Informe C#



Kevin Clark David Job Jiménez Suárez

<u>Índice</u>

Cuaderno de Batalla	
Código de interés	pág. 1
Ejecución del programa	pág. 2
Explicación MCV	pág. 4
Usabilidad	pág. 5
Excepciones y Errores	pág. 5
Opinión personal	pág. 6
Bibliografí	pág. 6

Cuaderno de batalla

El primer día, martes, estuvimos toda la clase instalando los componentes necesario del visual installer para poder iniciar el proyecto.

El miércoles ya teníamos todo a punto para iniciar, estábamos un poco perdidos por que no sabíamos muy bien que tipo de proyecto porque había muchos tipo de soluciones en C#. Después de preguntar por clase elegimos la aplicación de Windows Forms(.NET Framework). El siguiente problema era que no sabíamos como crear la clases, ya que teníamos que hacer un Modelo Vista Controlador. El Jueves pudimos crear la solución con sus tres proyectos, uno con la vista(aplicación de Windows Forms(.NET Framework)), otro con el modelo(Biblioteca de clases(.NET standart)) y para el controlador(.NET standart)). El resto del día estuvimos mirando vídeo de la plataforma para poder "navegar" entre proyectos así poder pasar variables, acceder a métodos, etc. El viernes resolvimos el problema de la navegabilidad añadiendo el namespace a todos los proyectos así como las referencias y trabajando con los returns, pudimos crear la conexión a la base de datos, mostrar la tabla en la vista y eliminar alumnos de la base de datos. El sábado añadimos alumnos a la base de datos a pesar de que este punto fue eliminado por la profesora. Nos quedaron puntos por resolver, el primero fue que al eliminar un alumno, si tenía un préstamo que no se pudiese eliminar, en mi proyecto se elimina cualquier alumno, el problema es que no conseguimos entender la base de datos, no sé que registro de la tabla préstamos hace referencia a un identificador de la tabla alumnos ya que ninguna tabla tiene claves foranéas.

Código de interés

Con este código me conectaba a la base de datos:

Con este código hacia el select a la base de datos para mostrar todos los alumnos en la tabla de la vista:

```
public static List<Alumno> Buscar()
        {
            conectar.Open();
            List<Alumno> lista = new List<Alumno>();
            MySqlCommand comando = new MySqlCommand(string.Format("Select * From alumnos"),
conectar);
            MySqlDataReader reader = comando.ExecuteReader();
            while (reader.Read())
                Alumno aux = new Alumno();
                aux.Registro = reader.GetInt32(0);
                aux.Dni = reader.GetString(1);
                aux.Nombre = reader.GetString(2);
                aux.Apellido1 = reader.GetString(3);
                aux.Apellido2 = reader.GetString(4);
                lista.Add(aux);
            conectar.Close();
            return lista;
```

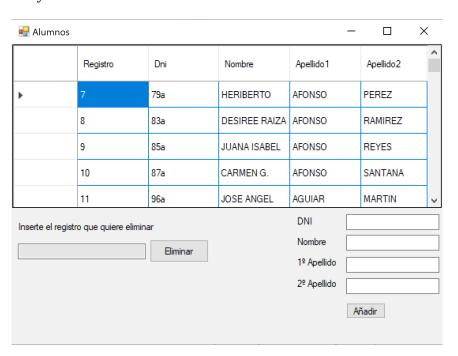
```
}
```

Aquí se borra un alumno con el registro:

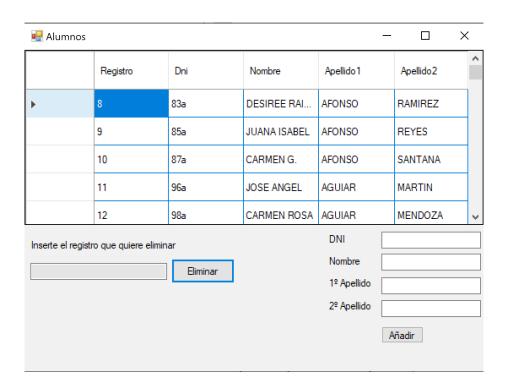
```
public static void Delete(int reg)
            conectar.Open();
            MySqlCommand comando = new MySqlCommand(string.Format("Delete From alumnos where
registro={0}", reg), conectar);
            comando.ExecuteNonQuery();
            conectar.Close();
        }
El código usado para añadir alumnos:
        public static void Anadir(string dni, string nombre, string apellido1, string
apellido2)
            conectar.Open();
            MySqlCommand comando = new MySqlCommand(string.Format("Insert into Alumnos
values (null, '{0}', '{1}', '{2}', '{3}')", dni, nombre, apellido1, apellido2), conectar);
            comando.ExecuteNonQuery();
            conectar.Close();
        }
    }
```

Capturas del programa en ejecución

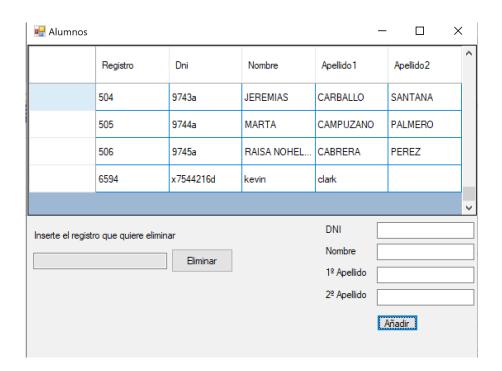
Esta es la primera y única vista.



