UNIVERSIDAD DE MÁLAGA
Departamento de Lenguajes y Ciencias de la Computación
UNIVERSIDAD DE MÁLAGA

Gestión de la Información
Práctica 8

Visual C#. Acceso a Base de datos con DataSet

Vamos continuar con la solución de la práctica anterior. En esta práctica vamos a desarrollar el Mantenimiento de clientes y la gestión de reservas.

Mantenimiento de Clientes.

Crear un Formulario llamado `FClientes` con un `Label`, `DataGridView` y un `Button` que tenga el siguiente aspecto:

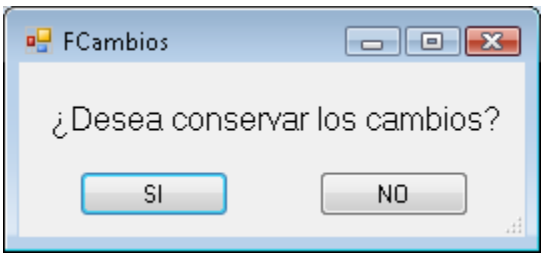
Cuando se inserte el DataGridView, Agregar un nuevo origen de datos al proyecto que será un origen de datos de base de datos (conjunto de datos) que tendrá todas las tablas de nuestra base de datos y cuyo `DataSet` se llamará `hotelDataSet` y seleccionar la Tabla `Clientes`.

El botón salir servirá para cerrar el formulario. (`this.Close();`)

Para actualizar los cambios que se hayan producido, cuando se vaya a cerrar el formulario habrá que ver si ha habido cambios y pedir confirmación para actualizarlos en la base de datos. Esto se consigue añadiendo un manejador al evento `FormClosed`.

```
private void FClientescs_FormClosed(object sender, FormClosedEventArgs e)
 {
     if (this.hotelDataSet.HasChanges())
     {
         FCambios guardar = new FCambios();
         if (guardar.ShowDialog() == System.Windows.Forms.DialogResult.Yes)
         {
             this.clientesTableAdapter.Update(this.hotelDataSet.clientes);
         }
     }
 }
```

El formulario `FCambios` es un formulario que pregunta si desea guardar los cambios. Tiene el siguiente aspecto:



Y debemos especiicar como propiedades del formulario que el `AcceptButton` es `bSI` y que el `CancelButton` es `bNO`. Además en el botón `bSI` hay que poner la propiedad `DialogResult` a `Yes` y en el botón `bNO` hay que poner la propiedad `DialogResult` a `No`.

Con eso debemos tener el mantenimiento de la tabla clientes terminado y podemos activarlo en el menú.

```
private void ActivarMenuClientes()
{
     this.Visible = false;
     FClientes clientes = new FClientes();
     clientes.ShowDialog();
     this.Visible = true;
}
```

Vamos ahora a realizar el proceso de mantenimiento de reservas. La primera restricción a tener en cuenta es que una reserva va siempre asociada a un cliente, por lo que antes de trabajar con cualquier reserva, debemos seleccionar el cliente sobre el cual vamos a trabajar. El formulario anterior nos podría servir si pudiéramos saber cuál es el cliente que está seleccionado.

Para ello vamos a añadir al formulario `FClientes` lo siguiente:

- Un atributo privado y una propiedad pública que sólo nos permita ver su contenido para almacenar el identificador del cliente que tengamos seleccionado.

```
private string selected;

 public string selectedClient
 {
     get
     {
         return selected;
     }
 }
```

- Un manejador del evento que se produce cuando hacemos click en una celda para seleccionar ese cliente

```
 private void dataGridView1_CellClick(object sender, DataGridViewCellEventArgs e)
 {
     if (dataGridView1.Rows.Count > 0)
         selected = dataGridView1.Rows[e.RowIndex].Cells[0].Value.ToString();
 }
```

