UNIVERSIDAD POLITÉCNICA INTERNACIONAL

ESCUELA DE INGENIERÍA INFORMÁTICA

Técnicas de Programación

Proyecto #1

SplitBuddies

Trabajo elaborado por

Daniel González Quirós

Kendal Meza Sánchez

Profesor: Luis Felipe Mora Umaña

Centro Universitario Heredia

30 de julio del 2025

Índice

1. Introducción

Mediante este entregable del proyecto, se tiene como objetivo presentar el producto ya hecho en base a lo que se ordenó en el PDF de las instrucciones del cliente. El cual se trata de un software desarrollada en C#, en el que permite a usuarios llevar el control de gastos en conjunto.

Se explicará la estructura de este, así mismo, de sus PBIs dentro de cada epic. Incluso, se confeccionará una explicación en cada línea del código. O si no, de la mayoría de las líneas del código.

2. Desarrollo

En el código se crean 6 FRM, los cuales son Grupo, Login, Gastos, ControlDeGatos, Principal y Usuario.

FRMGrupo:

En este apartado se crea las líneas de importaciones, las cuales nos van a ayudar a usar funcionalidades básicas cosas fechas, por ejemplo, además nos permite conectar los comandos de la base de datos, en este caso SQL, permitiendo trabajar con archivos y manejar imágenes.

- using System;
- using System.Data;
- using System.Data.SqlClient;
- using System.Drawing;
- using System.IO;
- using System.Windows.Forms;

Estas son las lineas de importaciones utilizadas para este frm.

Luego, se declarael formulario utilizando

namespace SplitBuddies.Presentation

{

public partial class FrmGrupo : Form

Se busca agrupar el formulario dentro del proyecto, teniendo una cadena de connexion con private readonly string _conn = "Server=.;Database=SplitBuddies;Trusted_Connection=True;";

Para cargar los grupos desde la base utilizamos el:

private void CargarGrupos()

```
{
    var dt = new DataTable();
    using (var da = new SqlDataAdapter("SELECT IdGrupo, Nombre FROM Grupo", _conn))
    {
        da.Fill(dt);
    }
    dgvGrupos.DataSource = dt;
}
```

En el que carga los grupos existentes de la base de datos, se crean las tablas vacias para los proximos datos añadidos y se llena la tabla con los datos que trajo el sqldataadapter.

En el programa pusimos que al crear el grupo sea un requisito poner la imagen de este, por lo que se elige la imagen desde el equipo

```
private void btnSeleccionarImagen_Click(object sender, EventArgs e)
{
    if (openFileDialog1.ShowDialog() != DialogResult.OK) return;
    pbImagen.Image = Image.FromFile(openFileDialog1.FileName);
}
```

Donde se ejecuta el metodo al darle click al elegir imagen, donde si se selecciona una la carga el picturebox pblmagen

A la hora de agregar el grupo con su respectiva imagen y nombre utilizamos:

```
private void btnAgregar_Click(object sender, EventArgs e)
{
    byte[] imp = null;
    if (pbImagen.Image != null)
    {
        using (var ms = new MemoryStream())
        {
            pbImagen.Image.Save(ms, pbImagen.Image.RawFormat);
            img = ms.ToArray();
        }
```

```
    }


    using (var cn = new SqlConnection(_conn))
    using (var cmd = new SqlCommand("INSERT INTO Grupo (Nombre, Imagen) VALUES
(@n, @i)", cn))
    {
        cmd.Parameters.AddWithValue("@n", txtNombre.Text);
        cmd.Parameters.AddWithValue("@i", (object)img ?? DBNull.Value);
        cn.Open();
        cmd.ExecuteNonQuery();
    }


    CargarGrupos();
    txtNombre.Clear();
    pbImagen.Image = null;
}
```

Esto se ejecuta al darle al botón agregar, se va a declarar la variable que guardara la imagen en forma de un arreglo, donde si el usuario selecciona una imagen se convierte en bytes, por lo que va al arreglo convertido en bytes dentro del using, ademas se crea una memoria para guardar la imagen, guardando la imagen en jpg y png.

Para guardar en la base de datos se usa la linea de codigo

using (var cn = new SqlConnection(_conn))

Usando el comando using (var cmd = new SqlCommand("INSERT INTO Grupo (Nombre, Imagen) VALUES (@n, @i)", cn)) se inserta un nuevo grupo con su nombre e imagen, luego abrimos la conexión a la base de datos utilizando cn.Open para poder ejecutar el INSERT agregando el grupo a la base.

