

DOCUMENTACIÓN:

C# y MySql

Índice

1. Batalla.....	3
2. Código de interés.....	3
3. Capturas en ejecución.....	4
4. Explicación MVC.....	5
5. Análisis de usabilidad y control de errores.....	6
6. Conclusiones y opiniones.....	6

1. Batalla

Comenzamos la tarea, el jueves 22 de octubre, nos vimos los vídeos en la plataforma para entender lo que se pedía y leímos a fondo el enunciado.

Del lunes al viernes estuvimos prácticamente estancados, el viernes por la mañana conseguimos arreglar un error que hacía que no pudiéramos conectarnos a la base de datos y por la tarde conseguimos que en la vista de lista se listaran los alumnos. Días anteriores habíamos preparado las 3 vistas con todos sus elementos.

El sábado por la mañana logramos que el botón buscar funcionara perfectamente, por la tarde nos centramos más en la usabilidad y en el control de excepciones y por la noche terminamos con el botón eliminar de la vista buscar.

Por último el domingo 1 de noviembre hemos hecho la documentación y hemos retocado el código para mejorar la usabilidad y el control de excepciones así como el funcionamiento de la propia aplicación.

2. Código de interés

```
public void LlenarTabla()
{
    try
    {
        vistaListar.getDataGrid.DataSource = conector.LlenarGrid().Tables[0];
    }
    catch (SqlNullValueException e)
    {
        MessageBox.Show("El valor es nulo \n" + e);
    }
    catch (InvalidOperationException e)
    {
        MessageBox.Show("La operación no es válida \n" + e);
    }
    catch (NotSupportedException e)
    {
        MessageBox.Show("el método no está soportado \n" + e);
    }
    catch (MySqlException e)
    {
        MessageBox.Show("ha habido un error con la obtencion de datos SQL \n" + e);
    }
    catch (Exception e)
    {
        MessageBox.Show("Ha ocurrido un error \n" + e);
    }
}
```

El método LlenarTabla agrega la fuente de datos del DataGrid que está en la vista de la lista, accedemos al método LlenarGrid de la conexión y le indicamos que cargue los datos de la primera tabla

```
public DataSet LlenarGrid()
{
    DataSet ds = new DataSet();
    try
    {
        consulta = "Select * from alumnos";
        MySqlDataAdapter adapter = new MySqlDataAdapter(consulta, conexionBD);

        adapter.Fill(ds);
    } catch (MySqlException e)
    {
        MessageBox.Show("Ha habido un error al llenar la tabla \n" + e);
    } catch (Exception e)
    {
        MessageBox.Show("Ha ocurrido un error \n" + e);
    }
    return ds;
}
```

El método mas complicado sin duda es el EliminarAlumno, el cual requiere no solo eliminar un alumno sino buscar un registro en la tabla préstamos y comprobar que no haya ninguno para el alumno que queramos eliminar

```
1 referencia
public void ComprobarYBorrarAlumno(string vCodDni, string vDni)
{
    int vNDni = Int32.Parse(vCodDni);
    string consulta = "Select * from prestamos where codAlumno= '" + vNDni + "'";
    MySqlCommand comando = new MySqlCommand(consulta, conexionBaseDatos);

    MySqlDataReader reader = comando.ExecuteReader();

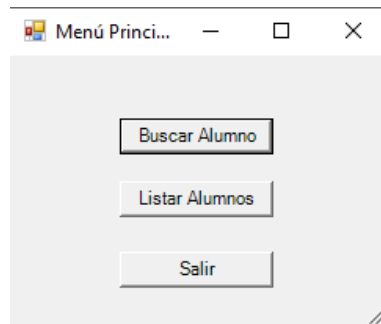
    if (reader.HasRows)
    {
        MessageBox.Show("No se puede eliminar un alumno con préstamo de libro");
        reader.Close();
    }
    else
    {
        reader.Close();
        string consultaBorrar = "Delete from alumnos where dni= '" + vDni + "'";
        using (MySqlCommand comandoBorrar = new MySqlCommand(consultaBorrar, conexionBaseDatos))
        {
            comandoBorrar.ExecuteNonQuery();
        }
        MessageBox.Show("El alumno ha sido eliminado");
        vistaBuscar.getTextBoxApe1.Text = "";
        vistaBuscar.getTextBoxApe2.Text = "";
        vistaBuscar.getTextBoxNombre.Text = "";
        vistaBuscar.getTextBoxDni.Text = "";
    }
}
```

