Herramientas de Programación 2

Tema Nº3

CONTROLADOR MVC (PARTE 2)

Indicador de logro Nº3:Desarrolla las aplicaciones web con Scaffolding usando las partes del proyecto aprendido.

**TEMA 01 Teoría de los**

Imagen que contiene Icono

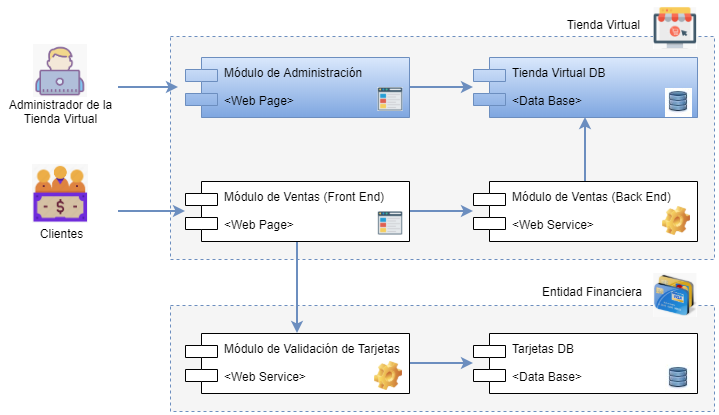
Descripción generada automáticamente

**TEMA Nº3:**

Controlador MVC (PARTE 2)

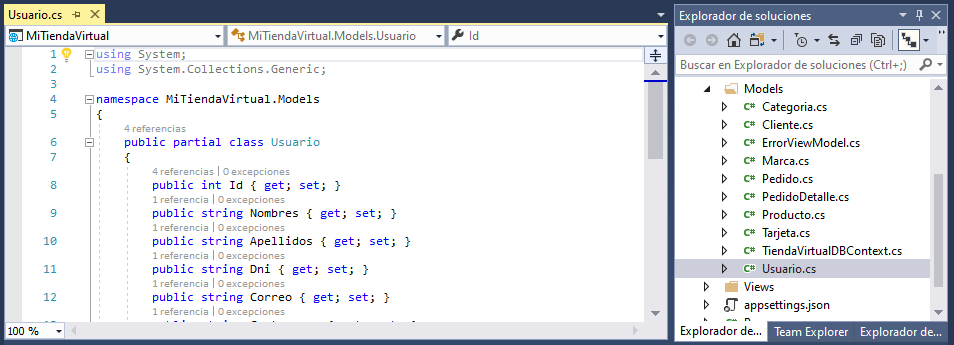
**Subtema 3.2:**

Proyecto Grupal de Administración



Paso 1. Implementar validaciones en el CRUD de Usuarios.

1. Abrimos la solución creada en el taller anterior.
2. Vamos a implementar las validaciones desde el Modelo, abrimos el archivo *Usuario.cs*.



1. Para validar que los campos Nombres, Apellidos y Dni sean obligatorios modificamos el código de acuerdo a:

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

namespace MiTiendaVirtual.Models

{

public partial class Usuario

{

public int Id { get; set; }

[Required]

public string Nombres { get; set; }

[Required]

public string Apellidos { get; set; }

[Required]

public string Dni { get; set; }

public string Correo { get; set; }

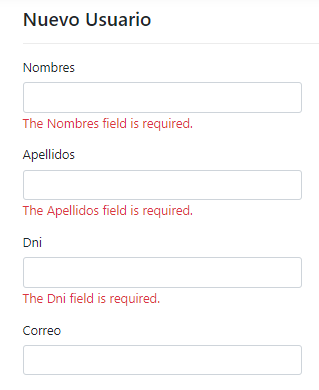
public string Contrasena { get; set; }

public bool? Activo { get; set; }

}

}

1. Ejecutamos la aplicación e ingresamos al formulario donde creamos un nuevo Usuario, sin ingresar dato alguno presionamos el botón *Guardar* y obtenemos:



1. Para limitar la cantidad máxima de caracteres a ingresar en cada campo modificamos el código de acuerdo a:

public partial class Usuario

{

public int Id { get; set; }

[Required]

