExisteCorreo = false;
    #region SeccionCritica_VerificarCorreoSQL
    conexion.Open();
    SqlCommand comandos = new SqlCommand(consulta, conexion);
    int contador = (int)comandos.ExecuteScalar();
    ExisteCorreo = contador > 0;
    conexion.Close();
    #endregion SeccionCritica_VerificarCorreoSQL
    return ExisteCorreo;
}

```

Explicación: Esta sección verifica la existencia de un correo. Es crítica porque un cambio en la tabla (agregar o eliminar un correo) durante la verificación podría llevar a una decisión incorrecta basada en un conteo obsoleto.

2.8 VerificacionLogin(CsLogin)

```

1 referencia
public bool VerificacionLogin(string clave)
{
    if (Usuario != string.Empty && contraseña != string.Empty)
    {
        #region SeccionCritica_VerificacionLogin
        conexion.Open();
        string query = "select * from CREDENCIAL where Usuario='" + Usuario + "' and Contraseña='" + contraseña + "'";
        SqlCommand comandos = new SqlCommand(query, conexion);
        SqlDataReader lector = comandos.ExecuteReader();
        if (lector.Read())
        {
            IdUsuario = lector["IdUsuario"].ToString();
            conexion.Close();
            return true;
        }
        #endregion SeccionCritica_VerificacionLogin
    }
    else { mensajes.MensajeCamposIncompletos(); }
    conexion.Close();
    return false;
}

```

Explicación: Esta sección verifica credenciales de inicio de sesión. Es crítica porque, si las credenciales cambian durante la verificación (por ejemplo, otra actualización de contraseña), podría denegar o autorizar acceso incorrectamente.

2.9 ActualizarContraseña(CsLogin)

```

1 referencia
public void ActualizarContraseña(string correo, string NuevaClave)
{
    string consulta = "select IdUsuario from USUARIO where Correo='" + correo + "'";
    idUsuario = Extraer(consulta, "IdUsuario");
    #region SeccionCritica_ActualizarContraseña
    string consulta01 = "update CREDENCIAL set Contraseña='" + NuevaClave + "' where IdUsuario='" + idUsuario + "'";
    Actualizar(consulta01);
    #endregion SeccionCritica_ActualizarContraseña
    MessageBox.Show("👍 Tu contraseña ha sido actualizada exitosamente. Puedes ahora acceder con tu nueva contraseña.");
}

```

Explicación: Esta sección actualiza una contraseña. Es crítica porque, si dos hilos intentan actualizar la misma contraseña simultáneamente, una podría sobrescribir a la otra, resultando en la pérdida de la actualización más reciente.

2.10 BusquedaPorCaracter(CsGestionPrestamos)

```
public void BusquedaPorCaracter(DataGridView dgvPrestamos, string busqueda)
{
    try
    {
        #region SeccionCritica_BusquedaPorCaracter
        string consulta = @"select P.IdPrestamo,P.IdLector, L.Nombres, LI.Titulo, P.FechaDevolucion, P.FechaConfir
                             from PRESTAMO P
                             join LECTOR L on P.IdLector = L.IdLector
                             join LIBRO LI on P.IdLibro = LI.IdLibro
                             where P.IdPrestamo like @busqueda or
                                   L.IdLector like @busqueda or
                                   L.Nombres like @busqueda or
                                   LI.Titulo like @busqueda";
        consulta = consulta.Replace("@busqueda", "%" + busqueda + "%");
        new csLlenarDataGridView().Mostrar(dgvPrestamos, consulta, 3);
        #endregion SeccionCritica_BusquedaPorCaracter
    }
    catch (Exception ex)
    {
        MessageBox.Show("Error " + ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

Explicación: Esta sección busca prestamos en la base de datos. Es crítica porque, aunque es una operación de lectura, un uso intensivo concurrente podría afectar el rendimiento o devolver datos obsoletos si los préstamos cambian durante la consulta.

3 Incluya el código de la aplicación en un documento de Word u otra aplicación de su preferencia.

[LINK DE GITHUB](#)

4 Incluya el código TRANSACT-SQL utilizado en el archivo documento.

4.1 Clase csConexionDataBase

Estas consultas son utilizadas en los métodos de la clase csConexionDataBase para realizar operaciones como actualizaciones, extracciones, conteos, y llenado de listas.

1. Consulta para actualizaciones (usada en Actualizar):

```
UPDATE [Tabla] SET [Columna] = @Valor WHERE [Condición] = @Condicion
```

La consulta específica depende del parámetro consulta pasado al método, por ejemplo, update CREDENCIAL set Contraseña=@NuevaClave where IdUsuario=@IdUsuario.

2. Consulta para extracción de un valor (usada en Extraer):

```
SELECT @columna FROM [Tabla] WHERE [Condición] = @Valor
```

Ejemplo específico: SELECT IdUsuario FROM USUARIO WHERE Correo=@Correo.

