PRÁCTICA 04 - MVP

Patrón Modelo-Vista-Presentador (MVP)

DESARROLLO RÁPIDO DE APLICACIONES

Por Juan José Camacho Hidalgo
UNIVERSIDAD DE ALMERÍA

Contenido

1. Introducción

Este documento tiene como objetivo la explicación y justificación del uso del patrón Modelo-Vista-Presentador, mediante un ejemplo programado en C#.

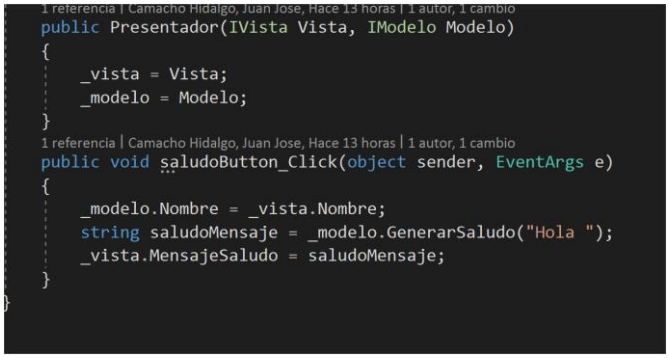
2. Patrón Modelo-Vista-Presentador (MVP)

El patrón Modelo-Vista-Controlador (MVP) se trata de un tipo de patrón derivado del patrón Modelo-Vista-Controlador (MVC). De igual forma que en el MVC, se separa el modelo, de la interfaz de usuario, pero en vez de usar un controlador, como intermediario, este hace uso de un presentador, cuyo objetivo es servir de enlace independientemente de cómo se programe la interfaz de usuario. La vista no conoce nada acerca del modelo.

En cuanto a la solución, vemos que se han mostrado dos ejemplos: uno de un saludo y otro de una suma. Ambos han hecho uso de dos puntos de vista desde la que se puede implementar el patrón MVP: vista pasiva y controlador de supervisión.

2.1. SaludoMVPPasivoWindowsFormsApp

En este ejemplo, se muestra un uso del patrón en forma de vista pasiva. Esto quiere decir que la vista no conoce los cambios en el modelo, y la interacción es gestionada exclusivamente por el presentador. El modelo corresponde a la clase Modelo que implementa IModelo. Su cometido es contener el nombre y el generarSaludo. La vista se trata de la clase SaludoForm y está montada mediante Windows Forms, implementando IVista, la interfaz definida para esta que tiene que contener lo necesario para mostrar ese saludo. Y lo más importante, la clase Presentador, sin la cuál no podrían interactuar la vista y el modelo. Hace uso de ambos, y como observamos en el código, realiza el saludo mediante la vista y el modelo:



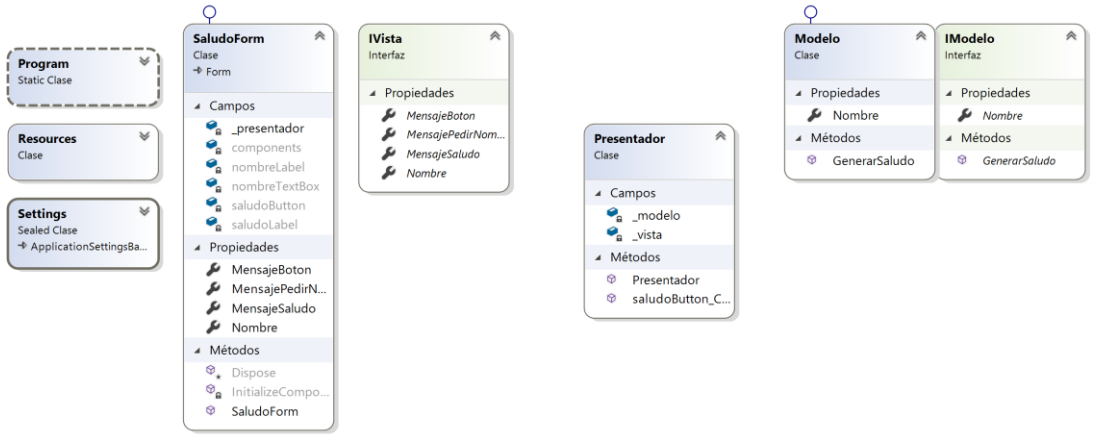
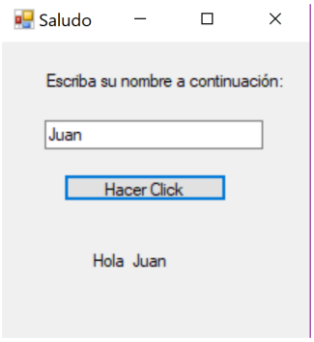


Ilustración 1. Diagrama de clases SaludoMVPPasivoWindowsFormsApp.

