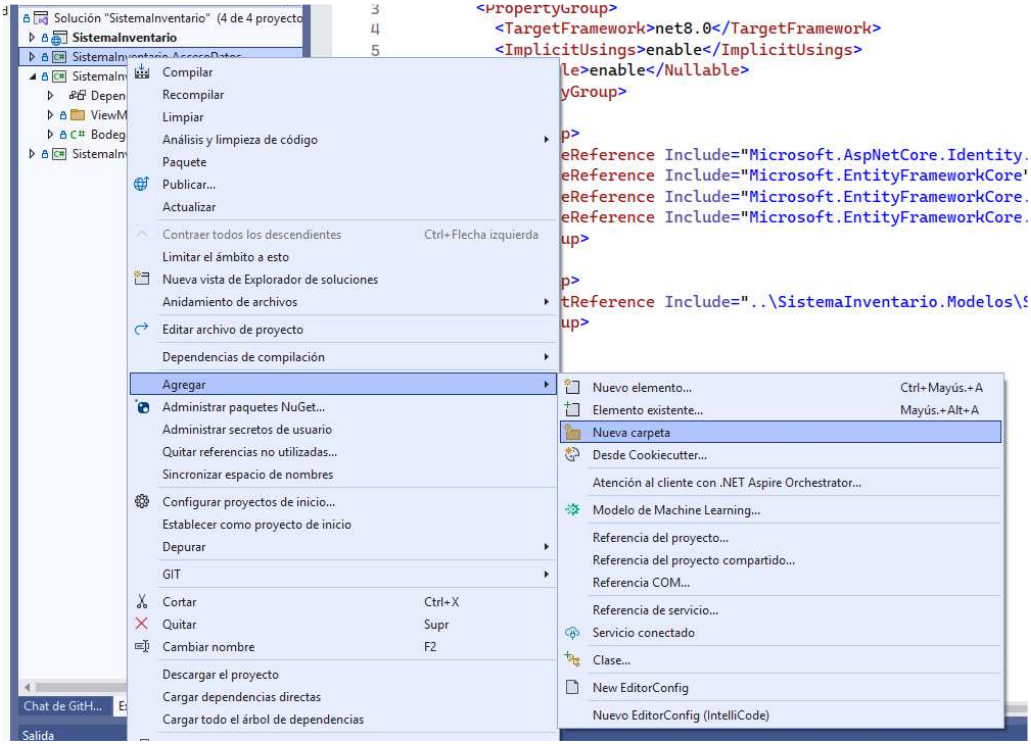
PROYECTO ASP.Net parte 2

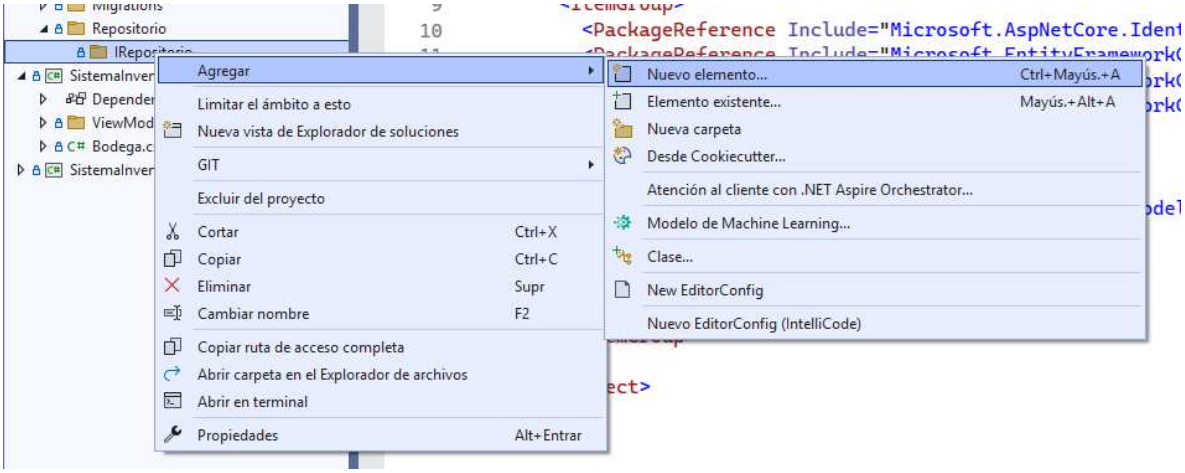
Patrón de Repositorio

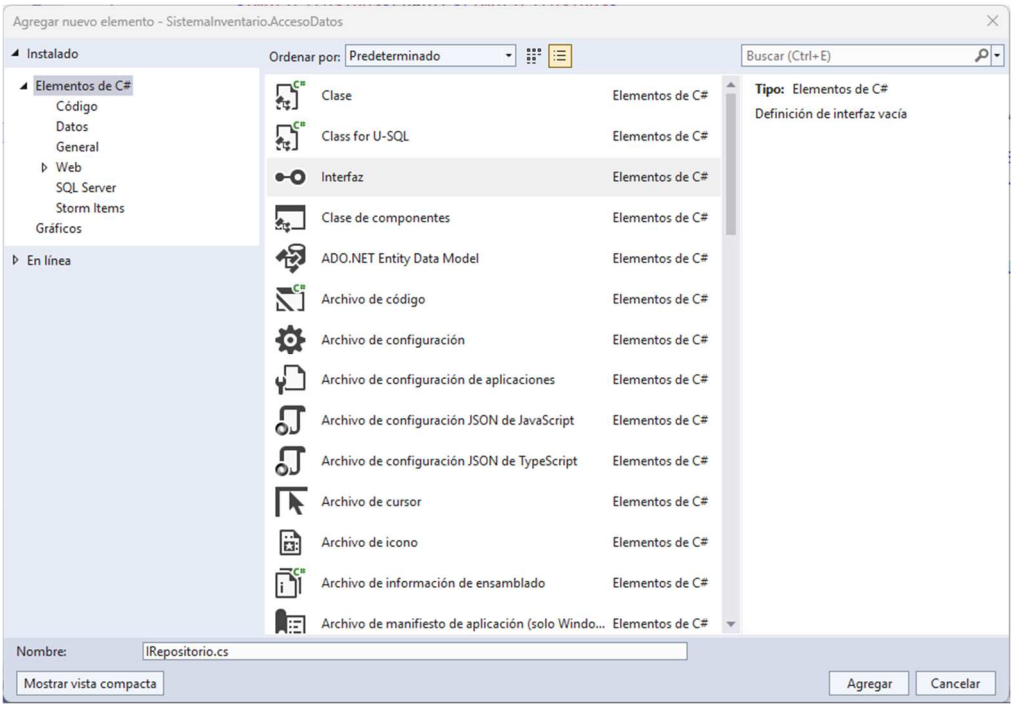
1. Interfaz Repositorio Generico

- Creamos una carpeta llamada Repositorio en el proyecto SistemaInventario.AccesoDatos



- Dentro de Repositorio creamos carpeta llamada IRepositorio, dentro de esta carpeta creamos la primera interfaz genérica del proyecto llamada IRepositorio.cs.






```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Linq.Expressions;
using System.Text;
using System.Threading.Tasks;

namespace SistemaInventario.AccesoDatos.Repositorio.IRepositorio
{
    public interface IRepositorio<T> where T : class
    {
        Task<T> Obtener(int id);

        Task<IEnumerable<T>> ObtenerTodos(
            Expression<Func<T, bool>> filtro = null,
```

```
        Func<IQueryable<T>, IOrderedQueryable<T>> orderBy = null,
        string incluirPropiedades = null,
        bool isTracking = true
        );

    Task<T> ObtenerPrimero(
        Expression<Func<T, bool>> filtro = null,
        string incluirPropiedades = null,
        bool isTracking = true
        );

    Task Agregar(T entidad);

    void Remover(T entidad);

    void RemoverRango(IEnumerable<T> entidad);

  }
}
```

2. Implementar Interfaz de Repositorio Generico

- En la carpeta Repositorio agregamos una clase llamada Repositorio.cs

```
using Microsoft.EntityFrameworkCore;
using SistemaInventario.AccesoDatos.Data;
using SistemaInventario.AccesoDatos.Repositorio.IRepositorio;
//using SistemaInventario.Modelos.Especificaciones;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Linq.Expressions;
using System.Text;
using System.Threading.Tasks;

namespace SistemaInventario.AccesoDatos.Repositorio
{
  public class Repositorio<T> : IRepositorio<T> where T : class
  {

    private readonly ApplicationDbContext _db;
    internal DbSet<T> dbSet;

    public Repositorio(ApplicationDbContext db)
    {
      _db = db;
      this.dbSet = _db.Set<T>();
    }


    public async Task Agregar(T entidad)
    {
```

```
        await dbSet.AddAsync(entidad);    // insert into Table
    }

    public async Task<T> Obtener(int id)
    {
        return await dbSet.FindAsync(id);    // select * from (Solo por Id)
    }


    public async Task<IEnumerable<T>> ObtenerTodos(Expression<Func<T, bool>> filtro = null,
        Func<IQueryable<T>, IOrderedQueryable<T>> orderBy = null, string incluirPropiedades = null, bool
isTracking = true)
    {
        IQueryable<T> query = dbSet;
        if (filtro != null)
        {
            query = query.Where(filtro);   //  select /* from where ....
        }
        if (incluirPropiedades != null)
        {
            foreach (var incluirProp in incluirPropiedades.Split(new char[] { ',' },
StringSplitOptions.RemoveEmptyEntries))
            {
                query = query.Include(incluirProp);    //  ejemplo "Categoria,Marca"
            }
        }
        if (orderBy != null)
        {
            query = orderBy(query);
        }
        if (!isTracking)
        {
            query = query.AsNoTracking();
        }
        return await query.ToListAsync();

    }

    public async Task<T> ObtenerPrimero(Expression<Func<T, bool>> filtro = null,
        string incluirPropiedades = null, bool isTracking = true)
    {

        IQueryable<T> query = dbSet;
        if (filtro != null)
        {
            query = query.Where(filtro);   //  select /* from where ....
        }
        if (incluirPropiedades != null)
        {
            foreach (var incluirProp in incluirPropiedades.Split(new char[] { ',' },
StringSplitOptions.RemoveEmptyEntries))
            {
                query = query.Include(incluirProp);    //  ejemplo "Categoria,Marca"
```

```
            }
        }

        if (!isTracking)
        {
            query = query.AsNoTracking();
        }
        return await query.FirstOrDefaultAsync();
    }



    public void Remover(T entidad)
    {
        dbSet.Remove(entidad);
    }

    public void RemoverRango(IEnumerable<T> entidad)
    {
        dbSet.RemoveRange(entidad);
    }


  }
}
```

3. Interfaz y Repositorio Bodega

- En la carpeta Repositorio/IRepositorio agregamos una interfaz llamada IBodegaRepositorio.cs

```
using SistemaInventario.Modelos;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SistemaInventario.AccesoDatos.Repositorio.IRepositorio
{
  public interface IBodegaRepositorio : IRepositorio<Bodega>
  {
    void Actualizar(Bodega bodega);
  }
}
```

- En la carpeta Repositorio agregamos una clase llamada BodegaRepositorio.cs

```
using SistemaInventario.AccesoDatos.Data;
using SistemaInventario.AccesoDatos.Repositorio.IRepositorio;
using SistemaInventario.Modelos;
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SistemaInventario.AccesoDatos.Repositorio
{
   public class BodegaRepositorio : Repositorio<Bodega>, IBodegaRepositorio
   {

      private readonly ApplicationDbContext _db;

      public BodegaRepositorio(ApplicationDbContext db) : base(db)
      {
         _db = db;
      }

      public void Actualizar(Bodega bodega)
      {
         var bodegaBD = _db.Bodegas.FirstOrDefault(b => b.Id == bodega.Id);
         if (bodegaBD != null)
         {
            bodegaBD.Nombre = bodega.Nombre;
            bodegaBD.Descripcion = bodega.Descripcion;
            bodegaBD.Estado = bodega.Estado;
            _db.SaveChanges();
         }
      }
   }
}
```

4. Unidad de Trabajo

- En la carpeta Repositorio/IRepositorio agregamos una interfaz llamada IUnidadtrabajo.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SistemaInventario.AccesoDatos.Repositorio.IRepositorio
{
    public interface IUnidadTrabajo : IDisposable
    {
        IBodegaRepositorio Bodega { get; }

        Task Guardar();
```

```
    }
}
```

- En la carpeta Repositorio agregamos una clase llamada Unidadtrabajo.cs

```
using SistemaInventario.AccesoDatos.Data;
using SistemaInventario.AccesoDatos.Repositorio.IRepositorio;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SistemaInventario.AccesoDatos.Repositorio
{
    public class UnidadTrabajo : IUnidadTrabajo
    {
        private readonly ApplicationDbContext _db;

        public IBodegaRepositorio Bodega { get; private set; }

        public UnidadTrabajo(ApplicationDbContext db)
        {
            _db = db;
            Bodega = new BodegaRepositorio(_db);
        }

        public void Dispose()
        {
            _db.Dispose();
        }

        public async Task Guardar()
        {
            await _db.SaveChangesAsync();
        }
    }
}
```

- Agregamos Unidadtrabajo.cs como un servicio en Program.cs.

