

UNIVERSIDAD AMERICANA
Facultad de Ingeniería y Arquitectura



Metodología y programación estructurada

Programación Visual en C#

Estudiante:

- Arelys Brigitte Obando Castillo
- Kimberly María Zapata Espinoza

Docente:

- Silvia Gigdalia Ticay López

Noviembre 14, 2024

Documentación del Proyecto de Registro Académico

1. Creación de la Interfaz de Usuario

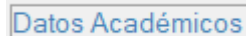
Etiquetas “Label”

Se añadieron etiquetas para guiar al usuario en el ingreso de información. A cada etiqueta se le asignó un nombre que representa su función:

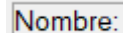
- “lblEncabezado”: Esta etiqueta se utiliza para mostrar el texto "Información del Estudiante".

A screenshot of a software interface showing a label with the text "Información del Estudiante" in a blue, sans-serif font. The label is enclosed in a light gray rectangular border.

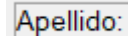
- “lblEncabezado2”: Esta etiqueta muestra el texto "Datos Académicos".

A screenshot of a software interface showing a label with the text "Datos Académicos" in a blue, sans-serif font. The label is enclosed in a light gray rectangular border.

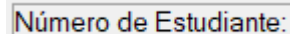
- “lblNombre”: Para indicar dónde se debe ingresar el nombre del estudiante.

A screenshot of a software interface showing a label with the text "Nombre:" in a blue, sans-serif font. The label is enclosed in a light gray rectangular border.

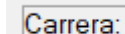
- “lblApellido”: Para el apellido del estudiante.

A screenshot of a software interface showing a label with the text "Apellido:" in a blue, sans-serif font. The label is enclosed in a light gray rectangular border.

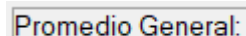
- “lblEstudianteID”: Para un identificador único del estudiante.

A screenshot of a software interface showing a label with the text "Número de Estudiante:" in a blue, sans-serif font. The label is enclosed in a light gray rectangular border.

- “lblCarrer””: Para seleccionar la carrera del estudiante en el ComboBox.

A screenshot of a software interface showing a label with the text "Carrera:" in a blue, sans-serif font. The label is enclosed in a light gray rectangular border.

- “lblPromedioGeneral”: Para el promedio acumulado del estudiante.

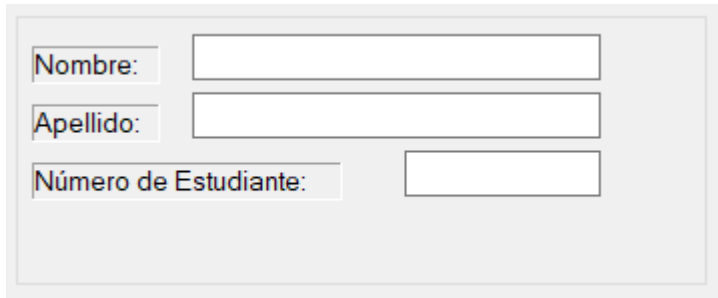
A screenshot of a software interface showing a label with the text "Promedio General:" in a blue, sans-serif font. The label is enclosed in a light gray rectangular border.

- “lblMateriasAprobadas”: Para la cantidad de materias aprobadas.

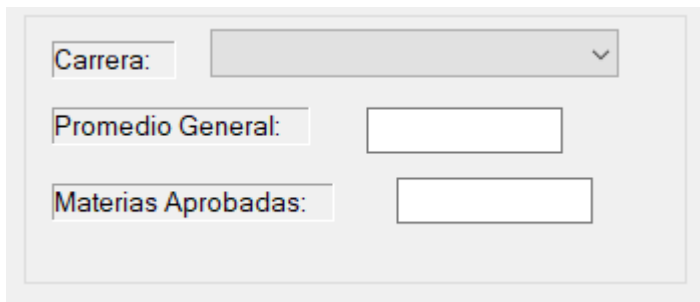
Materias Aprobadas:

Contenedor “GroupBox”

- “grpInformacionEstudiante”: agrupa todos los controles relacionados con la información personal del estudiante.



