

Caso Integrador

Presentado por:

Andrea Paola Alvarez Sánchez
David Santiago Noguera Hernández
Erik Brandon Mora Aguirre

Docente:

David Bohórquez Caro



Bogotá, 03-06-2024

**FACULTAD DE TÉCNICAS DE INGENIERÍA
Programa de Desarrollo de Software
Aplicaciones I**

1. ¿Qué ventajas trae usar este patrón/ arquitectura MVC de desarrollo de software?

El patrón Modelo-Vista-Controlador (MVC) ofrece múltiples ventajas clave para el desarrollo del software de cálculo de notas para el politécnico internacional. En primer lugar, MVC facilita la separación de preocupaciones, dividiendo la aplicación en Modelo (lógica de negocio), Vista (interfaz de usuario) y Controlador (manejo de solicitudes). Esto permite una mejor organización del código, facilitando su mantenimiento y actualización.

Además, la modularidad del MVC promueve la reutilización de código y el desarrollo paralelo, donde diferentes equipos pueden trabajar simultáneamente en distintas partes del proyecto, acelerando el proceso de desarrollo. La clara distinción entre componentes también mejora la experiencia del usuario, ya que permite enfocarse en la optimización de la interfaz sin interferir con la lógica de negocio.

La estructura MVC facilita las pruebas unitarias y de integración, ya que cada componente puede ser probado de forma independiente, asegurando una mayor calidad del software. Asimismo, la flexibilidad para gestionar cambios en los requisitos del negocio o en la interfaz de usuario es crucial en entornos ágiles como SCRUM, permitiendo iteraciones rápidas y entregas incrementales.

En resumen, MVC mejora la mantenibilidad, escalabilidad y eficiencia del desarrollo, alineándose perfectamente con las necesidades del proyecto y la metodología ágil adoptada.

2. ¿Existen otros patrones de desarrollo de software que puedan dar solución al problema planteado?

Modelo-Vista-Presentador (MVP)

Similar a MVC, pero el Presentador maneja toda la lógica de presentación y actúa como un intermediario entre la Vista y el Modelo.

Ventajas/Beneficios:

- Mejor separación de responsabilidades: El Presentador maneja toda la lógica de presentación, simplificando la Vista y facilitando su prueba y mantenimiento.
- Facilita las pruebas unitarias: El Presentador no depende de la interfaz de usuario, permitiendo probar la lógica de presentación de manera aislada.
- Mayor flexibilidad en la actualización de la UI: La Vista puede cambiar sin afectar la lógica de negocio, facilitando la adaptación a nuevos requerimientos de interfaz.

Modelo-Vista-VistaModelo (MVVM)

Descripción: Utiliza la vinculación de datos para conectar la Vista con el ViewModel, simplificando la sincronización entre la interfaz de usuario y el estado de la aplicación.

Ventajas/Beneficios:

- Data Binding: Automatiza la sincronización entre la Vista y el ViewModel, reduciendo el código necesario para actualizar la interfaz.
- Desarrollo paralelo: Los desarrolladores de UI y lógica de negocio pueden trabajar de manera independiente.
- Pruebas más sencillas: Al igual que en MVP, el ViewModel puede probarse sin necesidad de la Vista, facilitando las pruebas unitarias.

Ambos patrones, MVP y MVVM, ofrecen una separación clara de responsabilidades y facilitan el mantenimiento, la escalabilidad y la capacidad de prueba del software, proporcionando soluciones efectivas para el desarrollo del prototipo de cálculo de notas.

3. ¿Qué patrón (uno solo) recomendaría usted para dar solución a la problemática planteada y por qué?

En el contexto del desarrollo del prototipo de software para el cálculo de notas del politécnico internacional, el patrón más recomendable sería el **Modelo-Vista-Controlador (MVC)**. Esto se debe a varias razones:

Adaptabilidad a la Metodología SCRUM: MVC se integra bien con metodologías ágiles como SCRUM, ya que facilita la entrega incremental y la mejora continua del software. Cada componente (Modelo, Vista y Controlador) puede ser desarrollado y probado de forma independiente, lo que permite iteraciones rápidas y entregas funcionales en cada sprint.

Claridad en la Separación de Responsabilidades: MVC proporciona una clara separación de responsabilidades entre la lógica de negocio (Modelo), la presentación de datos (Vista) y el control del flujo de la aplicación (Controlador). Esto facilita la comprensión del código, el mantenimiento y la evolución del sistema a medida que los requisitos cambian.

Flexibilidad y Escalabilidad: La estructura modular de MVC permite que cada componente pueda ser modificado o reemplazado sin afectar los demás. Esto proporciona flexibilidad para adaptar el sistema a nuevas necesidades o para escalar el proyecto a medida que crece en funcionalidad y complejidad.

Compatibilidad con Windows Forms y C#: Dado que el front-end debe ser desarrollado con Windows Forms y C#, MVC es una elección natural, ya que es un patrón ampliamente utilizado en el desarrollo de aplicaciones basadas en .NET.

