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 #2: Registro de Feligreses y sus diezmos en un rango de meses

1-Clase objeto de tipo feligrense y sus atributos

```
v using System;
using System.Collections.Generic;
       using System.Ling;
       using System.Text;
       using System. Threading. Tasks;
     v namespace AEDLaboratorio2.Entities.Church
        {
            15 referencias
            public class Parishioner
                public int ID { get; set; }
                6 referencias
                public string? Name { get; set; }
                public string? Address { get; set; }
                public string? PhoneNumber { get; set; }
                3 referencias
                public string? PeriodStart { get; set; }
                3 referencias
                public string? PeriodEnd { get; set; }
                public decimal Amount { get; set; }
                public DateTime DateRegister { get; set; }
       * Entidades en la cual estaremos trabajando durante el manejo del arreglo de objetos
22
```

```
public class Validations
   private ErrorProvider errorProvider = new ErrorProvider();
    18 referencias
    public bool TextNullOrEmpty(Control control)
        errorProvider.SetError(control, "");
        if (control is ComboBox comboBox)
            if (comboBox.SelectedIndex == -1)
                errorProvider.SetError(comboBox, "Cannot be null or empty");
                return false;
        if (control is TextBox textBox)
            if (string.IsNullOrWhiteSpace(textBox.Text))
                errorProvider.SetError(textBox, "Cannot be null or empty");
                return false;
        if (control is MaskedTextBox maskedTextBox)
            if (!maskedTextBox.MaskFull)
                errorProvider.SetError(maskedTextBox, "Cannot be null or empty");
                return false;
       return true;
```

2-En formulario de ParishionerView se crean los metodos a utilizar

```
public partial class ParishionerView : Form
    private readonly Validations validations;
   private Parishioner? _parishioner;
     * Instancia del objeto que quardaremos en el arreglo
    public ParishionerView()
        InitializeComponent();
        validations = new Validations();
        _parishioner = new Parishioner();
        //Pasamos las instancias en el constructor para su uso
    int autoIncrementID = 0;//variable autoincrementable para tener el conteo de registros y mostrarlos en el datagridview y su previa busueda
    private void PrintParishioners()
        dgPrintParishioner.DataSource = null;
        dgPrintParishioner.DataSource = ParishionerModel.arrayLogic.GetParisioner();//metodo para actualilzar el datagriview en funcion del arreglo
   private void SaveParishioner()
        _parishioner = new Parishioner
            ID = ++autoIncrementID,
            Name = TxtName.Text,
Address = TxtAddress.Text,
           PhoneNumber = TxtMaskPhone.Text,
Amount = decimal.Parse(textCant.Text),
            PeriodStart = CmbPStart.Text,
            PeriodEnd = cmbPEnd.Text,
            DateRegister = DateTime.Now
        ParishionerModel.arrayLogic.AddParishioner(_parishioner);//metodo para guardar a los feligrenses inicializando un nuevo feligrense y usadndo el metodo en models
```

```
private void UpdateParishioner(Parishioner parishioner)
    var _parishionerUpdate = new Parishioner
        ID = parishioner.ID,
       Name = TxtName.Text,
Address = TxtAddress.Text,
        PhoneNumber = TxtMaskPhone.Text,
        Amount = decimal.Parse(textCant.Text),
       PeriodStart = CmbPStart.Text,
        PeriodEnd = cmbPEnd.Text,
        DateRegister = DateTime.Now
   };
ParishionerModel.arrayLogic.UpdateParishioner(_parishionerUpdate);
public void ValidationsTools()
   bool isValid = true;
    isValid &= validations.TextNullOrEmpty(TxtAddress);
    isValid &= validations.TextNullOrEmpty(TxtName);
    isValid &= validations.TextNullOrEmpty(TxtMaskPhone);
    isValid &= validations.TextNullOrEmpty(textCant);
    isValid &= validations.TextNullOrEmpty(CmbPStart);
    isValid &= validations.TextNullOrEmpty(cmbPEnd);
    if (!isValid)
        MessageBox.Show("Information null or empty", "Error", MessageBoxButtons.OK, MessageBoxIcon.Warning);
 * Validacion para los campos de texto nulos o invalidos, &= si el metodo retorna false, el booleano isvalid se vuelve false y salta la alerta
```