- “groupBoxDatosAcademicos”: agrupa los controles relacionados con los datos académicos del estudiante.



Cuadros de texto “TextBox”

Se añadieron campos de texto para permitir el ingreso de información específica del estudiante, con nombres descriptivos:

- “txtNombre”: Campo para el nombre del estudiante.



```
1 reference
private void txtNombre_KeyPress(object sender, KeyPressEventArgs e)
{
    if (!char.IsLetter(e.KeyChar) && e.KeyChar != (char)Keys.Back)
    {
        e.Handled = true;
        MessageBox.Show("Solo se permiten letras en este campo.", "Entrada inválida", MessageBoxButtons.OK, MessageBoxIcon.Warning);
    }
}
```

Se agregó la validación mediante el método **KeyPress** para el campo txtNombre para asegurarse de que solo se ingresen letras. En este código, se utiliza `char.IsLetter(e.KeyChar)` para verificar si el carácter presionado es una letra. Si no lo es, se bloquea la entrada usando `e.Handled = true`. Además, se permite que el usuario presione la tecla de retroceso (`(char)Keys.Back`) para borrar lo que ha escrito. Si el usuario intenta ingresar algo que no es una letra, se muestra un mensaje de advertencia con `MessageBox.Show` para indicar que solo se permiten letras en ese campo.

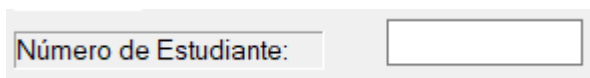
- "txtApellido": Campo para el apellido del estudiante.



```
1 reference
private void txtApellido_KeyPress(object sender, KeyPressEventArgs e)
{
    if (!char.IsLetter(e.KeyChar) && e.KeyChar != (char)Keys.Back)
    {
        e.Handled = true;
        MessageBox.Show("Solo se permiten letras en este campo.", "Entrada inválida", MessageBoxButtons.OK, MessageBoxIcon.Warning);
    }
}
```

El mismo tipo de validación se implementó en el campo txtApellido para garantizar que solo se ingresen letras. Al igual que en el txtNombre, el método `char.IsLetter(e.KeyChar)` verifica si el carácter presionado es una letra. Si no lo es, se bloquea la entrada con `e.Handled = true`. La tecla de retroceso también se permite para que el usuario pueda borrar. Si el usuario intenta ingresar un carácter no permitido, se muestra un mensaje de advertencia, informando que solo se pueden ingresar letras en este campo.

- "txtEstudianteID": Campo para el número único de identificación.



```
1 reference
private void txtEstudianteID_KeyPress(object sender, KeyPressEventArgs e)
{
    if (!char.IsDigit(e.KeyChar) && e.KeyChar != (char)Keys.Back)
    {
        e.Handled = true;
        MessageBox.Show("Solo se permiten números enteros en este campo.", "Entrada inválida", MessageBoxButtons.OK, MessageBoxIcon.Warning);
    }
}
```

En el campo txtEstudianteID, se implementó una validación para que solo se permitan números enteros. Usando `char.IsDigit(e.KeyChar)`, se verifica si el carácter presionado es un número. Si no lo es, se bloquea la entrada con `e.Handled = true`. Además, se permite la tecla de retroceso para borrar el contenido del campo. Si el

usuario intenta ingresar cualquier otro tipo de carácter, aparece un mensaje de advertencia informando que solo se permiten números en este campo.

- “txtPromedioGeneral”: Campo para el promedio general acumulado.

