

UNIVERSIDAD NACIONAL DE INGENIERÍA
(UNI)



Materia: Matemática para computación II

Ing. En computación.

Tema: Redimensión de arreglos.

Laboratorio #1

Integrantes:

Erick Hazael Morales Jarquin 2023-0862U

Denis Josué Wong S. 2019-0919U

Grupo: 2M1-CO.

Programa, Registro de estudiantes hecho por: Erick Morales.

Programa, Registro de trabajadores hecho por: Denis Wong.

Laboratorio #1: Registro de estudiantes.

Un placer y gracias por su atención, es bien sabido que c# es un amplio lenguaje de programación y complejo de entender en un corto periodo de tiempo pero por eso es nuestro deber compartir la belleza de este lenguaje para trabajar. Antes de pasar a nuevas tecnologías es importante tener dominio de las bajas, en este caso de Windows forms, Windows forms es una tecnología que no se usa ya demasiado pero es una buena alternativa para empezar a desarrollar tu aplicación de escritorio, la ventaja de Windows forms es que tiene una manera de hacer interfaces graficas más eficiente y atractiva a la vista que del propio Java.

En Java necesitas saber hasta como programar un pequeño JFrame, esto puede causar un proceso lento en desarrollar tu de escritorio. Además, sus interfaces no serán muy atractivas si de tiempo no dispones, si eres un programador aficionado en código y no en diseño, c# te gustara muchísimo, aquí tienes una pequeña guía de como realizar un diseño rápido y sencillo de un sistema cualquiera ya sea de ventas o manejo de fabricas, eso depende de tu creatividad:

Aprender a diseñar tu primera app de escritorio:

<https://youtu.be/eCSbUCL4teE?si=oo--ixrjHHFylwhJ>

(Copiar y pegar en el buscador)

Aprender a usar formularios secundarios para crear vistas es crucial si quieres tener una reestructura decente en tu aplicación, porque habrá una que otra ocasión por no decir siempre que necesitaras una nueva vista para realizar el proceso correctamente, aparte que será mas como de usar y atractivo a la vista, aparte de tener un buen orden, es algo practico de hacer, aquí hay una guía de como crear las vistas secundarias donde se estarán mostrando las distintas secciones de la solución:

Aprender a crear vistas en Windows forms:

<https://youtu.be/lx7aaNazwwU?si=xG9QXnV8OV2AlzU6>

(Copiar y pegar en el buscador)

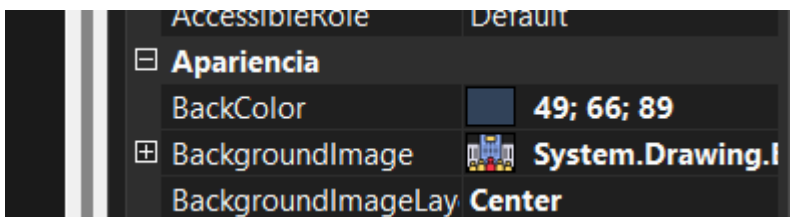
Si tienes claro de como diseñar un formulario, ahora pasemos a la parte interesante, todo lo que pasa por encima y se mira es lo de menos. Empieza diseñando el formulario padre guiándote del video anterior, crea paneles y asignales un color a tu preferencia. Uno para los botones que cambiaran las

vistas, otro para los 3 botones responsables del cambio de tamaño de tu formulario y por ultimo y menos importante, el panel donde se mostraran las vistas.

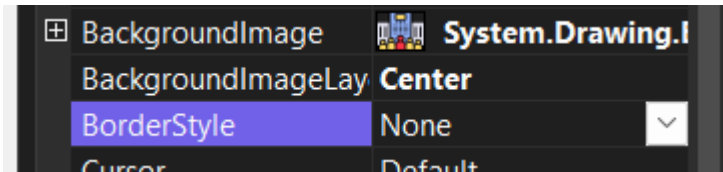
Modelo a seguir para avanzar al código:



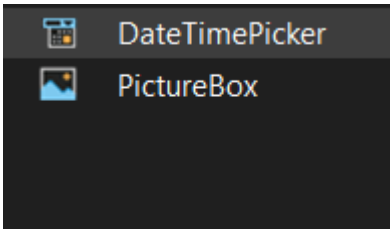
Si te queda la duda de como se han puesto imágenes como por ejemplo en el fondo del panel, ubicándote en propiedades hay una propiedad que te dice:



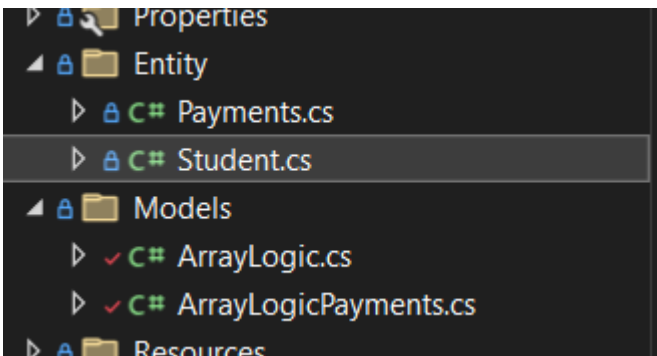
Es importante haber descargado la imagen para exportarla y elegir un fondo adecuado, si un dado caso la imagen esta como no debe de estar, es crucial ajustarla usando la propiedad :



Ajusta la imagen a tu gusto. Sin embargo, puedes poner un logo a tu preferencia usando:



Si te gusta ser creativo puedes darte gusto con eso, aunque es la parte bonita, programar también tiene su toque si estas un poco mal de la cabeza. A continuación se explicara la lógica para guardar, buscar, borrar y actualizar estudiante usando una clase en models:



Anteriormente se ha explicado la estructura para guardar tus clases y métodos, junto a sus vistas. Creamos 2 clases dentro de la carpeta Entity que servirán de nuestros objetos y luego 2 clases mas en la carpeta models donde se manejaran los arreglos como una colección de objeto.

```

17 referencias
public class ArrayLogic
{
    private static readonly ArrayLogic logic = new();
    13 referencias
    public static ArrayLogic Arraylog { get { return logic; } }
    private Student[] students;
    private int size = 0, quantity = 0;

    1 referencia
    public ArrayLogic()
    {
        students = new Student[size];
    }

    1 referencia
    public void AddStudent(Student newstudent)
    {
        if (quantity >= size)
        {
            size = size == 0 ? 1 : size + 1;
            Array.Resize(ref students, size);
        }
        students[quantity] = newstudent;
        ++quantity;
    }

    4 referencias
    public Student[] GetStudents()
    {
        for (int i = 0; i < students.Length; ++i)
        {
            for (int j = 0; j < students.Length - i - 1; j++)
            {
                if (students[j].Finalgrades < students[j + 1].Finalgrades)
                {
                    Student BestGrades = students[j];
                    students[j] = students[j + 1];
                    students[j + 1] = BestGrades;
                }
            }
        }
        return students;
    }
}

```

Para poder guardar un estudiante se requiere de métodos ubicados en la clase Studentlogic, esta sección se divide en guardar, actualizar, eliminar y buscar estudiando por medio de su carnet. En el método guardar se agrega una condición que comprueba si la cantidad es mayor al almacenamiento del arreglo, y si no, procede a guardar dicha información. En el siguiente método está construido con el objeto de agrupar a los estudiantes de mayor a menor nota e imprimir.

```

public bool RemoveStudent(string Carnet)
{
    var index = Array.FindIndex(students, e => Carnet.Equals(e.Carnet));
    if (index < 0) return false;
    for (int i = index; i < quantity - 1; i++)
    {
        students[i] = students[i + 1];
    }

    --quantity;
    Array.Resize(ref students, quantity);
    return true;
}

0 referencias
public bool UpdateStudent(Student student)
{
    if (string.IsNullOrEmpty(student.Carnet)) return false;
    var index = Array.FindIndex(students, s => student.Carnet.Equals(s.Carnet));
    if (index < 0) return false;

    students[index] = student;
    return true;
}

1 referencia
public bool SaveGrades(Student student)
{
    if (string.IsNullOrEmpty(student.Carnet)) return false;
    var index = Array.FindIndex(students, s => student.Carnet.Equals(s.Carnet));
    if (index < 0) return false;
    students[index].IPar = student.IPar;
    students[index].IIPar = student.IIPar;
    students[index].Project = student.Project;
    return true;
}

8 referencias
public Student[] SearchStudent(string carnet)
{
    var student = Array.Find(students, e => carnet.Equals(e.Carnet));

    return student != null ? new[] { student } : Array.Empty<Student>();
}

```

Método para eliminar, actualizar y buscar estudiante por medio del carnet siguiendo la misma logica

```

1 referencia
public bool SaveGrades(Student student)
{
    if (string.IsNullOrEmpty(student.Carnet)) return false;
    var index = Array.FindIndex(students, s => student.Carnet.Equals(s.Carnet));
    if (index < 0) return false;
    students[index].IPar = student.IPar;
    students[index].IIPar = student.IIPar;
    students[index].Project = student.Project;
    return true;
}

8 referencias
public Student[] SearchStudent(string carnet)
{
    var student = Array.Find(students, e => carnet.Equals(e.Carnet));

    return student != null ? new[] { student } : Array.Empty<Student>();
}

0 referencias
public bool CarnetUnique(string carnet)
{
    return SearchStudent(carnet).Length == 0;
}

0 referencias
public bool IdtUnique(string identification)
{
    return SearchStudent(identification).Length == 0;
}
}

```

Se guardan las notas ubicando al estudiante por su carnet.

Existen 2 métodos útiles para validar la entrada de datos, nada mas puede haber un carnet y una cédula dentro del sistema de registro.


```

9 referencias
public class ArrayLogicPayments
{
    private static readonly ArrayLogicPayments logicPay = new();
    5 referencias
    public static ArrayLogicPayments ArraylogPay { get { return logicPay; } }
    private Payments[] _payments;

    private int size = 0, quantity = 0;

    1 referencia
    public ArrayLogicPayments()
    {
        _payments = new Payments[size];
    }

    1 referencia
    public void AddPayment(Payments newPayment)
    {
        if (!CanPayment(newPayment.Carnet))
        {
            MessageBox.Show("No Puede pagar sin haber registrado un estudiante", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
            return;
        }
        AutoFillCamp(newPayment);

        if (quantity >= size)
        {
            size = size == 0 ? 1 : size + 1;
            Array.Resize(ref _payments, size);
        }
        _payments[quantity] = newPayment;
        ++quantity;
    }

    3 referencias
    public Payments[] GetPayments() => _payments;

    1 referencia
    public bool RemovePayment(string carnet)
    {
        var index = Array.FindIndex(_payments, p => carnet.Equals(p.Carnet));
        if (index < 0) return false;
        for (int i = index; i < quantity - 1; i++)
        {
            _payments[i] = _payments[i + 1];
        }
        _payments[quantity - 1] = null;
    }
}