3-Crear eventos de botones y formulario para el funcionamiento de la aplicacion

```
# referencies

private void CleanBox()

{
    TxtName.Text = textCant.Text = TxtMaskPhone.Text = TxtAddress.Text = string.Empty;
    CmbPStart.SelectedIndex = -1;
    cmbPEnd.SelectedIndex = -1;
    //funcionpara limpiar los campos de texto
}

private void ParishionerView_Load(object sender, EventArgs e)
{
    PrintParishioners();//uso de una funcion
}

totalencies

private void BtnSave_Click(object sender, EventArgs e)
{
    ValidationsTools();//aplicacion del metodo validationstools y si uno falla no permite avanzar

if (!validations.TextNullOrEmpty(TxtAddress) || !validations.TextNullOrEmpty(TxtName) ||
    ivalidations.TextNullOrEmpty(TxtAddress) || !validations.TextNullOrEmpty(textCant) ||
    ivalidations.TextNullOrEmpty(CmbPStart) || !validations.TextNullOrEmpty(cmbPEnd))
{
    return; // No proceder si hay errores de validación
}

SaveParishioners();//uso de funciones
CleanBox();
}
```

```
private void BtnClearForm_Click(object sender, EventArgs e)
{
    CleanBox();//uso de funciones
}

!referencia
private void BtnPrevious_Click(object sender, EventArgs e)
{
    if (dgPrintParishioner.CurrentRow == null) return;
    var currentIndex = dgPrintParishioner.CurrentRow.Index;
    if (currentIndex > 0)
    {
        dgPrintParishioner.ClearSelection();
        dgPrintParishioner.Rows[currentIndex - 1].Selected = true;
        dgPrintParishioner.CurrentCell = dgPrintParishioner.Rows[currentIndex - 1].Cells[0];
    }
}

* Verifica si el indice es nulo y si el indice actual es mayor a 0, si es mayor a 0 retrocede una celda en el data gridview
*/
```

```
private void BtnDelete_Click(object sender, EventArgs e)
   if (dgPrintParishioner.SelectedRows.Count == \theta)
       MessageBox.Show("No row selected. Please select a row to update.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
       return;
   var parishionerSelected = dgPrintParishioner.SelectedRows[θ].DataBoundItem as Parishioner;
   if (parishionerSelected == null)
       MessageBox.Show("Option selected doesn't exist or is invalid.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
    var dlgTest = MessageBox.Shom("Are you sure delete: " + parishionerSelected.Name + " with ID : " + parishionerSelected.ID, "Error", MessageBoxButtons.YesNo, MessageBoxIcon.Question);
   if (dlgTest == DialogResult.No) return;
   var proveDelete = ParishionerModel.arrayLogic.DeleteParishioner(parishionerSelected.ID);
   if (!proveDelete)
       MessageBox.Show("Cannot delete parishioner: " + parishionerSelected.Name + " with ID : " + parishionerSelected.ID + "?", "Alert", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
   btnSaveUpdate.Visible = false;
   BtnSave.Visible = true;
* Si no se ha seleccionado una celda muestra el mensaje de error, si no se obtiene la informacion de esa celda de tipo objeto pariishioner respectivamente se usa su id para buscar en el arreglo el objeto con ese id
private void BtnShowAll_Click(object sender, EventArgs e)
   PrintParishioners();//uso de funciones
```

```
private void btnSaveUpdate_Click(object sender, EventArgs e)
     if (dgPrintParishioner.SelectedRows.Count == \theta)
         MessageBox.Show("No row selected. Please select a row to update.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
         return:
    var parishionerSelected = dgPrintParishioner.SelectedRows[0].DataBoundItem as Parishioner;
if (parishionerSelected == null)