3. Consulta para extracción de imagen (usada en ExtraerImagen):

```
SELECT @columna FROM LIBRO WHERE IdLibro = @IdLibro
```

La columna @columna suele ser Foto, que contiene datos binarios.

4. Consulta para llenar listas (usada en LLenarLista):

```
SELECT @columna FROM [Tabla] WHERE Estado = 1
```

Ejemplos específicos: `SELECT Genero FROM GENERO WHERE Estado = 1, SELECT Editorial FROM EDITORIAL WHERE Estado = 1.`

5. Consulta para contar registros (usada en Contar):

`SELECT COUNT (*) FROM [Tabla] WHERE [Condición] = @Valor`

Usada en VerificarCorreoSQL con una consulta como `SELECT COUNT (*) FROM USUARIO WHERE Correo=@Correo.`

6. Consulta para guardar imagen (usada en GuardarImagen):

`UPDATE LIBRO SET Foto = @imagen WHERE IdLibro = @IdLibro`

7. Consulta para llenar una lista (usada en Lista):

`SELECT [Columna] FROM [Tabla] WHERE [Condición] = @Valor`

Ejemplo: `SELECT Nombres FROM LECTOR WHERE Estado = 1.`

8. Consulta para extraer dos parámetros (usada en Extraer2Parametros):

`SELECT [Columna1], [Columna2] FROM [Tabla] WHERE [Condición] = @Valor`

Ejemplo: `SELECT IdLibro, Titulo FROM LIBRO WHERE Estado = 1`

9. Consulta para verificar correo (usada en VerificarCorreoSQL):

`SELECT COUNT (*) FROM USUARIO WHERE Correo = @Correo`

4.2 Clase csLogin

Estas consultas están relacionadas con la autenticación y gestión de contraseñas en las tablas CREDENCIAL y USUARIO.

1. Consulta para verificación de login (usada en VerificacionLogin):

`SELECT * FROM CREDENCIAL WHERE Usuario = @Usuario AND Contraseña = @Clave`

2. Consulta para obtener IdUsuario (usada en ActualizarContraseña):

`SELECT IdUsuario FROM USUARIO WHERE Correo = @Correo`

3. Consulta para actualizar contraseña (usada en ActualizarContraseña):

`UPDATE CREDENCIAL SET Contraseña = @NuevaClave WHERE IdUsuario = @IdUsuario`

4.3 Clase csGestionPrestamos

1. Consulta para búsqueda por carácter (usada en BusquedaPorCaracter):

```
SELECT P.IdPrestamo, P.IdLector, L.Nombres, LI.Titulo,
P.FechaDevolucion, P.FechaConfirmacionDevolucion, P.Estado
FROM PRESTAMO P
JOIN LECTOR L ON P.IdLector = L.IdLector
JOIN LIBRO LI ON P.IdLibro = LI.IdLibro
WHERE P.IdPrestamo LIKE @Busqueda OR
      L.IdLector LIKE @Busqueda OR
      L.Nombres LIKE @Busqueda OR
      LI.Titulo LIKE @Busqueda
```


@Busqueda se reemplaza dinámicamente con '%' + busqueda + '%'.

2. Consulta para extraer estado (usada en ExtraerEstado):

```
SELECT @columna FROM PRESTAMO WHERE IdPrestamo = @IdPrestamo
```

@columna es un parámetro dinámico, por ejemplo, Estado.


4. Consulta base para filtro de préstamos (usada en GenerarConsultaFiltro):

```
SELECT P.IdPrestamo, P.IdLector, L.Nombres, LI.Titulo,  
P.FechaDevolucion, P.FechaConfirmacionDevolucion, P.Estado  
FROM PRESTAMO P  
JOIN LECTOR L ON P.IdLector = L.IdLector  
JOIN LIBRO LI ON P.IdLibro = LI.IdLibro
```

Se pueden agregar condiciones WHERE dinámicamente, como L.Nombres = @IdLector o P.Estado = @Estado.


5 Incluya capturas de pantalla de la tabla utilizada en SQL SERVER en el documento

5.1 Credencial

CREDENCIAL			
	Nombre de columna	Tipo de datos	Permitir valores ...
	IdCredencial	varchar(8)	<input type="checkbox"/>
	IdUsuario	varchar(8)	<input checked="" type="checkbox"/>
	Usuario	nchar(16)	<input checked="" type="checkbox"/>
	Contraseña	nchar(100)	<input checked="" type="checkbox"/>
			<input type="checkbox"/>

	IdCredencial	IdUsuario	Usuario	Contraseña
1	000001	000001	admin	pQ2Zygl/Go8=

5.2 Usuario

USUARIO			
	Nombre de columna	Tipo de datos	Permitir valores ...
	IdUsuario	varchar(8)	<input type="checkbox"/>
	Nombres	varchar(50)	<input checked="" type="checkbox"/>
	Apellidos	varchar(50)	<input checked="" type="checkbox"/>
	Correo	varchar(50)	<input checked="" type="checkbox"/>
	IdTipoPersona	varchar(8)	<input checked="" type="checkbox"/>
	Estado	bit	<input checked="" type="checkbox"/>
	FechaCreacion	varchar(10)	<input checked="" type="checkbox"/>
			<input type="checkbox"/>