```

Mismo proceso para la extracción de información categoría pagos, se guardan los datos, se eliminan, se actualizan y se extraen a un archivo Excel en función del carnet estudiantil junto a su información como el nombre, carnet y fecha de pago

```

        _payments[i] = _payments[i + 1];
    }
    _payments[quantity - 1] = null;
    --quantity;
    return true;
}

0 referencias
public bool UpdatePayment(Payments payment)
{
    if (string.IsNullOrEmpty(payment.Carnet)) return false;
    var index = Array.FindIndex(_payments, p => payment.Carnet.Equals(p.Carnet));
    if (index < 0) return false;

    _payments[index] = payment;
    return true;
}

0 referencias
public Payments[] SearchPayment(string carnet)
{
    var pay = Array.Find(_payments, p => carnet.Equals(p.Carnet));
    return pay != null ? new[] { pay } : Array.Empty<Payments>();
}

1 referencia
public bool CanPayment(string Carnet)
{
    var StudentExist = ArrayLogic.Arraylog.SearchStudent(Carnet).Any();
    return StudentExist;
}

1 referencia
public void AutoFillCamp(Payments payments)
{
    var StudentFills = ArrayLogic.Arraylog.SearchStudent(payments.Carnet).FirstOrDefault();
    if (payments != null)
    {
        payments.Name = StudentFills.Name;
        payments.Surname = StudentFills.Surname;
        payments.Identification = StudentFills.Identification;
    }
}
}

```

Método para rellenar los campos de nombre, apellido y carnet para enlazar los datos del usuario y el pago, de este modo se ubica de quien y cuando hizo el pago .

```

public void ExportCloseXml(DataGridView information, string filePath)
{
    try
    {
        using (var workbook = new XLWorkbook())
        {
            var workSheet = workbook.Worksheets.Add("Sheet1");

            for (int i = 0; i < information.Columns.Count; i++)
            {
                workSheet.Cell(1, i + 1).Value = information.Columns[i].HeaderText;
            }

            for (int i = 0; i < information.Rows.Count; i++)
            {
                if (!information.Rows[i].IsNewRow)
                {
                    for (int j = 0; j < information.Columns.Count; j++)
                    {
                        var cellValue = information.Rows[i].Cells[j].Value;
                        workSheet.Cell(i + 2, j + 1).Value = cellValue != null ? cellValue.ToString() : string.Empty;
                    }
                }
            }

            workbook.SaveAs(filePath);
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Error al exportar a Excel: {ex.Message}", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}

```

Clase para exportar la información de los estudiantes y pagos en un archivo Excel.

```
public bool TextNullOrEmpty(TextBox[] textBoxes)
{
    bool TextNull = false;

    foreach (TextBox txt in textBoxes)
    {
        if (string.IsNullOrEmpty(txt.Text))
        {
            errorProvider.SetError(txt, "Este campo es obligatorio.");
            TextNull = true;
        }
        else
        {
            errorProvider.SetError(txt, "");
        }
    }

    if (TextNull)
    {
        System.Media.SystemSounds.Exclamation.Play();
    }

    return TextNull;
}
```

Validaciones: Campo vacío.

```

2 referencias
public bool IsValidId(TextBox[] identification)
{
    bool idWrong = false;
    string pattern = @"^d{3}-d{6}-d{4}[A-Z]$";

    Regex regex = new Regex(pattern);
    foreach (TextBox txt in identification)
    {
        if (!regex.IsMatch(txt.Text))
        {
            errorProvider.SetError(txt, "Esta identificacion no es valida");
            idWrong = true;
        }
        else
        {
            errorProvider.SetError(txt, "");
        }
    }

    if (idWrong)
    {
        System.Media.SystemSounds.Exclamation.Play();
    }

    return idWrong;
}

4 referencias
public bool IsValidCarnet(TextBox[] carnet)
{
    bool carnetWrong = false;

    string patterncarnet = @"^20\d{2}-d{4}[A-Z]$";
    Regex regexcarnet = new Regex(patterncarnet);

    foreach (TextBox txtcarnet in carnet)
    {
        if (!regexcarnet.IsMatch(txtcarnet.Text))
        {
            errorProvider.SetError(txtcarnet, "Este carnet no es valido");
            carnetWrong = true;
        }
        else
        {
            errorProvider.SetError(txtcarnet, "");
        }
    }

    if (carnetWrong)
    {
        System.Media.SystemSounds.Exclamation.Play();
    }

    return carnetWrong;
}

9 referencias

```

Validaciones: Carnet y cédula válida.

```

private void Form1_Load(object sender, EventArgs e)
{
}
5 referencias
void OpenForms(Form form)
{
    while (PanelFather.Controls.Count > 0)
    {
        PanelFather.Controls.RemoveAt(0);
    }
    Form formHijo = form;
    form.TopLevel = false;
    formHijo.FormBorderStyle = FormBorderStyle.None;
    formHijo.Dock = DockStyle.Fill;
    PanelFather.Controls.Add(formHijo);
    formHijo.Show();
}

1 referencia
private void btnMin_Click(object sender, EventArgs e)
{
    this.WindowState = FormWindowState.Minimized;
}

1 referencia
private void btnMaxim_Click(object sender, EventArgs e)
{
    this.WindowState = FormWindowState.Maximized;
    btnMaxim.Visible = false;
    btnNormal.Visible = true;
}

1 referencia
private void button4_Click(object sender, EventArgs e)
{
    this.WindowState = FormWindowState.Normal;
    btnMaxim.Visible = true;
    btnNormal.Visible = false;
}

1 referencia
private void btnClose_Click_1(object sender, EventArgs e)
{
    Application.Exit();
}

1 referencia
private void btnRegister_Click(object sender, EventArgs e)
{
    OpenForms(new RegisterView());
}