Promedio General:

```
1 reference
private void txtPromedioGeneral_KeyPress(object sender, KeyPressEventArgs e)
{
    if (!Char.IsDigit(e.KeyChar) && e.KeyChar != '.' && e.KeyChar != (char)8)
    {
        e.Handled = true;
    }

    // Si se presiona el punto decimal
    if (e.KeyChar == '.')
    {
        // Si ya hay un punto en el texto, no permitir otro
        if (txtPromedioGeneral.Text.Contains("."))
        {
            e.Handled = true;
        }
    }

    // Validar que no se ingresen más de 3 cifras después del punto
    if (txtPromedioGeneral.Text.Contains(".") && e.KeyChar != (char)8) // Excluir la tecla de retroceso
    {
        // Contar la cantidad de cifras después del punto
        int decimalCount = txtPromedioGeneral.Text.Substring(txtPromedioGeneral.Text.IndexOf('.') + 1).Length;

        if (decimalCount >= 3)
        {
            e.Handled = true;
            MessageBox.Show("Solo se permiten hasta 3 cifras después del punto decimal.", "Advertencia", MessageBoxButtons.OK, MessageBoxIcon.Warning);
        }
    }
}
```

En el campo txtPromedioGeneral, se implementó una validación que permite ingresar solo números y un punto decimal, para asegurar que se puedan ingresar promedios. Se usa Char.IsDigit(e.KeyChar) para permitir solo números, y se permite el punto decimal con e.KeyChar == '.'. Además, para evitar que el usuario ingrese más de un punto decimal, se verifica si ya existe un punto en el texto con txtPromedioGeneral.Text.Contains("."). Si el texto ya contiene un punto y el usuario intenta ingresar otro, se bloquea la entrada. También se limitó el número de decimales a tres después del punto, contando los caracteres después del punto con txtPromedioGeneral.Text.Substring(txtPromedioGeneral.Text.IndexOf('.') + 1).Length.

- “txtMateriasA”: Campo para la cantidad de materias aprobadas.

Materias Aprobadas:

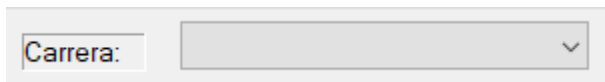
```
1 reference
private void txtMateriasA_KeyPress(object sender, KeyPressEventArgs e)
{
    if (!Char.IsDigit(e.KeyChar) && e.KeyChar != (char)Keys.Back)
    {
        e.Handled = true;
        MessageBox.Show("Solo se permiten números enteros en este campo.", "Entrada inválida", MessageBoxButtons.OK, MessageBoxIcon.Warning);
    }
}
```

Finalmente, en el campo txtMateriasA, se agregó una validación similar a la de txtEstudianteID, permitiendo solo la entrada de números enteros. De nuevo, se utiliza char.IsDigit(e.KeyChar) para verificar que el carácter presionado sea un número. Si no es un número, se bloquea la entrada con e.Handled = true. También se permite la tecla de retroceso para borrar cualquier valor ingresado. Si el usuario intenta ingresar un carácter no numérico, se muestra un mensaje de advertencia, informando que solo se permiten números enteros.

Lista desplegable “ComboBox”

Se agregó un `ComboBox` para la selección de carrera con opciones predefinidas ("Ingeniería", "Medicina", "Derecho", "Administración", etc.)

- Elemento “cmbCarrera”

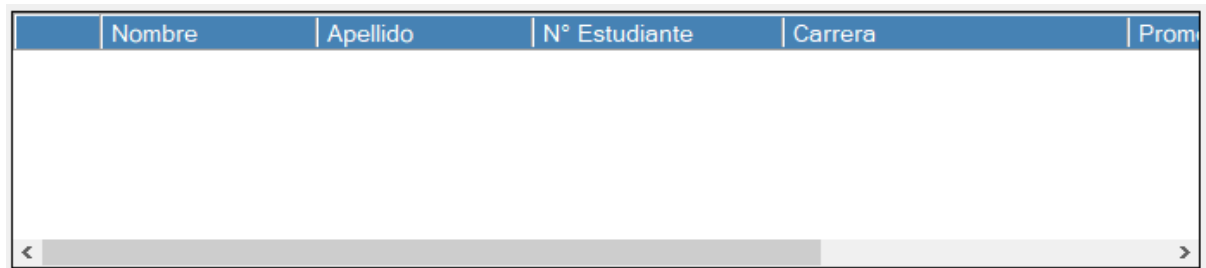


Se deshabilitó la opción de escritura en el ComboBox configurando su propiedad **DropDownStyle** a **DropDownList**, lo que permite solo seleccionar opciones predefinidas y evita que el usuario ingrese texto manualmente.