- Un manejador del evento que se produce cuando hacemos click en la pestaña de seleccionar un registro para seleccionar ese cliente

```
 private void dataGridView1_SelectionChanged(object sender, EventArgs e)
 {
     if (dataGridView1.SelectedRows.Count > 0)
         selected = dataGridView1.SelectedRows[0].Cells[0].Value.ToString();
 }
```

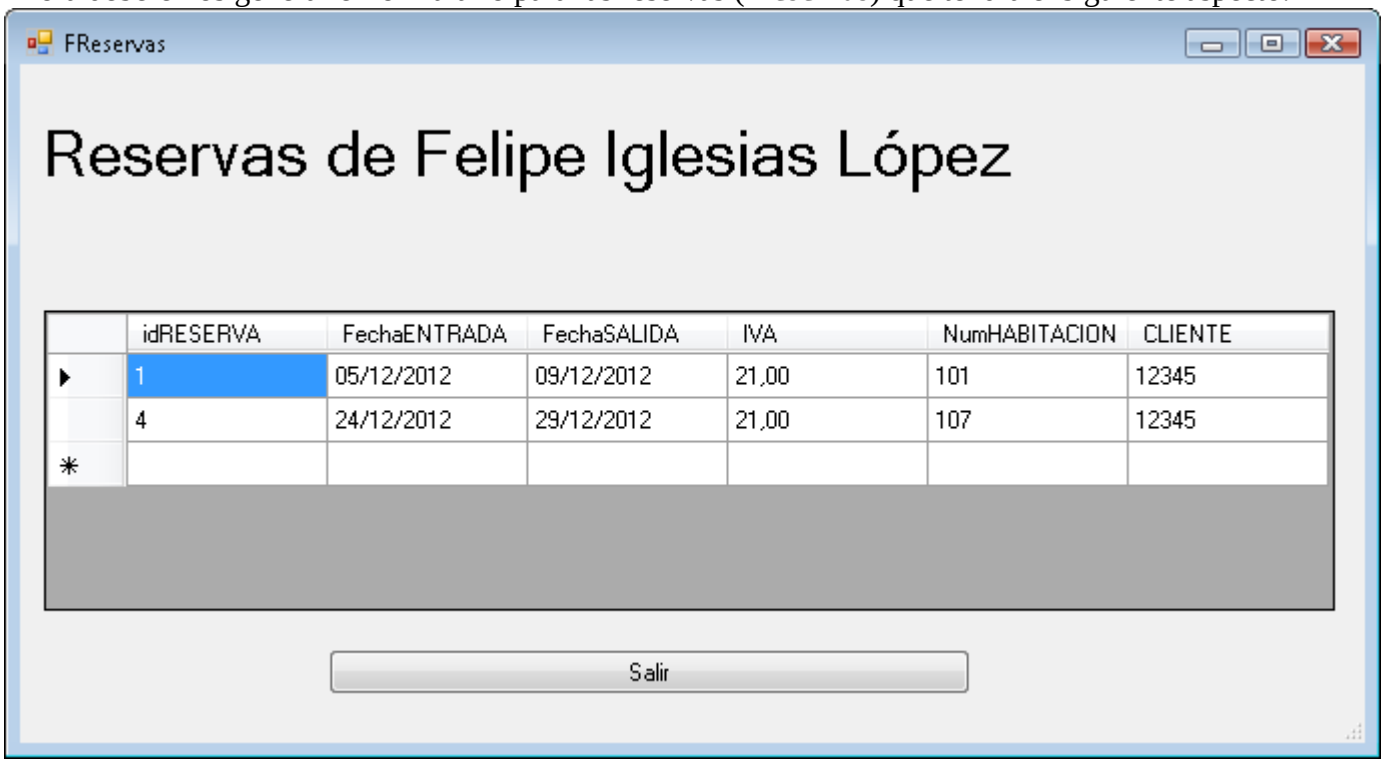
Y en el menú activaremos el formulario de reservas de la siguiente forma:

```
 private void ActivarMenuReservas()
 {
     FClientes clientes;
     this.Visible = false;
```

```
            do
            {
                MessageBox.Show("Primero debe seleccionar un Cliente");
                clientes = new FClientes();
                clientes.ShowDialog();
            } while (clientes.selectedClient == null);

            FReservas reservas = new FReservas(clientes.selectedClient);
            reservas.ShowDialog();
            this.Visible = true;
        }
```