Como podemos ver, este metodo recibe variables del codigo dni y del dni, la variable codDni es el dni pero sin letra (`string codDni = dni.Remove(dni.Length - 1)`) dentro del método la llamo vCodDni para evitar confusiones, con esta variable ejecutamos un select a prestamos, y al ejecutar un reader podemos confirmar si devuelve valores o es null, si es null significa que no hay préstamo y podemos borrar, si no es null significa que hay un préstamo, por lo que muestra un error y cancela la operación, aquí está como quedaría aplicado al método correspondiente

```
public void EliminarAlumno()
{
    string dni = vistaBuscar.getTxtDni;
    string codDni = dni.Remove(dni.Length - 1);
    try
    {
        ComprobarYBorrarAlumno( codDni, dni);
    }
    catch (FormatException e)
    {
        MessageBox.Show("Error de formato\n" + e);
    }
    catch (SqlNullValueException e)
    {
        MessageBox.Show("El valor es nulo \n" + e);
    }
    catch (IndexOutOfRangeException e)
    {
        MessageBox.Show("El valor actual no existe \n" + e);
    }
}
```

Como se puede observar, generamos dos string, siendo una el texto que esté en la vista en el apartado del dni, y otra el mismo texto pero sin letra final, y con eso ya funcionaría nuestra búsqueda y borrado con comprobación de tablas

3. Capturas en ejecución



Datos del Alumno

Dni a buscar:

Nombre:

1er Apellido:

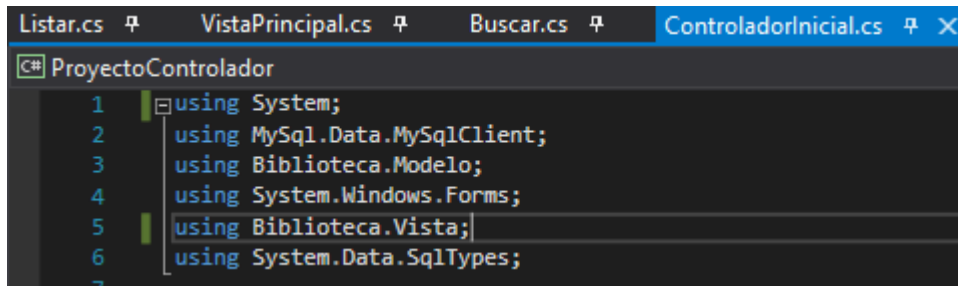
2do Apellido:

Listar todos los Alumnos

	registro	dni	nombre	apellido1	apellido2
▶	4	64a	JUAN MIGUEL	AFONSO	ALONSO
	5	73a	SERGIO	AFONSO	MARRERO
	6	74a	MAR...	AFONSO	MEDINA
	7	79a	HERIBERTO	AFONSO	PEREZ
	8	83a	DESIREE RAIZA	AFONSO	RAMIREZ
	9	85a	JUANA ISABEL	AFONSO	REYES
	10	87a	CARMEN G.	AFONSO	SANTANA
	11	96a	JOSE ANGEL	AGUIAR	MARTIN
	12	98a	CARMEN ROSA	AGUIAR	MENDOZA
	13	99a	OLIVERIO CAR...	AGUIAR	PONCE
	14	100a	ISRAEL SERGIO	AGUIAR	RODR
	15	121a	IRENE DE LA LUZ	ALAMO	CACERES
	16	122a	ADELAIDA	ALAMO	GONZALEZ
	17	132a	NEREIDA MARIA	ALAMO	MORENO
	18	135a	FRANCISCO EC...	ALAMO	PEREZ
	19	148a	INES MARIA	ALBA	RODRIGUEZ
	20	154a	ALICIA	ALCEDO	REYES
	21	158a	GLORIA	ALEM	AIZP