Tabla interactiva DataGridView

Se configuró un “DataGridView” para mostrar una lista completa de estudiantes registrados, que permite visualizar rápidamente sus datos básicos como Nombre, Apellido, Número de Estudiante, Carrera, Promedio General y Materias Aprobadas.

- Elemento “dataGridView1”:



Nombre	Apellido	N° Estudiante	Carrera	Promedio	Materias Aprobadas
--------	----------	---------------	---------	----------	--------------------

```

1 reference
private void Form1_Load(object sender, EventArgs e)
{
    if (!File.Exists("registro.txt"))
    {
        StreamWriter archivo = new StreamWriter("registro.txt");
        archivo.Close();
    }
    else
    {
        StreamReader archivo = new StreamReader("registro.txt");
        while (!archivo.EndOfStream)
        {
            string nombre = archivo.ReadLine();
            string apellido = archivo.ReadLine();
            string estudianteID = archivo.ReadLine();
            string carrera = archivo.ReadLine();
            string promedioGeneral = archivo.ReadLine();
            string materiasAprobadas = archivo.ReadLine();
            dataGridView1.Rows.Add(nombre, apellido, estudianteID, carrera, promedioGeneral, materiasAprobadas);
        }
        archivo.Close();
    }
}

```

El método Form1_Load se ejecuta al iniciar la aplicación. Lo primero que hace es verificar si el archivo "registro.txt" ya existe. Si no existe, lo crea usando StreamWriter para asegurarse de que haya un archivo disponible para guardar datos. En caso de que el archivo ya exista, lo abre con StreamReader y comienza a leer sus líneas. Cada línea representa un dato específico de un estudiante (como nombre, apellido, ID de estudiante, carrera, promedio y materias aprobadas), los cuales se añaden al DataGridView con el método dataGridView1.Rows.Add(). Esto permite que, al cargar el formulario, se muestren los registros de estudiantes que fueron guardados previamente. Finalmente, el archivo se cierra con archivo.Close() para liberar recursos.

```

5 references
private void GrabarDatos()
{
    // Sobrescribir el archivo de texto con los datos actuales del DataGridView
    StreamWriter archivo = new StreamWriter("registro.txt", false); // false para sobrescribir el archivo
    for (int i = 0; i < dataGridView1.Rows.Count; i++)
    {
        archivo.WriteLine(dataGridView1.Rows[i].Cells[0].Value.ToString());
        archivo.WriteLine(dataGridView1.Rows[i].Cells[1].Value.ToString());
        archivo.WriteLine(dataGridView1.Rows[i].Cells[2].Value.ToString());
        archivo.WriteLine(dataGridView1.Rows[i].Cells[3].Value.ToString());
        archivo.WriteLine(dataGridView1.Rows[i].Cells[4].Value.ToString());
        archivo.WriteLine(dataGridView1.Rows[i].Cells[5].Value.ToString());
    }
    archivo.Close();
}

```

El método `GrabarDatos` se encarga de guardar los registros del `DataGridView` en el archivo "registro.txt". Primero, se crea un `StreamWriter` con la opción `false` para sobrescribir el archivo cada vez que se guarda la información. Luego, se recorre cada fila del `DataGridView` con un bucle `for`. En cada iteración, se extraen los valores de cada celda de la fila (de la columna 0 a la columna 5) y se escriben en el archivo con `archivo.WriteLine()`. Esto asegura que cada dato se guarde en una nueva línea del archivo. Al finalizar, se cierra el archivo con `archivo.Close()`,

asegurando que los cambios se guarden correctamente. Este proceso asegura que los datos de los estudiantes se mantengan actualizados en el archivo de texto.

```
4 references
private void GrabarBorrado()
{
    StreamWriter archivo = new StreamWriter("registro.txt"); // Se borra el archivo y se actualiza con los datos que quedan en el GridView
    for (int i = 0; i < dataGridView1.Rows.Count; i++)
    {
        archivo.WriteLine(dataGridView1.Rows[i].Cells[0].Value.ToString());
        archivo.WriteLine(dataGridView1.Rows[i].Cells[1].Value.ToString());
        archivo.WriteLine(dataGridView1.Rows[i].Cells[2].Value.ToString());
        archivo.WriteLine(dataGridView1.Rows[i].Cells[3].Value.ToString());
        archivo.WriteLine(dataGridView1.Rows[i].Cells[4].Value.ToString());
        archivo.WriteLine(dataGridView1.Rows[i].Cells[5].Value.ToString());
    }
    archivo.Close();
}
```