```
builder.Services.AddScoped<IUnidadTrabajo, UnidadTrabajo>();
```

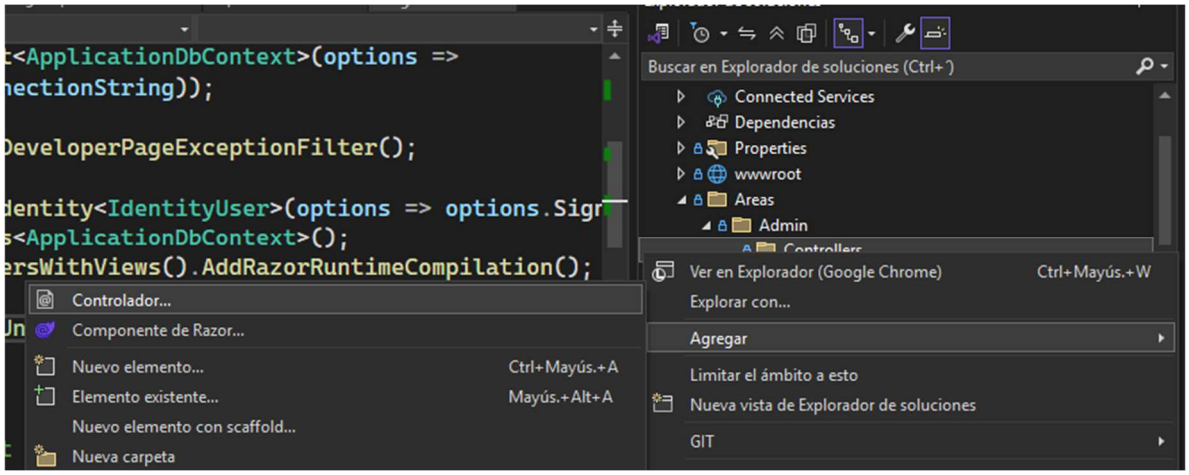
5. Subir cambios a github (opcional)

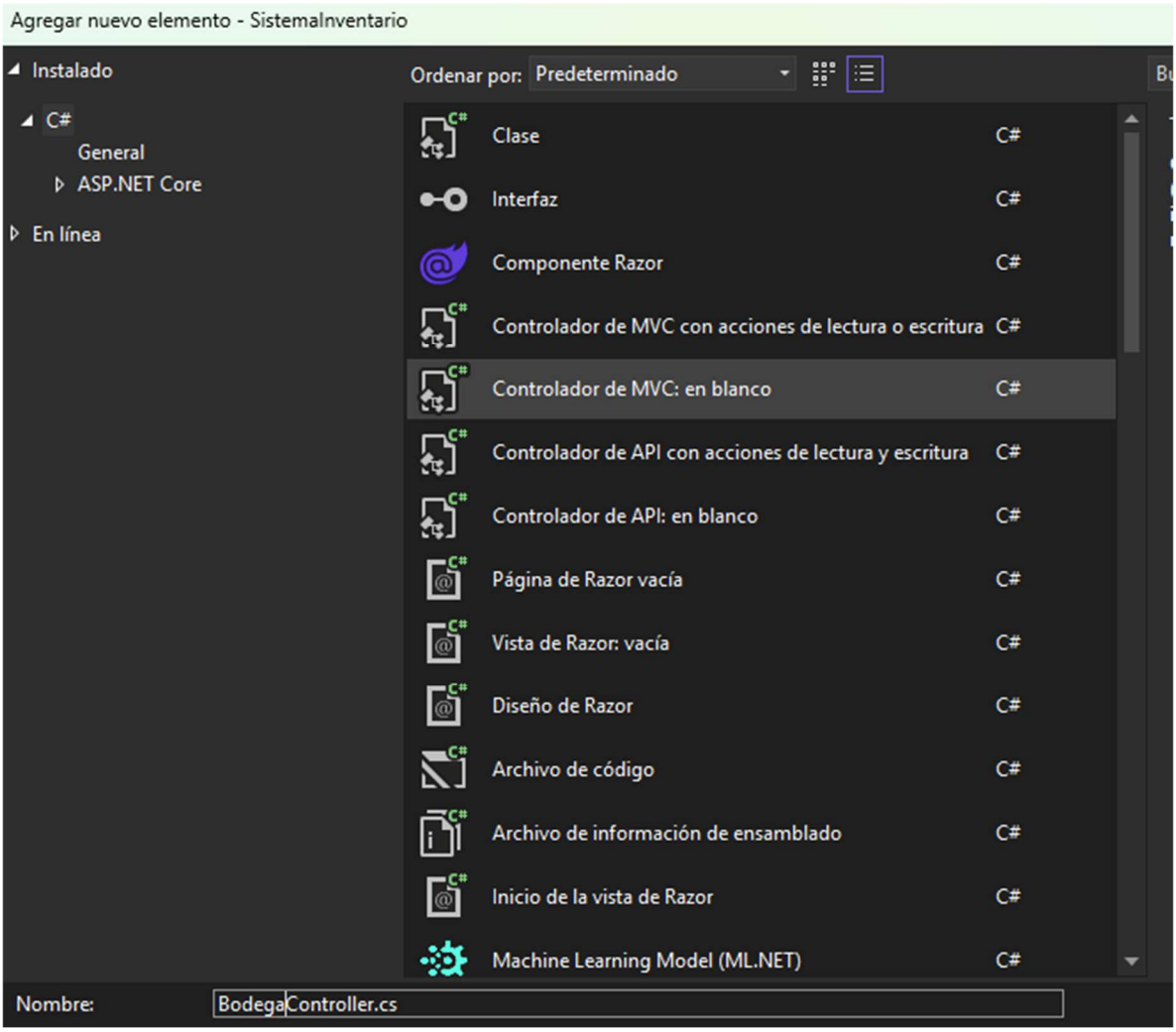
- git add .
- git commit -m “Patron de Repositorio”
- git push-u origin main

Bodega CRUD

1. Crear Controlador Bodega

- En la carpeta Areas/Admin/Controllers agregamos un controlador de MVC llamado BodegaController.cs.



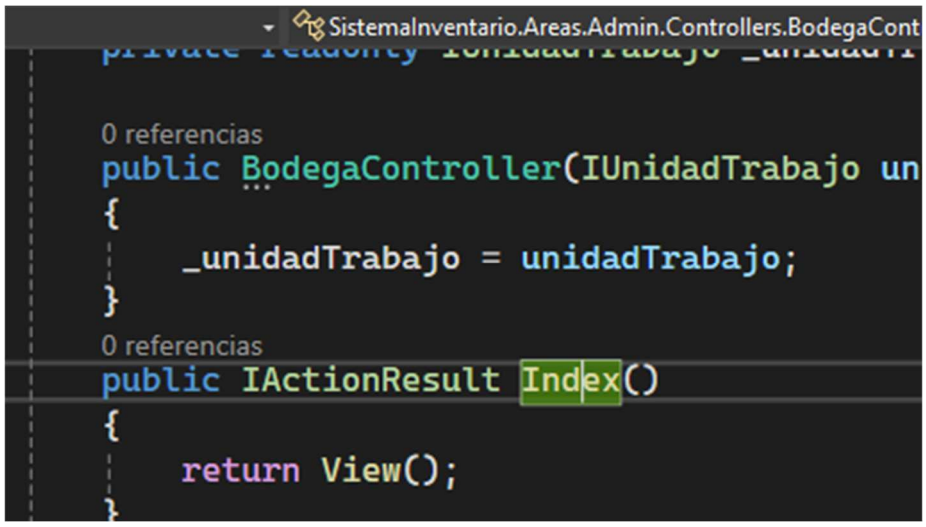


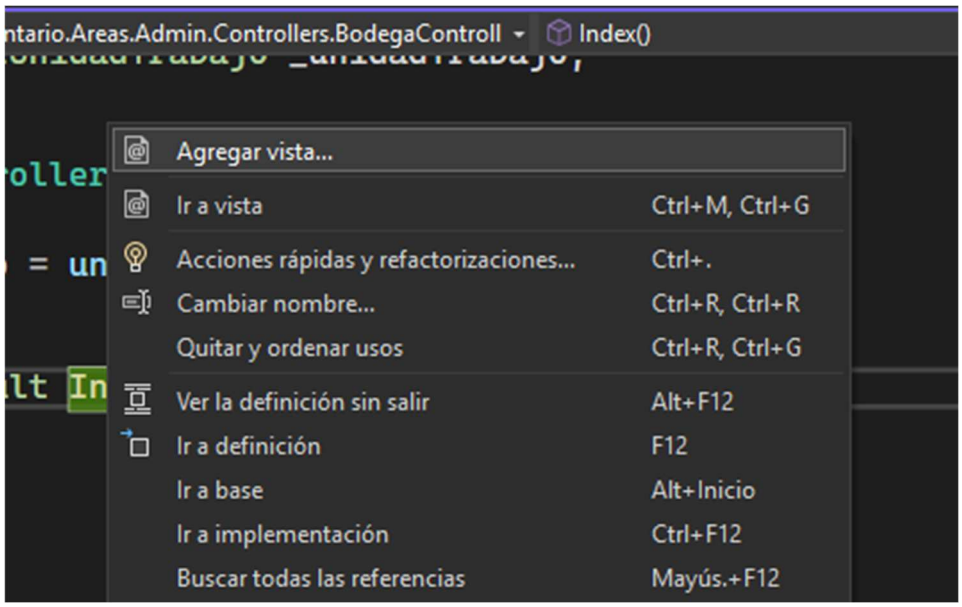
BodegaController.cs

```
using Microsoft.AspNetCore.Mvc;
using SistemaInventario.AccesoDatos.Repositorio.IRepositorio;

namespace SistemaInventario.Areas.Admin.Controllers
{
    public class BodegaController : Controller
    {
        private readonly IUnidadTrabajo _unidadTrabajo;

        public BodegaController(IUnidadTrabajo unidadTrabajo)
        {
            _unidadTrabajo = unidadTrabajo;
        }
        public IActionResult Index()
        {
            return View();
        }
```

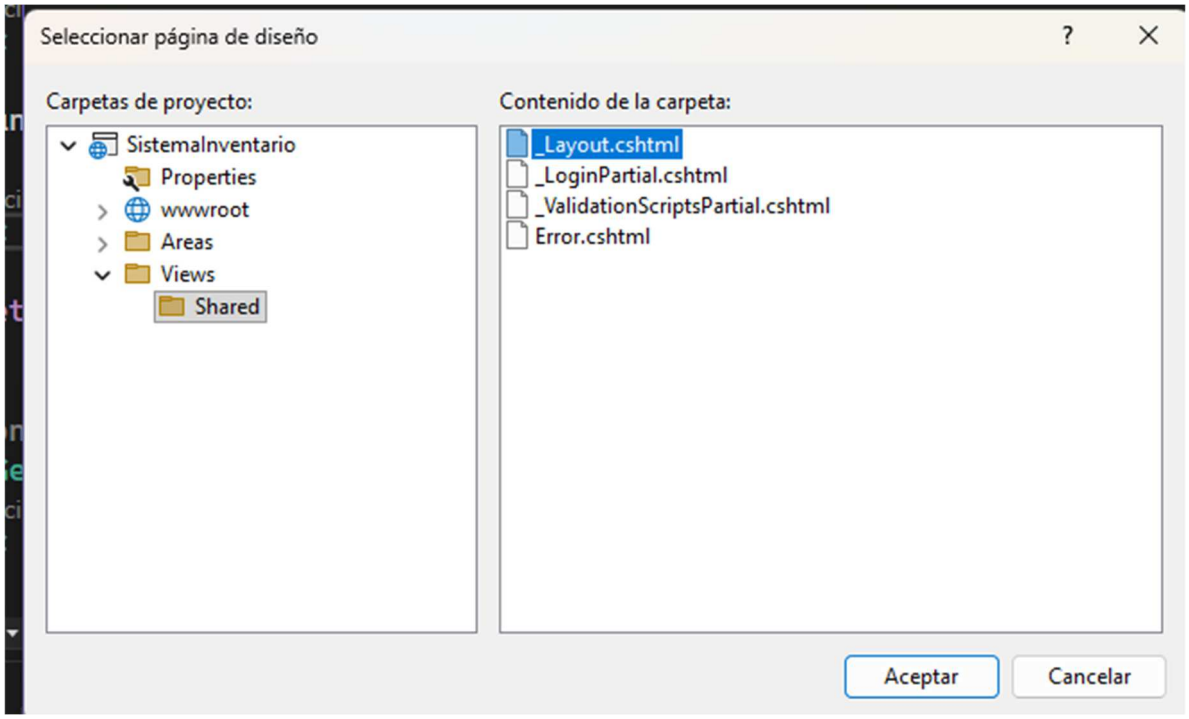
```
        #region API
        [HttpGet]
        public async Task<IActionResult> ObtenerTodos()
        {
            var todos = await _unidadTrabajo.Bodega.ObtenerTodos();
            return Json(new { data = todos });
        }