         MessageBox.Show("Option selected doesn't exist or is invalid.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
    BtnPrevious.Enabled = true;
    BtnNext.Enabled = true;
btnSaveUpdate.Visible = false;
     BtnSave.Visible = true;
     ValidationsTools();
    if (!validations.TextNullOrEmpty(TxtAddress) || !validations.TextNullOrEmpty(TxtName) ||
    !validations.TextNullOrEmpty(TxtMaskPhone) || !validations.TextNullOrEmpty(textCant) ||
    !validations.TextNullOrEmpty(CmbPStart) || !validations.TextNullOrEmpty(cmbPEnd))
    UpdateParishioner(parishionerSelected):
    CleanBox();
     MessageBox.Show("Parishioner update sucessfully", "Success", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
 * Se vuelve a validar si la celda seleccionada es valida y se manda a llamar el metodo actualizar, ahora con la informacion actualizada en los campos de text,

* (Antes se habian rellenado los campos de texto con los datos del feligrense ubicado por su id para su respectiva edicion) se envia el id del feligrense y el metodo hace su accion
    private void BtnUpdate_Click(object sender, EventArgs e)
        if (dgPrintParishioner.SelectedRows.Count == θ)
             MessageBox.Show("No row selected. Please select a row to update.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
             return;
        var parishionerSelected = dgPrintParishioner.SelectedRows[\theta].DataBoundItem as Parishioner;
             MessageBox.Show("Option selected doesn't exist or is invalid.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
             return;
        var dlgTest = MessageBox.Show($"Are you sure you want to update: {parishionerSelected.Name} with ID: {parishionerSelected.ID}?", "Alert", MessageBoxButtons.YesNo, MessageBoxIcon.Question);
if (dlgTest == DialogResult.No) return;
        TxtName.Text = parishionerSelected.Name;
        TxtAddress.Text = parishionerSelected.Address;
        TxtMaskPhone.Text = parishionerSelected.PhoneNumber;
dateTimeRegister.Value = parishionerSelected.DateRegister;
        CmbPStart.Text = parishionerSelected.PeriodStart;
        cmbPEnd.Text = parishionerSelected.PeriodEnd;
        textCant.Text = parishionerSelected.Amount.ToString();
        BtnPrevious.Enabled = false;
        BtnNext.Enabled = false;
        btnSaveUpdate.Visible = true:
        BtnSave.Visible = false;
```

```
private void BtnSearchById_Click(object sender, EventArgs e) {
     int idParishioner = (int)TxtNumSearchID.Value;
var specificParioshioner = ParishionerModel.arrayLogic.SearchParishioner(idParishioner);
if (specificParioshioner == null)
         MessageBox.Show("ID Doesn't Exist", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error); return;
    dgPrintParishioner.DataSource = new Parishioner[] { specificParioshioner };
/*

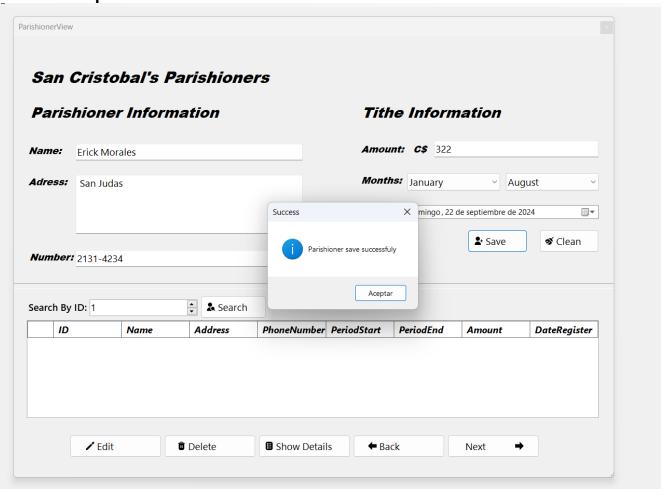
* Se convierte a entero el valor recibido de numericuptodown debido que se requiere buscar el feligrense con ese id e imprimirlo en el datagridview, si no se

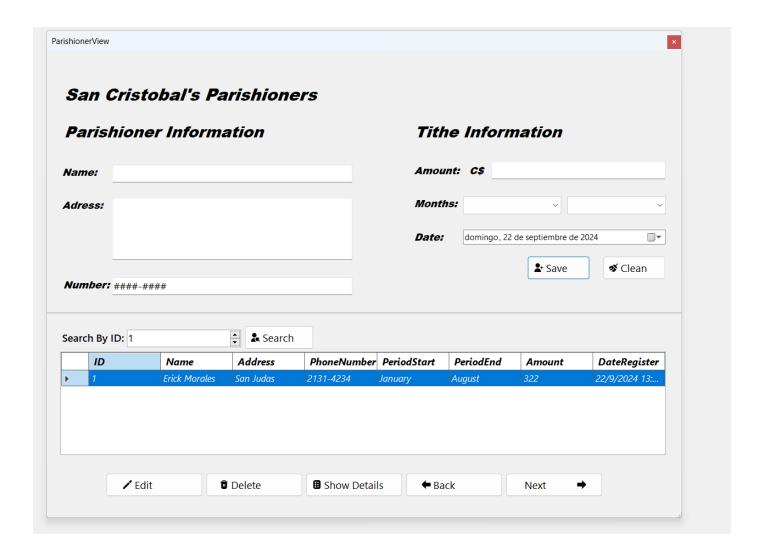
* encuentra se envia un mensaje de error
private void textCant_KeyPress(object sender, KeyPressEventArgs e) {
    char index = e.KeyChar;
if (index == 8)
         e.Handled = false;//permite backspace
    if (!char.IsDigit(index))
{
         e.Handled = true;//Permite solo numeros
         return;
private void TxtName_KeyPress(object sender, KeyPressEventArgs e) {
     char index = e.KeyChar;
if (index == 8)
         e.Handled = false;//permite backspace
     if (char.IsDigit(index))
         e.Handled = true;//permite solo letras
```