El método GrabarBorrado actualiza el archivo "registro.txt" después de eliminar una fila del DataGridView. Borra el contenido anterior del archivo al crear un nuevo StreamWriter, luego recorre las filas restantes y guarda los datos de cada celda en el archivo con archivo.WriteLine(). Finalmente, cierra el archivo con archivo.Close(), asegurando que los cambios se guarden correctamente.

Botones "Button"

Los botones "Guardar", "Eliminar" y "Actualizar" permiten al usuario añadir nuevos registros, eliminar información existente y actualizar datos ya guardados, asegurando que el sistema mantenga la información actualizada y organizada.

- Botón "btnGuardar": se utiliza para registrar los datos ingresados por el usuario en los campos de texto (como nombre, apellido, número de estudiante, etc.). Al presionar este botón, los datos se añaden a la lista mostrada en el DataGridView y se actualiza el archivo donde se almacenan los registros. Su acción también incluye la validación de los campos para asegurarse de que no falte ninguna información antes de guardar.

Guardar


```

private void btnGuardar_Click(object sender, EventArgs e)
{
    // Verificar si todos los TextBox y el ComboBox están completos
    if (string.IsNullOrEmpty(txtNombre.Text) ||
        string.IsNullOrEmpty(txtApellido.Text) ||
        string.IsNullOrEmpty(txtEstudianteID.Text) ||
        string.IsNullOrEmpty(txtPromedioGeneral.Text) ||
        string.IsNullOrEmpty(txtMateriasA.Text) ||
        string.IsNullOrEmpty(cmbCarrera.Text))
    {
        MessageBox.Show("Por favor, complete todos los campos antes de guardar.", "Campos incompletos", MessageBoxButtons.OK, MessageBoxIcon.Warning);
        return; // No continuar si hay campos vacíos
    }

    // Agregar una nueva fila al DataGridView
    dataGridView1.Rows.Add(txtNombre.Text, txtApellido.Text, txtEstudianteID.Text, cmbCarrera.Text, txtPromedioGeneral.Text, txtMateriasA.Text);

    // Limpiar los cuadros de texto y deseleccionar el combo
    txtNombre.Text = "";
    txtApellido.Text = "";
    txtEstudianteID.Text = "";
    txtPromedioGeneral.Text = "";
    txtMateriasA.Text = "";
    cmbCarrera.SelectedIndex = -1;

    // Actualizar el archivo de texto con los datos actuales
    GrabarDatos();

    MessageBox.Show("Registro guardado exitosamente.", "Éxito", MessageBoxButtons.OK, MessageBoxIcon.Information);
}

```

Este bloque de código se ejecuta cuando el usuario intenta guardar un nuevo registro. Primero, verifica que todos los campos de texto (como txtNombre, txtApellido, txtEstudianteID, txtPromedioGeneral, txtMateriasA) y el ComboBox (cmbCarrera) no estén vacíos, utilizando la función string.IsNullOrEmpty(). Si algún campo está vacío, muestra un mensaje de advertencia pidiendo que se completen todos los campos y termina la ejecución del método con return, para evitar que se guarde el registro incompleto. Si todos los campos están completos, el programa agrega una nueva fila al DataGridView con los datos ingresados. Luego, limpia los cuadros de texto y deselecciona el ComboBox, dejándolo listo para ingresar nuevos datos. A continuación, actualiza el archivo de texto con los datos actuales mediante el método GrabarDatos(), y finalmente muestra un mensaje informando que el registro se ha guardado correctamente.