        #endregion
    }
}
```

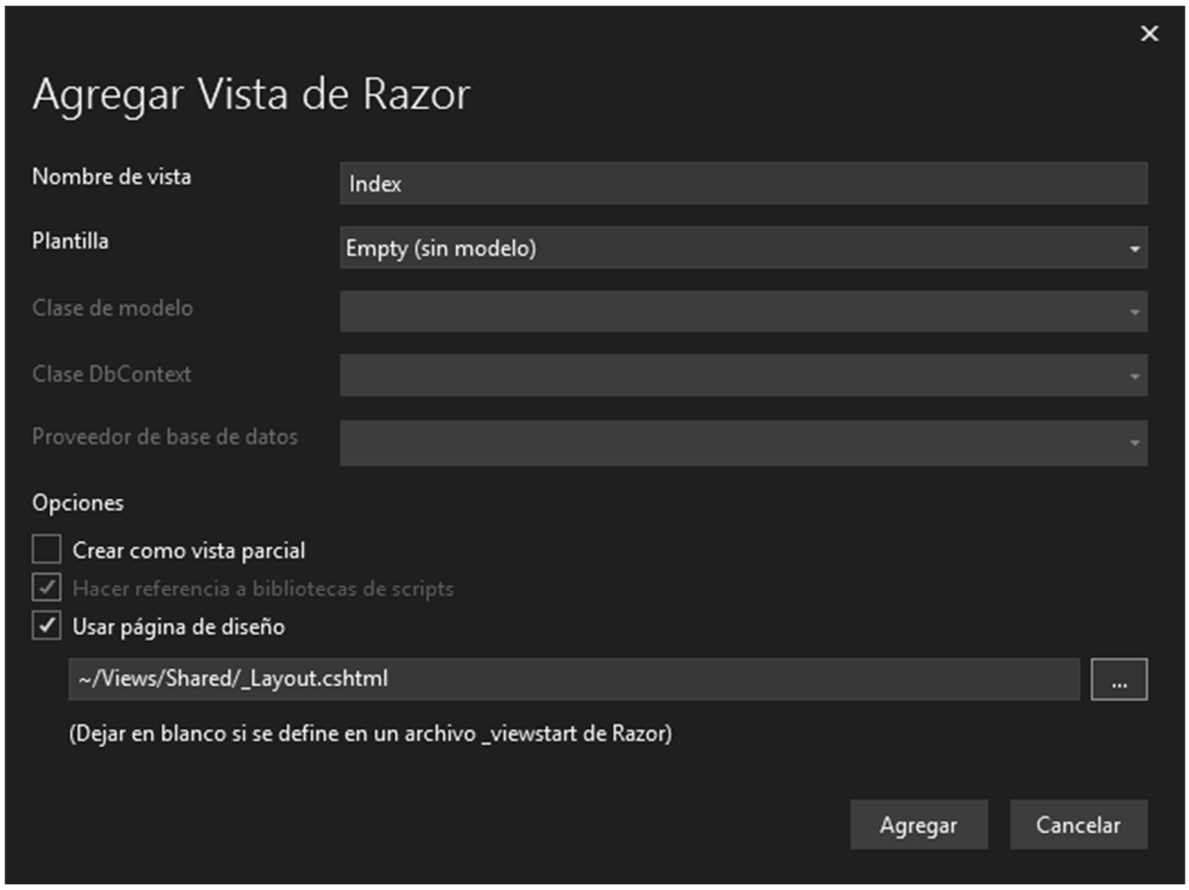
- Desde BodegaController.cs crear vista Index en base al método.






Agregar nuevo elemento con scaffolding
Instalado
Común
API
Blazor
MVC
Controlador
Ver
Páginas de Razor
Diseño
Vista de Razor: vacía
Vista de Razor

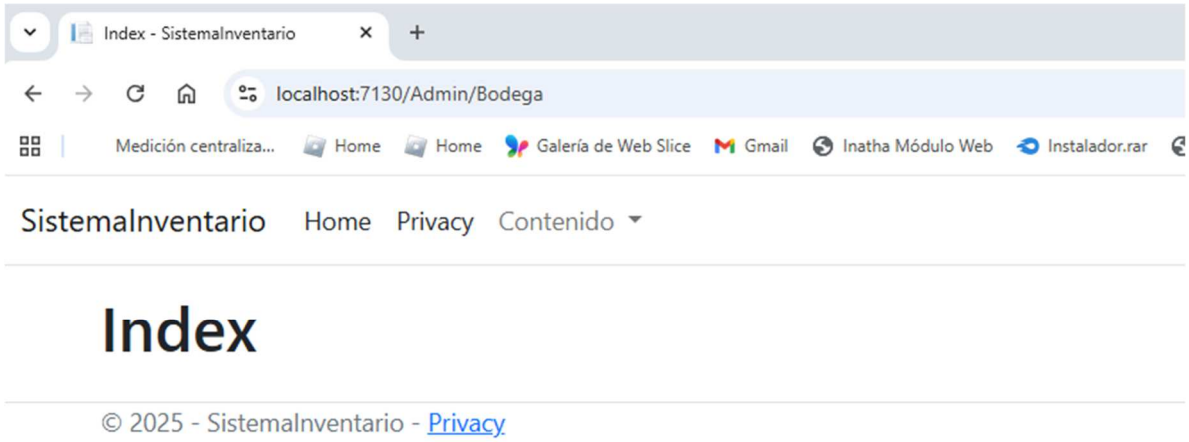

Seleccionar página de diseño
Carpetas de proyecto:
SistemaInventario
Properties
wwwroot
Areas
Views
Shared
Contenido de la carpeta:
_Layout.cshtml
_LoginPartial.cshtml
_ValidationScriptsPartial.cshtml
Error.cshtml
Aceptar
Cancelar



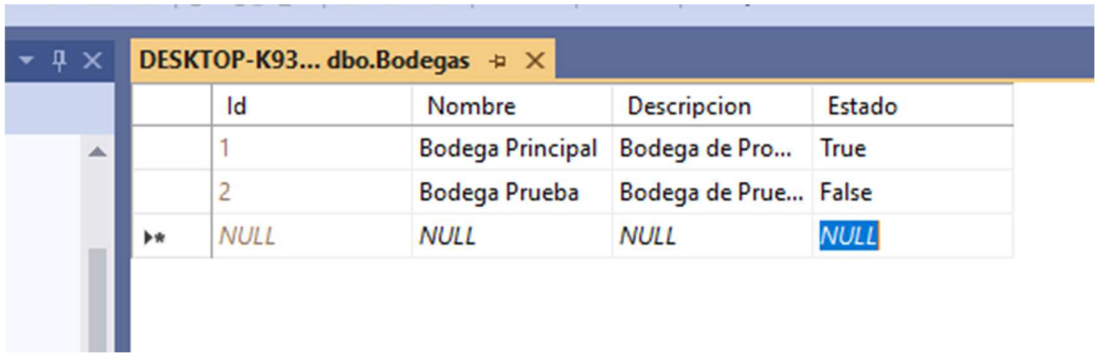
- En BodegaController.cs adicionamos el Area.

```
namespace SistemaInventario.Areas.Admin.Controllers
{
   [Area("Admin")]
.
.
.
```

- Probamos la aplicación.



- En la tabla Bodega agregamos dos registros.



- Probamos en el navegador

2. Herramientas de Terceros

- datatables.net
- js.org/?xn--sweetalert-dz6e
- cdnjs.com/libraries/toastr.js/latest
- icons.getbootstrap.com/

- Adicionamos las librerias en el archivo SistemaInventario/Views/Share/_Layout.cshtml

Los css en la parte superior

```
<link rel="stylesheet" href="https://cdn.datatables.net/1.13.3/css/jquery.dataTables.min.css" />
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/toastr.js/latest/toastr.min.css" />
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.10.3/font/bootstrap-icons.css">
```

Los js en la parte inferior

```
<script src="https://cdn.datatables.net/1.13.3/js/jquery.dataTables.min.js"></script>
<script src="https://unpkg.com/sweetalert/dist/sweetalert.min.js"></script>
<script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/toastr.js/latest/toastr.min.js"></script>
```

3. Bodega Vista Index

- En Areas/Admin/Views/Bodega/Index.cshtml agregamos el diseño de la vista.

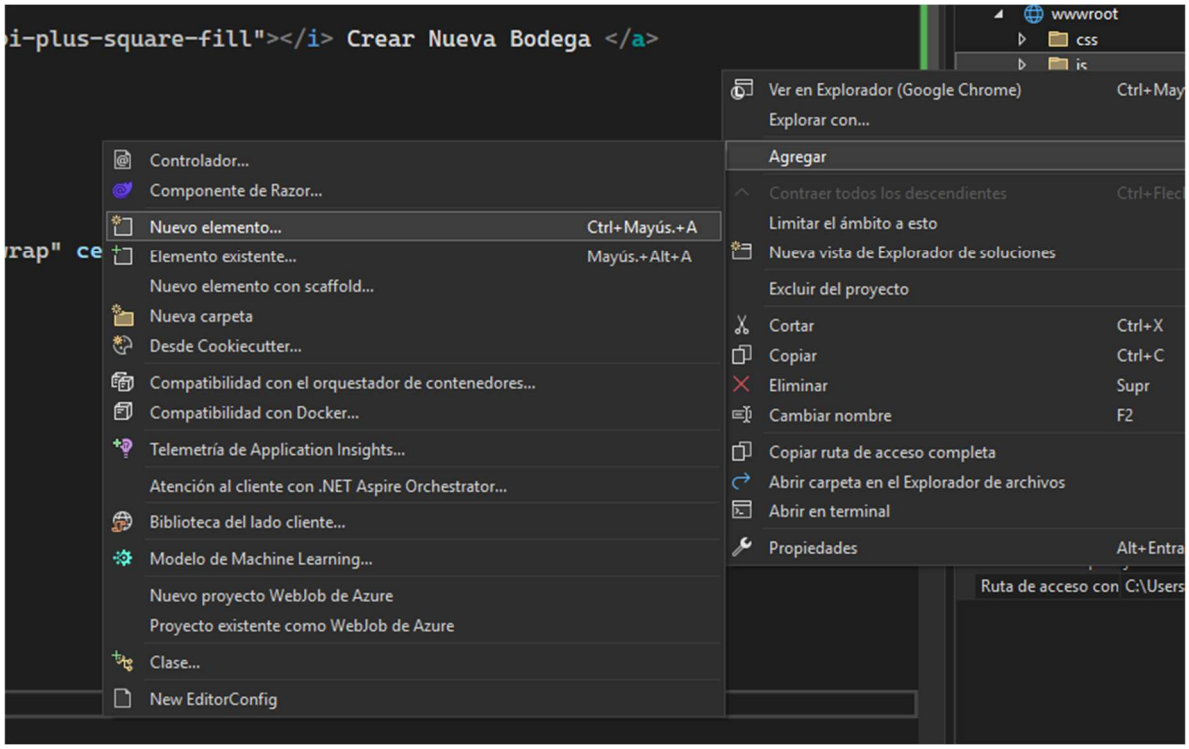
Index.cshtml

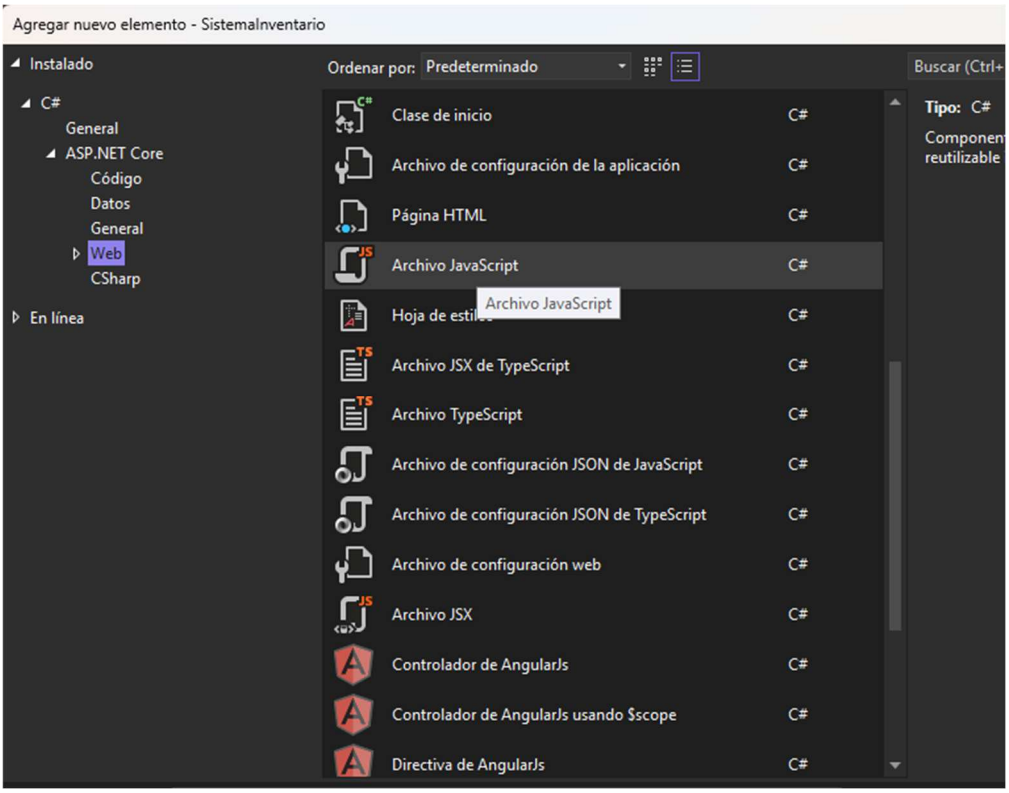
```
@{
  ViewData["Title"] = "Index";
  Layout = "~/Views/Shared/_Layout.cshtml";
}

<div class="row">
  <div class="col-lg-6">
    <h2 class="text-primary">Lista de Bodegas</h2>
  </div>
  <div class="col-lg-6">
    <a class="btn btn-outline-primary" asp-action="Upsert"> <i class="bi bi-plus-square-fill"></i> Crear Nueva Bodega </a>
  </div>
</div>

<br />

<div class="p-4 border rounded bg-light">
  <table id="tblDatos" class="table table-responsive table-hover display nowrap" cellspacing="0" width="100%">
    <thead class="table-dark">
      <tr>
        <th>Nombre</th>
        <th>Descripcion</th>
        <th>Estado</th>
        <th></th>
      </tr>
    </thead>
  </table>
```

```
</div>

@section Scripts {
    <script src="~/js/bodega.js"></script>
}
```

4. Bodega JavaScript

- Creamos un js llamado Bodega.js en la carpeta wwwroot/js





Bodega.js

```
let datatable;