En resumen, MVC ofrece una combinación de adaptabilidad, claridad en la separación de responsabilidades, flexibilidad y compatibilidad con las tecnologías requeridas en el proyecto, lo que lo convierte en el patrón más recomendable para el desarrollo del prototipo de software para el cálculo de notas del politécnico internacional.

Resultado Final de la Interfaz

Se agregan algunos datos, y confirmamos a través de un Select si los datos están siendo guardados en la BD y así probamos que la conexión funciona.

The image displays three screenshots of a web application interface for managing student grades, along with a SQL query window and its results.

Form 1 (Top Left): Shows a student named Juanita Perez with grades: Nota 1: 1,2, Nota 2: 3,3, Nota 3: 3,8. The calculated final grade is 2,715.

Form 1 (Top Right): Shows a student named Manuel Franz with grades: Nota 1: 4,3, Nota 2: 3,9, Nota 3: 4,7. The calculated final grade is 4,28.

Form 1 (Bottom Left): Shows a student named Manuel Franz with grades: Nota 1: 4,8, Nota 2: 3,2, Nota 3: 4,1. The calculated final grade is 4,029999999999999.

SQL Query Window: The query is as follows:

```
use notasPolitecnicoInternacional
Drop table tecnologiaDesarrolloSoftware
create table tecnologiaDesarrolloSoftware (Id INT PRIMARY KEY IDENTITY(1,1),
Nombre NVARCHAR(100) NOT NULL,
Nota1 FLOAT NOT NULL,
Nota2 FLOAT NOT NULL,
Nota3 FLOAT NOT NULL,
NotaFinal FLOAT NOT NULL);
Select *FROM tecnologiaDesarrolloSoftware
```

Results: The query results are displayed in a table:

Id	Nombre	Nota1	Nota2	Nota3	NotaFinal
1	Juanita Perez	1,2	3,3	3,8	2,715
2	Manuel Franz	4,3	3,9	4,7	4,28
3	Manuel Franz	4,8	3,2	4,1	4,03

Clase ConectorBD

Esta clase maneja la conexión a la base de datos SQL Server.

Propiedad BaseDatos: Contiene la cadena de conexión a la base de datos SQL Server.

Método GetConnection: Retorna una nueva conexión SQL utilizando la cadena de conexión proporcionada.



```
Form1.cs [Diseño] | Program.cs | Form1.cs | Controlador.cs | ConectorBD.cs | Form1.Designer.cs
C# | calificarNotas | calificarNotas.ConectorBD | GetConnection()

1  using System;
2  using System.Collections.Generic;
3  using System.Data.SqlClient;
4  using System.Linq;
5  using System.Text;
6  using System.Threading.Tasks;
7
8  namespace calificarNotas
9  {
10     2 referencias
11     internal class ConectorBD
12     {
13         private string BaseDatos = "Server=LAPTOP-ANDREA\\SQLEXPRESS; DATABASE=notasPolitecnicoInternacional;Integrated Security=True";
14
15         //Metodo para hacer la conexion
16
17         1 referencia
18         public SqlConnection GetConnection()
19         {
20             return new SqlConnection(BaseDatos);
21         }
22     }
23 }
```

Clase Controlador

Esta clase maneja la lógica del negocio y la interacción con la base de datos.

Método CalcularNotaFinal: Calcula la nota final basada en las ponderaciones de los tres cortes (35%, 35%, 30%).