- Botón "btnEliminar": permite eliminar un estudiante seleccionado en el `DataGridView`. Antes de realizar la eliminación, el sistema solicita una confirmación para evitar eliminaciones accidentales. Una vez confirmada la acción, la fila correspondiente al estudiante se elimina tanto en la interfaz como en los registros almacenados.

Eliminar

```

1 reference
private void btnEliminar_Click(object sender, EventArgs e)
{
    // Verificar si se ha seleccionado una fila en el DataGridView
    if (dataGridView1.SelectedRows.Count > 0)
    {
        // Confirmar si se desea eliminar el registro
        var confirmResult = MessageBox.Show("¿Está seguro de que desea eliminar este estudiante?", "Confirmar Eliminación", MessageBoxButtons.YesNo, MessageBoxIcon.Warning);
        if (confirmResult == DialogResult.Yes)
        {
            // Eliminar la fila seleccionada
            int selectedIndex = dataGridView1.SelectedRows[0].Index; // Obtener el índice de la fila seleccionada
            dataGridView1.Rows.RemoveAt(selectedIndex);

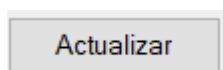
            // Llamar a la función para guardar los cambios (GrabarBorrado)
            GrabarBorrado();

            MessageBox.Show("Datos del estudiante seleccionado eliminados", "Eliminar Registro", MessageBoxButtons.OK, MessageBoxIcon.Information);
        }
    }
    else
    {
        MessageBox.Show("Por favor, seleccione un estudiante para eliminar.", "Sin selección", MessageBoxButtons.OK, MessageBoxIcon.Warning);
    }
}

```

Este código permite eliminar un estudiante del DataGridView. Primero, verifica si se ha seleccionado alguna fila. Si no se ha seleccionado ninguna, muestra un mensaje pidiendo que se elija un estudiante. Si se ha seleccionado una fila, muestra una ventana de confirmación para asegurarse de que el usuario desea eliminar el registro. Si el usuario confirma, elimina la fila seleccionada y luego actualiza el archivo de texto con los cambios utilizando la función GrabarBorrado(). Finalmente, muestra un mensaje informando que el registro ha sido eliminado con éxito.

- Botón "btnActualizar": Este botón se utiliza para modificar los datos de un estudiante ya registrado. Después de seleccionar al estudiante cuya información se desea actualizar, el usuario puede cambiar los valores en los campos de texto y presionar "Actualizar" para actualizar esos datos en el `DataGridView` y en el archivo de registros. Este botón se deshabilita temporalmente si no se está editando un registro existente.



```

1 reference
private void btnActualizar_Click(object sender, EventArgs e)
{
    if (btnActualizar.Tag != null) // Si hay un Tag asignado, sabemos que estamos editando un registro existente
    {
        int rowIndex = (int)btnActualizar.Tag;

        // Actualizar los valores de la fila seleccionada en el DataGridView
        dataGridView1.Rows[rowIndex].Cells[0].Value = txtNombre.Text;
        dataGridView1.Rows[rowIndex].Cells[1].Value = txtApellido.Text;
        dataGridView1.Rows[rowIndex].Cells[2].Value = txtEstudianteID.Text;
        dataGridView1.Rows[rowIndex].Cells[3].Value = cmbCarrera.Text;
        dataGridView1.Rows[rowIndex].Cells[4].Value = txtPromedioGeneral.Text;
        dataGridView1.Rows[rowIndex].Cells[5].Value = txtMateriasA.Text;

        // Limpiar los cuadros de texto
        txtNombre.Text = "";
        txtApellido.Text = "";
        txtEstudianteID.Text = "";
        cmbCarrera.Text = "";
        txtPromedioGeneral.Text = "";
        txtMateriasA.Text = "";

        btnActualizar.Tag = null;

        // Deshabilitar el botón "Actualizar" y habilitar el botón "Guardar" nuevamente
        btnActualizar.Enabled = false;
        btnGuardar.Enabled = true;

        GrabarDatos();

        MessageBox.Show("Registro actualizado exitosamente.", "Actualizar", MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
}

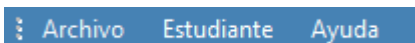
```

Aquí se actualiza un registro en el `DataGridView` cuando el botón `btnActualizar` tiene un `Tag` asignado. Primero, actualiza la fila seleccionada con los datos de los cuadros de texto y el `ComboBox`. Luego, limpia los cuadros de texto, desactiva el botón "Actualizar", activa el botón "Guardar" y llama a `GrabarDatos()` para guardar los cambios. Finalmente, muestra un mensaje indicando que el registro se ha actualizado con éxito.