$(document).ready(function () {
    loadDataTable();
});

function loadDataTable() {
    datatable = $('#tblDatos').DataTable({
        "ajax": {
            "url": "/Admin/Bodega/ObtenerTodos"
        },
        "columns": [
            { "data": "nombre", "width": "20%" },
            { "data": "descripcion", "width": "40%" },
            {
                "data": "estado",
                "render": function (data) {
                    if (data == true) {
                        return "Activo";
                    }
                    else {
                        return "Inactivo";
                    }
                }, "width": "20%"
            },
```

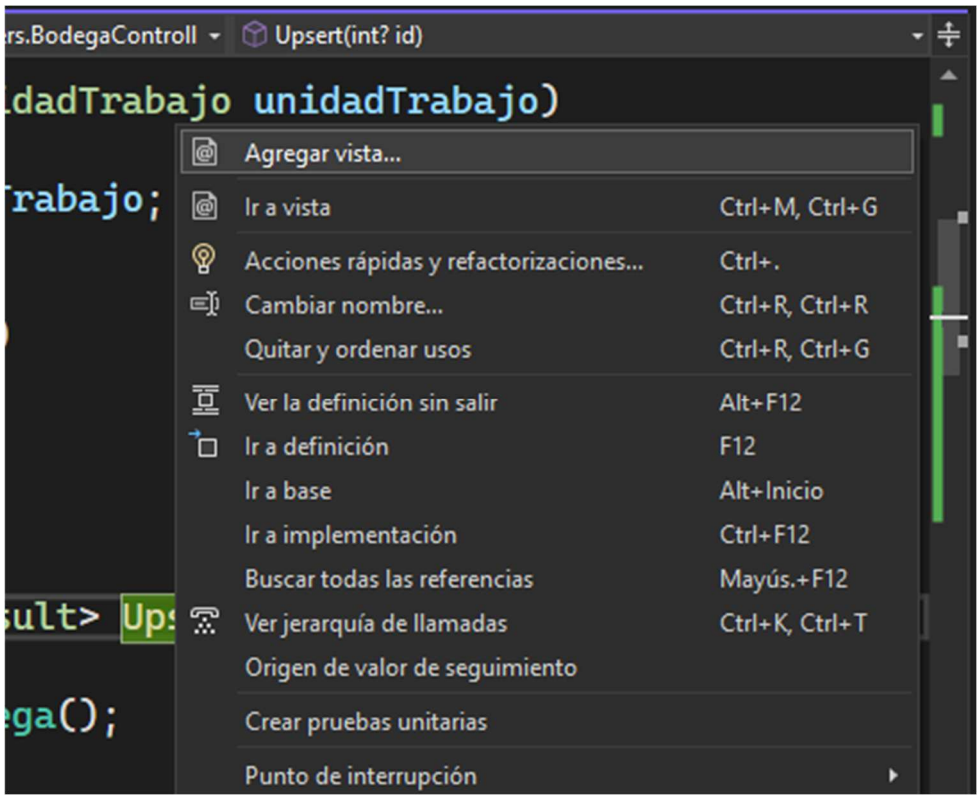
```
        {
            "data": "id",
            "render": function (data) {
                return `
                    <div>
                        <a href="/Admin/Bodega/Upsert/${data}" class="btn btn-success text-white"
style="cursor:pointer">
                        <i class="bi bi-pencil-square"></i>
                        </a>
                        <a onclick=Delete("/Admin/Bodega/Delete/${data}") class="btn btn-danger text-white"
style="cursor:pointer">
                        <i class="bi bi-trash3-fill"></i>
                        </a>
                    </div>
                `;
            }, "width": "20%"
        }
    ]
  });
}

function Delete(url) {

  swal({
    title: "Esta seguro de eliminar la Bodega",
    text: "Este registro no se podra recuperar",
    icon: "warning",
    buttons: true,
    dangerMode: true
  }).then((borrar) => {
    if (borrar) {
      $.ajax({
        type: "POST",
        url: url,
        success: function (data) {
          if (data.success) {
            toastr.success(data.message);
            datatable.ajax.reload();
          }
          else {
            toastr.error(data.message);
          }
        }
      });
    }
  });
}
```

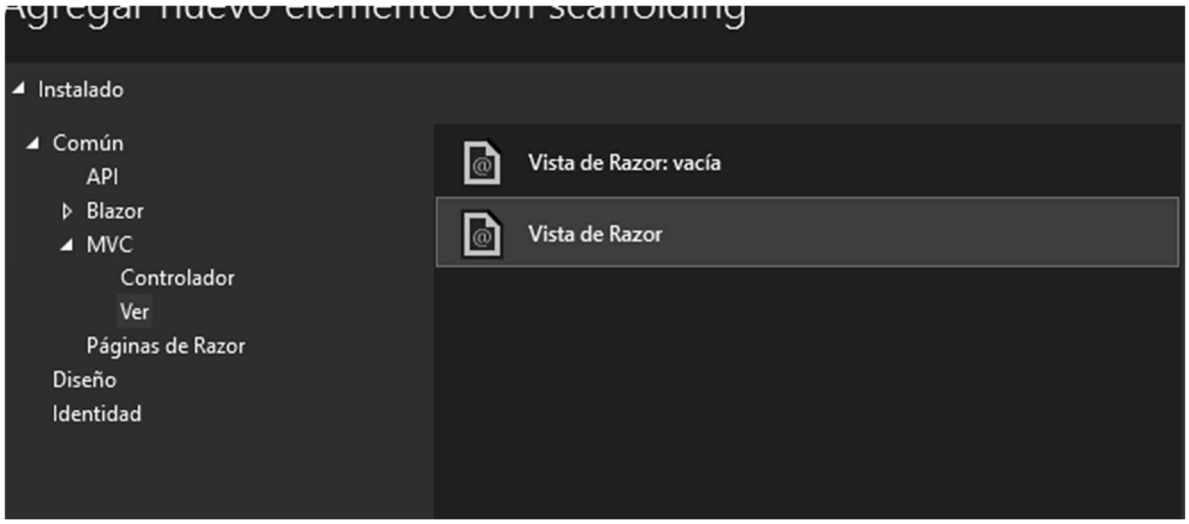
5. Bodega Upsert Metodo Get

- En Areas/Admin/Controller/BodegaController.cs creamos el método Upsert

```
public async Task<IActionResult> Upsert(int? id)
{
   Bodega bodega = new Bodega();

   if(id== null)
   {
      // Crear una nueva Bodega
      bodega.Estado = true;
      return View(bodega);
   }
   // Actualizamos Bodega
   bodega = await _unidadTrabajo.Bodega.Obtener(id.GetValueOrDefault());
   if(bodega ==null)
   {
      return NotFound();
   }
   return View(bodega);
}
```

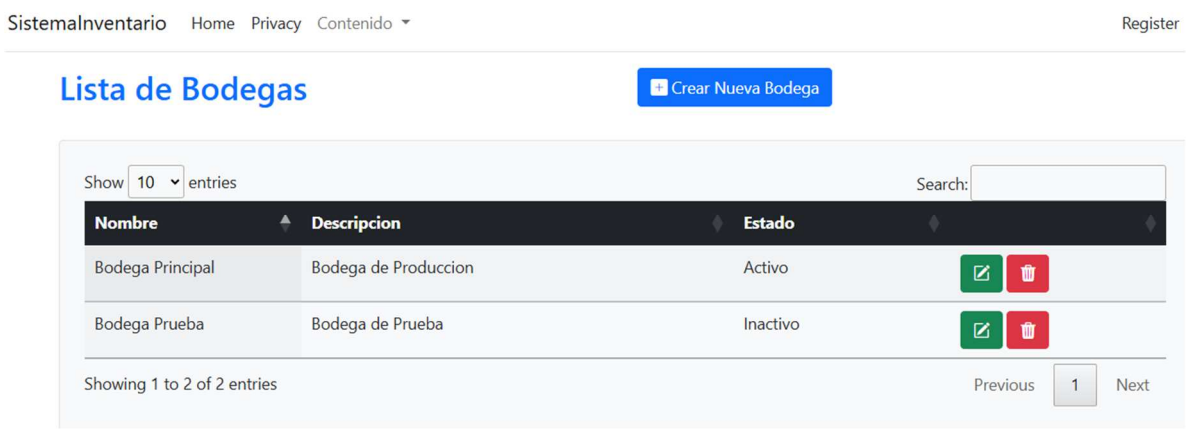
- Agregamos la vista del método Upsert

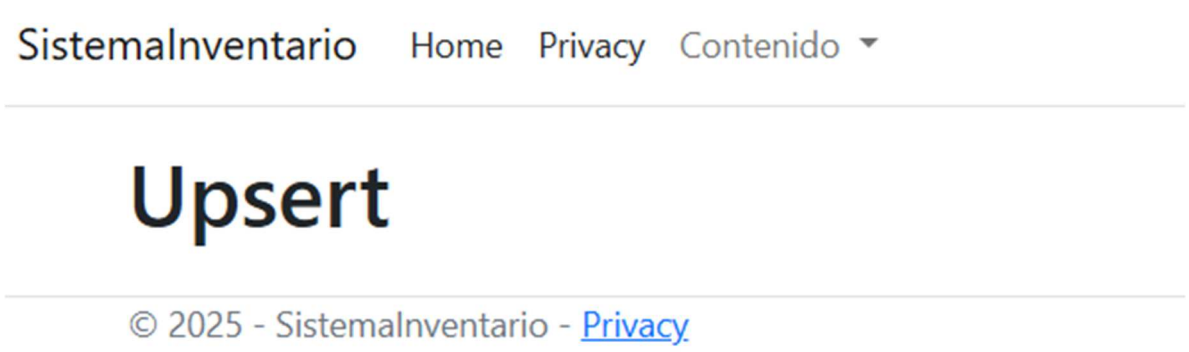





```
@{
   Layout = "~/Views/Shared/_Layout.cshtml";
}

<h1>Upsert</h1>
```

- Ejecutamos la aplicación.





6. Partial Views

- Modificamos la vista del método Upsert y creamos las Partial Views

```
@model SistemaInventario.Modelos.Bodega

@{
  Layout = "~/Views/Shared/_Layout.cshtml";
}

<h1>Upsert</h1>
```

- En View/Share agregamos una Vista de Razor llamada _BotonesCrearyRegresar y marcamos la casilla Crear como vista parcial

```
<div class="d-grid gap-2 d-md-block">
  <button type="submit" class="btn btn-primary" onfocus="false">  <i class="bi bi-plus-square-fill"></i> Crear
</button>
  <a asp-action="Index" class="btn btn-success"> <i class="bi bi-arrow-return-left"></i> Regresar </a>
</div>
```

- En View/Share agregamos una Vista de Razor llamada _BotonesActualizaryRegresar y marcamos la casilla Crear como vista parcial

```
@model int

<div class="d-grid gap-2 d-md-block">
  <button type="submit" class="btn btn-primary" onfocus="false" asp-route-id="@Model">  <i class="bi bi-
pencil-square"></i> Guardar Cambios </button>
  <a asp-action="Index" class="btn btn-success"> <i class="bi bi-arrow-return-left"></i> Regresar </a>
</div>
```

7. Bodega Vista Upsert

- Modificamos el archivo Upsert.schtml

```
@model SistemaInventario.Modelos.Bodega

@{
  Layout = "~/Views/Shared/_Layout.cshtml";
  var titulo = "Crear Bodega";
}

<form method="post">
  <div style="padding-left:15%; padding-right:15%; padding-bottom:inherit.4rem;">
    <div class="row border-0">
      <div asp-validation-summary="ModelOnly" class="text-danger"></div>
    </div>

    @if (Model.Id != 0)
    {
      // Actualizar
      titulo = "Actualizar Bodega";
      <input type="hidden" asp-for="Id" id="id" />
    }
    <div class="col-12 border-bottom p-0">
      <h2 class="text-primary">@titulo</h2>
    </div>

    <div class="row mb-2 mt-2">
      <div class="form-group col-md-6">
        <label>Nombre</label>
        <input type="text" asp-for="Nombre" class="form-control" placeholder="Nombre de la Bodega"
id="nombre" />
        <span asp-validation-for="Nombre" class="text-danger"></span>
      </div>
    </div>
    <div class="row mb-2 mt-2">
      <div class="form-group col-md-6">
        <label>Descripcion</label>
        <input type="text" asp-for="Descripcion" class="form-control" placeholder="Descripcion de la
Bodega" />
        <span asp-validation-for="Descripcion" class="text-danger"></span>
      </div>
    </div>
    <div class="row mb-2 mt-2">
      <div class="form-group col-md-6">
        <label>Estado</label>
        <select asp-for="Estado" class="form-select">
          <option value=true>Activo</option>
          <option value=false>Inactivo</option>
        </select>
        <span asp-validation-for="Estado" class="text-danger"></span>
      </div>
```

```
        </div>
        <br />
        <div>
            @if (Model.Id != 0)
            {
                <partial name="_BotonesActualizarYRegresar" model="Model.Id" />
            }
            else
            {
                <partial name="_BotonesCrearYRegresar" />
            }
        </div>

    </div>


</form>

@section Scripts{
    <partial name="_ValidationScriptsPartial" />
}
```

- Ejecutamos la aplicación para verificar el botón Crear Nueva Bodega y Actualizar

8. Bodega Upsert Post Action

- En Areas/Admin/Controller/BodegaController.cs creamos el metodo Upsert de tipo Post

```
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Upsert(Bodega bodega)
{
    if (ModelState.IsValid)
    {
        if (bodega.Id == 0)
        {
            await _unidadTrabajo.Bodega.Agregar(bodega);
        }
        else
        {
            _unidadTrabajo.Bodega.Actualizar(bodega);
        }
        await _unidadTrabajo.Guardar();
        return RedirectToAction(nameof(Index));
    }
    return View(bodega);
}
```

- Ejecutamos la aplicación para verificar el ingreso y la actualización de una bodega en la base de datos.

9. Bodega Delete API Call y JavaScript

- En Areas/Admin/Controller/BodegaController.cs creamos el metodo Delete y lo pegamos dentro de la región API.