Como vemos en la ejecución, al introducir el nombre en la vista, el presentador haciendo uso del modelo nos muestra el saludo.



2.2. SaludoMVPSupervisingWindowsFormsApp

Este segundo ejemplo de saludo, lo hace bajo el controlador de supervisión. Esto significa que la vista y el modelo si interactúan a pesar del uso del presentador.

```
public partial class SaludoForm : Form, IVista
{
    private Presentador _presentador;
    1 referencia | Camacho Hidalgo, Juan Jose, Hace 13 horas | 1 autor, 1 cambio
    public SaludoForm(IModelo modelo)
    {
        InitializeComponent();
        _presentador = new Presentador(this, modelo);
        saludoButton.Click += SaludoButton_Click;
        modelo.Saludo += Modelo_Saludo;
    }
    1 referencia | Camacho Hidalgo, Juan Jose, Hace 13 horas | 1 autor, 1 cambio
```

Como vemos, la vista SaludoForm, realiza en su constructor una llamada al saludo del Modelo, un evento, por el cual, por defecto muestra un Hola en pantalla, antes de conocer el nombre de la persona a quien se saluda.

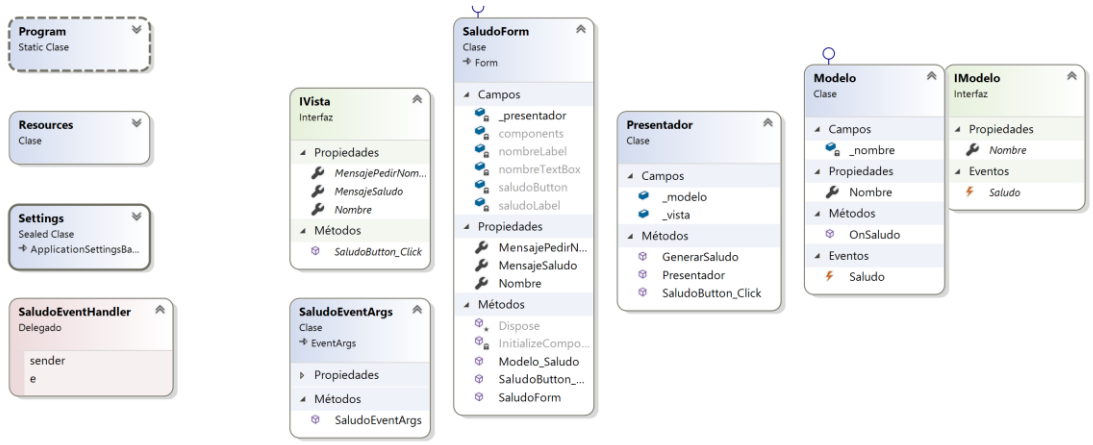
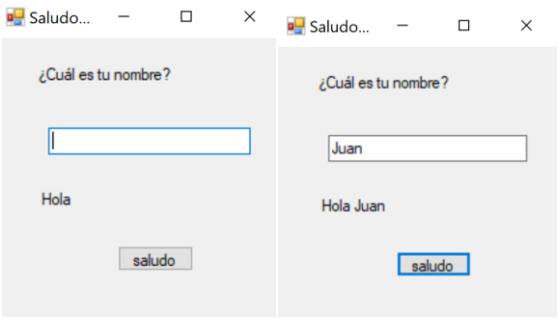


Ilustración 2. Diagrama de clases de SaludoMVPSupervisingWindowsFormsApp.

Como vemos en la ejecución, antes de escribir el nombre, la vista ya ha consultado al modelo, para imprimir la palabra Hola, mientras que cuando se introduce el nombre, ya lo sobreescribe con el nombre de entrada.



2.3. SumaMVPPasivoWindowsFormsApp

Esta segunda aplicación es muy similar al saludo, solo que en vez de realizar un solo saludo, se incorpora una entrada más en la vista, y una operación de suma. Esta hace uso de la vista pasiva, por lo que el modelo no interactua con la vista. El modelo es la clase Modelo, y la vista SumaForm, mientras que la clase Presentador se encarga de la interacción.