```

Aplicamos el tutorial sobre conectar vistas o forma hijos/secundarios a un formulario padre/principal

```

public RegisterView()
{
    InitializeComponent();
    validations = new Validations();
    errorProvider1 = new ErrorProvider();
}

2 referencias
private void PrintStudents()
{
    dgStudents.DataSource = null;
    dgStudents.DataSource = ArrayLogic.Arraylog.GetStudents();
    dgStudents.Columns["IPar"].Visible = false;
    dgStudents.Columns["IIPar"].Visible = false;
    dgStudents.Columns["project"].Visible = false;
    dgStudents.Columns["Finalgrades"].Visible = false;
    dgStudents.Columns["test"].Visible = false;
}

1 referencia
void RegisterStudents()
{
    if (ArrayLogic.Arraylog.SearchStudent(textCarnet.Text).Any())
    {
        MessageBox.Show("El carnet ya está registrado. Por favor, ingrese un carnet único.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
        return;
    }

    if (ArrayLogic.Arraylog.SearchStudent(textIdentification.Text).Any())
    {
        MessageBox.Show("La cedula ya está registrado. Por favor, ingrese una cedula única.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
        return;
    }

    Student student = new Student
    {
        Name = textName.Text,
        Surname = textSurname.Text,
        Number = textNumMask.Text,
        Carnet = textCarnet.Text,
        Identification = textIdentification.Text,
        DateRegister = dateTimePicker1.Value,
    };

    ArrayLogic.Arraylog.AddStudent(student);
}

```

Aplica la lógica de guardar estudiantes en tu formulario junto a sus validaciones.

```

private void RegisterView_Load(object sender, EventArgs e)
{
    PrintStudents();
}

1 referencia
private void btnRegister_Click(object sender, EventArgs e)
{
    TextBox[] textBoxes = { textCarnet, textSurname, textName, textIdentification };
    TextBox[] textBid = { textIdentification };
    TextBox[] _textCarnet = { textCarnet };

    if (validations.IsValidCarnet(_textCarnet))
    {
        MessageBox.Show("Por favor, ingrese un carnet valido", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
        return;
    }
    if (validations.IsValidId(textBid))
    {
        MessageBox.Show("Por favor, ingrese un id valido", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
        return;
    }
    if (validations.TextNullOrEmpty(textBoxes))
    {
        MessageBox.Show("Por favor, rellena todos los campos", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
        return;
    }
    if (!textNumMask.MaskFull)
    {
        errorProvider1.SetError(textNumMask, "Numero incompleto");
        return;
    }

    RegisterStudents();
    PrintStudents();
}

```

Validaciones y método guardar.


```

1 referencia
public GradesView()
{
    InitializeComponent();
    validations = new Validations();
    textP1.KeyPress += new KeyPressEventHandler(validations.ValidateNumberInput);
    textPar2.KeyPress += new KeyPressEventHandler(validations.ValidateNumberInput);
    textPro.KeyPress += new KeyPressEventHandler(validations.ValidateNumberInput);
    textTest.KeyPress += new KeyPressEventHandler(validations.ValidateNumberInput);
}

1 referencia
private void btnSearch_Click(object sender, EventArgs e)
{
    var Carnet = textSearch.Text;
    var _students = ArrayLogic.Arraylog.SearchStudent(Carnet);

    if (_students.Length > 0)
    {
        dataGridView1.DataSource = _students;
    }
    else
    {
        MessageBox.Show("No se encontro al estudiante con carnet:" + Carnet, "Atencion", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}

3 referencias
private void PrintStudents()
{
    dataGridView1.DataSource = null;
    dataGridView1.DataSource = ArrayLogic.Arraylog.GetStudents();

    dataGridView1.Columns["identification"].Visible = false;
    dataGridView1.Columns["Number"].Visible = false;
    dataGridView1.Columns["dateRegister"].Visible = false;
}

```

Llamar validaciones en el constructor.

Validaciones que requieren de un buttom event se pueden instanciar de ese modo, ahorramos líneas de código, esfuerzo y obtenemos una mejor experiencia con el programa, en este caso es para no ingresar algo que sea diferente a un número.

```

1 referencia
private void btnSave_Click(object sender, EventArgs e)
{
    if (dataGridView1.SelectedRows.Count == 0)
    {
        MessageBox.Show("No se seleccionado un estudiante", "Atencion", MessageBoxButtons.OK, MessageBoxIcon.Error);
        return;
    }
    var IndexSelectDg = dataGridView1.SelectedRows[0].Index;
    if (dataGridView1.Rows[IndexSelectDg].DataBoundItem is not Student studentselect)
    {
        MessageBox.Show("No se seleccionado un estudiante", "Atencion", MessageBoxButtons.OK, MessageBoxIcon.Error);
        return;
    }

    if (string.IsNullOrEmpty(studentselect.Carnet))
    {
        MessageBox.Show("No se ha encontrado el carnet estudiantil", "Atencion", MessageBoxButtons.OK, MessageBoxIcon.Error);
        return;
    }

    TextBox[] textBoxes = { textPar2, textP1, textTest, textPro };
    if (validations.TextNullOrEmpty(textBoxes))
    {
        MessageBox.Show("Por favor, rellena todos los campos", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
        return;
    }

    studentselect.IPar = float.Parse(textP1.Text);
    studentselect.IIPar = float.Parse(textPar2.Text);
    studentselect.Project = float.Parse(textPro.Text);
    studentselect.Test = float.Parse(textTest.Text);
    studentselect.Finalgrades = studentselect.finalgrades();
    ArrayLogic.Arraylog.SaveGrades(studentselect);
    PrintStudents();
}