```
[HttpPost]
public async Task<IActionResult> Delete(int id)
{
   var bodegaDb = await _unidadTrabajo.Bodega.Obtener(id);
   if(bodegaDb ==null)
   {
      return Json(new { success = false, message = "Error al borrar Bodega" });
   }
   _unidadTrabajo.Bodega.Remover(bodegaDb);
   await _unidadTrabajo.Guardar();
   return Json(new { success = true, message = "Bodega borrada exitosamente" });
}
```

- Ejecutamos la aplicación para verificar el borrado de una bodega en la base de datos

10. Notificaciones Toastr Partial View

- En View/Share agregamos una Vista de Razor llamada _Notificaciones.schtml y marcamos la casilla Crear como vista parcial

_Notificaciones.schtml

```
@using SistemaInventario.Utilidades

@if (TempData[DS.Exitosa] != null)
{
   <script src="~/lib/jquery/dist/jquery.min.js"></script>
   <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/toastr.js/latest/toastr.min.js"></script>
   <script type="text/javascript">
      toastr.success('@TempData[DS.Exitosa]');
   </script>
}


@if (TempData[DS.Error] != null)
{
   <script src="~/lib/jquery/dist/jquery.min.js"></script>
   <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/toastr.js/latest/toastr.min.js"></script>
   <script type="text/javascript">
```

```
        toastr.error('@TempData[DS.Error]');
    </script>
}
```

- En SistemaInventario.Utilidades actualizar DS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SistemaInventario.Utilidades
{
    public static class DS
    {
        public const string Exitosa = "Exitosa";
        public const string Error = "Error";

        public const string ImagenRuta = @"\imagenes\producto\";
        public const string ssCarroCompras = "Sesion carro Compras";

        public const string Role_Admin = "Admin";
        public const string Role_Cliente = "Cliente";
        public const string Role_Inventario = "Inventario";

        // Estados de la Orden
        public const string EstadoPendiente = "Pendiente";
        public const string EstadoAprobado = "Aprobado";
        public const string EstadoEnProceso = "Procesando";
        public const string EstadoEnviado = "Enviado";
        public const string EstadoCancelado = "Cancelado";
        public const string EstadoDevuelto = "Devuelto";
        // Estado del Pago de la Orden
        public const string PagoEstadoPendiente = "Pendiente";
        public const string PagoEstadoAprobado = "Aprobado";
        public const string PagoEstadoRetrasado = "Retrasado";
        public const string PagoEstadoRechazado = "Rechazado";

    }
}
```

- En Views/Shared_Layout.cshtml adicionamos las notificaciones.

```
<div class="container">
    <main role="main" class="pb-3">
        <partial name="_Notificaciones" />
        @RenderBody()
    </main>
</div>
```

- Actualizamos Areas/Admin/Controller/BodegaController.cs con el TempData

```
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Upsert(Bodega bodega)
{
   if (ModelState.IsValid)
   {
      if (bodega.Id == 0)
      {
         await _unidadTrabajo.Bodega.Agregar(bodega);
         TempData[DS.Exitosa] = "Bodega creada Exitosamente";
      }
      else
      {
         _unidadTrabajo.Bodega.Actualizar(bodega);
         TempData[DS.Exitosa] = "Bodega actualizada Exitosamente";

      }
      await _unidadTrabajo.Guardar();
      return RedirectToAction(nameof(Index));
   }
   TempData[DS.Error] = "Error al grabar Bodega";
   return View(bodega);
}
```

11. Validar Nombre

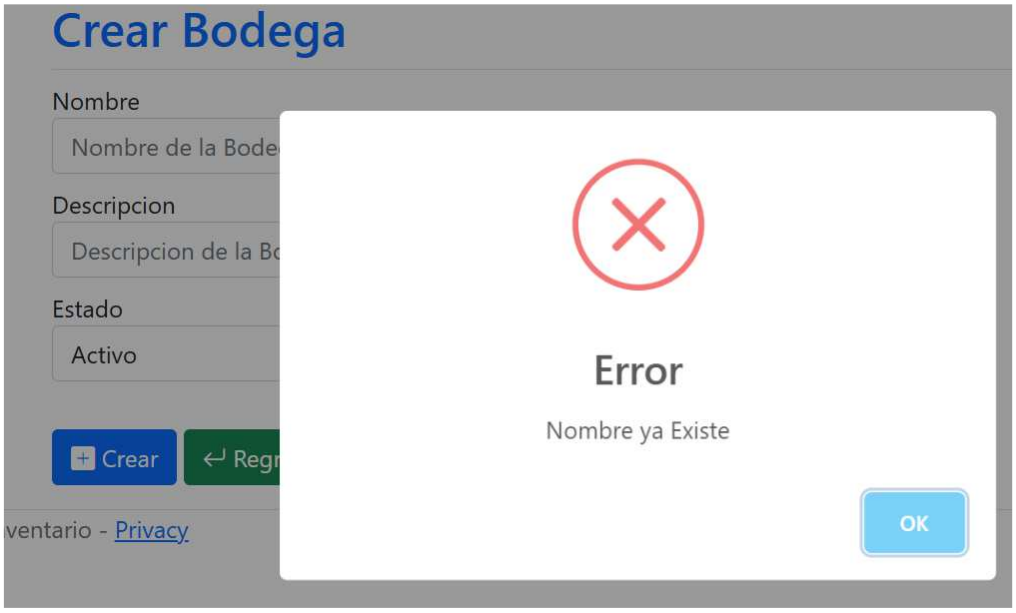
- En BodegaController.cs creamos un método llamado validarNombre en la region API

```
[ActionName("ValidarNombre")]
public async Task<IActionResult> ValidarNombre(string nombre, int id =0)
{
   bool valor = false;
   var lista = await _unidadTrabajo.Bodega.ObtenerTodos();
   if(id==0)
   {
      valor = lista.Any(b => b.Nombre.ToLower().Trim() == nombre.ToLower().Trim());
   }
   else
   {
      valor = lista.Any(b => b.Nombre.ToLower().Trim() == nombre.ToLower().Trim() && b.Id !=id);
   }
   if(valor)
   {
      return Json(new { data = true });
   }
   return Json(new { data = false });

}
```

- Modificamos Areas/Admin/Views/Bodega/Upsert.cshtml adicionando en la @section Scripts un Scripts

```
<script>

   document.getElementById("nombre").addEventListener('change', function () {
      validarNombre();
   });

   function validarNombre()
   {
      var url = '@Url.Content("~/")' + "Admin/Bodega/ValidarNombre";
      var id = '#id';
      var nombre = '#nombre';

      $.getJSON(url, { id: $(id).val(), nombre: $(nombre).val() }, function (data) {
         $.each(data, function (key, value) {
            if(value) {
               var nombre = document.getElementById("nombre");
               swal("Error", "Nombre ya Existe", "error");
               nombre.value="";
            }
         });
      })

   }
</script>
```

- Ejecutamos la aplicación para verificar los nombres de las bodegas



Categoría CRUD

1. Categoria Modelo y Repositorio

- En SistemaInventario.Modelos agregamos una Clase llamada Categoría.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SistemaInventario.Modelos
{
   public class Categoria
   {
      [Key]
      public int Id { get; set; }

      [Required(ErrorMessage ="Nombre es Requerido")]
      [MaxLength(60, ErrorMessage ="Nombre debe ser Maximo 60 Caracteres")]
      public string Nombre { get; set; }

      [Required(ErrorMessage = "Descripcion es Requerido")]
      [MaxLength(100, ErrorMessage = "Descripcion debe ser Maximo 100 Caracteres")]
      public string Descripcion { get; set; }

      [Required(ErrorMessage = "Estado es Requerido")]
      public bool Estado { get; set; }
   }
}
```

- En SistemaInventario.AccesoDatos/Configuracion le hacemos una copia a BodegaConfiguracion.cs y le cambiamos el nombre a CategoriaConfiguracion.cs

```
using Microsoft.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore.Metadata.Builders;
using SistemaInventario.Modelos;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SistemaInventario.AccesoDatos.Configuracion
{
   public class CategoriaConfiguracion : IEntityTypeConfiguration<Categoria>
   {
      public void Configure(EntityTypeBuilder<Categoria> builder)
      {
```

```
        builder.Property(x => x.Id).IsRequired();
        builder.Property(x => x.Nombre).IsRequired().HasMaxLength(60);
        builder.Property(x => x.Descripcion).IsRequired().HasMaxLength(100);
        builder.Property(x => x.Estado).IsRequired();
    }
  }
}
```

- Actualizamos SistemaInventario.AccesoDatos/Data/ApplicationDbContext.cs con la Categoria

```
public DbSet<Categoria> Categorias { get; set; }
```

- Realizamos la migración a la base de datos.
    - add-migration AgregarCategoriaMigracion
    - update-database

- En SistemaInventario.AccesoDatos/Repositorio/IRepositorio le hacemos una copia a IBodega Repositorio.cs y le cambiamos el nombre a ICategoriaRepositorio.cs

```
using SistemaInventario.Modelos;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SistemaInventario.AccesoDatos.Repositorio.IRepositorio
{
  public interface ICategoriaRepositorio : IRepositorio<Categoria>
  {
    void Actualizar(Categoria categoria);
  }
}
```

- En SistemaInventario.AccesoDatos/Repositorio le hacemos una copia a Bodega Repositorio.cs y le cambiamos el nombre a CategoriaRepositorio.cs

```
using SistemaInventario.AccesoDatos.Data;
using SistemaInventario.AccesoDatos.Repositorio.IRepositorio;
using SistemaInventario.Modelos;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SistemaInventario.AccesoDatos.Repositorio
{
```

```
  public class CategoriaRepositorio : Repositorio<Categoria>, ICategoriaRepositorio
  {

    private readonly ApplicationDbContext _db;

    public CategoriaRepositorio(ApplicationDbContext db) : base(db)
    {
      _db = db;
    }

    public void Actualizar(Categoria categoria)
    {
      var categoriaBD = _db.Categorias.FirstOrDefault(b => b.Id == categoria.Id);
      if (categoriaBD != null)
      {
        categoriaBD.Nombre = categoria.Nombre;
        categoriaBD.Descripcion = categoria.Descripcion;
        categoriaBD.Estado = categoria.Estado;
        _db.SaveChanges();
      }
    }
  }
}
```

- En SistemaInventario.AccesoDatos/Repositorio/IRepositorio modificamos IUnidadTrabajo.cs

```
ICategoriaRepositorio Categoria { get; }
```

- En SistemaInventario.AccesoDatos/Repositorio/Repositorio modificamos UnidadTrabajo.cs

```
    public IBodegaRepositorio Bodega { get; private set; }
    public ICategoriaRepositorio Categoria { get; private set; }

    public UnidadTrabajo(ApplicationDbContext db)
    {
      _db = db;
      Bodega = new BodegaRepositorio(_db);
      Categoria = new CategoriaRepositorio(_db);
    }
```

2. Categoria Controlador y Vistas

- En SistemaInventario/Areas/Admin/Controllers le hacemos una copia a BodegaController.cs y le cambiamos el nombre a CategoriaController.cs

- Actualizar la información en CategoriaController.cs

- En SistemaInventario/Areas/Admin/Views duplicar la carpeta Bodega y le cambiamos el nombre a Categoria; modificamos los archivos haciendo referencia a Categoria

- En SistemaInventario/wwwroot le hacemos una copia a bodega.js y le cambiamos el nombre a categoria.js; realizamos los cambios respectivos.

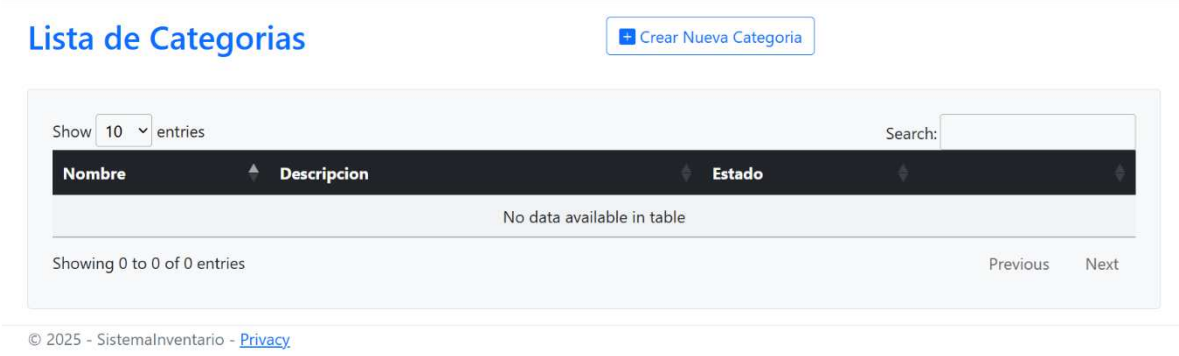
- Actualizamos el archivo SistemaInventario/Views/Shared/_Layout.cshtml

```
<ul class="dropdown-menu">
  <li><a class="dropdown-item" asp-area="Admin" asp-controller="Bodega" asp-action="Index">Bodegas</a></li>
  <li><a class="dropdown-item" asp-area="Admin" asp-controller="Categoria" asp-action="Index">Categorias</a></li>
</ul>
```

- Subir Cambios a GitHub (Opcional)

  - git add .
  - git commit-m "CRUD de Categoria"
  - git push-u origin main

- Ejecutamos la aplicación para verificar la funcionalidad de las categorías.



Marca CRUD

1. Marca - Modelo y Repositorio

- En SistemaInventario.Modelos agregamos una Clase llamada Marca.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
namespace SistemaInventario.Modelos
{
   public class Marca
   {
      [Key]
      public int Id { get; set; }

      [Required(ErrorMessage ="Nombre es Requerido")]
      [MaxLength(60, ErrorMessage ="Nombre debe ser Maximo 60 caracteres")]
      public string Nombre { get; set; }

      [Required(ErrorMessage = "Descripcion es Requerido")]
      [MaxLength(100, ErrorMessage = "Descripcion debe ser Maximo 60 caracteres")]
      public string Descripcion { get; set; }

      [Required(ErrorMessage = "Estado es Requerido")]
      public bool Estado { get; set; }
   }
}
```

- En SistemaInventario.AccesoDatos/Configuracion le hacemos una copia a CategoriaConfiguracion.cs y le cambiamos el nombre a MarcaConfiguracion.cs y cambiamos la información correspondiente.

- Actualizamos SistemaInventario.AccesoDatos/Data/ApplicationDbContext.cs con la Marca

```
public DbSet<Marca> Marcas { get; set; }
```

- Realizamos la migración a la base de datos.
    - add-migration AgregarMarcaMigracion
    - update-database

- En SistemaInventario.AccesoDatos/Repositorio/IRepositorio le hacemos una copia a ICategoriaRepositorio.cs y le cambiamos el nombre a IMarcaRepositorio.cs y cambiamos la información correspondiente.

- En SistemaInventario.AccesoDatos/Repositorio le hacemos una copia a CategoriaRepositorio.cs y le cambiamos el nombre a MarcaRepositorio.cs y cambiamos la información correspondiente.

- En SistemaInventario.AccesoDatos/Repositorio/IRepositorio modificamos IUnidadTrabajo.cs

```
IMarcaRepositorio Marca { get; }
```

- En SistemaInventario.AccesoDatos/Repositorio/Repositorio modificamos UnidadTrabajo.cs

2. Marca Controlador y Vistas

- En SistemaInventario/Areas/Admin/Controllers le hacemos una copia a CategoriaController.cs y le cambiamos el nombre a MarcaController.cs y cambiamos la información correspondiente.