4-Creacion de la carpeta models con los procesos para guardar, listar, eliminar y actualizar en el orden de mayor monto de diezmo a menor monto de diezmo.

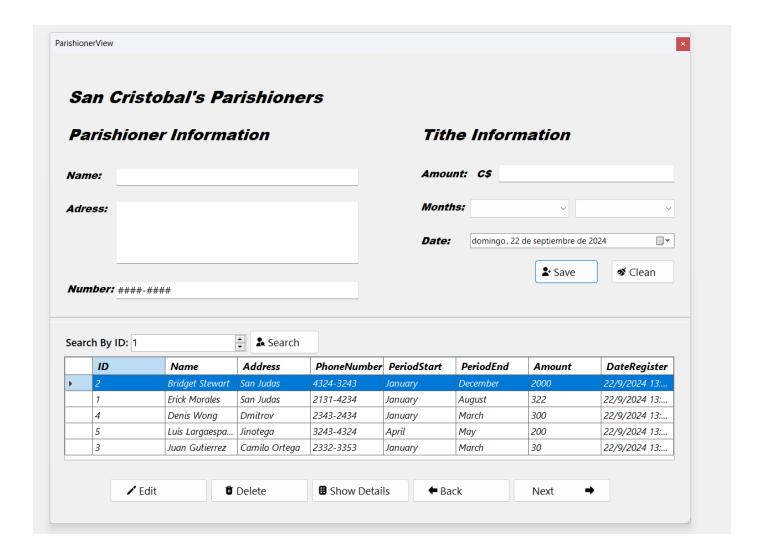
```
1 meferencia
public Parishioner[] GetParisioner() { return parishionerArray; }//devuelve el arreglo de objetos para su visualizacion
     public bool DeleteParishioner(int id)//usando el id se ubica al objeto que se desea eliminar
         var index = Array.FindIndex(parishionerArray, p \Rightarrow p.ID.Equals(id) == true); if (index < 0) return false;
         for (int i = index; i < Quantity - 1; i++)
            parishionerArray[i] = parishionerArray[i + 1];//desplaza todos los objetos hacia la izquieda menos al deseado dejandolo de ultimo a la derecha y se disminuye el tamanio para desechar el objeto elimminado
         --Quantity;
Array.Resize(ref parishionerArray, Quantity);
         return Array.Find(parishionerArray, p => p.ID.Equals(id));//metodo para filtrar un feligrense por su id
          public bool UpdateParishioner(Parishioner? parishioner)
              if (parishioner == null) return false;
var index = Array,FindIndex(parishionerArray, p => p.ID.Equals(parishioner?.ID) == true);//Recibe el nuevo objeto con la informacion actualizada y se guarda en la misma posicion if (index < 0) return false;
parishionerArray[index] = parishioner;
public static readonly ParishionerModel InstanceParishioner = newQ;//Se crea una instancia unica, mas que todo para evitar errores a futuro y asegurar un mejor manejo de la clase
public static ParishionerModel arraylogic { get { return InstanceParishioner; } }//se vuelve publica la instancia public Parishioner[] parishionerArray;//arreglo donde se guardaran los objetos int Size = 0, Quantity = 0;//Cantidad de registros y tamanio del arreglo
public ParishionerModel()
   parishionerArray = new Parishioner[Size]://se le asigna el tamanio al arreglo
public void AddParishioner(Parishioner parishioner)//Se verifica si el tamanio es menor a la cantidad de registro, de caso contrario se redomensiona el arreglo y se almeneca el objeto y la cantidad aumenta
   if (Size <= Quantity)
       Size = Size == 0 ? 1 : Size + 1;
Array.Resize(ref parishionerArray, Size);
```

5-Diseño de la aplicaicion con validaciones de entrada.