```

Obligatoriamente si desea guardar las calificaciones debe de haber ingresado algo.

```

3 referencias
public partial class PaymentsView : Form
{
    private Validations validations;
    private Payments payments;
    private Student student;
    1 referencia
    public PaymentsView()
    {
        InitializeComponent();
        validations = new Validations();
        payments = new Payments();
        student = new Student();
        textCordobas.KeyPress += new KeyPressEventHandler(validations.ValidateNumberInput);
        comboMotive.SelectedIndex = 0;
        PrintStudents();
    }

    private int Quantity = 1;
    4 referencias
    private void PrintStudents()
    {
        dgStudents.DataSource = null;
        dgStudents.DataSource = ArrayLogicPayments.ArraylogPay.GetPayments();
    }

    1 referencia
    void RegisterStudents()
    {
        string CarnetPayment = textCarnet.Text;
        Payments payment = new Payments
        {
            Cordobas = float.Parse(textCordobas.Text),
            Motive = comboMotive.Text,
            Carnet = CarnetPayment,
            DatePay = dateTimePicker1.Value,
            CantPayments = Quantity
        };

        ArrayLogicPayments.ArraylogPay.AddPayment(payment);
    }
}

```

Registro de pagos usando ArrayLogicPayments

```

1 referencia
private void btnRegister_Click(object sender, EventArgs e)
{
    TextBox[] textBoxes = { textCarnet, textCordobas };
    TextBox[] textCarnetValid = { textCarnet };
    if (validations.IsValidCarnet(textCarnetValid))
    {
        MessageBox.Show("Por favor, rellena todos los campos", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
        return;
    }

    if (validations.TextNullOrEmpty(textBoxes))
    {
        MessageBox.Show("Por favor, rellena todos los campos", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
        return;
    }
    RegisterStudents();
    ++Quantity;
    PrintStudents();
}
1 referencia
private void PaymentsView_Load(object sender, EventArgs e)
{
    PrintStudents();
}

```

Debe de haber algo valido para guardar, de casi contrario el método no funcionará.

```

1 referencia
private void button2_Click(object sender, EventArgs e)
{
    if (dgStudents.SelectedRows.Count == 0)
    {
        MessageBox.Show("No se seleccionado un estudiante", "Atencion", MessageBoxButtons.OK, MessageBoxIcon.Error);
        return;
    }
    var IndexSelectDg = dgStudents.SelectedRows[0].Index;
    if (dgStudents.Rows[IndexSelectDg].DataBoundItem is not Payments studentselect)
    {
        MessageBox.Show("No se seleccionado un estudiante", "Atencion", MessageBoxButtons.OK, MessageBoxIcon.Error);
        return;
    }

    if (string.IsNullOrEmpty(studentselect.Carnet))
    {
        MessageBox.Show("No se ha encontrado el carnet estudiantil", "Atencion", MessageBoxButtons.OK, MessageBoxIcon.Error);
        return;
    }

    var resulMessage = MessageBox.Show("Seguro que quiere eliminar el registro?", "Atencion", MessageBoxButtons.YesNo, MessageBoxIcon.Exclamation);
    if (resulMessage == DialogResult.No)
    {
        return;
    }
    ArrayLogicPayments.ArraylogPay.RemovePayment(studentselect.Carnet);
    dgStudents.DataSource = null;
    dgStudents.DataSource = ArrayLogicPayments.ArraylogPay.GetPayments();
    --Quantity;
}

1 referencia
private void button1_Click(object sender, EventArgs e)
{
    if (dgStudents.SelectedRows.Count == 0)
    {
        MessageBox.Show("No se seleccionado un estudiante", "Atencion", MessageBoxButtons.OK, MessageBoxIcon.Error);
        return;
    }
    var IndexSelectDg = dgStudents.SelectedRows[0].Index;
    if (dgStudents.Rows[IndexSelectDg].DataBoundItem is not Payments studentselect)
    {
        MessageBox.Show("No se seleccionado un estudiante", "Atencion", MessageBoxButtons.OK, MessageBoxIcon.Error);
        return;
    }

    UpdatePayView updateViewpay = new UpdatePayView(studentselect);
    updateViewpay.OnDataUpdate += PrintStudents;
    updateViewpay.ShowDialog();
}

1 referencia

```

Eliminar registro de pago extrayendo la información a través del carnet de estudiante, se elimina sus pagos pero si desea actualizar se llama otro formulario que lanza un evento al cerrarse, pero al abrirse genera un evento que extrae los datos del pago y los pega en campos de texto para su edición.

```

private ExportToExcel ExportToExcel;
1 referencia
public ReportView()
{
    InitializeComponent();
}
1 referencia
private void PrintStudents()
{
    dataGridView1.DataSource = null;
    dataGridView1.DataSource = ArrayLogicPayments.ArraylogPay.GetPayments();
    ExportToExcel = new ExportToExcel();
}

1 referencia
private void ReportView_Load(object sender, EventArgs e)
{
    PrintStudents();
}

1 referencia
private void btnExport_Click(object sender, EventArgs e)
{
    SaveFileDialog dlg = new SaveFileDialog()
    {
        Filter = "Excel Files (*.xlsx)|*.xlsx",
        FileName = "PaymentsReport.xlsx"
    };

    if (dlg.ShowDialog() == DialogResult.OK)
    {
        ExportToExcel.ExportCloseXml(dataGridView1, dlg.FileName);
        MessageBox.Show("Datos exportados exitosamente", "Éxito", MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
}