- En SistemaInventario/Areas/Admin/Views duplicar la carpeta Categoria y le cambiamos el nombre a Marca; modificamos los archivos haciendo referencia a Marca.

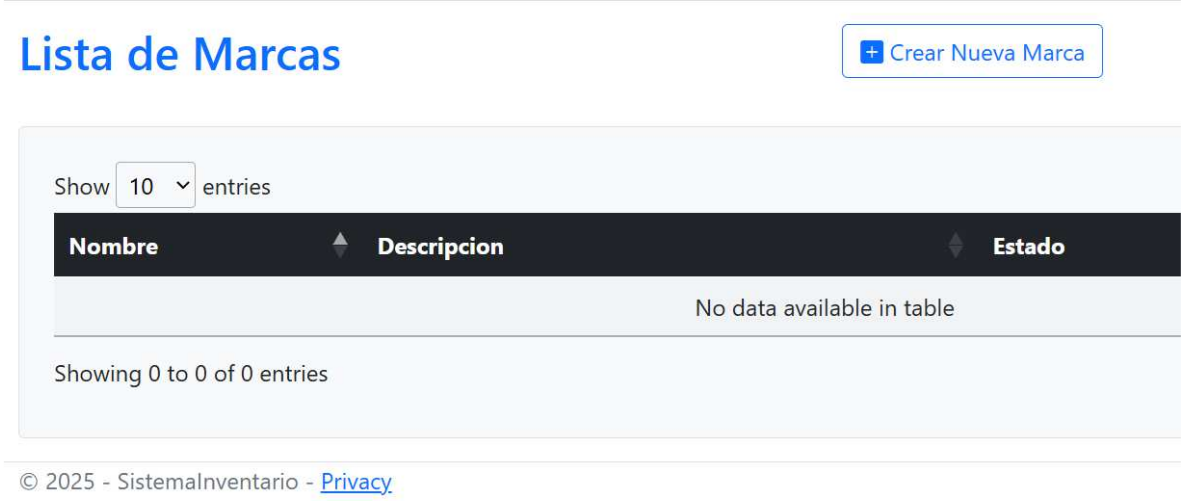
- En SistemaInventario/wwwroot le hacemos una copia a categoria.js y le cambiamos el nombre a marca.js; realizamos los cambios respectivos.

- Actualizamos el archivo SistemaInventario/Views/Shared/_Layout.cshtml

```
<li><a class="dropdown-item" asp-area="Admin" asp-controller="Bodega" asp-action="Index">Bodegas</a></li>
<li><a class="dropdown-item" asp-area="Admin" asp-controller="Categoria" asp-action="Index">Categorias</a></li>
<li><a class="dropdown-item" asp-area="Admin" asp-controller="Marca" asp-action="Index">Marcas</a></li>
```

- Subir Cambios a GitHub (Opcional)
  - git add .
  - git commit-m "CRUD de Marca"
  - git push-u origin main

- Ejecutamos la aplicación para verificar la funcionalidad de las marcas.



Producto CRUD

1. Producto Modelo

- Abrir los archivos de los proyectos y eliminar la línea.

<Nullable>enable</Nullable>

- En SistemaInventario.Modelos agregamos una Clase llamada Producto.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SistemaInventario.Modelos
{
   public class Producto
   {
      [Key]
      public int Id { get; set; }

      [Required(ErrorMessage ="Numero de Serie es Requerido")]
      [MaxLength(60)]
      public string NumeroSerie { get; set; }

      [Required(ErrorMessage = "Descripcion es Requerido")]
      [MaxLength(60)]
      public string Descripcion { get; set; }

      [Required(ErrorMessage = "Precio es Requerido")]
      public double Precio { get; set; }

      [Required(ErrorMessage = "Costo es Requerido")]
      public double Costo { get; set; }

      public string ImagenUrl { get; set; }

      [Required(ErrorMessage = "Estado es Requerido")]
      public bool Estado { get; set; }

      [Required(ErrorMessage = "Categoria es Requerido")]
      public int CategoriaId { get; set; }

      [ForeignKey("CategoriaId")]
      public Categoria Categoria { get; set; }

      [Required(ErrorMessage = "Marca es Requerido")]
```

```
        public int MarcaId { get; set; }

        [ForeignKey("MarcaId")]
        public Marca Marca { get; set; }

        public int? PadreId { get; set; }
        public virtual Producto  Padre { get; set; }

    }
}
```

- Actualizamos SistemaInventario.AccesoDatos/Data/ApplicationDbContext.cs con el Producto

```
public DbSet<Producto> Productos { get; set; }
```

- En SistemaInventario.AccesoDatos/Configuracion le hacemos una copia a MarcaConfiguracion.cs y le cambiamos el nombre a ProductoConfiguracion.cs.

```
using Microsoft.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore.Metadata.Builders;
using SistemaInventario.Modelos;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SistemaInventario.AccesoDatos.Configuracion
{
    public class ProductoConfiguracion : IEntityTypeConfiguration<Producto>
    {
        public void Configure(EntityTypeBuilder<Producto> builder)
        {
            builder.Property(x => x.Id).IsRequired();
            builder.Property(x =>x.NumeroSerie).IsRequired().HasMaxLength(60);
            builder.Property(x =>x.Descripcion).IsRequired().HasMaxLength(100);
            builder.Property(x => x.Estado).IsRequired();
            builder.Property(x => x.Precio).IsRequired();
            builder.Property(x => x.Costo).IsRequired();
            builder.Property(x => x.CategoriaId).IsRequired();
            builder.Property(x => x.MarcaId).IsRequired();
            builder.Property(x => x.ImagenUrl).IsRequired(false);
            builder.Property(x => x.PadreId).IsRequired(false);

            /* Relaciones*/

            builder.HasOne(x => x.Categoria).WithMany()
                .HasForeignKey(x => x.CategoriaId)
                .OnDelete(DeleteBehavior.NoAction);
```

```
            builder.HasOne(x => x.Marca).WithMany()
                .HasForeignKey(x => x.MarcaId)
                .OnDelete(DeleteBehavior.NoAction);

            builder.HasOne(x => x.Padre).WithMany()
                .HasForeignKey(x => x.PadreId)
                .OnDelete(DeleteBehavior.NoAction);

        }
    }
}
```

- Realizamos la migración a la base de datos.
    - add-migration AgregarProductoMigracion
    - update-database

2. Producto Repositorio

- En SistemaInventario.AccesoDatos/Repositorio/IRepositorio le hacemos una copia a IMarcaRepositorio.cs y le cambiamos el nombre a IProductoRepositorio.cs y cambiamos la información correspondiente.

- En SistemaInventario.AccesoDatos/Repositorio le hacemos una copia a MarcaRepositorio.cs y le cambiamos el nombre a ProductoRepositorio.cs y cambiamos la información correspondiente.

```
using SistemaInventario.AccesoDatos.Data;
using SistemaInventario.AccesoDatos.Repositorio.IRepositorio;
using SistemaInventario.Modelos;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SistemaInventario.AccesoDatos.Repositorio
{
    public class ProductoRepositorio : Repositorio<Producto>, IProductoRepositorio
    {

        private readonly ApplicationDbContext _db;

        public ProductoRepositorio(ApplicationDbContext db) : base(db)
        {
            _db = db;
        }

        public void Actualizar(Producto producto)
```

```
        {
            var productoBD = _db.Productos.FirstOrDefault(b => b.Id == producto.Id);
            if (productoBD != null)
            {
                if (producto.ImagenUrl != null)
                {
                    productoBD.ImagenUrl = producto.ImagenUrl;
                }
                productoBD.NumeroSerie = producto.NumeroSerie;
                productoBD.Descripcion = producto.Descripcion;
                productoBD.Precio = producto.Precio;
                productoBD.Costo = producto.Costo;
                productoBD.CategoriaId = producto.CategoriaId;
                productoBD.MarcaId = producto.MarcaId;
                productoBD.PadreId = producto.PadreId;
                productoBD.Estado = producto.Estado;

                _db.SaveChanges();
            }
        }
    }
}
```

- En SistemaInventario.AccesoDatos/Repositorio/IRepositorio modificamos IUnidadTrabajo.cs

```
IProductoRepositorio Producto { get; }
```

- En SistemaInventario.AccesoDatos/Repositorio/Repositorio modificamos UnidadTrabajo.cs

```
public IProductoRepositorio Producto { get; private set; }
.
.
.
public UnidadTrabajo(ApplicationDbContext db)
{
    _db = db;
    Bodega = new BodegaRepositorio(_db);
    Categoria = new CategoriaRepositorio(_db);
    Marca = new MarcaRepositorio(_db);
    Producto = new ProductoRepositorio(_db);
}
```

3. Producto Controlador

- En SistemaInventario/Areas/Admin/Controllers le hacemos una copia a MarcaController.cs y le cambiamos el nombre a ProductoController.cs y cambiamos la información correspondiente.

```
using Microsoft.AspNetCore.Mvc;
using SistemaInventario.AccesoDatos.Repositorio.IRepositorio;
using SistemaInventario.Modelos;
using SistemaInventario.Utilidades;

namespace SistemaInventario.Areas.Admin.Controllers
{
    [Area("Admin")]
    public class ProductoController : Controller
    {
        private readonly IUnidadTrabajo _unidadTrabajo;

        public ProductoController(IUnidadTrabajo unidadTrabajo)
        {
            _unidadTrabajo = unidadTrabajo;
        }
        public IActionResult Index()
        {
            return View();
        }

        public async Task<IActionResult> Upsert(int? id)
        {
            return View();
        }



        #region API
        [HttpGet]
        public async Task<IActionResult> ObtenerTodos()
        {
            var todos = await _unidadTrabajo.Producto.ObtenerTodos(incluirPropiedades:"Categoria,Marca");
            return Json(new { data = todos });
        }

        [HttpPost]
        public async Task<IActionResult> Delete(int id)
        {
            var productoDb = await _unidadTrabajo.Producto.Obtener(id);
            if (productoDb == null)
            {
                return Json(new { success = false, message = "Error al borrar Producto" });
            }
            _unidadTrabajo.Producto.Remover(productoDb);
            await _unidadTrabajo.Guardar();
            return Json(new { success = true, message = "Producto borrado exitosamente" });
        }

        [ActionName("ValidarSerie")]
        public async Task<IActionResult> ValidarSerie(string serie, int id = 0)
        {
            bool valor = false;
```

```
            var lista = await _unidadTrabajo.Producto.ObtenerTodos();
            if (id == 0)
            {
                valor = lista.Any(b => b.NumeroSerie.ToLower().Trim() == serie.ToLower().Trim());
            }
            else
            {
                valor = lista.Any(b => b.NumeroSerie.ToLower().Trim() == serie.ToLower().Trim() && b.Id != id);
            }
            if (valor)
            {
                return Json(new { data = true });
            }
            return Json(new { data = false });

        }

        #endregion
    }
}
```

- En SistemaInventario/Areas/Admin/Views creamos la carpeta Producto; y copiamos de la carpeta Marca el archivo index.cshtml; modificamos el archivo haciendo referencia a Producto.

```
@{
    ViewData["Title"] = "Index";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<div class="row">
    <div class="col-lg-6">
        <h2 class="text-primary">Lista de Productos</h2>
    </div>
    <div class="col-lg-6">
        <a class="btn btn-outline-primary" asp-action="Upsert"> <i class="bi bi-plus-square-fill"></i> Crear Nueva Producto </a>
    </div>
</div>

<br />

<div class="p-4 border rounded bg-light">
    <table id="tblDatos" class="table table-responsive table-hover display nowrap" cellspacing="0" width="100%">
        <thead class="table-dark">
            <tr>
                <th>Numero Serie</th>
                <th>Descripcion</th>
                <th>Categoria</th>
                <th>Marca</th>
```

```
            <th>Precio</th>
            <th>Estado</th>
            <th></th>
          </tr>
        </thead>
    </table>

</div>

@section Scripts {
    <script src="~/js/producto.js"></script>
}
```

- En SistemaInventario/wwwroot le hacemos una copia a marca.js y le cambiamos el nombre a producto.js; realizamos los cambios respectivos.