[MaxLength(50)]

public string Nombres { get; set; }

[Required]

[MaxLength(50)]

public string Apellidos { get; set; }

[Required]

[MaxLength(8)]

public string Dni { get; set; }

[MaxLength(50)]

public string Correo { get; set; }

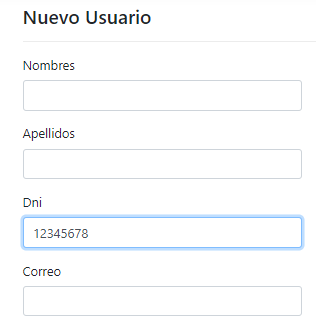
[MaxLength(20)]

public string Contrasena { get; set; }

public bool? Activo { get; set; }

}

1. Ejecutamos la aplicación para verificar los cambios, por ejemplo en el campo *Dni* no podemos escribir más de 8 caracteres.



1. Para validar el formato del campo *Correo* modificamos el código de acuerdo a:

public partial class Usuario

{

public int Id { get; set; }

[Required]

[MaxLength(50)]

public string Nombres { get; set; }

[Required]

[MaxLength(50)]

public string Apellidos { get; set; }

[Required]

[MaxLength(8)]

public string Dni { get; set; }

[MaxLength(50)]

[RegularExpression(@"^[a-zA-Z0-9\_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+$")]

public string Correo { get; set; }

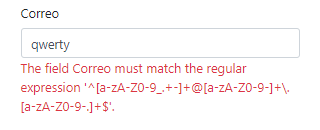
[MaxLength(20)]

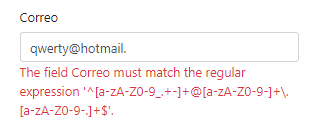
public string Contrasena { get; set; }

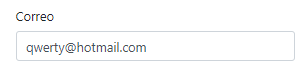
public bool? Activo { get; set; }

}

1. Ejecutamos la aplicación y verificamos los cambios.







1. Ahora vamos a personalizar los mensajes, modificamos el código de acuerdo a:

public partial class Usuario

{

public int Id { get; set; }

[Required(ErrorMessage = "El campo 'Nombres' es obligatorio")]

[MaxLength(50)]

public string Nombres { get; set; }

[Required(ErrorMessage = "El campo 'Apellidos' es obligatorio")]

[MaxLength(50)]

public string Apellidos { get; set; }

[Required(ErrorMessage = "El campo 'Dni' es obligatorio")]

[MaxLength(8)]

public string Dni { get; set; }

[MaxLength(50)]

[RegularExpression(@"^[a-zA-Z0-9\_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+$", ErrorMessage = "Ingrese el formato correcto.")]

public string Correo { get; set; }

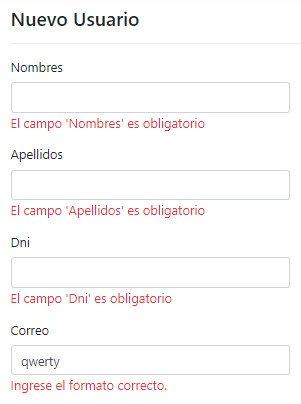
[MaxLength(20)]

public string Contrasena { get; set; }

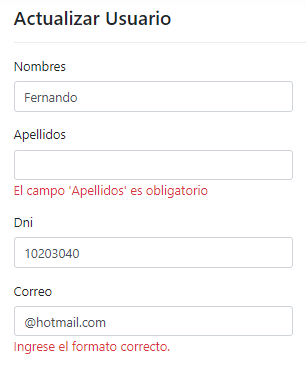
public bool? Activo { get; set; }

}

1. Ejecutamos la aplicación y verificamos los cambios.

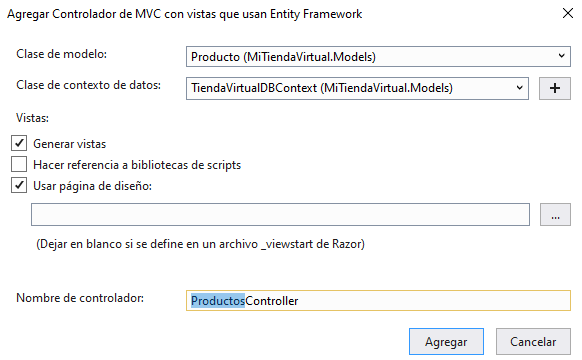


1. Verificamos que las validaciones también se dan en la actualización de datos del Usuario.



Paso 2. CRUD de la entidad (tabla) *Producto*.