```

Uso de la clase para guardar un archivo Excel con el registro. Indicas que será un archivo xlsx y el nombre del archivo .

```

InitializeComponent();
validations = new Validations();
student = payments;
UpdateInformation();
}

1 referencia
public void UpdateInformation()
{
    if (student != null)
    {
        textCarnet.Text = student.Carnet;
        textCordobas.Text = student.Cordobas.ToString();
        comboMotive.Text = student.Motive;
        dateTimePicker1.Value = student.DatePay;
    }
}

1 referencia
private void btnUpdate_Click(object sender, EventArgs e)
{
    TextBox[] textBoxes = { textCarnet, textCordobas };
    TextBox[] textCarnetValid = { textCarnet };
    if (validations.IsValidCarnet(textCarnetValid))
    {
        MessageBox.Show("Por favor, rellena todos los campos", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
        return;
    }

    if (validations.TextNullOrEmpty(textBoxes))
    {
        MessageBox.Show("Por favor, rellena todos los campos", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
        return;
    }

    if (student != null)
    {
        student.Carnet = textCarnet.Text;
        student.Cordobas = float.Parse(textCordobas.Text);
        student.Motive = comboMotive.Text;
        student.DatePay = dateTimePicker1.Value;
    }

    OnDataUpdate?.Invoke();
    this.Close();
}

```

Ubicados en el formulario de actualización, es como cualquier otra excepto por “.Invoke();” y “this.close”, esto dispara el evento antes mencionado la actualización de los datos sin haber una intervención.

```

public partial class UpdateView : Form
{
    private Validations validations;
    public event Action OnDataUpdate;
    private Student _student;
    1 referencia
    public UpdateView(Student _student)
    {
        InitializeComponent();
        student = _student;
        UpdateInformation();
        validations = new Validations();
        texPar1.KeyPress += new KeyPressEventHandler(validations.ValidateNumberInput);
        textPar2.KeyPress += new KeyPressEventHandler(validations.ValidateNumberInput);
        textPro.KeyPress += new KeyPressEventHandler(validations.ValidateNumberInput);
        textTest.KeyPress += new KeyPressEventHandler(validations.ValidateNumberInput);
    }
    1 referencia
    public void UpdateInformation()
    {
        if (student != null)
        {
            texName.Text = student.Name;
            textSurname.Text = student.Surname;
            textCarnet.Text = student.Carnet;
            dateTimePicker1.Value = (DateTime)student.DateRegister;
            textNum.Text = student.Number;
            texPar1.Text = student.IPar.ToString();
            textPar2.Text = student.IIPar.ToString();
            textPro.Text = student.Project.ToString();
            textIdentification.Text = student.Identification;
            textTest.Text = student.Test.ToString();
        }
    }
    1 referencia
}

```

Usamos el evento keypress en el constructor para su validación al ingresar información. Y actualizamos lo que se haya cambiado .


```

1 referencia
private void btnUpdate_Click(object sender, EventArgs e)
{
    TextBox[] textBoxes = { textCarnet, textNum, textSurname, texName, textIdentification, texPar1, textPar2, textPro, textTest };
    TextBox[] textBid = { textIdentification };
    TextBox[] _textCarnet = { textCarnet };

    if (validations.IsValidCarnet(_textCarnet))
    {
        MessageBox.Show("Por favor, ingrese un carnet valido", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
        return;
    }
    if (validations.IsValidId(textBid))
    {
        MessageBox.Show("Por favor, ingrese un id valido", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
        return;
    }
    if (validations.TextNullOrEmpty(textBoxes))
    {
        MessageBox.Show("Por favor, rellena todos los campos", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
        return;
    }
    if (student != null)
    {
        student.Name = texName.Text;
        student.Surname = textSurname.Text;
        student.Carnet = textCarnet.Text;
        student.DateRegister = dateTimePicker1.Value;
        student.Number = textNum.Text;
        student.IPar = float.Parse(texPar1.Text);
        student.IIPar = float.Parse(textPar2.Text);
        student.Project = float.Parse(textPro.Text);
        student.Identification = textIdentification.Text;
        student.Test = float.Parse(textTest.Text);
    }
    OnDataUpdate.Invoke();
    this.Close();
}

```

Nuevamente el evento Invoke es usado, cuando se cierra se extrae la nueva versión y se almacenan en la colección de objetos y en donde corresponde

```
private ExportToExcel ExportToExcel;
public Student[] students;
1 referencia
public StudentsView()
{
}
```

```
2 referencia
private void PrintStudents()
{
    dgPrint.DataSource = null;
    dgPrint.DataSource = ArrayLogic.Arraylog.GetStudents();
}

1 referencia
private void StudentsView_Load(object sender, EventArgs e)
{
    PrintStudents();
}

1 referencia
private void btnDelete_Click(object sender, EventArgs e)
{
    if (dgPrint.SelectedRows.Count == 0)
    {
        MessageBox.Show("No se seleccionado un estudiante", "Atencion", MessageBoxButtons.OK, MessageBoxIcon.Error);
        return;
    }
    var IndexSelectDg = dgPrint.SelectedRows[0].Index;
    if (dgPrint.Rows[IndexSelectDg].DataBoundItem is not Student studentselect)
    {
        MessageBox.Show("No se seleccionado un estudiante", "Atencion", MessageBoxButtons.OK, MessageBoxIcon.Error);
        return;
    }
    if (string.IsNullOrEmpty(studentselect.Carnet))
    {
        MessageBox.Show("No se ha encontrado el carnet estudiantil", "Atencion", MessageBoxButtons.OK, MessageBoxIcon.Error);
        return;
    }
    var resulMessage = MessageBox.Show("Seguro que quiere eliminar a " + studentselect.Name + " " + studentselect.Surname, "Atencion", MessageBoxButtons.YesNo, MessageBoxIcon.Exclamation);
    if (resulMessage == DialogResult.No)
    {
        return;
    }
    ArrayLogic.Arraylog.RemoveStudent(studentselect.Carnet);
    dgPrint.DataSource = null;
    dgPrint.DataSource = ArrayLogic.Arraylog.GetStudents();
}
```