```
let datatable;

$(document).ready(function () {
    loadDataTable();
});

function loadDataTable() {
    datatable = $('#tblDatos').DataTable({
        "ajax": {
            "url": "/Admin/Producto/ObtenerTodos"
        },
        "columns": [
            { "data": "numeroSerie" },
            { "data": "descripcion" },
            { "data": "categoria.nombre" },
            { "data": "marca.nombre" },
            {
                "data": "precio", "className": "text-end",
                "render": function (data) {
                    var d = data.toFixed(2).replace(/\d(?=(\d{3})+\.)/g, '$&,');
                    return d;
                }
            },
            {
                "data": "estado",
                "render": function (data) {
                    if (data == true) {
                        return "Activo";
                    }
                    else {
                        return "Inactivo";
                    }
                }
            },
            {
                "data": "id",
```

```
            "render": function (data) {
                return `
                    <div class="text-center">
                      <a href="/Admin/Producto/Upsert/${data}" class="btn btn-success text-white"
style="cursor:pointer">
                        <i class="bi bi-pencil-square"></i>
                      </a>
                      <a onclick=Delete("/Admin/Producto/Delete/${data}") class="btn btn-danger text-white"
style="cursor:pointer">
                        <i class="bi bi-trash3-fill"></i>
                      </a>
                    </div>
                `;
            }, "width": "20%"
        }
      ]

  });
}

function Delete(url) {

  swal({
    title: "Esta seguro de Eliminar el Producto?",
    text: "Este registro no se podra recuperar",
    icon: "warning",
    buttons: true,
    dangerMode: true
  }).then((borrar) => {
    if (borrar) {
      $.ajax({
        type: "POST",
        url: url,
        success: function (data) {
          if (data.success) {
            toastr.success(data.message);
            datatable.ajax.reload();
          }
          else {
            toastr.error(data.message);
          }
        }
      });
    }
  });
}
```

- Actualizamos el archivo SistemaInventario/Views/Shared/_Layout.cshtml

```
<ul class="dropdown-menu">
  <li><a class="dropdown-item" asp-area="Admin" asp-controller="Bodega" asp-
action="Index">Bodegas</a></li>
```

```
  <li><a class="dropdown-item" asp-area="Admin" asp-controller="Categoria" asp-action="Index">Categorias</a></li>
  <li><a class="dropdown-item" asp-area="Admin" asp-controller="Marca" asp-action="Index">Marcas</a></li>
  <li><a class="dropdown-item" asp-area="Admin" asp-controller="Producto" asp-action="Index">Productos</a></li>
</ul>
```

- Ejecutamos la aplicación para verificar la funcionalidad de los productos.

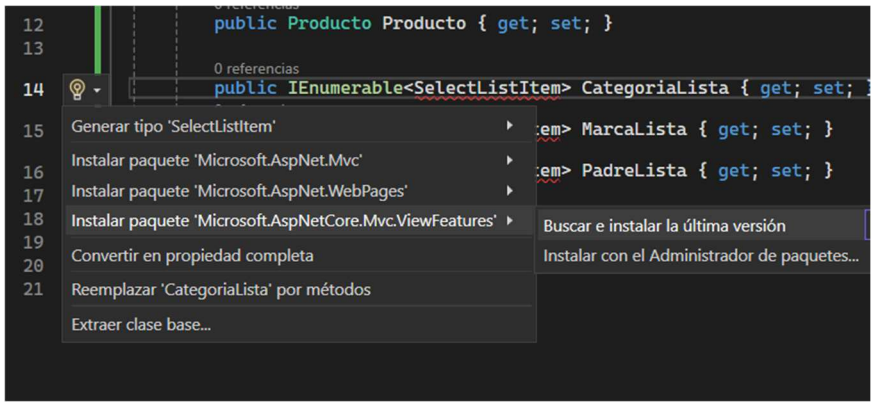
4. Producto ViewModel

- En SistemaInventario.Modelo creamos una carpeta llamada ErrorViewModels y pasamos el archivo ViewModels/ ErrorViewModel.cs a esa carpeta.

- En SistemaInventario.Modelo/ViewModels creamos una clase llamada ProductoVM.cs

```
using Microsoft.AspNetCore.Mvc.Rendering;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SistemaInventario.Modelos.ViewModels
{
  public class ProductoVM
  {
    public Producto Producto { get; set; }

    public IEnumerable<SelectListItem> CategoriaLista { get; set; }
    public IEnumerable<SelectListItem> MarcaLista { get; set; }
    public IEnumerable<SelectListItem> PadreLista { get; set; }

  }
}
```

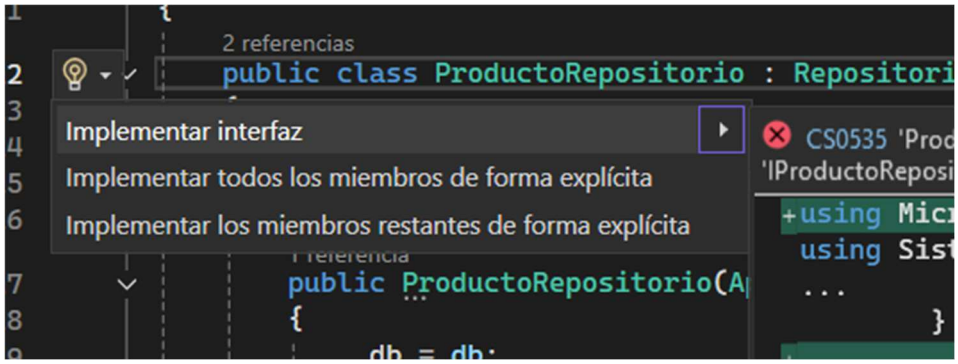
- Para corregir los errores instalamos paquete Mvc.ViewFeatures



- Actualizamos el archivo SistemaInventario.AccesoDatos/Repositorio/IRepositorio/IProductoRepositorio.cs

```
public interface IProductoRepositorio : IRepositorio<Producto>
{
   void Actualizar(Producto producto);
   IEnumerable<SelectListItem> ObtenerTodosDropdownLista(string obj);

}
```

- Actualizamos el archivo SistemaInventario.AccesoDatos/Repositorio/ProductoRepositorio.cs; primero con implementar interfaz, y actualizamos la interfaz.




```
public IEnumerable<SelectListItem> ObtenerTodosDropdownLista(string obj)
{
   if (obj == "Categoria")
   {
      return _db.Categorias.Where(c => c.Estado == true).Select(c => new SelectListItem
      {
         Text = c.Nombre,
         Value = c.Id.ToString()
      });
   }
   if (obj == "Marca")
   {
```

```
        return _db.Marcas.Where(c => c.Estado == true).Select(c => new SelectListItem
        {
            Text = c.Nombre,
            Value = c.Id.ToString()
        });
    }
    return null;
}
```

- En SistemaInventario/Areas/Admin/Crontrollers/ProductoController.cs actualizamos el método Upsert

```
public async Task<IActionResult> Upsert(int? id)
{

    ProductoVM productoVM = new ProductoVM()
    {
        Producto = new Producto(),
        CategoriaLista = _unidadTrabajo.Producto.ObtenerTodosDropdownLista("Categoria"),
        MarcaLista = _unidadTrabajo.Producto.ObtenerTodosDropdownLista("Marca"),
        PadreLista = _unidadTrabajo.Producto.ObtenerTodosDropdownLista("Producto")

    };

    if(id == null)
    {
        // Crear nuevo Producto
        productoVM.Producto.Estado = true;
        return View(productoVM);
    }
    else
    {
        productoVM.Producto = await _unidadTrabajo.Producto.Obtener(id.GetValueOrDefault());
        if(productoVM.Producto == null)
        {
            return NotFound();
        }
        return View(productoVM);
    }
}
```

5. Producto Upsert View

- En SistemaInventario/wwwroot creamos una carpeta llamada imagenes/producto para alojar las imágenes de los productos.

- En SistemaInventario/Areas/Admin/Crontrollers/ProductoController.cs seleccionamos el método Upsert para crear una Vista Razor, desmarcamos la parte de "Crear como vista parcial"

```
@model SistemaInventario.Modelos.ViewModels.ProductoVM
@using SistemaInventario.Utilidades
@{
   Layout = "~/Views/Shared/_Layout.cshtml";
   var titulo = "Crear Nuevo Producto";
}

<form method="post" enctype="multipart/form-data">
   @if(Model.Producto.Id !=0)
   {
      titulo = "Editar Producto";
      <input asp-for="Producto.Id" hidden />
      <input asp-for="Producto.ImagenUrl" hidden />
   }

   <div class="border p-3">
      <div asp-validation-summary="ModelOnly" class="text-danger"></div>
      <div class="row">
         <h2 class="text-info">@titulo</h2>
      </div>

      <div class="row">
         <div class="col-8">
            <div class="container">
               <div class="row">
                  <div class="col-md-6">
                     <label>Numero de Serie</label>
                     <input asp-for="Producto.NumeroSerie" class="form-control" />
                     <span asp-validation-for="Producto.NumeroSerie" class="text-danger"></span>
                  </div>
               </div>

               <div class="row">
                  <div class="col-md-6">
                     <label>Descripcion</label>
                     <input asp-for="Producto.Descripcion" class="form-control" />
                     <span asp-validation-for="Producto.Descripcion" class="text-danger"></span>
                  </div>
               </div>

               <div class="row">
                  <div class="col-md-6">
                     <label>Costo</label>
                     <input asp-for="Producto.Costo" class="form-control" />
                     <span asp-validation-for="Producto.Costo" class="text-danger"></span>
                  </div>
               </div>

                <div class="row">
                  <div class="col-md-6">
                     <label>Precio</label>
                     <input asp-for="Producto.Precio" class="form-control" />
```

```
                <span asp-validation-for="Producto.Precio" class="text-danger"></span>
              </div>
            </div>

            <div class="row">
              <div class="col-md-6">
                <label>Imagen</label>
                <input type="file" accept="image/png, image/gif, image/jpeg" name="files" id="imagenId" class="form-control" />
                <span asp-validation-for="Producto.Precio" class="text-danger"></span>
              </div>
            </div>

            <div class="row">
              <div class="col-md-6">
                <label>Categoria</label>
                <select asp-for="Producto.CategoriaId" asp-items="@Model.CategoriaLista" class="form-select">
                  <option disabled selected>-- Seleccione una Categoria--</option>
                </select>
                <span asp-validation-for="Producto.CategoriaId" class="text-danger"></span>
              </div>
            </div>

            <div class="row">
              <div class="col-md-6">
                <label>Marca</label>
                <select asp-for="Producto.MarcaId" asp-items="@Model.MarcaLista" class="form-select">
                  <option disabled selected>-- Seleccione una Marca--</option>
                </select>
                <span asp-validation-for="Producto.MarcaId" class="text-danger"></span>
              </div>
            </div>


            <div class="row">
              <div class="col-md-6">
                <label>Producto Padre</label>
                <select asp-for="Producto.PadreId" asp-items="@Model.PadreLista" class="form-select">
                  <option disabled selected>-- Producto Padre--</option>
                </select>
              </div>
            </div>

            <div class="row">
              <div class="col-md-6">
                <label>Estado</label>
                <select asp-for="Producto.Estado" class="form-select">
                  <option value="true">Activo</option>
                  <option value="false">Inactivo</option>
                </select>
                <span asp-validation-for="Producto.Estado" class="text-danger"></span>
              </div>
```

```
            </div>



            <br />

            <div class="d-grid gap-2 d-md-block">
                @if(Model.Producto.Id !=0)
                {
                    <partial name="_BotonesActualizarYRegresar" model="Model.Producto.Id" />
                }
                else
                {
                    <button type="submit" onclick="return validarImagen()" class="btn btn-primary"><i class="bi bi-plus-square-fill"></i> Crear</button>
                    <a asp-action="Index" class="btn btn-success "><i class="bi bi-arrow-return-left"></i> Regresar</a>
                }

            </div>


        </div>
    </div>

    <div class="col-4">
        @if(Model.Producto.Id !=0)
        {
            <img src="@DS.ImagenRuta@Model.Producto.ImagenUrl" width="100%" style="border-radius:5px; border:1px solid #bbb" />
        }
    </div>


</div>


</div>


</form>

@section Scripts {
    <partial name="_ValidationScriptsPartial"/>

    <script>

        function validarImagen() {
            if (document.getElementById("imagenId").value=="")
            {
                swal("Error", "Seleccione una Imagen!", "error")
                return false;
            }
```

```
            return true;
        }

    </script>

}
```

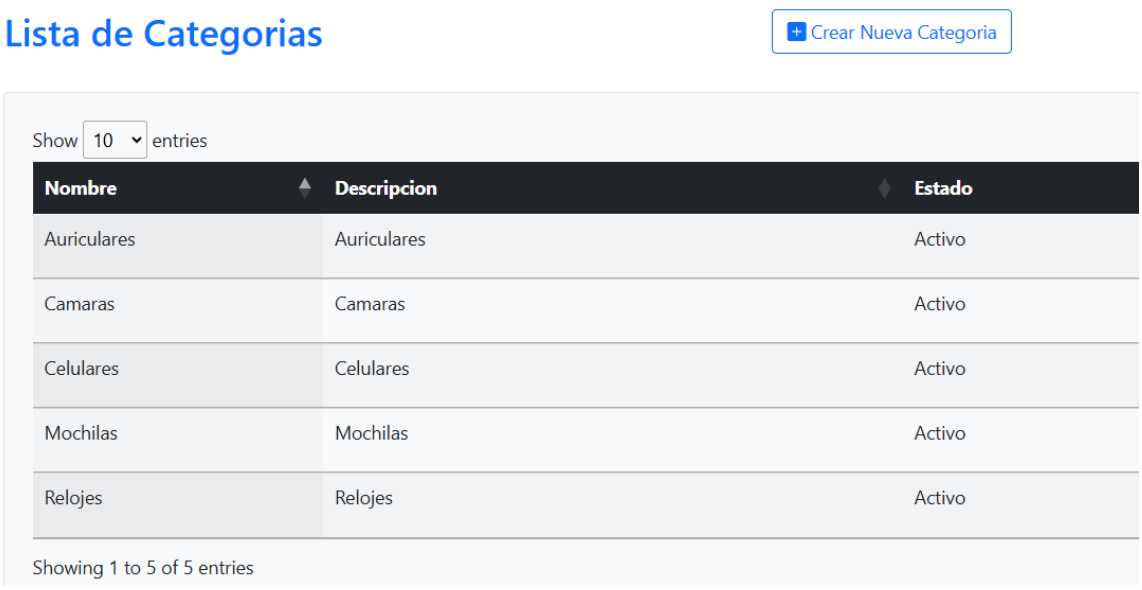
6. **Upsert Post**

- Actualizamos SistemaInventario/Areas/Admin/Controllers/ProductoController.cs y creamos el método Upsert Http Post.

