

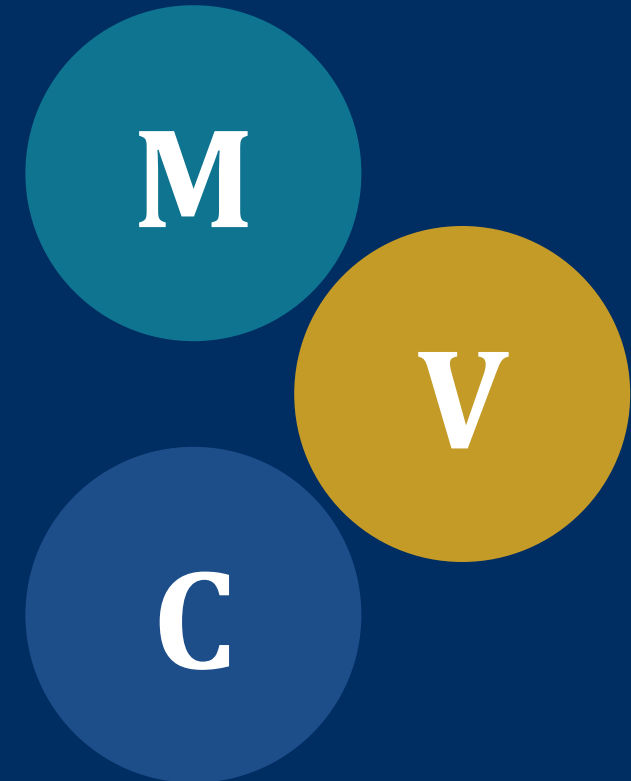
DESARROLLO DE APLICACIONES WEB · UNIDAD 3 · TEMA 3.3

Patrón MVC

Cómo ordenar el back-end de una aplicación web con
Modelo, Vista y Controlador

Dr. Juan Carlos de la Cruz Maldonado

Licenciatura en Tecnologías de la Información · Periodo 2026-3



MOMENTO 1 • EL DILEMA

Un cambio de precio que rompió la facturación

1 archivo

con SQL, HTML y reglas de negocio mezclados en la misma página

3 días

para encontrar por qué la factura cobraba un 10 % de más

```
Productos.cshtml (todo en un archivo)

@{
    var cn = new SqlConnection("Server=...");
    var lista = cn.Query("SELECT * FROM Productos");
    foreach (var p in lista)
    {
        if (p.Stock < 5) p.Precio *= 1.10m; // ¿regla?
        <tr><td>@p.Nombre</td>
            <td>@p.Precio</td></tr>
    }
}
```

Caso simulado: Ferretería del Norte, Cd. Victoria. *¿Quién debería saber cuánto cuesta un producto: la página o el negocio?*

MOMENTO 2 · FUNDAMENTO

MVC: tres responsabilidades, tres lugares



Modelo

Los datos y las reglas del negocio

- Clases como Producto, Venta o Cliente
- Valida y calcula; no sabe nada de HTML

En ASP.NET Core: **Models/**



Vista

Lo que ve el usuario

- Plantillas Razor (.cshtml)
- Muestra datos; nunca consulta la base de datos

En ASP.NET Core: **Views/**



Controlador

El que coordina

- Recibe la petición y decide qué responder
- Pide datos al modelo y se los entrega a la vista

En ASP.NET Core: **Controllers/**

MOMENTO 2 • FUNDAMENTO

El viaje de una petición en ASP.NET Core MVC



Ruta por defecto

convierte la URL en controlador y acción

`{controller=Home}/{action=Index}/{id?}`

MOMENTO 2 • FUNDAMENTO

Todo en la página vs. separar con MVC

Todo en la página

- ✗ Un cambio visual puede romper la lógica
- ✗ Las reglas solo se prueban abriendo el navegador
- ✗ Dos personas no pueden trabajar sin chocar
- ✗ El mismo SQL se repite en cada página