1. Abrimos la solución creada en el taller anterior.
2. Vamos a generar, de forma automática, El Controlador y las Vistas que nos permitan hacer el mantenimiento de la tabla *Producto*, de la misma forma que hicimos para la tabla *Usuario*.



1. Editamos el archivo *\_Layout.cshtml*, enlazamos la opción de menú *Productos* a la vista que lista los Productos.
2. Personalizamos el archivo *Index.cshtml* de acuerdo a:

@model IEnumerable<MiTiendaVirtual.Models.Producto>

@{

ViewData["Title"] = "Productos";

}

<h4>Productos</h4>

<p>

<**a** **asp-action**="Create">Nuevo Producto</**a**>

</p>

<table class="table table-hover table-sm">

<thead class="thead-dark">

<tr>

<th>

Producto

</th>

<th>

Descripción

</th>

<th>

@Html.DisplayNameFor(model => model.Precio)

</th>

<th>

@Html.DisplayNameFor(model => model.Destacado)

</th>

<th>

@Html.DisplayNameFor(model => model.Activo)

</th>

<th>

Marca

</th>

<th>

Categoría

</th>

<th></th>

</tr>

</thead>

<tbody>

@foreach (var item in Model) {

<tr>

<td>

@Html.DisplayFor(modelItem => item.Nombre)

</td>

<td>

@Html.DisplayFor(modelItem => item.Descripcion)

</td>

<td>

@Html.DisplayFor(modelItem => item.Precio)

</td>

<td>

<input type="checkbox" checked="@item.Destacado.GetValueOrDefault()" disabled>

</td>

<td>

<input type="checkbox" checked="@item.Activo.GetValueOrDefault()" disabled>

</td>

<td>

@Html.DisplayFor(modelItem => item.IdMarcaNavigation.Nombre)

</td>

<td>

@Html.DisplayFor(modelItem => item.IdCategoriaNavigation.Nombre)

</td>

<td>

<**a** **asp-action**="Edit" **asp-route-id**="@item.Id">Actualizar</**a**> |

<**a** **asp-action**="Delete" **asp-route-id**="@item.Id">Eliminar</**a**>

</td>

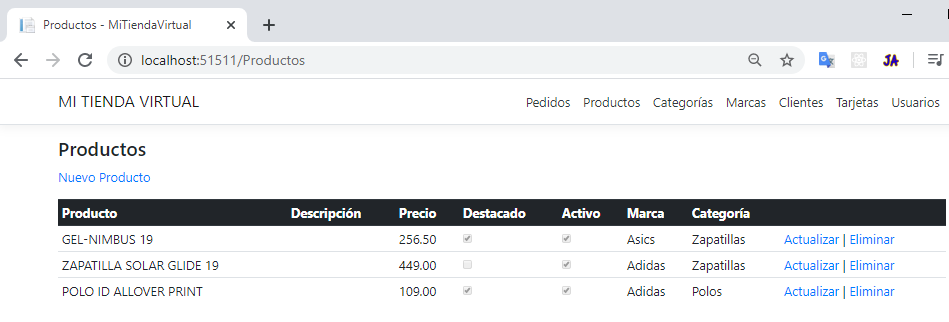
</tr>

}

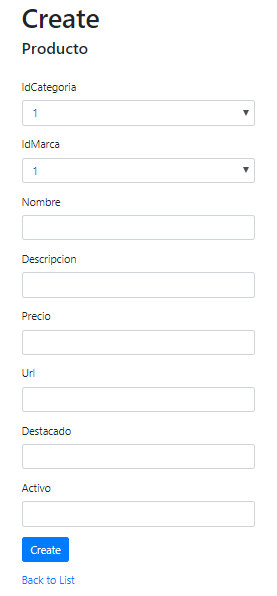
</tbody>

</table>

1. Ejecutamos la aplicación y verificamos los cambios.



1. Hacemos click en el link Nuevo Producto.



1. Personalizamos esta vista (*Create.cshtml*) de acuerdo a.

@model MiTiendaVirtual.Models.Producto

@{

ViewData["Title"] = "Nuevo Producto";