FRMLogin

Para lo que es el login agrupamos el código como en una carpeta, más conocida como una carpeta lógica.

Se declara el formulario que hereda de form, esto con el comando public partial class FrmGrupo : Form

Cadena de conexión:

Aquí es donde se guarda la cadena de conexión a la base de datos splitBuddies la cual se usa en el servidor actual

Constructor del formulario

```
public FrmGrupo()
{
    InitializeComponent();
    CargarGrupos();
}
```

El constructor se ejecutará al abirir la ventana, cargando los textbox y botones en el frm.

La función para cargar los grupos existentes se utilizo

CopiarEditar

```
private void CargarGrupos()
{
    var dt = new DataTable();
    using (var da = new SqlDataAdapter("SELECT IdGrupo, Nombre FROM Grupo", _conn))
    {
        da.Fill(dt);
    }
    dgvGrupos.DataSource = dt;
}
```

Se traen todos los grupos de la base de datos, luego se crea una tabla para guardar los datos de los grupos y se usa finalmente el DataGridView dgvGrupos para mostrar los datos de manera visual

FrmGasto

Para definir la clase del frm que hereda

```
namespace SplitBuddies.Presentation
```

```
{
    public partial class FrmGasto : Form
```

Constructor del formulario

Se cargan todos los cnroles visuales, llamando a la funcion comboboxcon los grupos y usuarios.

```
public FrmGasto()
{
    InitializeComponent();
    CargarGrupos();
    CargarUsuarios();
    cmbGrupo.SelectionChangeCommitted += cmbGrupo_SelectionChangeCommitted;
}
```

Para crear una datatable y llenar los datos desde la tabla grupo con su id se utilizó el comando cmbGrupo.DataSource = dt;

Para observar el nombre en pantalla se utilizó DisplayMember = "Nombre"

Función CargarUsuarios()

```
private void CargarUsuarios()
{
    var dt = new DataTable();
    using (var da = new SqlDataAdapter("SELECT IdUsuario, Nombre FROM Usuario", _conn))
    {
        da.Fill(dt);
    }
    cmbUsuario.DataSource = dt;
    cmbUsuario.DisplayMember = "Nombre";
    cmbUsuario.ValueMember = "IdUsuario";
    cmbUsuario.SelectedIndex = -1;
}
```

Hace lo mismo que CargarGrupos, pero para el ComboBox cmbUsuario.

Actualizar usuarios según grupo seleccionado

Cuando el usuario selecciona un grupo, esta función actualiza los usuarios del ComboBox según ese grupo.
Si no hay grupo seleccionado (SelectedIndex < 0), se borra el contenido del ComboBox cmbUsuario.
Si hay un grupo, se obtiene su Id y se hace una consulta para traer solo los usuarios de ese grupo.
Luego, se muestra esa lista en cmbUsuario.

Botón para agregar un gasto

private void btnAgregar_Click(object sender, EventArgs e)

Esta función se ejecuta cuando se hace clic en el botón "Agregar gasto".

Recolectar información

```
decimal total = Convert.ToDecimal(txtMonto.Text);

int grupo = (int)cmbGrupo.SelectedValue;

int pagador = (int)cmbUsuario.SelectedValue;

string nombre = txtNombre.Text;

string desc = txtDescripcion.Text;
```

Obtiene los datos ingresados en los controles:
El monto del gasto (txtMonto)
El grupo seleccionado (cmbGrupo)
El usuario que pagó (cmbUsuario)
El nombre y la descripción del gasto.

Insertar el gasto en la base de datos

```
using (var cn = new SqlConnection(_conn))

{

   cn.Open();

   using (var tran = cn.BeginTransaction())

   {
```

```
        // 1. Insertar Gasto
        int idGasto;
        using (var cmd = new SqlCommand(
            "INSERT INTO Gasto (Nombre, Descripcion, Enlace, MontoTotal, IdUsuarioPago,
IdGrupo) " +
            "VALUES (@n, @d, @e, @m, @p, @g); SELECT SCOPE_IDENTITY()",
            cn, tran))
        {
            cmd.Parameters.AddWithValue("@n", nombre);
            cmd.Parameters.AddWithValue("@d", desc);
            cmd.Parameters.AddWithValue("@e", DBNull.Value);
            cmd.Parameters.AddWithValue("@m", total);
            cmd.Parameters.AddWithValue("@p", pagador);
            cmd.Parameters.AddWithValue("@g", grupo);
            idGasto = Convert.ToInt32(cmd.ExecuteScalar());
        }

        tran.Commit();
    }
}
```

Se abre la conexión a la base de datos.
Se inicia una transacción para asegurar que todo el proceso sea exitoso o se cancele todo.
Se crea el INSERT en la tabla Gasto y se insertan los valores recibidos.