Ahora deberemos generar el formulario para las reservas (`FReservas`) que tendrá el siguiente aspecto:



Para ello debemos usar el origen de datos que ya tenemos en el proyecto y del DataSet seleccionar la tabla de Reservas.
Además hay que añadirle que pregunte si queremos guardar los cambios al cerrar el formulario.

```
        private void FReservas_FormClosed(object sender, FormClosedEventArgs e)
        {
            if (this.hotelDataSet.HasChanges())
            {
                FCambios guardar = new FCambios();
                if (guardar.ShowDialog() == System.Windows.Forms.DialogResult.Yes)
                {
                    this.reserva_habitacTableAdapter.Update(this.hotelDataSet.reserva_habitac);
                }
            }
        }
```

Hay varios detalles importantes que debemos tener en cuenta:

1. El texto del Label lleva el nombre y apellidos del cliente seleccionado.
2. Sólo aparecen las reservas del cliente seleccionado y no de todos los clientes.
3. Las fechas deben ser correctas y la fecha de entrada debe ser menor que la de salida.
4. El identificador de reserva debe ser único y se generará de forma automática.
5. El número de habitación debe ser válido y la habitación no puede estar ocupada en las fechas que deseamos realizar la reserva.

1. **Texto del Label**: Lo primero que hay que hacer es modificar el constructor para que admita como parámetro el string con el identificador del cliente y almacenarlo en un atributo privado. Posteriormente podemos hacer una consulta a la base de datos usando la BDLibrary.

```
private string id_cliente;

public FReservas(string id_cliente)
{
    InitializeComponent();
    this.id_cliente = id_cliente.TrimEnd();
    SQLSERVERDB bd = new SQLSERVERDB("localhost", "hotel");
    object[]res = bd.Select("SELECT nombre, apellido1, apellido2 FROM clientes ”
            + “WHERE Identificacion = '" + this.id_cliente + "';")[0];
    label1.Text += res[0].ToString().TrimEnd() + " "
            + res[1].ToString().TrimEnd() + " " + res[2].ToString().TrimEnd();

}
```

2. **Reservas del cliente seleccionado**: Para que aparecen las reservas del cliente seleccionado y no de todos los clientes, debemos irnos al DataGridView y pinchando sobre la pestaña inteligente de *Tareas de DataGridView* y seleccionar Agregar Consulta. El código de la consulta será:

```
SELECT idRESERVA, FechaENTRADA, FechaSALIDA, IVA, NumHABITACION, CLIENTE
FROM reserva_habitac
WHERE CLIENTE = @cli
```

`@cli` es el parámetro de la consulta. Por defecto, lo añade al formulario en una barra de menú (`fillByToolStrip`) que deberemos eliminar de nuestro formulario.
Posteriormente deberemos modificar la carga del formulario para que en vez de rellenar el dataTable con toda la tabla lo haga con nuestra consulta usando la función que hemos generado `FillBy` en vez de `Fill` y le pasaremos como parámetro el identificador del cliente.

```
private void FReservas_Load(object sender, EventArgs e)
{
    this.reserva_habitacTableAdapter.FillBy(this.hotelDataSet.reserva_habitac,id_cliente);

}
```

3. **Las fechas**: Para gestionar las fechas hay que crearse un formulario que tenga un único componente `MonthCalendar` con un botón de OK y otro de CANCEL y que tenga una propiedad que me devuelva la fecha seleccionada.

```
public DateTime selectedDate
{
    get
    {
        return monthCalendar1.SelectionStart;
    }
}
```