}

<h4>Nuevo Producto</h4>

<hr />

<**form** **asp-action**="Create">

<div class="form-row">

<**div** **asp-validation-summary**="ModelOnly" class="text-danger"></**div**>

<div class="form-group col-md-8">

<**label** **asp-for**="Nombre" class="control-label"></**label**>

<**input** **asp-for**="Nombre" class="form-control" />

<**span** **asp-validation-for**="Nombre" class="text-danger"></**span**>

</div>

<div class="form-group col-md-4">

<**label** **asp-for**="IdCategoria" class="control-label">Categoría</**label**>

<**select** **asp-for**="IdCategoria" class="form-control" **asp-items**="ViewBag.IdCategoria"></**select**>

</div>

</div>

<div class="form-row">

<div class="form-group col-md-8">

<**label** **asp-for**="Descripcion" class="control-label">Descripción</**label**>

<**input** **asp-for**="Descripcion" class="form-control" />

<**span** **asp-validation-for**="Descripcion" class="text-danger"></**span**>

</div>

<div class="form-group col-md-4">

<**label** **asp-for**="IdMarca" class="control-label">Marca</**label**>

<**select** **asp-for**="IdMarca" class="form-control" **asp-items**="ViewBag.IdMarca"></**select**>

</div>

</div>

<div class="form-row">

<div class="form-group col-md-8">

<**label** **asp-for**="Precio" class="control-label"></**label**>

<**input** **asp-for**="Precio" class="form-control" />

<**span** **asp-validation-for**="Precio" class="text-danger"></**span**>

</div>

<div class="form-group col-md-4">

<**label** **asp-for**="Destacado" class="control-label"></**label**><br />

@Html.CheckBox("Destacado")

</div>

</div>

<div class="form-row">

<div class="form-group col-md-8">

<**label** **asp-for**="Url" class="control-label"></**label**>

<**input** **asp-for**="Url" class="form-control" />

<**span** **asp-validation-for**="Url" class="text-danger"></**span**>

</div>

<div class="form-group col-md-4">

<**label** **asp-for**="Activo" class="control-label"></**label**><br />

@Html.CheckBox("Activo")

</div>

</div>

<div class="form-group">

<input type="submit" value="Guardar" class="btn btn-primary" /> |

<**a** **asp-action**="Index">Cancelar</**a**>

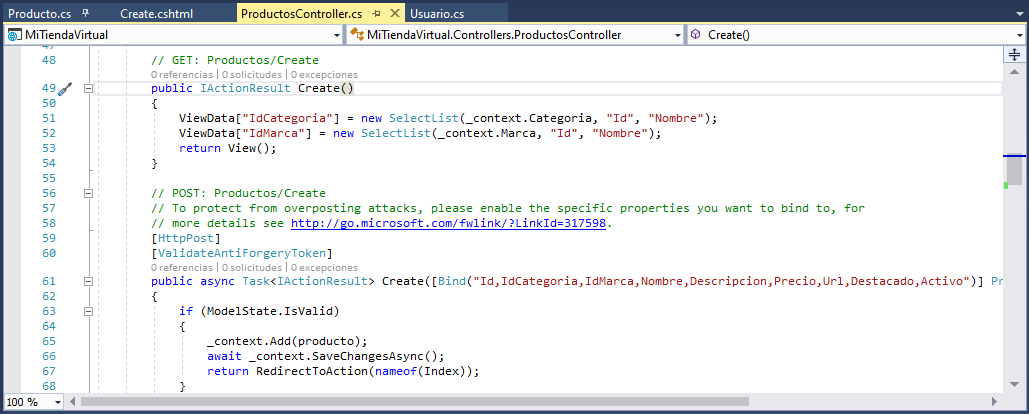
</div>

</**form**>

1. Ejecutamos la aplicación y verificamos los cambios.



1. Observamos que en los campos *Categoría* y *Marca* se listan los *Id* y no los *Nombres*, para corregir esto abrimos el controlador *ProductosController.cs* y nos ubicaos en las funciones *Create().*