Mensaje de éxito

MessageBox.Show("Gasto registrado con éxito.");

Muestra una ventana emergente avisando que el gasto fue agregado correctamente.

FrmControlGastos

Importación de librerías

```
using System;

using System.Data;

using System.Data.SqlClient;

using System.Windows.Forms;
```

Declaración de la clase y conexión

```
namespace PtoyectoDanielKendall

{

   public partial class FrmControlGastos : Form

   {

      private readonly string _conn =
"Server=.;Database=SplitBuddies;Trusted_Connection=True;";
```

Se define un formulario llamado FrmControlGastos dentro del proyecto PtoyectoDanielKendall. Form es la clase base que representa un formulario de Windows.

Constructor del formulario

InitializeComponent();: Inicializa todos los elementos visuales.

CargarGrupos();: Llama a una función para llenar el combo de grupos (cmbGrupo) desde la base de datos.

Cargar los grupos en el ComboBox

Se crea una tabla en memoria (dt).
SqlDataAdapter ejecuta la consulta SQL que trae todos los grupos con su ID y Nombre.
Con el comando da.Fill(dt);: llena la tabla dt con esos datos.

```
   cmbGrupo.DataSource = dt;

   cmbGrupo.DisplayMember = "Nombre";

   cmbGrupo.ValueMember = "IdGrupo";

   cmbGrupo.SelectedIndex = -1;

}
```

DisplayMember = "Nombre": el nombre del grupo que verá el usuario.

ValueMember = "IdGrupo": el valor que se usará internamente.
SelectedIndex = -1: para que no haya ningún grupo seleccionado al principio

5. Evento al seleccionar un grupo

```
private void cmbGrupo_SelectionChangeCommitted(object sender, EventArgs e)
{
    if (cmbGrupo.SelectedIndex < 0)
    {
        dgvGastos.DataSource = null;
        lblTotal.Text = "Total: ₡0.00";
        return;
    }
```

Este método se ejecuta cuando el usuario cambia la selección de grupo.

Si no se selecciona ningún grupo, se limpian

6. Obtener y mostrar gastos del grupo

```
int grupoId = (int)cmbGrupo.SelectedValue;
var dt = new DataTable();
using (var da = new SqlDataAdapter(
    "SELECT * FROM Gasto WHERE IdGrupo = @g",
    _conn))
{
    da.SelectCommand.Parameters.AddWithValue("@g", grupoId);
    da.Fill(dt);
}
```

Se toma el IdGrupo seleccionado en el ComboBox.
Se ejecuta una consulta SQL que obtiene todos los gastos de ese grupo.
Se llenan los datos en la tabla dt.

Mostrar gastos en el formulario

```
dgvGastos.DataSource = dt;
```

8. Calcular el total de los gastos

```
decimal suma = 0m;

foreach (DataRow row in dt.Rows)

{

   if (row["MontoTotal"] != DBNull.Value)

      suma += Convert.ToDecimal(row["MontoTotal"]);

}
```

Se crea una variable para llevar el total de los gastos.
Se recorre cada fila del DataTable.
Si el valor de MontoTotal no está vacío, se convierte a decimal y se suma.

frmPrincipal

Definición de la clase principal
Esta clase frmPrincipal hereda de Form, lo que la hace un formulario Windows.
childFormNumber: se usa para numerar formularios hijos que se creen dinámicamente.

```
private void ShowNewForm(object sender, EventArgs e)

{

   Form childForm = new Form();

   childForm.MdiParent = this;

   childForm.Text = "Ventana " + childFormNumber++;

   childForm.Show();

}
```

Crea una nueva ventana vacía dentro del formulario principal MDI.