Cada vez que uses un método de una clase, en dado caso que al usar dicho método te muestre un error de objeto, una alternativa es crear un objeto de esa clase e instanciarlo.

```

1 referencia
private void button3_Click(object sender, EventArgs e)
{
    if (dgPrint.SelectedRows.Count == 0)
    {
        MessageBox.Show("No se seleccionado un estudiante", "Atencion", MessageBoxButtons.OK, MessageBoxIcon.Error);
        return;
    }
    var IndexSelectDg = dgPrint.SelectedRows[0].Index;
    if (dgPrint.Rows[IndexSelectDg].DataBoundItem is not Student studentselect)
    {
        MessageBox.Show("No se seleccionado un estudiante", "Atencion", MessageBoxButtons.OK, MessageBoxIcon.Error);
        return;
    }
    UpdateView updateView = new UpdateView(studentselect);
    updateView.OnDataUpdate += PrintStudents;
    updateView.ShowDialog();
}

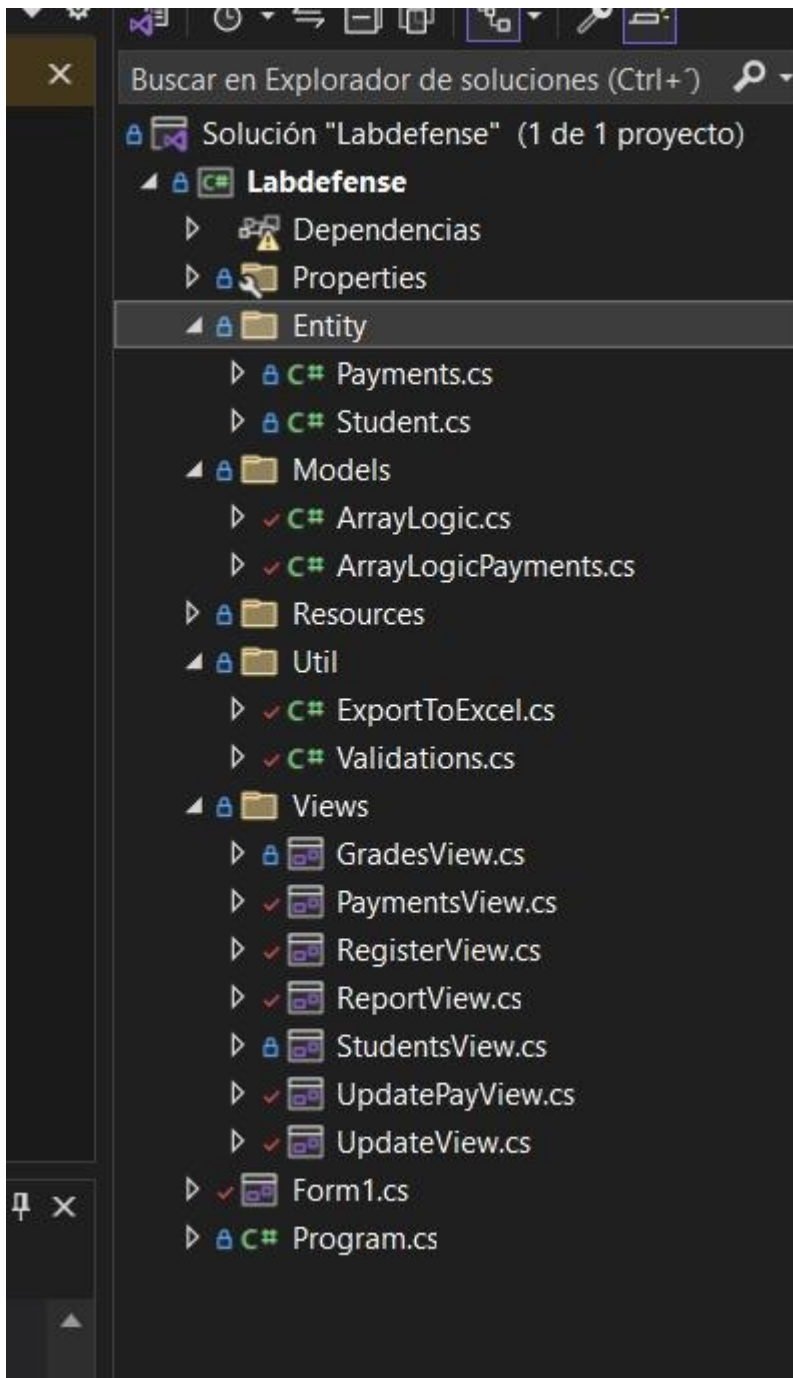
1 referencia
private void button1_Click(object sender, EventArgs e)
{
    var Carnet = textSearch.Text;
    var _students = ArrayLogic.Arraylog.SearchStudent(Carnet);


    if (_students.Length > 0)
    {
        dgPrint.DataSource = _students;
    }
    else
    {
        MessageBox.Show("No se encontro al estudiante con carnet:" + Carnet, "Atencion", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}

```

Uso del evento cerrar formulario para la extracción de datos, por consiguiente si es necesario buscar un estudiante por su carnet Se utiliza el siguiente botón que ocupa el método SearchStudent y manda el objetivo encontrado.