1. Modificamos el código de ambas funciones de acuerdo a:

// GET: Productos/Create

public IActionResult Create()

{

ViewData["IdCategoria"] = new SelectList(\_context.Categoria, "Id", "Nombre");

ViewData["IdMarca"] = new SelectList(\_context.Marca, "Id", "Nombre");

return View();

}

// POST: Productos/Create

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Create([Bind("Id,IdCategoria,IdMarca,Nombre,Descripcion,Precio,Url,Destacado,Activo")] Producto producto)

{

if (ModelState.IsValid)

{

\_context.Add(producto);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

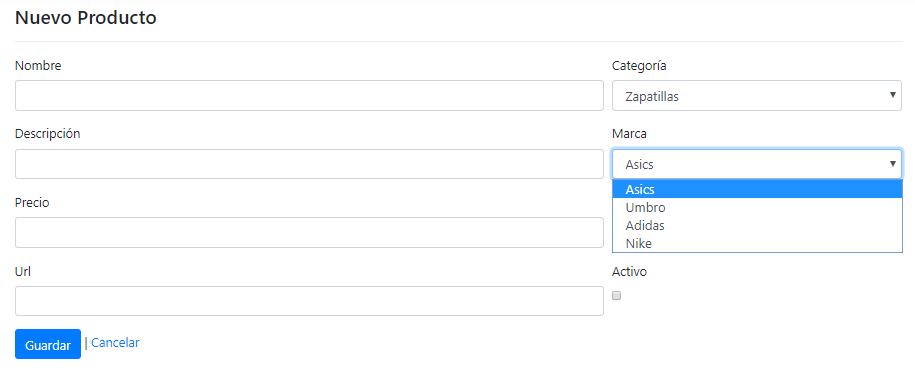
ViewData["IdCategoria"] = new SelectList(\_context.Categoria, "Id", "Nombre", producto.IdCategoria);

ViewData["IdMarca"] = new SelectList(\_context.Marca, "Id", "Nombre", producto.IdMarca);

return View(producto);

}

1. Ejecutamos la aplicación.



1. Agregamos las validaciones en el Modelo (*Producto.cs*) de acuerdo a:

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

namespace MiTiendaVirtual.Models

{

public partial class Producto

{

public Producto()

{

PedidoDetalle = new HashSet<PedidoDetalle>();

}

public int Id { get; set; }

public int? IdCategoria { get; set; }

public int? IdMarca { get; set; }

[Required(ErrorMessage = "El campo 'Nombre' es obligatorio.")]

[MaxLength(50)]

public string Nombre { get; set; }

[Required(ErrorMessage = "El campo 'Descripcion' es obligatorio.")]

[MaxLength(50)]

public string Descripcion { get; set; }

[Required(ErrorMessage = "El campo 'Precio' es obligatorio.")]

public decimal? Precio { get; set; }

[Required(ErrorMessage = "El campo 'Url' es obligatorio.")]