Menú “MenuStrip”:

El menú principal de la aplicación está diseñado para facilitar la navegación y el acceso a las diferentes funcionalidades del sistema.

- “menuStrip1”: es el control principal que contiene el menú de la aplicación. Este menú se muestra en la parte superior de la ventana de la aplicación y organiza todas las opciones disponibles para el usuario.



Dentro de este menú principal existen varios submenús que agrupan las acciones de manera organizada

- Elemento “archivoToolStripMenuItem” (menuArchivo)

Este es un menú principal que contiene opciones relacionadas con la gestión general de la aplicación. Las sub-opciones dentro de este menú son:

- “nuevoToolStripMenuItem”: Esta opción permite limpiar todos los campos del formulario para registrar un nuevo estudiante. Al hacer clic en ella, todos los datos previamente ingresados en los cuadros de texto se eliminan.

Nuevo

```

1 reference
private void nuevoToolStripMenuItem_Click(object sender, EventArgs e)
{
    MessageBox.Show("Nuevo Archivo", "Archivo", MessageBoxButtons.OK);

    txtNombre.Text = "";
    txtApellido.Text = "";
    txtEstudianteID.Text = "";
    txtPromedioGeneral.Text = "";
    txtMateriasA.Text = "";
    cmbCarrera.SelectedIndex = -1;

    // Limpiar cualquier selección del DataGridView, si aplica
    dataGridView1.ClearSelection();
}

```

Aquí se ejecuta cuando se hace clic en la opción "Nuevo" del menú. Muestra un mensaje indicando "Nuevo Archivo", y luego limpia todos los cuadros de texto (txtNombre, txtApellido, txtEstudianteID, txtPromedioGeneral, txtMateriasA) y restablece el ComboBox a su estado inicial (sin selección). Además, se asegura de que cualquier selección actual en el DataGridView sea eliminada utilizando el método ClearSelection(), dejando todo listo para ingresar nuevos datos.

- “salirToolStripMenuItem”: Esta opción cierra la aplicación. Al hacer clic en ella, la aplicación se cierra, mostrando un mensaje de confirmación.

Salir

```

1 reference
private void salirToolStripMenuItem_Click(object sender, EventArgs e)
{
    Close();
}

```

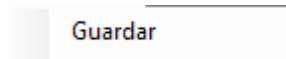
Aquí se cierra el formulario

- **Elemento “estudianteToolStripMenuItem” (menuEstudiante)**

Este es otro menú que contiene opciones para gestionar los registros de los estudiantes. Sus sub-opciones son:

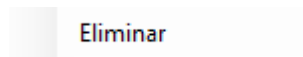
- “guardarToolStripMenuItem”: Esta opción registra los datos del estudiante en el sistema. Al hacer clic en ella, el programa verifica que todos los campos

estén completos y luego guarda los datos del estudiante en el archivo de texto.



Aquí se utiliza el mismo código del botón guardar.

- “eliminarToolStripMenuItem”: Permite eliminar un estudiante seleccionado de la lista de estudiantes. Cuando se selecciona un estudiante en el formulario, se elimina al hacer clic en esta opción.

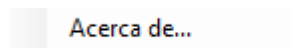


Aquí se utiliza el mismo código del botón eliminar.

- **Elemento “ayudaToolStripMenuItem” (menuAyuda)**

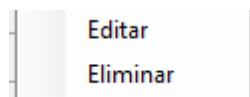
Este menú ofrece una opción para mostrar información relacionada con el sistema. La sub-opción es:

- “acercaDeToolStripMenuItem”: Muestra un cuadro de mensaje con información breve sobre el sistema. Es útil para que el usuario entienda el propósito de la interfaz.



