



Patrón de diseño: Modelo Vista Controlador

Programación Web Avanzada



Conceptos a comprender

Patrón de diseño:

Soluciones probadas, expresivas y fáciles de mantener para resolver problemas comunes en el desarrollo de software.

Arquitectura de software:

“las estructuras de un sistema, compuestas de elementos con propiedades visibles de forma externa y las relaciones que existen entre ellos.”



¿Que es MVC?

Patrón de diseño que separa la lógica de la aplicación, la vista y los datos.

Ventajas:

- Mantenable a largo plazo.
- Pauta para desarrollo colaborativo.
- Estandarización de código
- Cambios independientes de acuerdo al problema




Modelo

Representación de los datos con los que la aplicación va a interactuar.

- Consulta a la base de datos.
- Consulta a apis externas.
- CRUD de datos.

Ejemplo de Modelo



```
7 references
public class Product
{
    2 references | 0 excepciones
    public long Id { get; set; }
    [Required(ErrorMessage = "El nombre es requerido")]

    0 references | 0 excepciones
    public string Name { get; set; }
    [Required(ErrorMessage = "La imagen es requerida")]
    [Url]

    0 references | 0 excepciones
    public string ImageUrl { get; set; }
    0 references | 0 excepciones
    public int Quantity { get; set; }
    0 references | 0 excepciones
    public double Price { get; set; }
    [Required(ErrorMessage = "El codigo es requerido")]
    0 references | 0 excepciones
    public string Code { get; set; }

    0 references | 0 excepciones
    public ICollection<OrderDetail> Orders { get; set; }
```



Vista

Representación gráfica de los datos al usuario.

- **Mostrar los datos al usuario final**
- **Otorgarle una interfaz al usuario para ingresar datos.**
- **Comunicación directa con el controlador.**



Ejemplo de Vista

```
@page  
@model IndexModel  
@{  
    ViewData["Title"] = "Home page";  
}
```

```
<div class="text-center">  
    <h1 class="display-4">Welcome</h1>  
    <p>Hello, world! The time on the  
server is @DateTime.Now</p>  
</div>
```