public string Url { get; set; }

public bool? Destacado { get; set; }

public bool? Activo { get; set; }

public virtual Categoria IdCategoriaNavigation { get; set; }

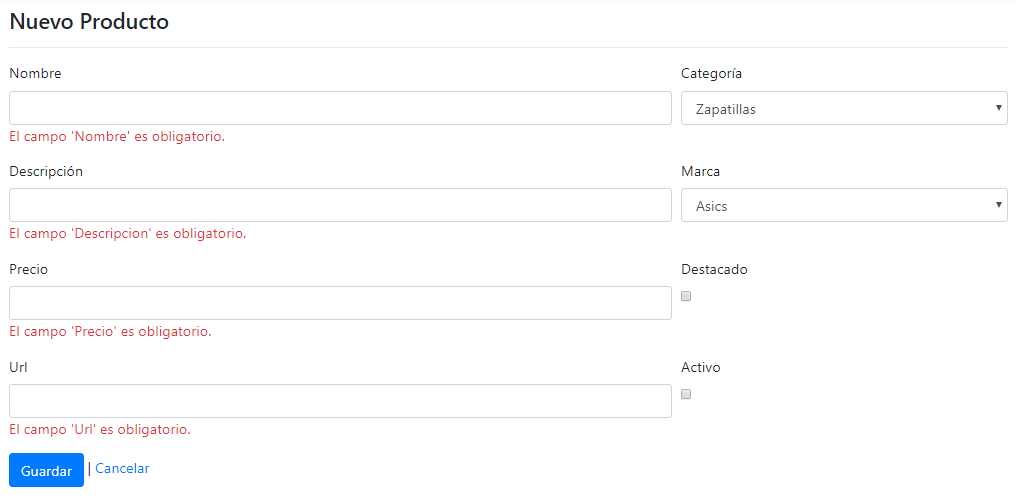
public virtual Marca IdMarcaNavigation { get; set; }

public virtual ICollection<PedidoDetalle> PedidoDetalle { get; set; }

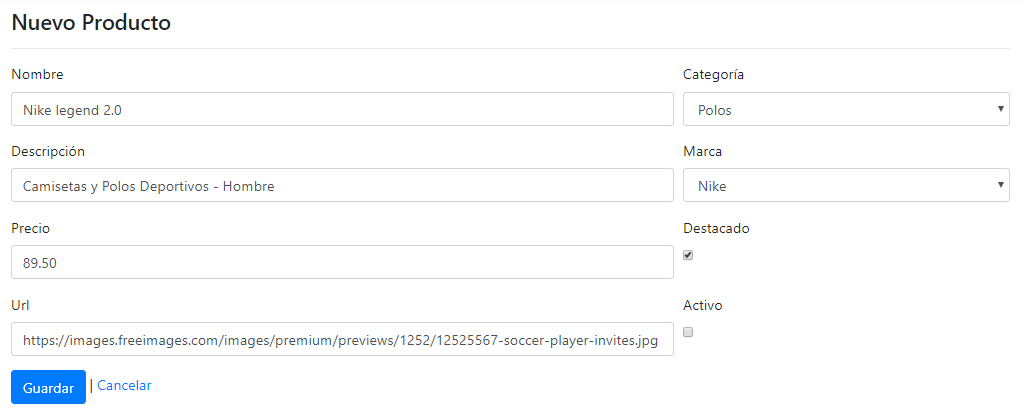
}

}

1. Ejecutamos la aplicación para verificar los cambios.



1. Agregamos un nuevo Producto.





ACTIVIDAD VIRTUAL:

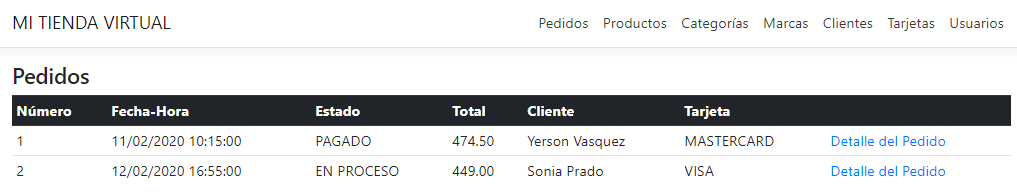
Ingresa a la plataforma virtual. Revisa y analiza el tema desarrollado en esta sesión, luego desarrolla las siguientes actividades propuestas:

1. CUESTIONARIO TÉCNICO
   * 1. Para completar el CRUD de Productos personalice las Vistas para:

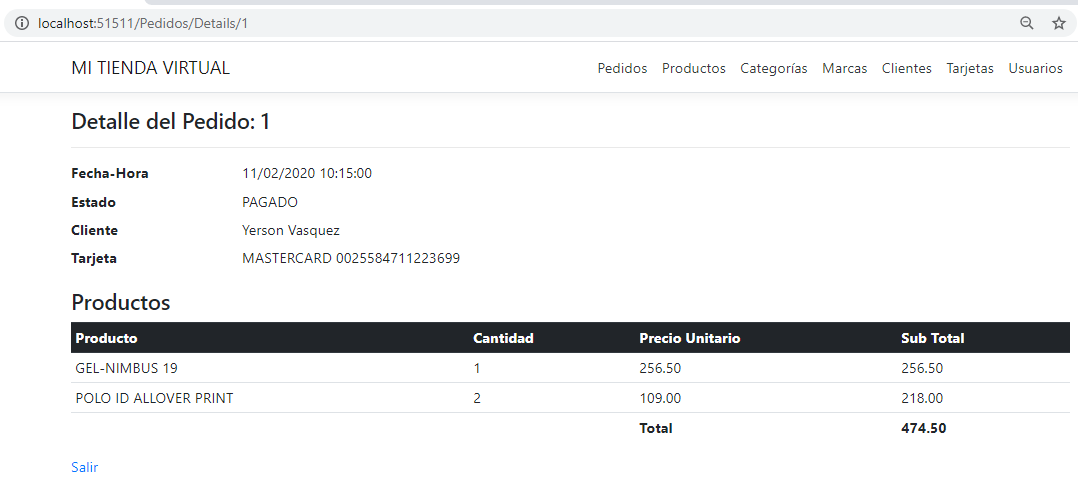
* Actualizar
* Eliminar
  + 1. Agregar validaciones en el mantenimiento de todas las entidades (tablas) del módulo de Administración, tanto al crear y actualizar datos.
    2. Crear con Scaffold Controladores y vistas para el mantenimiento (CRUD) de la entidad (Tabla) *Pedido*.

Esta entidad debe ser de solo lectura, esto es, no debe permitir crear, actualizar ni eliminar Pedidos.

En la lista de Pedidos, por cada Pedido debe agregar el link *Detalle del Pedido* cuya funcionalidad se explica en la siguiente actividad.



* + 1. Acondicionar en el Controlador *PedidosController.cs* una función que permita ver el detalle de un Pedido, la vista del *Detalle del Pedido* debe tener el siguiente diseño.



* + 1. Investigar la forma de agregar la funcionalidad de acceso al sistema de Administración, esto es, al iniciar la aplicación debe mostrarse la siguiente página:



Debe validarse el Correo y Contraseña (encriptada) con las que están en la tabla Usuario.

1. CONCLUSIONES DE LA EXPERIENCIA

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ACTIVIDAD** **VIRTUAL**:

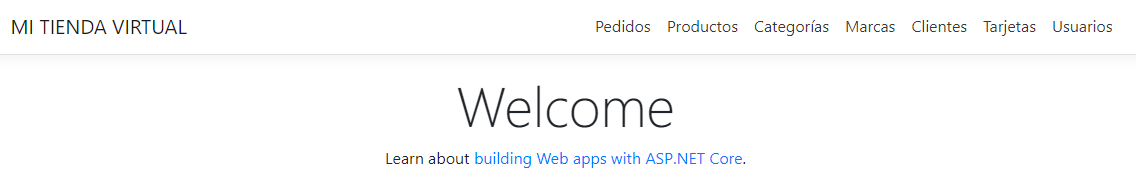
Ingresa a la plataforma virtual. Revisa y analiza el tema desarrollado en esta sesión, luego desarrolla las siguientes actividades propuestas:

1. CUESTIONARIO TÉCNICO
   * 1. Crear con Scaffold Controladores y vistas para el mantenimiento (CRUD) de las siguientes entidades (Tablas):