Menú contextual “ContextMenuStrip”

Se utiliza para proporcionar opciones adicionales cuando el usuario hace clic derecho sobre un registro en el DataGridView. Las sub-opciones disponibles son:



- “toolStripMenuEditar”: Esta opción permite editar los datos del estudiante seleccionado. Al hacer clic en ella, los campos de texto se llenan con los datos del estudiante, permitiendo su actualización.



```

private void toolStripMenuEditar_Click(object sender, EventArgs e)
{
    // Verificar si se ha seleccionado una fila
    if (dataGridView1.SelectedRows.Count > 0)
    {
        int rowIndex = dataGridView1.SelectedRows[0].Index;

        // Cargar los valores de la fila seleccionada en los controles de entrada
        txtNombre.Text = dataGridView1.Rows[rowIndex].Cells[0].Value.ToString();
        txtApellido.Text = dataGridView1.Rows[rowIndex].Cells[1].Value.ToString();
        txtEstudianteID.Text = dataGridView1.Rows[rowIndex].Cells[2].Value.ToString();
        cmbCarrera.Text = dataGridView1.Rows[rowIndex].Cells[3].Value.ToString();
        txtPromedioGeneral.Text = dataGridView1.Rows[rowIndex].Cells[4].Value.ToString();
        txtMateriasA.Text = dataGridView1.Rows[rowIndex].Cells[5].Value.ToString();

        btnActualizar.Tag = rowIndex;

        // Habilitar el botón "Actualizar" y deshabilitar el botón "Guardar"
        btnGuardar.Enabled = false;
        btnActualizar.Enabled = true;
    }
    else
    {
        MessageBox.Show("Seleccione una fila para editar.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}

```

Aquí se ejecuta cuando se desea editar una fila en el `DataGridView`.

Primero, verifica si hay una fila seleccionada en el `DataGridView` mediante `dataGridView1.SelectedRows.Count > 0`. Si es así, obtiene el índice de la fila seleccionada (`rowIndex`) y carga los valores de esa fila en los controles de entrada, como `txtNombre`, `txtApellido`, `txtEstudianteID`, `cmbCarrera`, `txtPromedioGeneral`, y `txtMateriasA`. Luego, asigna el índice de la fila seleccionada al `Tag` del botón `btnActualizar`, lo que permite saber qué fila se está editando. Después, se deshabilita el botón "Guardar" y se habilita el botón "Actualizar", para que el usuario pueda actualizar los datos. Si no se ha seleccionado ninguna fila, muestra un mensaje de error pidiendo que se seleccione una fila para editar.

- "toolStripMenuEliminar": Permite eliminar el registro del estudiante seleccionado. Al hacer clic en esta opción, el estudiante es eliminado del DataGridView y los datos se actualizan en el sistema.

Eliminar

Aquí se utiliza el mismo código del botón eliminar.

Barras de Herramientas "ToolStrip"

- "tIsOpciones": Es una barra de herramientas ubicada en la parte superior de la ventana del programa. Contiene íconos que representan las acciones más utilizadas, facilitando su acceso rápido. Los íconos incluyen:



- “toolGuardar”: Permite guardar el registro actual del estudiante. Al hacer clic en este ícono, los datos se guardan en el sistema.



Aquí se utiliza el mismo código del botón guardar.

- “tIsbtnEliminar”: Elimina el estudiante seleccionado de la lista. Este ícono permite eliminar rápidamente un registro sin necesidad de usar el menú contextual o los botones.



Aquí se utiliza el mismo código del botón eliminar.

- “tIsbtnActualizar”: Actualiza la información de un estudiante. Este ícono permite refrescar los datos del estudiante, ya sea para agregar cambios o para sincronizar los datos del sistema.



Aquí se utiliza el mismo código del botón actualizar.

Cada uno de estos íconos tiene un “**ToolTipText**” asociado, que muestra una breve descripción de la función cuando el usuario pasa el cursor sobre el ícono.