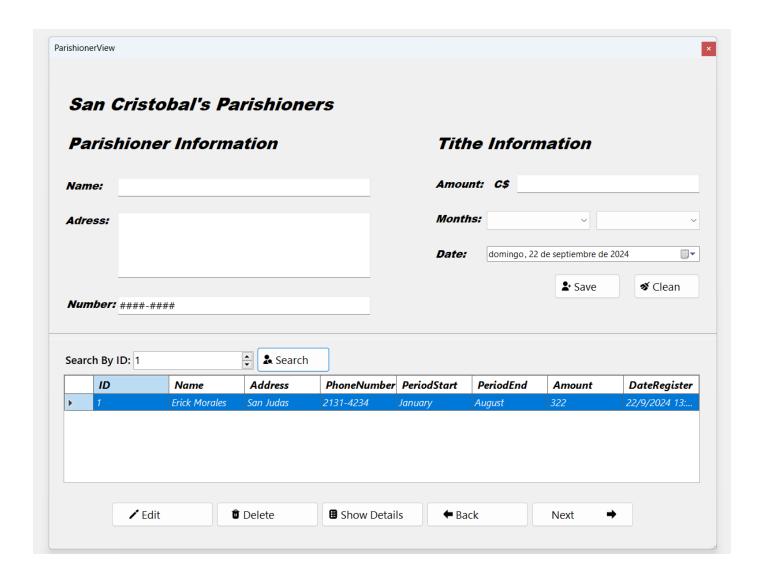




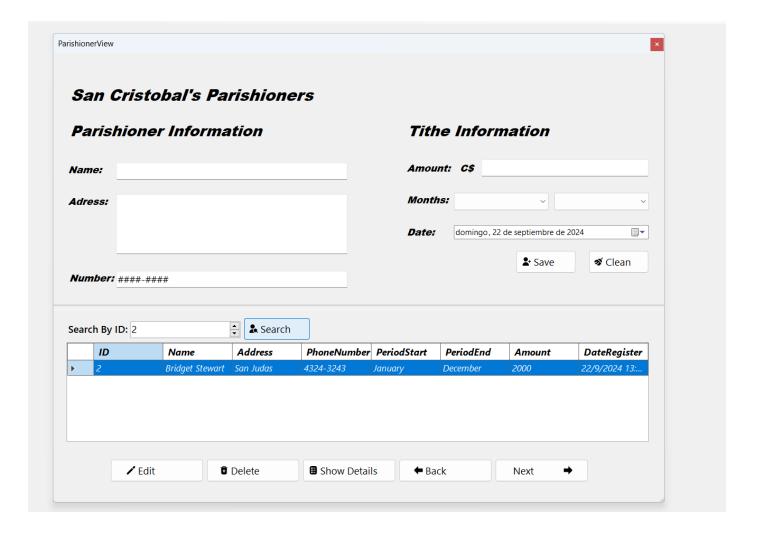
7-Usuario guardado si el proceso es valido