Y utilizaremos una función auxiliar para comprobar las fechas de la fila en la que estemos:

```
private bool FechasOk(int row)
{
    try
    {
        DateTime FechaEntrada = (DateTime)dataGridView1.Rows[row].Cells[1].Value;
        DateTime FechaESalida = (DateTime)dataGridView1.Rows[row].Cells[2].Value;
        return FechaESalida > FechaEntrada;
    }
    catch
    {
        return true;
        // Si falta por especificar una fecha se considera que de momento está bien
    }
}
```

4. **El identificador de reserva:** lo que debemos hacer es asignarle 1 + el máximo entre los que haya en la base de datos y los que tengamos en el dataGridView (por si aun no lo hemos actualizado en la base de datos). Para ello podemos usar la siguiente función auxiliar:

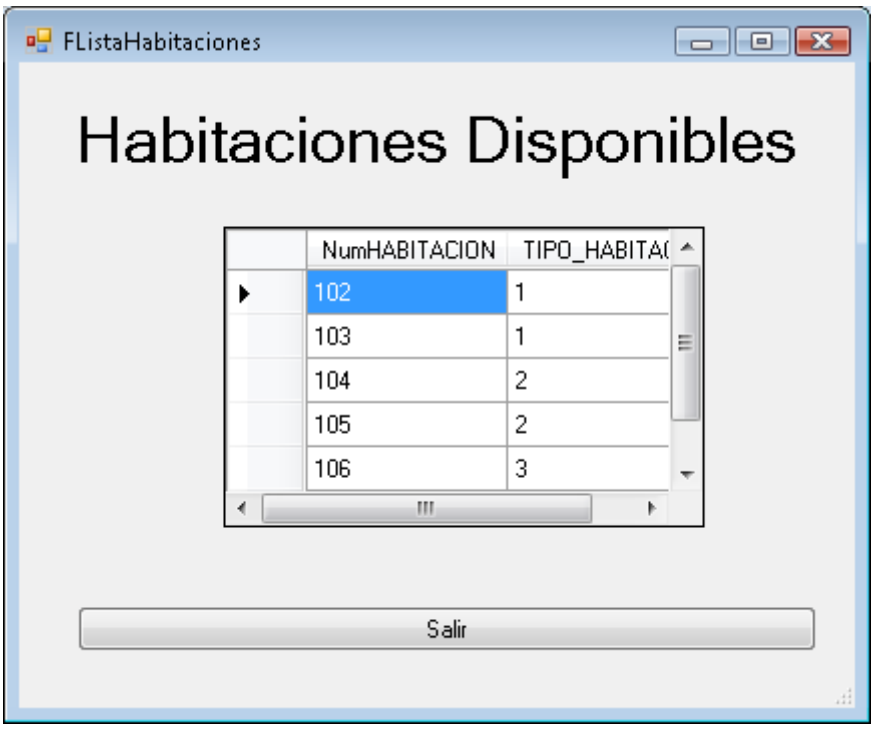
```
private int getIDReserva()
{
    SQLSERVERDB bd = new SQLSERVERDB("localhost", "hotel");
    object o = bd.SelectScalar("SELECT MAX(idRESERVA) FROM reserva_habitac;");
    int cuenta=0;

    if (o != null) cuenta = int.Parse(o.ToString());

    foreach (DataGridViewRow fila in dataGridView1.Rows)
    {
        string celda = fila.Cells[0].Value.ToString();
        if (celda != null && celda.Length > 0)
        {
            int id = int.Parse(celda);
            if (id > cuenta) cuenta = id;
        }
    }

    return 1 + cuenta;
}
```

5. **Números de Habitaciones**. Lo más sencillo es darle al usuario a elegir sólo entre las habitaciones que estén disponibles. Para ello crearemos un formulario con un `DataGridView` y un botón como el que se muestra a continuación. El origen de datos del `DataGridView` será la tabla habitaciones del origen de datos de nuestro proyecto.