	IdUsuario	Nombres	Apellidos	Correo	IdTipoPersona	Estado	FechaCreacion
1	000001	Acceso Default	Default Acceso	Default@gmail.com	000001	1	Mar 18 202

5.3 Libro

LIBRO			
	Nombre de columna	Tipo de datos	Permitir valores ...
	IdLibro	varchar(8)	<input type="checkbox"/>
	Titulo	varchar(100)	<input checked="" type="checkbox"/>
	IdGenero	varchar(8)	<input checked="" type="checkbox"/>
	IdEditorial	varchar(8)	<input checked="" type="checkbox"/>
	Ubicacion	varchar(50)	<input checked="" type="checkbox"/>
	Cantidad	int	<input checked="" type="checkbox"/>
	Estado	bit	<input checked="" type="checkbox"/>
	Foto	varbinary(MAX)	<input checked="" type="checkbox"/>
	FechaCreacion	varchar(10)	<input checked="" type="checkbox"/>
			<input type="checkbox"/>

	IdLibro	Titulo	IdGenero	IdEditorial	Ubicacion	Cantidad	Estado	Foto	FechaCreacion
1	000001	Libro	000001	000001	Pasillo	1	1	NULL	18-03-2025

5.4 Préstamo

PRESTAMO			
	Nombre de columna	Tipo de datos	Permitir valores ...
	IdPrestamo	varchar(8)	<input type="checkbox"/>
	IdLector	varchar(8)	<input checked="" type="checkbox"/>
	IdLibro	varchar(8)	<input checked="" type="checkbox"/>
	FechaDevolucion	varchar(10)	<input checked="" type="checkbox"/>
	FechaConfirmacionDevolucion	varchar(10)	<input checked="" type="checkbox"/>
	EstadoEntregado	varchar(100)	<input checked="" type="checkbox"/>
	EstadoRecibido	varchar(100)	<input checked="" type="checkbox"/>
	Estado	bit	<input checked="" type="checkbox"/>
	FechaCreacion	varchar(10)	<input checked="" type="checkbox"/>
	Aviso	bit	<input checked="" type="checkbox"/>
	AvisoPrestamo	bit	<input checked="" type="checkbox"/>
			<input type="checkbox"/>


	IdPrestamo	IdLector	IdLibro	FechaDevolucion	FechaConfirmacionDevolucion	EstadoEntregado	EstadoRecibido	Estado	FechaCreacion	Aviso	AvisoPrestamo
1	000001	000001	000001	17-06-2025	NULL	Buen estado	NULL	1	2025-06-15	0	0

5.5 Lector

LECTOR			
	Nombre de columna	Tipo de datos	Permitir valores ...
	IdLector	varchar(8)	<input type="checkbox"/>
	Nombres	varchar(50)	<input checked="" type="checkbox"/>
	Apellidos	varchar(50)	<input checked="" type="checkbox"/>
	Correo	varchar(50)	<input checked="" type="checkbox"/>
	Estado	bit	<input checked="" type="checkbox"/>
	FechaCreacion	varchar(10)	<input checked="" type="checkbox"/>


Resultados		Mensajes				
	IdLector	Nombres	Apellidos	Correo	Estado	FechaCreacion
1	000001	Niurca	Bone	arlethbone8@gmail.com	1	2025-06-15

5.6 Género

GENERO			
	Nombre de columna	Tipo de datos	Permitir valores ...
	IdGenero	varchar(8)	<input type="checkbox"/>
	Genero	varchar(50)	<input checked="" type="checkbox"/>
	Estado	bit	<input checked="" type="checkbox"/>
	FechaCreacion	varchar(10)	<input checked="" type="checkbox"/>

Resultados		Mensajes			
	IdGenero	Genero	Estado	FechaCreacion	
1	000001	Genero	1	18-03-2025	
2	000002	Fantasia	1	15-06-2025	
3	000003	Ficción	1	15-06-2025	

5.7 Editorial

EDITORIAL			
	Nombre de columna	Tipo de datos	Permitir valores ...
	IdEditorial	varchar(8)	<input type="checkbox"/>
	Editorial	varchar(50)	<input checked="" type="checkbox"/>
	Estado	bit	<input checked="" type="checkbox"/>
	FechaCreacion	varchar(10)	<input checked="" type="checkbox"/>

Resultados		Mensajes			
	IdEditorial	Editorial	Estado	FechaCreacion	
1	000001	Editorial	1	18-03-2025	
2	000002	Pearson	1	15-06-2025	
3	000003	McKlein	1	15-06-2025	