VS

Con MVC

- ✓ Cambias la vista sin tocar las reglas
- ✓ Pruebas unitarias del modelo y del controlador
- ✓ Front-end y back-end avanzan en paralelo
- ✓ Cada consulta vive en un solo lugar

MOMENTO 3 · CASO PRÁCTICO

Ferretería del Norte: el modelo y el controlador

Paso 1 · Modelo

```
Models/Producto.cs

public class Producto
{
    public int Id { get; set; }
    public string Nombre { get; set; } = "";
    public decimal Precio { get; set; }
    public int Stock { get; set; }

    // Regla de negocio: vive en el modelo
    public bool RequiereResurtido
        => Stock < 5;
}
```

Paso 2 · Controlador

```
Controllers/ProductosController.cs

public class ProductosController : Controller
{
    private readonly FerreteriaContext _db;
    public ProductosController(FerreteriaContext db)
        => _db = db;

    // GET /Productos
    public IActionResult Index()
    {
        var productos = _db.Productos
            .OrderBy(p => p.Nombre).ToList();
        return View(productos);
    }
}
```

MOMENTO 3 · CASO PRÁCTICO

Paso 3: la vista muestra, no decide

```

Views/Productos/Index.cshtml

@model IEnumerable<Producto>
<h2>Inventario</h2>
<table class="table">
  @foreach (var p in Model)
  {
    <tr>
      <td>@p.Nombre</td>
      <td>@p.Precio.ToString("C")</td>
      <td>@p.Stock</td>
      <td>@(p.RequiereResurtido
        ? "Resurtir" : "OK")</td>
    </tr>
  }
</table>
  
```

Resultado en el navegador

Inventario

Producto	Precio	Stock	Estado
Cinta métrica 5 m	\$95.50	3	Resurtir
Desarmador plano	\$45.00	20	OK
Martillo 16 oz	\$189.00	12	OK
Pinzas de presión	\$210.00	2	Resurtir

Datos simulados con fines didácticos

MOMENTO 4 • DEBATE

¿Dónde pondrías esta regla?

«No se puede registrar una venta si deja el stock de un producto en negativo.»

A

En la vista

Validar en el formulario
antes de enviar

B

En el controlador

Revisar el stock en la acción
Vender()

C

En el modelo

Un método Descontar() en la
clase Producto

D

En un servicio

Una clase VentasService que
coordina

En equipos, 5 minutos: elijan una opción y defiéndanla con una consecuencia concreta para el sistema.

MOMENTO 5 • SÍNTESIS

Tres ideas para llevarte

01

Cada cosa en su lugar

Datos y reglas en el Modelo,
presentación en la Vista y
coordinación en el Controlador.

02

Controladores delgados

Si un controlador tiene mucha
lógica, esa lógica pertenece al
modelo o a un servicio.

03

La base de tu proyecto

Roles, CRUD y seguridad del
proyecto integrador se organizan
sobre esta estructura.

Para profundizar: Freeman, A. (2023). *Pro ASP.NET Core 7* (10.^a ed.). Manning Publications.

MOMENTO 5 · ACTIVIDAD

Actividad: módulo de Proveedores con MVC

- 1 Crea el modelo Proveedor (Id, Nombre, Ciudad, DiasCredito) con la regla TieneCredito.
- 2 Crea ProveedoresController con la acción Index.
- 3 Crea la vista Index.cshtml que liste los proveedores.
- 4 Entrega el proyecto, una captura y tu declaración de uso de IA (herramienta y prompts).

Criterio (%)	Excelente	Bueno	Básico	Insuficiente
Modelo (25)	Propiedades y regla correctas	Regla con fallas menores	Sin regla de negocio	No compila
Controlador (25)	Delgado; usa el modelo	Funciona con lógica de más	Acción incompleta	No existe
Vista (25)	Lista completa, sin reglas	Lista con detalles menores	Datos incompletos	No se muestra
Separación y entrega (25)	Cada cosa en su lugar; IA declarada	Una mezcla menor	Varias mezclas	Entrega incompleta

Niveles: Excelente 100 % · Bueno 80 % · Básico 60 % · Insuficiente 0–40 % del valor del criterio