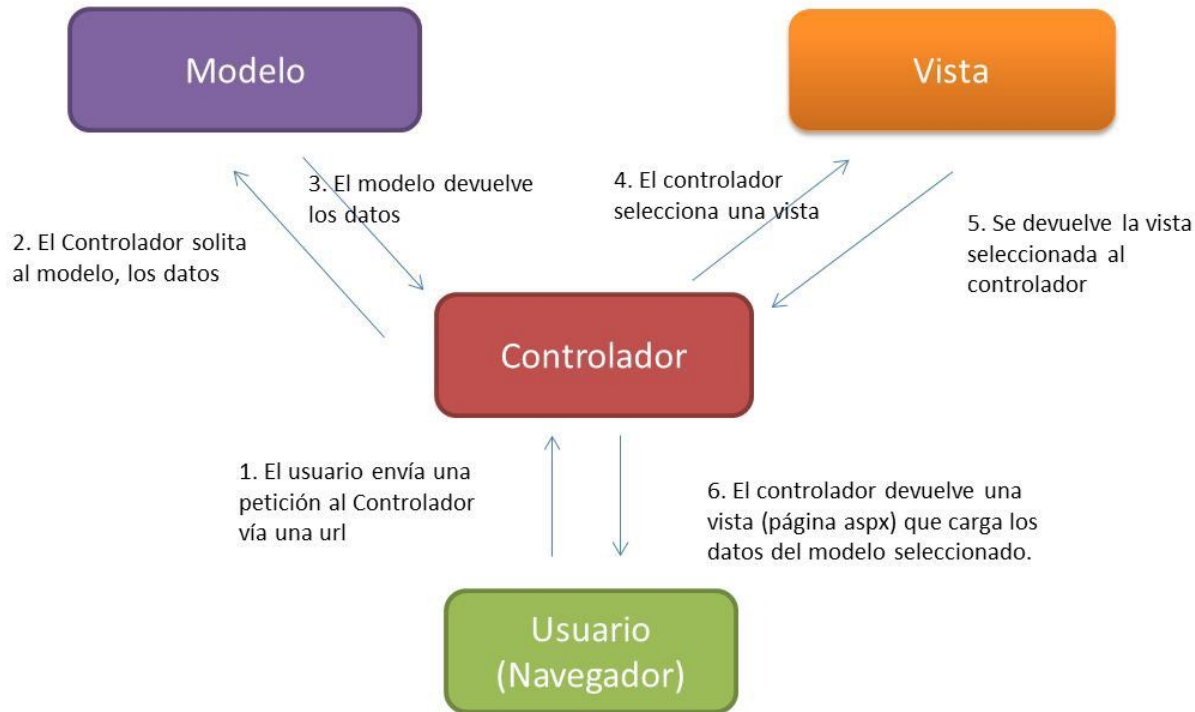


Controlador

Gestiona el flujo de información entre el modelo y la vista.

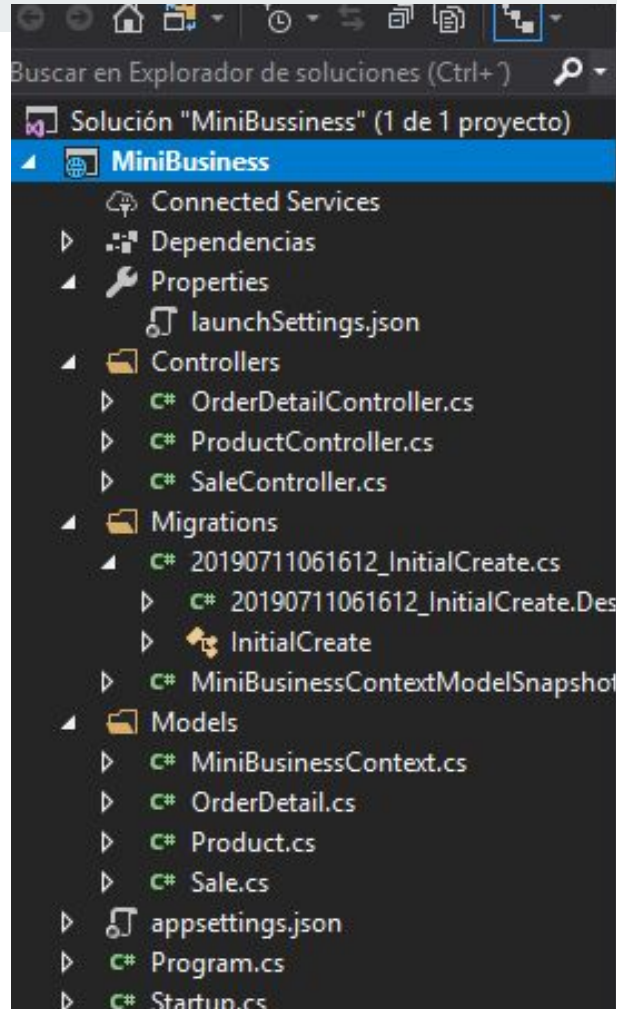
- Contesta peticiones del usuario (vista)
- Responde información estructurada por el modelo.

Ejemplo de petición



Estructura

C# (ASP.NET CORE MVC API)





Fundamentos de MVC

- **Convención sobre configuración**
 - Preocupate por codificar no por configurar :)
- **No Repitas (principio “DRY”)**
 - Su principal ventaja es mantener el código centralizado.