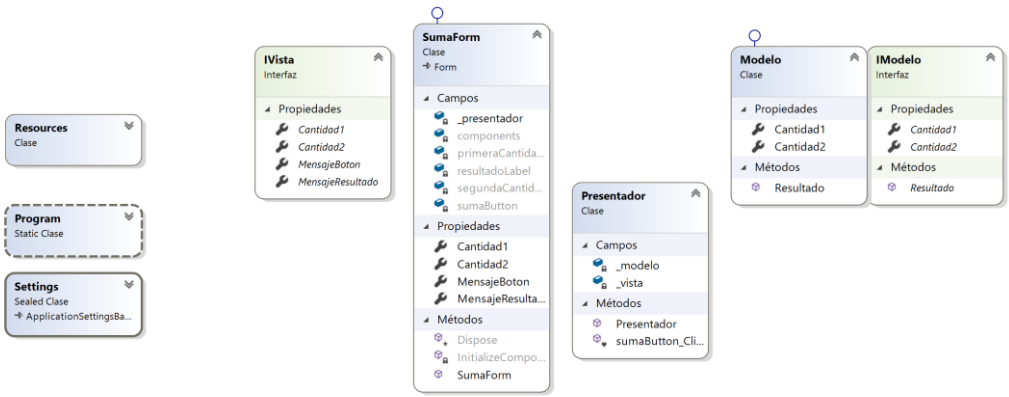
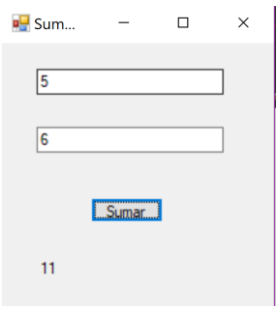


Ilustración 3. Diagrarma de clases de SumaMVPPasivoWindowsFormsApp.

En la ejecución podemos apreciar que directamente se muestra el resultado por parte del Presentador.



2.4. SumaMVPPasivoWindowsFormsApp

Esta segunda aplicación de suma, hace uso del controlador de supervisión, y se encarga de realizar una llamada desde la vista (SumaForm) a Modelo, antes de hacer uso del Presentador para mostrar la suma por pantalla, para posteriormente hacer uso de él y mostrar la suma.

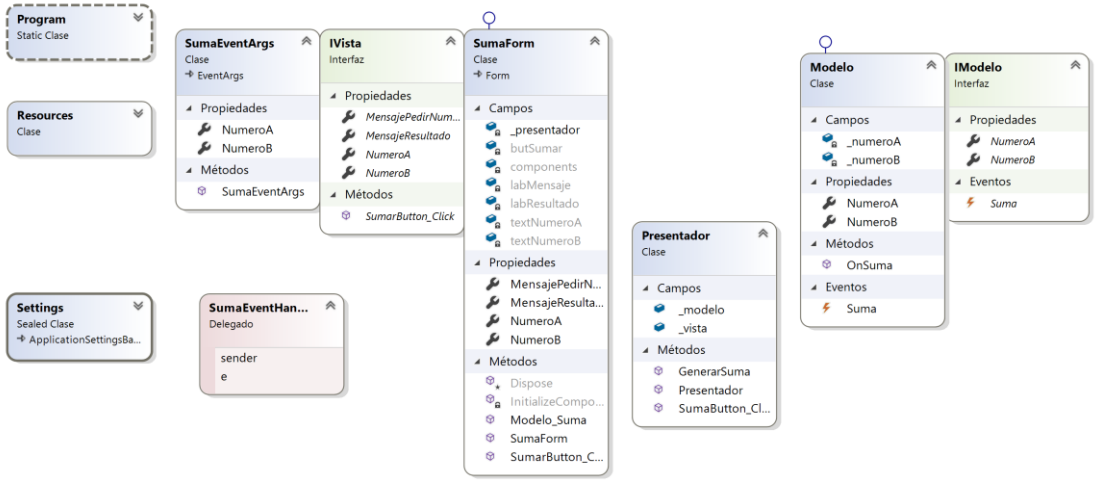


Ilustración 4. Diagrama de clases de SumaMVPPasivoWindowsFormsApp.

En la ejecución podemos observar como mediante el evento Suma, la vista muestra por pantalla “La suma es” para luego llamar al presentador e incorporar el 11.