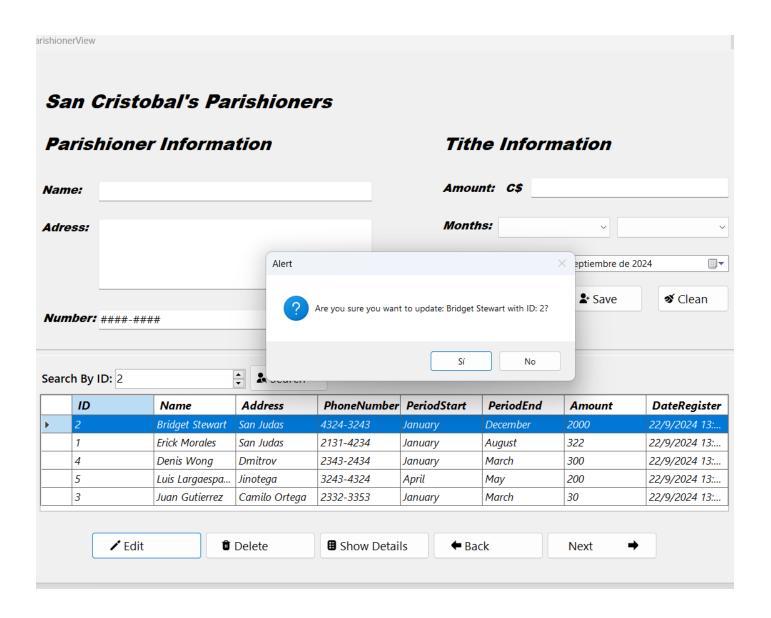


8-Guardar a los feligrenses en orden de mayor monto a menor

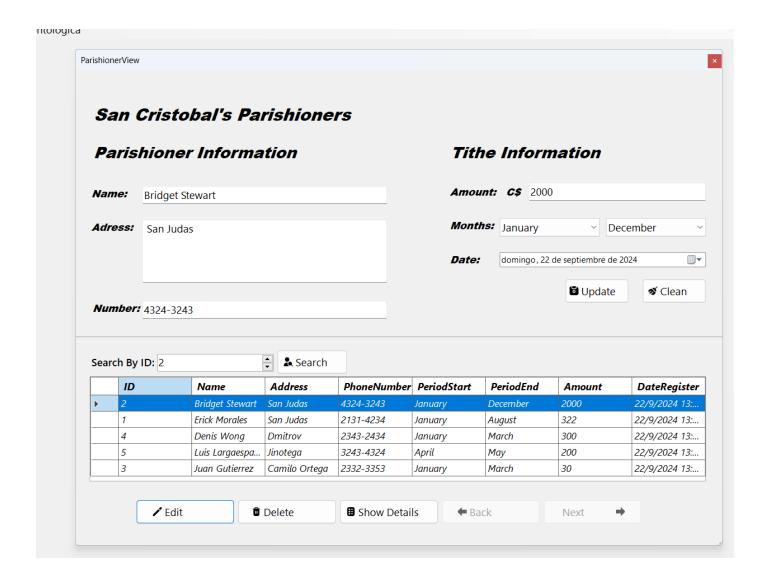


9-Listar el feligrense que se ha buscado por su ID

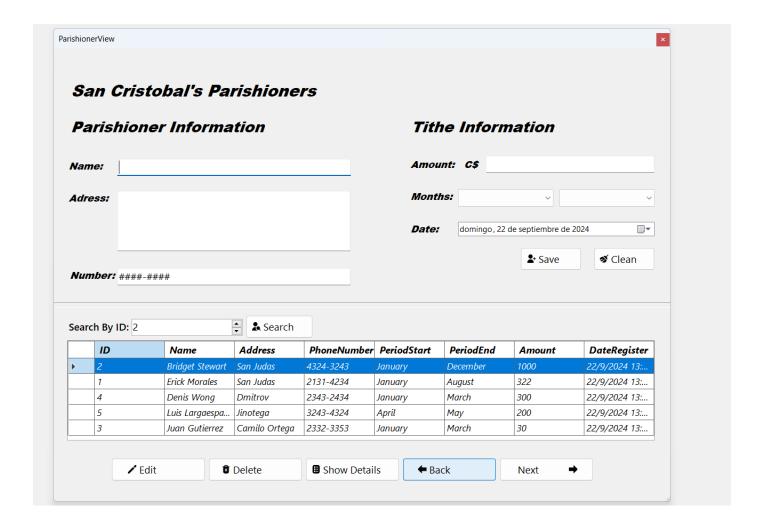




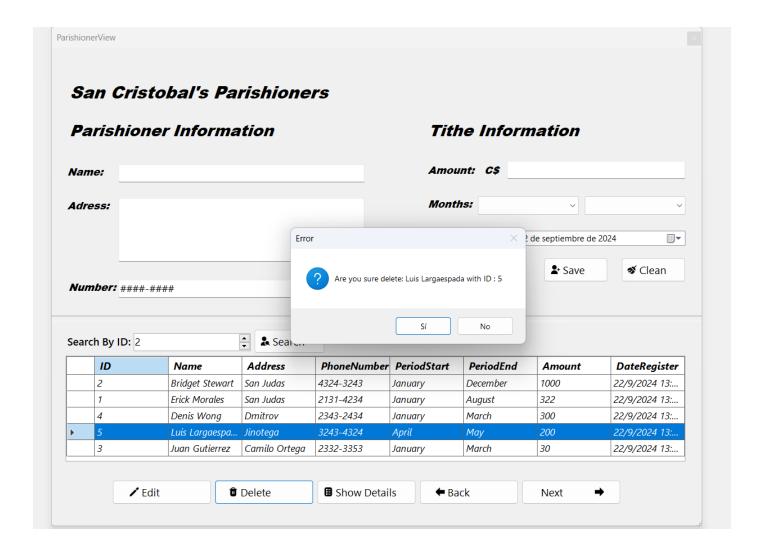
9-Actualizar seleccionando la celda



10-Se rellenan los campos con la información del feligrese seleccionado



11- Se actualiza los datos, en este caso el monto



12-Se elimina el feligrés seleccionado