Al constructor del formulario le pasaremos la lista de habitaciones que están ocupadas para que las quite del dataGridView. El código de dicha clase será:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
```

```
namespace hotel
{
    public partial class FListaHabitaciones : Form
    {
        private int selected = -1;
        private List<int> ocupadas;

        public int SelectedRoom
        {
            get
            {
                return selected;
            }
        }

        public FListaHabitaciones(List<int> ocupadas)
        {
            InitializeComponent();
            this.ocupadas = ocupadas;
        }

        private void quitaHabitacion(int num)
        {
            for (int i = 0; i < dataGridView1.Rows.Count; ++i)
            {
                if (int.Parse(dataGridView1.Rows[i].Cells[0].Value.ToString()) == num)
                {
                    dataGridView1.Rows.RemoveAt(i);
                    return;
                }
            }
        }

        private void FListaHabitaciones_Load(object sender, EventArgs e)
        {
            this.habitacionesTableAdapter.Fill(this.hotelDataSet.habitaciones);
            foreach (int num in ocupadas) quitaHabitacion(num);
        }

        private void bExit_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        private void dataGridView1_SelectionChanged(object sender, EventArgs e)
        {
            if (dataGridView1.SelectedRows.Count>0)
                selected = int.Parse(dataGridView1.SelectedRows[0].Cells[0].Value.ToString());
        }

        private void dataGridView1_CellClick(object sender, DataGridViewCellEventArgs e)
        {
            selected = int.Parse(dataGridView1.Rows[e.RowIndex].Cells[0].Value.ToString());
        }
    }
}
```

Finalmente, para gestionar todos los controles de la reserva, se harán cuando se haga click sobre una celda del dataGridview:

```
        private void dataGridView1_CellClick(object sender, DataGridViewCellEventArgs e)
        {
            if (dataGridView1.Rows[e.RowIndex].Cells[5].Value.ToString().Length == 0)
                dataGridView1.Rows[e.RowIndex].Cells[5].Value=id_cliente;
```

```
            if (dataGridView1.Rows[e.RowIndex].Cells[0].Value.ToString().Length == 0)
                dataGridView1.Rows[e.RowIndex].Cells[0].Value = getIDReserva();

            if (e.ColumnIndex >= 0)
            {

                DataGridViewColumn columna = dataGridView1.Columns[e.ColumnIndex];
                DataGridViewCell celda = dataGridView1.Rows[e.RowIndex].Cells[e.ColumnIndex];
                if (columna.HeaderCell.Value.ToString().IndexOf("Fecha") != -1)
                {
                    FFecha fecha = new FFecha();
                    if (fecha.ShowDialog() == System.Windows.Forms.DialogResult.OK)
                    {
                        object oldFecha = celda.Value;
                        celda.Value = fecha.selectedDate;
                        if (!FechasOk(e.RowIndex))
                        {
                            celda.Value = oldFecha;
                            MessageBox.Show("La Fecha de salida debe ser posterior a la de
entrada");
                        }
                    }
                }
                else if (columna.HeaderCell.Value.ToString() == "NumHABITACION")
                {
                    SQLSERVERDB bd = new SQLSERVERDB("localhost", "hotel");
                    DateTime FechaEntrada = (DateTime)dataGridView1.Rows[e.RowIndex].Cells[1].Value;
                    DateTime FechaESalida = (DateTime)dataGridView1.Rows[e.RowIndex].Cells[2].Value;
                    List<object[]> res = bd.Select("SELECT NumHABITACION FROM reserva_habitac WHERE
(FechaENTRADA BETWEEN '" +
                                    FechaEntrada.ToShortDateString() + "' AND '" +
FechaESalida.ToShortDateString() + "') AND (FechaSALIDA BETWEEN '" +
                                    FechaEntrada.ToShortDateString() + "' AND '" +
FechaESalida.ToShortDateString() + "');");

                    List<int> ocupadas = new List<int>();
                    foreach (object[] fila in res) ocupadas.Add(int.Parse(fila[0].ToString()));
                    FListaHabitaciones hab = new FListaHabitaciones(ocupadas);
                    hab.ShowDialog();
                    celda.Value = hab.SelectedRoom;
                }

            }
        }
```