4. Explicación MVC

El espacio de nombres común en nuestros proyectos es “Biblioteca” siendo así que el espacio de nombres de cada proyecto es “Biblioteca.Vista”, “Biblioteca.Controlador” y “Biblioteca.Modelo”. En el proyecto del controlador se añadieron las dependencias de los otros dos proyectos y el conector de MySQL a parte de la dependencia System.Windows.Forms.

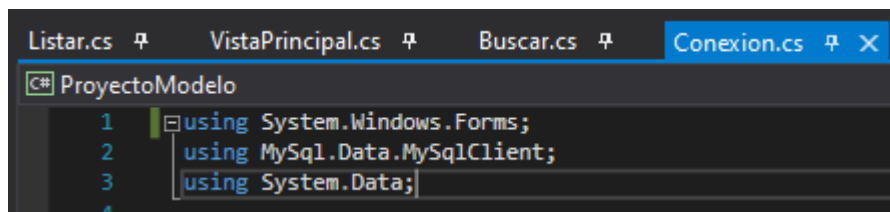


```

1  using System;
2  using MySql.Data.MySqlClient;
3  using Biblioteca.Modelo;
4  using System.Windows.Forms;
5  using Biblioteca.Vista;
6  using System.Data.SqlTypes;

```

En el proyecto del modelo sólo se añadió a las dependencias el conector de MySQL y System.Windows.Forms.

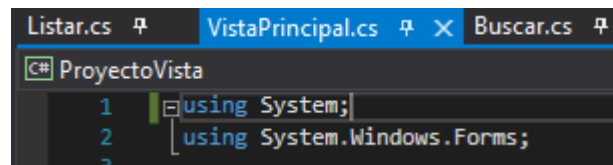


```

1  using System.Windows.Forms;
2  using MySql.Data.MySqlClient;
3  using System.Data;

```

En el proyecto de vista no añadimos ninguna referencia, solo las necesarias.



```

1  using System;
2  using System.Windows.Forms;

```


5. Análisis de usabilidad y control de errores

En un principio sabíamos en que lugares necesitaríamos ayudar al usuario, los botones y los campos de texto tienen nombres inconfundibles para facilitar el entendimiento del usuario, las interfaces son sencillas y fáciles de aprender, no están sobrecargadas y muestran la información necesaria. En cuanto al control de errores hemos hecho bastante hincapié en ellos además de unirlos con la usabilidad mostrando un mensaje explicatorio sobre la excepción ocurrida a parte del código de la excepción para que los desarrolladores o alguien que entienda el código de error pueda solucionarlo.

6. Conclusiones y opiniones

La división de la tarea en diferentes proyectos dentro de una misma solución en Visual Studio es una forma de tener más organizada la aplicación final pero creemos que es más simple crear un único proyecto con los paquetes necesarios, ahorras el trabajo que supone cambiar el espacio de nombres y hacer las referencias necesarias y tienes prácticamente el mismo orden sin afectar al rendimiento de la aplicación. Por otra parte nos ha gustado bastante trabajar con C#, a pesar de no tener una base con este lenguaje se hace bastante fácil programar con él debido a la cantidad de documentación que hay y su similitud en ciertos aspectos con Java.

7. Otros

La mayoría de métodos y ayudas las encontramos en [Stack Overflow](#) además de otros foros de ayuda, también hemos encontrado algunos vídeos que nos han ayudado bastante.

8. Enlace de GitHub

<https://github.com/acoidan-santana/tarea5busquedaDniAlumno.git>