Estado del arte

Lenguaje	Framework	ORM	Motor de plantillas
C#	ASP.NET CORE 2.X	ENTITY	RAZOR
PHP	LARAVEL	ELOQUENT	BLADE
Ruby	RUBY ON RAILS	ACTIVERECORD	SLIM
Python	DJANGO	SQLALCHEMY	JINGA2
Java	SPRING	HIBERNATE	THYMELEAF
JavaScript (Nodejs)	EXPRESS	SEQUELIZE	PUG



Patrones derivados

- MVP - Modelo vista presentador
- MVA - Modelo vista adaptador



¿Que es MVVM? (Modelo - Vista - Vista Modelo)

Patrón de diseño derivado del patrón MVC que intenta separar toda la UI de todo lo demás.

Características:

- Permite realizar con mayor facilidad los test unitarios
- Separación de lógica de negocio de la vista.



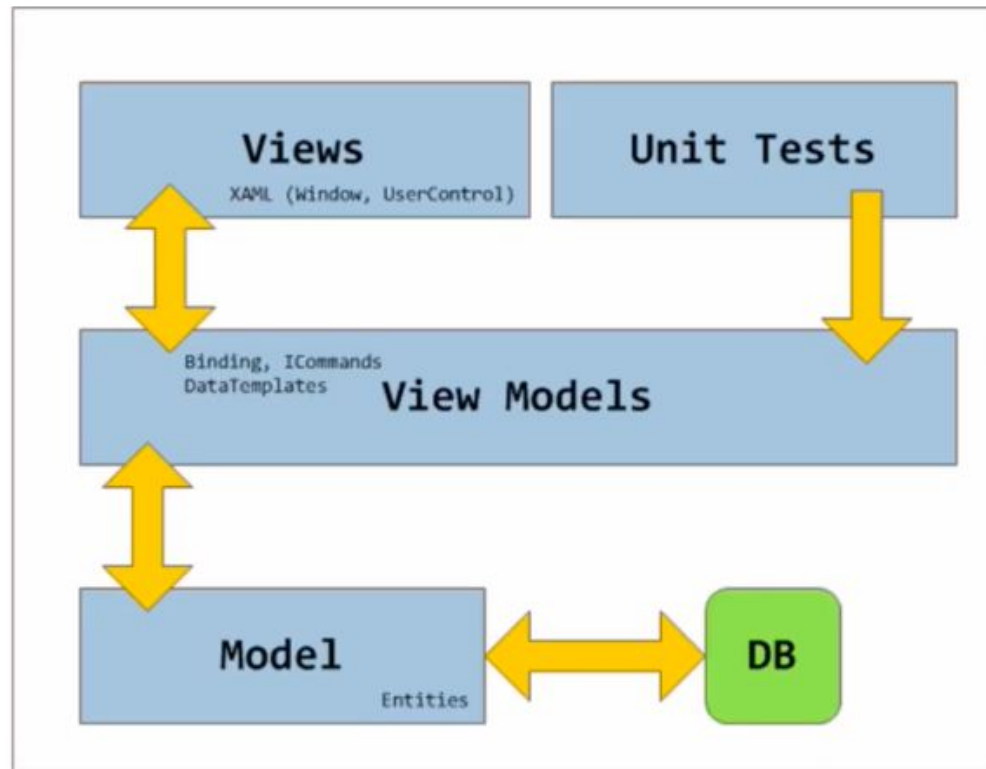
Modelo

Representación de los datos con los que la aplicación va a interactuar.

Vista

Representación gráfica de los datos al usuario.

Esquema de interacción



ACTIVIDAD



Un cliente (Caja de préstamo) tiene la necesidad de una plataforma para gestionar las transacciones de préstamos. La idea es empezar con un Módulo que permita:

- CRUD Clientes (Nombre, Dirección, No. Cliente).
- CRUD Préstamos (Monto, Fecha de pago, más 3 campos que consideres necesarios).

El arquitecto de software debe diseñar el modelo MVC que permita al cliente realizar las acciones mencionadas.

- Define las vistas necesarias.
- Define los controladores que crees necesitar.
- Define los modelos de datos.
- Crea un diagrama de ejemplode una petición.
- Define una estructura de carpetas básica que consideres organizada.