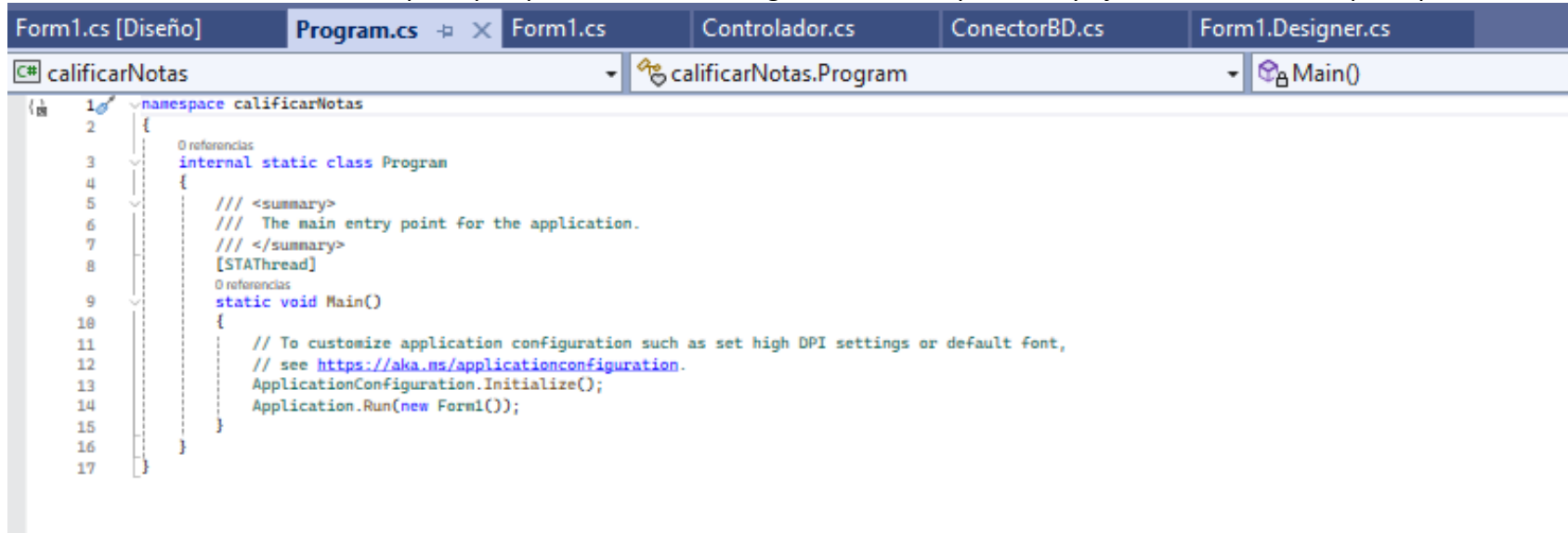
Método GuardarEnBD: Guarda el nombre del estudiante y sus notas en la base de datos.

```
Form1.cs [Diseño] | Program.cs | Form1.cs | Controlador.cs | ConectorBD.cs | Form1.Designer.cs
C# | calificarNotas | calificarNotas.Controlador | GuardarEnBD(string nombre, double nota1, double r
{
    1 using System;
    2 using System.Collections.Generic;
    3 using System.Data.SqlClient;
    4 using System.Linq;
    5 using System.Text;
    6 using System.Threading.Tasks;
    7
    8 namespace calificarNotas
    9 {
    10     2 referencias
    11     internal class Controlador
    12     {
    13         1 referencia
    14         public double CalcularNotaFinal(double nota1, double nota2, double nota3)
    15         {
    16             return (nota1 * 0.35) + (nota2 * 0.35) + (nota3 * 0.30);
    17         }
    18
    19         1 referencia
    20         public void GuardarEnBD(string nombre, double nota1, double nota2, double nota3, double notaFinal)
    21         {
    22             ConectorBD db = new ConectorBD();
    23             SqlConnection connection = db.GetConnection();
    24             try
    25             {
    26                 connection.Open();
    27                 string query = "INSERT INTO tecnologiaDesarrolloSoftware (NombreEstudiante, Nota1, Nota2, Nota3, NotaFinal) VALUES (@Nombre, @Nota1, @Nota2, @Nota3, @NotaFinal)";
    28                 SqlCommand cmd = new SqlCommand(query, connection);
    29                 cmd.Parameters.AddWithValue("@Nombre", nombre);
    30                 cmd.Parameters.AddWithValue("@Nota1", nota1);
    31                 cmd.Parameters.AddWithValue("@Nota2", nota2);
    32                 cmd.Parameters.AddWithValue("@Nota3", nota3);
    33                 cmd.Parameters.AddWithValue("@NotaFinal", notaFinal);
    34                 cmd.ExecuteNonQuery();
    35             }
    36             finally
    37             {
    38                 connection.Close();
    39             }
    40         }
    41     }
    42 }
```

Clase Program

Esta es la clase de entrada para la aplicación Windows Forms.

Método Main: Punto de entrada principal que inicializa la configuración de la aplicación y ejecuta el formulario principal Form1.



```
1 namespace calificarNotas
2 {
3     internal static class Program
4     {
5         /// <summary>
6         /// The main entry point for the application.
7         /// </summary>
8         [STAThread]
9         static void Main()
10        {
11            // To customize application configuration such as set high DPI settings or default font,
12            // see https://aka.ms/applicationconfiguration.
13            ApplicationConfiguration.Initialize();
14            Application.Run(new Form1());
15        }
16    }
17 }
```

Funcionalidad del Prototipo

Interfaz Gráfica (Front-End):

- Utiliza Windows Forms para crear un formulario que permite ingresar las notas de los tres cortes para un solo estudiante.
- Los datos se ingresan en campos TextBox y la nota final se muestra en un TextBox o Label.

Lógica del Negocio (Controlador):

- Calcula la nota final utilizando las ponderaciones especificadas.
- Maneja la conexión a la base de datos y almacena los datos ingresados y calculados en la tabla tecnologiaDesarrolloSoftware.

Almacenamiento de Datos (Modelo):

- Usa SQL Server para almacenar las notas y la información del estudiante.

- La clase ConectorBD facilita la conexión a la base de datos y Controlador maneja las operaciones de inserción.

Flujo de Información

1. El usuario ingresa las notas en el formulario de Windows Forms.
2. El formulario envía las notas al Controlador.
3. El Controlador calcula la nota final.
4. El Controlador guarda las notas y la nota final en la base de datos.
5. El resultado se muestra al usuario en la interfaz gráfica.

Este prototipo demuestra cómo calcular y almacenar las notas de un estudiante usando un patrón MVC en un entorno de desarrollo ágil (SCRUM), y está diseñado para ser escalable a futuro

La versión e Visual Studio utilizada es la 2022.

La versión de SQL Server Management Studio es la 19.

El código fue desarrollado por todo el equipo de trabajo en sincronía.