Resultado:





UNIVersidad Nacional de Ingenieros

REGISTER

GRADES

PAYMENTS

REPORTS

STUDENTS

/30 I Par

/30 II Par

/20 Proyecto

/20 Pruebas

Name	Surname	Carnet	IPar	IIPar	Project	Test	Finalgrades
Kelvin	Aleman	2023-08628	29	12	18	17	76
Kelvin	Aleman	2023-0862L	29	12	18	16	75
Erick Hazael	Morales Jarquin	2023-0862X	29	11	18	16	74
Erick Hazael	Morales Jarquin	2023-0862U	29	15	12	15	71



UNIVersidad Nacional de Ingenieros

REGISTER


GRADES

PAYMENTS

REPORTS

STUDENTS

Name	Surname	Carnet	DateRegister	Number	Identification	IPar	IIPar	Project	Test	Finalgrades
Kelvin	Aleman	2023-08628	9/9/2024 15:50	1112-1312	123-123456-123...	29	12	18	17	76
Kelvin	Aleman	2023-0862L	9/9/2024 15:50	1112-1312	123-123456-123...	29	12	18	16	75
Erick Hazael	Morales Jarquin	2023-0862X	9/9/2024 15:50	1112-1312	123-123456-123...	29	11	18	16	74
Erick Hazael	Morales Jarquin	2023-0862U	9/9/2024 15:50	1112-1312	123-123456-123...	29	15	12	15	71


UNIVERSIDAD NACIONAL DE INGENIERIA

REGISTER

GRADES

PAYMENTS

REPORTS

STUDENTS

Update

Nombre:

Identificacion:

/30 I Par

Apellido:

Numero:

/30 II Par


Carnet:

Fecha:

/30 Proyecto

/20 Pruebas

II Par	Project	Test	Finalgrades
12	18	17	76
12	18	16	75
11	18	16	74
15	12	15	71


UNIVERSIDAD NACIONAL DE INGENIERIA

REGISTER

GRADES

PAYMENTS

REPORTS

STUDENTS

Motivo

Matricula

Matricula

Aranceles

Biblioteca


Constancia de notas

Otros...

Carnet

Date

	Name	Surname	Carnet	Identification	Motive	Cordobas	DatePay	CantPayments
	Erick Hazael	Morales Jarq...	2023-0862U	123-123456-...	Matricula	143	9/9/2024 15:...	1
	Erick Hazael	Morales Jarq...	2023-0862U	123-123456-...	Matricula	143	9/9/2024 15:...	2


UNIVERSIDAD NACIONAL DE INGENIERIA

REGISTER

GRADES

PAYMENTS

REPORTS

STUDENTS

29

/30 I Par

12

/30 II Par


18

/20 Proyecto

17

/20 Pruebas

	Name	Surname	Carnet	IPar	IIPar	Project	Test	Finalgrades
	Kelvin	Aleman	2023-0862B	29	12	18	17	76
	Kelvin	Aleman	2023-0862L	29	12	18	16	75
	Erick Hazael	Morales Jar...	2023-0862X	29	11	18	16	74
	Erick Hazael	Morales Jar...	2023-0862U	29	15	12	15	71


UNIVERSIDAD NACIONAL DE INGENIERIA

REGISTER


GRADES

PAYMENTS

REPORTS

STUDENTS

	Name	Surname	Carnet	Identification	Motive	Cordobas	DatePay	ContPayments
	Erick Hazael	Morales Jarquin	2023-0862U	123-123456-1234Q	Matricula	143	9/9/2024 15:52	1
	Erick Hazael	Morales Jarquin	2023-0862U	123-123456-1234Q	Matricula	143	9/9/2024 15:52	2
	Erick Hazael	Morales Jarquin	2023-0862U	123-123456-1234Q	Aranceles	1443	9/9/2024 15:52	3
	Erick Hazael	Morales Jarquin	2023-0862U	123-123456-1234Q	Otras...	1443	9/9/2024 15:52	4


Universidad Nacional de Ingeniería

REGISTER

GRADES

PAYMENTS

REPORTS

STUDENTS


Motivo:

Carnet:

Date:

lunes , 9 de septiembre de 2024

	Name	Surname	Carnet	Identification	Motive	Cordobas	DatePay	CantPayments
+	Erick Hazael	Morales Jarquin	2023-0862U	123-123456-1234Q	Matricula	143	9/9/2024 15:52	1
	Erick Hazael	Morales Jarquin	2023-0862U	123-123456-1234Q	Matricula	143	9/9/2024 15:52	2
	Erick Hazael	Morales Jarquin	2023-0862U	123-123456-1234Q	Aranceles	1443	9/9/2024 15:52	3
	Erick Hazael	Morales Jarquin	2023-0862U	123-123456-1234Q	Otras...	1443	9/9/2024 15:52	4


Universidad Nacional de Ingeniería

REGISTER

GRADES

PAYMENTS

REPORTS

STUDENTS

Name:

Carnet:

Surname:

Identification:

Number:

Date:

lunes , 9 de septiembre de 2024

	Name	Surname	Carnet	DateRegister	Number	Identification
+	Kelvin	Aleman	2023-0862B	9/9/2024 15:50	1112-1312	123-123456-1234J
	Kelvin	Aleman	2023-0862L	9/9/2024 15:50	1112-1312	123-123456-1234M
	Erick Hazael	Morales Jarquin	2023-0862X	9/9/2024 15:50	1112-1312	123-123456-1234W
	Erick Hazael	Morales Jarquin	2023-0862U	9/9/2024 15:50	1112-1312	123-123456-1234Q