Guardar como

```
private void SaveAsToolStripMenuItem_Click(object sender, EventArgs e)
{
   SaveFileDialog saveFileDialog = new SaveFileDialog();
   saveFileDialog.InitialDirectory =
Environment.GetFolderPath(Environment.SpecialFolder.Personal);
   saveFileDialog.Filter = "Archivos de texto (.txt)|.txt|Todos los archivos (.)|.";
   if (saveFileDialog.ShowDialog(this) == DialogResult.OK)
   {
      string FileName = saveFileDialog.FileName;
   }
}
```

5. Opciones de edición

```
private void CutToolStripMenuItem_Click(object sender, EventArgs e) {}
private void CopyToolStripMenuItem_Click(object sender, EventArgs e) {}
private void PasteToolStripMenuItem_Click(object sender, EventArgs e) {}
```

Están definidos, pero no implementados.

6. Organización de ventanas MDI

```
private void CascadeToolStripMenuItem_Click(object sender, EventArgs e)
{
```

Organiza formularios hijos en cascada.

Organiza formularios hijos en cascada.

```
private void TileVerticalToolStripMenuItem_Click(object sender, EventArgs e)
{
   LayoutMdi(MdiLayout.TileVertical);
}

private void TileHorizontalToolStripMenuItem_Click(object sender, EventArgs e)
{
   LayoutMdi(MdiLayout.TileHorizontal);
}

private void ArrangeIconsToolStripMenuItem_Click(object sender, EventArgs e)
{
   LayoutMdi(MdiLayout.ArrangeIcons);
}
private void sodexoToolStripMenuItem_Click(object sender, EventArgs e)
{
   FrmUsuario frm = new FrmUsuario();
   frm.MdiParent = this;
   frm.Show();
}

private void sodexoToolStripMenuItem1_Click(object sender, EventArgs e)
{
   FrmGrupo frm = new FrmGrupo();
   frm.MdiParent = this;
   frm.Show();
```

```
}

private void porClaveNumericaToolStripMenuItem_Click(object sender, EventArgs e)
{
    FrmGasto frm = new FrmGasto();
    frm.MdiParent = this;
    frm.Show();
}

private void frmControlGastosToolStripMenuItem_Click(object sender, EventArgs e)
{
    FrmControlGastos frm = new FrmControlGastos();
    frm.MdiParent = this;
    frm.Show();
}
```

Cada uno de estos métodos abre un formulario específico relacionado con el control de gastos del sistema.

frmUsuario

Este formulario permite:

- Cargar y mostrar la lista de usuarios.
- Cargar grupos desde la base de datos en un ComboBox.
- Registrar un nuevo usuario asociado a un grupo.

Conexión

```
private readonly string _conn = "Server=.;Database=SplitBuddies;Trusted_Connection=True;";
```

Define la cadena de conexión a la base de datos local SplitBuddies.

Cuando se abre el formulario:

- Se cargan los usuarios en el DataGridView (dgvUsuarios).
- Se llena el ComboBox (cmbGrupo) con los grupos disponibles.

CargarUsuarios()

```
private void CargarUsuarios()
{
    var dt = new DataTable();
    using (var da = new SqlDataAdapter("SELECT * FROM Usuario", _conn))
    {
        da.Fill(dt);
    }
    dgvUsuarios.DataSource = dt;
}
```

Consulta todos los usuarios desde la tabla Usuario y los muestra en el dgvUsuarios.

CargarGrupos()

Llena el ComboBox de grupos (cmbGrupo) para que el usuario pueda seleccionar uno al registrar un nuevo usuario.

btnAgregar_Click — Agregar un nuevo usuario
Verifica que se haya seleccionado un grupo.
Crea una conexión y comando SQL.
Inserta un nuevo usuario en la base de datos.
Recarga los usuarios para reflejar el nuevo.

3. Conclusión

Este entregable demuestra cómo, a partir de las instrucciones del cliente, se construyó un software en C# que permite a varios usuarios llevar un control colaborativo de gastos. Se validaron la estructura modular del proyecto, y se ofreció una explicación detallada de la mayoría de las líneas de código para garantizar transparencia y mantenibilidad.

El resultado final del proyecto aplica buenas prácticas de programación orientada a objetos y facilita la colaboración entre usuarios mediante una arquitectura clara y extensible, de manera en que se pueda mejorar a futuro para las adaptaciones del cliente.