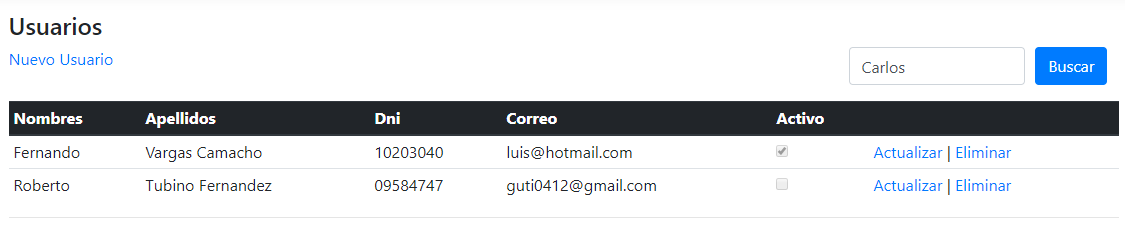
* Marca
* Categoría
* Producto
  + 1. Crear con Scaffold Controladores y vistas para el mantenimiento (CRUD) de las siguientes entidades (Tablas):
* Cliente
* Tarjeta

Estas dos entidades deben ser de solo lectura, esto es, no debe permitir crear, actualizar ni eliminar registros.

* + 1. Para los nuevos mantenimientos, desarrollar las personalizaciones de forma análoga al CRUD de Usuarios.
    2. Agregar en el menú superior links que permitan acceder a los nuevos mantenimientos. Además, alinear el menú a la derecha.



* + 1. Investigar la forma de agregar la funcionalidad de búsqueda en la lista de Usuarios, debe buscar por Nombres y Apellidos.



1. CONCLUSIONES DE LA EXPERIENCIA

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Un motor paso a paso siempre debemos considerar su corriente de operación y los grados, como indica el ejemplo de la siguiente placa característica de la imagen mostrada:

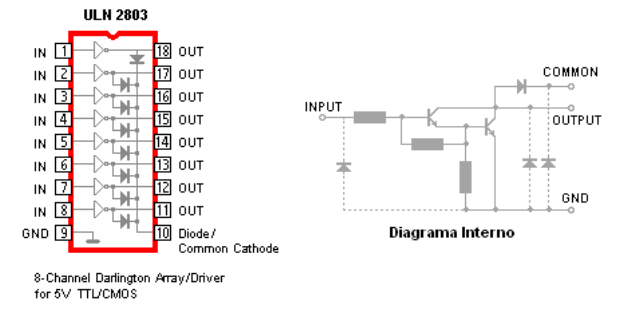


**Subtema 1.2:**

Transistores de potencia

Para operaciones con motores de bajo consumo tenemos circuitos integrados como el circuito ULN2803, el cual es un driver con transistores darlintong , con soporte de hasta 500 mA.

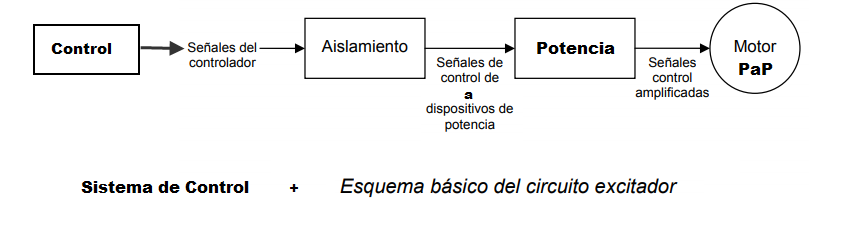
Por lo tanto si deseamos mayores corrientes debemos utilizar otros drivers o implementar uno concomponentes discretos , es decir con transistores de potencia.



**Subtema 1.3:**

Control del Motor

Las etapas para controlar un motor Paso a Paso



**Actividad:**

Los estudiantes desarrollarán ejercicios de Teoría de exponentes y radicales, propuestos por el docente. según lo aprendido en clase. a) CUESTIONARIO TÉCNICO

• ¿Qué diferencia hay entre motores paso a paso unipolares y bipolares?

• ¿Por qué es importante usar driver para motores paso a paso?

• ¿Qué características posee un motor paso a paso?

• ¿Para qué me sirve el puente H?

• Investigue sobre cómo controlar los motores paso a paso con PWM y LOGO! 8.