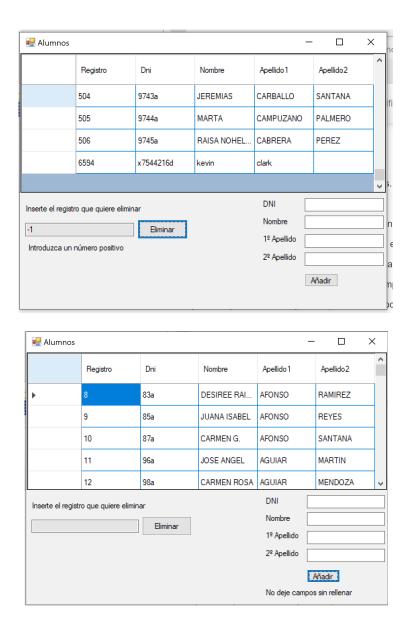
Como se veía en la imagen anterior el primer registro era el 7, lo eliminé y el primer registro que aparece el 8.



Aquí añado mi nombre a la tabla.



Y en estas dos capturas muestro un poco el control de errores que hicimos.



Explicación MVC

Como explicaba un poco en el aparatado del cuaderno de batalla, el MVC que hicimos es diferente al de java, ya que aquí trabajamos con tres proyecto, uno para cada parte del MCV, en java trabajamos en un sólo proyecto con tres clases, y al estar en el mismo proyecto a la hora de acceder a variable o métodos que se encuentran en otra clase es mucho más fácil, en C# tuvimos que agregar referencia a los proyectos, nosotros a la vista de dimos referencia del controlador y al controlador del modelo y así conseguimos movernos entre proyectos, el problema es que la navegabilidad es unidireccional, yo puedo acceder a el controlador desde la vista pero el controlador no puede acceder a la vista, esto lo resolvimos con los returns de los métodos, también agregamos el mismo namespace a todos los proyectos.

Usabilidad

En cuanto a la usabilidad el proyecto cumple las tareas para el que fue diseñado, es fácil de usar, tiene un diseño simple, como solo tiene una sola vista no está todo apretado y no tiene tantos elementos.

Excepciones y errores

Los errores que yo pude capturar y manejar eran errores de el usuario de tipo escribe un dato erróneo o que deja datos en blanco.

```
private void anadir_button_Click(object sender, EventArgs e)
{
   if (textBox_dni.Text.Equals("") || textBox_nombre.Text.Equals("") ||
        textBox_apellido1.Text.Equals("") )
   {
        data_error_label.Text = "No deje campos sin rellenar";
   }
   else
   {
        Controller.AnadirAlumno(textBox_dni.Text, textBox_nombre.Text, textBox_apellido1.Text,
        dataGridView1.DataSource = Controller.MostrarAlumnos();
        textBox_dni.Text = "";
        textBox_apellido1.Text = "";
        textBox_apellido2.Text = "";
        data_error_label.Text = "";
        data_error_label.Text = "";
    }
}
```

```
private void button_eliminar_Click(object sender, EventArgs e)
{
  int n;
  try
  {
    n = Int32.Parse(textBox1.Text);
    if (n <= 0)
    {
        label_ERROR.Text = "Introduzca un número positivo";
    }
    else
    {
        Controller.EliminarAlumno(n);
        dataGridView1.DataSource = Controller.MostrarAlumnos();
        label_ERROR.Text = "";
        textBox1.Text = "";
    }
}
catch (FormatException)
    {
        label_ERROR.Text = "Introduzca un valor númerico";
    }
}</pre>
```

Opinión personal

La verdad que fue una buena experiencia trabajar con c# y con Visual Studio, no había utilizado ni el lenguaje ni el IDE hasta ahora, a la hora de buscar información se encuentra fácilmente, la única pega que le veo es que trabajamos en un IDE muy actual y todos los vídeos que encontré, incluso los que nos facilitó la profesora, estaban realizados en versiones anteriores del Visual Studio y la interfaz ha cambiado.

Bibliografía:

- http://www3.gobiernodecanarias.org/medusa/eforma/campus/course/view.php?id=20598
- https://docs.microsoft.com/es-es/dotnet/csharp/