```
public class ProductoController : Controller
 {
    private readonly IUnidadTrabajo _unidadTrabajo;
    private readonly IWebHostEnvironment _webHostEnvironment;

    public ProductoController(IUnidadTrabajo unidadTrabajo, IWebHostEnvironment webHostEnvironment)
    {
       _unidadTrabajo = unidadTrabajo;
       _webHostEnvironment = webHostEnvironment;
    }
.
.
.
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Upsert(ProductoVM productoVM)
{
   if(ModelState.IsValid)
   {
      var files = HttpContext.Request.Form.Files;
      string webRootPath = _webHostEnvironment.WebRootPath;

      if(productoVM.Producto.Id == 0)
      {
         // Crear
         string upload = webRootPath + DS.ImagenRuta;
         string fileName = Guid.NewGuid().ToString();
         string extension = Path.GetExtension(files[0].FileName);

        using(var fileStream = new FileStream(Path.Combine(upload,fileName+extension),FileMode.Create))
         {
            files[0].CopyTo(fileStream);
         }
         productoVM.Producto.ImagenUrl = fileName + extension;
         await _unidadTrabajo.Producto.Agregar(productoVM.Producto);
      }
      else
      {
         // Actualizar
         var objProducto = await _unidadTrabajo.Producto.ObtenerPrimero(p=>p.Id ==
productoVM.Producto.Id, isTracking:false);
         if(files.Count >0)  // Si se carga una nueva Imagen para el producto existente
         {
            string upload = webRootPath + DS.ImagenRuta;
            string fileNAme = Guid.NewGuid().ToString();
            string extension = Path.GetExtension(files[0].FileName);
```

```
            //Borrar la imagen anterior
            var anteriorFile = Path.Combine(upload, objProducto.ImagenUrl);
            if(System.IO.File.Exists(anteriorFile))
            {
               System.IO.File.Delete(anteriorFile);
            }
            using(var fileStream = new FileStream(Path.Combine(upload, fileNAme + extension),
FileMode.Create))
            {
               files[0].CopyTo(fileStream);
            }
            productoVM.Producto.ImagenUrl = fileNAme + extension;
         } // Caso contrario no se carga una nueva imagen
         else
         {
            productoVM.Producto.ImagenUrl = objProducto.ImagenUrl;
         }
         _unidadTrabajo.Producto.Actualizar(productoVM.Producto);
      }
      TempData[DS.Exitosa] = "Transaccion Exitosa!";
      await _unidadTrabajo.Guardar();
      //return View("Index");
      return RedirectToAction("Index");

   } // If not Valid
   productoVM.CategoriaLista = _unidadTrabajo.Producto.ObtenerTodosDropdownLista("Categoria");
   productoVM.MarcaLista = _unidadTrabajo.Producto.ObtenerTodosDropdownLista("Marca");
   productoVM.PadreLista = _unidadTrabajo.Producto.ObtenerTodosDropdownLista("Producto");
   return View(productoVM);
}
```

- Ejecutar la aplicación y crear Categorías, Marcas y Productos



Lista de Marcas

Crear Nueva Marca

Show 10 entries

Nombre	Descripcion	Estado
Android	Android	Activo
Apple	Apple	Activo
Cannon	Cannon	Activo
Casio	Casio	Activo
GoPro	GoPro	Activo
Logitech	Logitech	Activo
Michael Kors	Michael Kors	Activo
Rolex	Rolex	Activo

Showing 1 to 8 of 8 entries

Lista de Productos

Crear Nuevo Producto

Show 10 entries

Numero Serie	Descripcion	Categoria	Marca	Precio	Estado
11111	Apple IPhone 12 Pro 6.1 Ram 6GB 512FB Unlocked	Celulares	Apple	900.00	Activo
22222	AppleWatchSeries 1 Sport Case 38mm Black	Relojes	Apple	800.00	Activo

Showing 1 to 2 of 2 entries

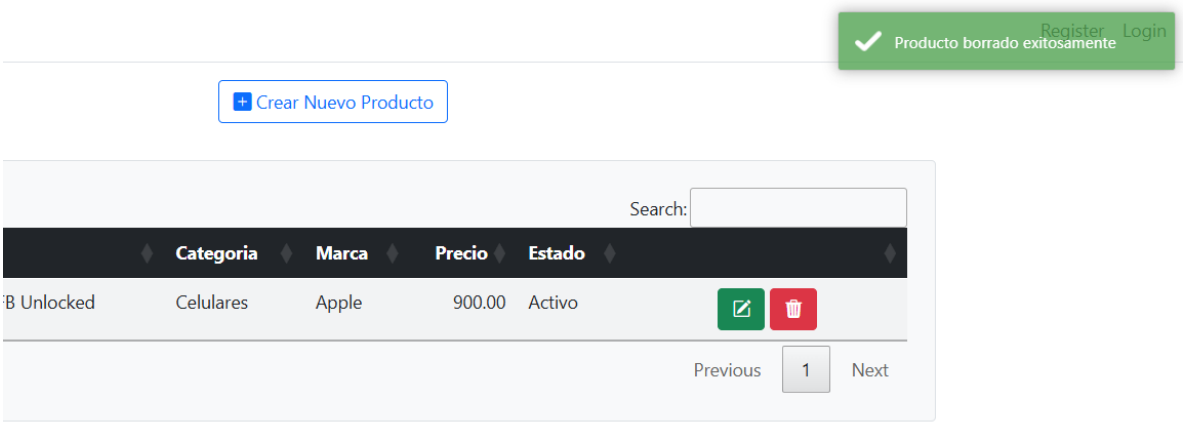
7. **Producto Delete**

- En SistemaInventario/Areas/Admin/Controllers/ProductoController.cs Creamos el método Delete dentro de la regio API

```
[HttpPost]
public async Task<IActionResult> Delete(int id)
{
   var productoDb = await _unidadTrabajo.Producto.Obtener(id);
   if(productoDb == null)
```

```
    {
        return Json(new { success = false, message = "Error al borrar Producto" });
    }

    // Remover imagen
    string upload = _webHostEnvironment.WebRootPath + DS.ImagenRuta;
    var anteriorFile = Path.Combine(upload, productoDb.ImagenUrl);
    if(System.IO.File.Exists(anteriorFile))
    {
        System.IO.File.Delete(anteriorFile);
    }

    _unidadTrabajo.Producto.Remover(productoDb);
    await _unidadTrabajo.Guardar();
    return Json(new { success = true, message = "Producto borrado exitosamente" });
}
```

- Ejecutamos la aplicación para validar Delete



8. **Producto Aplicar Recursividad**

- Actualizamos el método ObtenerTodosDropdownLista de SistemaInventario.AccesoDatos/Repositorio/ProductoRepositorio.cs

```
public IEnumerable<SelectListItem> ObtenerTodosDropdownLista(string obj)
{
    if (obj == "Categoria")
    {
        return _db.Categorias.Where(c => c.Estado == true).Select(c => new SelectListItem
        {
            Text = c.Nombre,
            Value = c.Id.ToString()
        });
    }
```

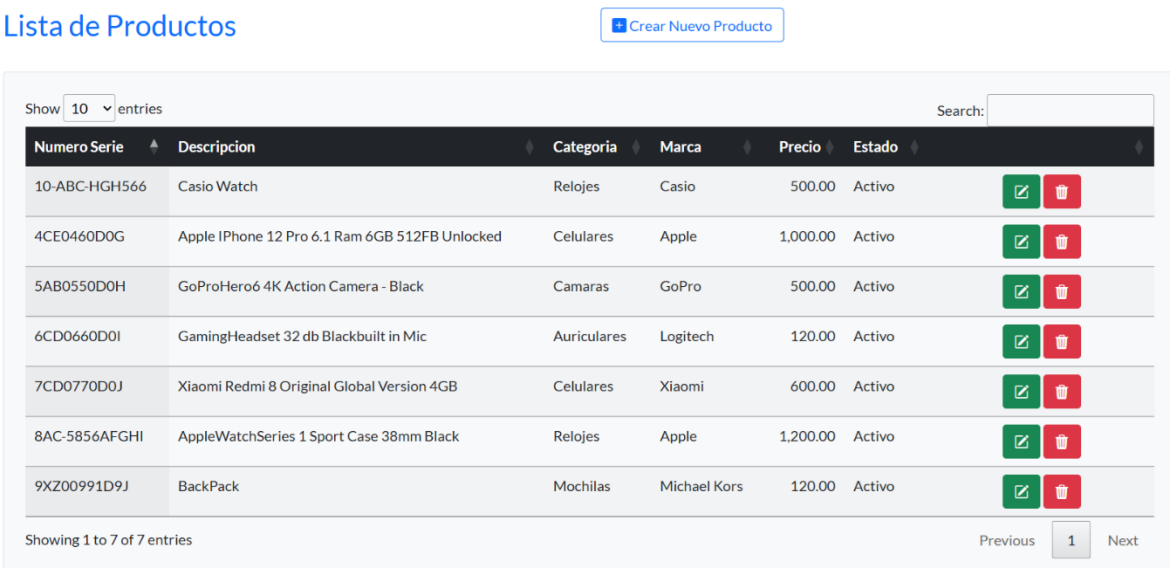
```
    if (obj == "Marca")
    {
        return _db.Marcas.Where(c => c.Estado == true).Select(c => new SelectListItem
        {
            Text = c.Nombre,
            Value = c.Id.ToString()
        });
    }
    if (obj == "Producto")
    {
        return _db.Productos.Where(c => c.Estado == true).Select(c => new SelectListItem
        {
            Text = c.Descripcion,
            Value = c.Id.ToString()
        });
    }
    return null;
}
```

- Ejecutamos la aplicación (no es obligatorio que un producto tenga un padre)

Diseño Página Principal

1. Diseño Página Principal. Desplegar Todos Los Productos

- Incluir listado de productos



- Actualizamos los link del archivo SistemaInventario/Views/Shared/ _Layout.cshtml

```
<link rel="shortcut icon" href="~/imagenes//icono.ico" />
<link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />
```

```
<link rel="stylesheet" href="~/css/site.css" asp-append-version="true" />
<link rel="stylesheet" href="~/SistemaInventario.styles.css" asp-append-version="true" />
<link rel="stylesheet" href="https://cdn.datatables.net/1.13.3/css/jquery.dataTables.min.css" />
<link rel="stylesheet" href="https://cdn.datatables.net/1.13.3/css/jquery.dataTables.min.css" />
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/toastr.js/latest/toastr.min.css" />
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.10.3/font/bootstrap-icons.css">
<link rel="preconnect" href="https://fonts.googleapis.com">
<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
<link
href="https://fonts.googleapis.com/css2?family=Lato:ital,wght@0,300;0,400;0,700;0,900;1,400&display=swap" rel="stylesheet">
```

- Actualizamos el archivo SistemaInventario/wwwroot/css/site.css con los estilos

```
html {
  font-size: 14px;
}

@media (min-width: 768px) {
  html {
    font-size: 16px;
  }
}

.btn:focus, .btn:active:focus, .btn-link.nav-link:focus, .form-control:focus, .form-check-input:focus {
  box-shadow: 0 0 0 0.1rem white, 0 0 0 0.25rem #258cfb;
}

html {
  position: relative;
  min-height: 100%;
}

body {
    margin-bottom: 10px;
    font-family: "Lato", sans-serif;

}


.header {
    height: 20vh;
    background: linear-gradient(to right, #2b34bae3, #d1d4f1ea);
    clip-path: polygon(0 0, 100% 0, 100% 75%, 0 100%);
}

.heading-primary {
    font-weight: 900;
    color: white;
    text-transform: uppercase;
    letter-spacing: 0.50rem;
}
```

```
@media only screen and (max-width: 51.25rem) {
  .heading-primary {
    font-weight: 400;
    letter-spacing: 0.1rem;
  }

  .header {
    height: 13vh;
    background: linear-gradient(to right, #2b34bae3, #d1d4f1ea);
    clip-path: polygon(0 0, 100% 0, 100% 75%, 0 100%);
  }
}

@media only screen and (max-width: 57rem) {
  .heading-primary {
    font-weight: 400;
    letter-spacing: 0.1rem;
  }

  .header {
    height: 13vh;
    background: linear-gradient(to right, #2b34bae3, #d1d4f1ea);
    clip-path: polygon(0 0, 100% 0, 100% 75%, 0 100%);
  }
}

@media only screen and (max-width: 37.5rem) {
  .heading-primary {
    font-weight: 400;
    letter-spacing: 0.1rem;
  }
  .header {
    height: 13vh;
    background: linear-gradient(to right, #2b34bae3, #d1d4f1ea);
    clip-path: polygon(0 0, 100% 0, 100% 75%, 0 100%);
  }
}

@media only screen and (max-width: 25.9rem) {
  .heading-primary {
    font-weight: 400;
    font-size: 0.7rem;
    padding-top: 0.5rem;
    letter-spacing: 0.1rem;
  }

  .header {
    height: 10vh;
    background: linear-gradient(to right, #2b34bae3, #d1d4f1ea);
    clip-path: polygon(0 0, 100% 0, 100% 75%, 0 100%);
  }
```

```
}

@media only screen and (max-width: 23.5rem) {
  .heading-primary {
    font-weight: 400;
    font-size: 0.7rem;
    padding-top: 0.5rem;
    letter-spacing: 0.1rem;
  }

  .header {
    height: 10vh;
    background: linear-gradient(to right, #2b34bae3, #d1d4f1ea);
    clip-path: polygon(0 0, 100% 0, 100% 75%, 0 100%);
  }
}
```

- En SistemaInventario/Views/Shared/_Layout.cshtml adicionamos el icono.png

```
<div class="container-fluid">
    <a class="navbar-brand" asp-area="" asp-controller="Home" asp-action="Index"><img
src="~/imagenes/icono.png" width="50px;/></a>
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target=".navbar-collapse"
aria-controls="navbarSupportedContent"
```

- Modificamos SistemaInventario/Areas/Inventario/Controllers/HomeController.cs para la página principal.

```
namespace SistemaInventario.Areas.Inventario.Controllers
{
  [Area("Inventario")]
  public class HomeController : Controller
  {
    private readonly ILogger<HomeController> _logger
    private readonly IUnidadTrabajo _unidadTrabajo;

    public HomeController(ILogger<HomeController> logger, IUnidadTrabajo unidadTrabajo)
    {
      _logger = logger;
      _unidadTrabajo = unidadTrabajo;
    }

    public async Task<IActionResult> Index()
    {
      IEnumerable<Producto> productoLista = await _unidadTrabajo.Producto.ObtenerTodos();
      return View(productoLista);
    }

    public IActionResult Privacy()
    {
      return View();
    }
```

```
    [ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]
    public IActionResult Error()
    {
        return View(new ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier });
    }
  }
}
```

- Modificamos SistemaInventario/Areas/Inventario/Views/Home/index.cshtml

```
@using SistemaInventario.Utilidades
@model IEnumerable<SistemaInventario.Modelos.Producto>
@{
  ViewData["Title"] = "Home Page";
}

<header class="header">
  <div class="container pt-sm-5">
    <div class="heading-primary">
      Los Mejores Productos & <br />
      Marcas en nuestra Tienda
    </div>
  </div>
</header>


<section>
  <div class="container my-2">
    <div class="col-lg-6 col-md-6 col-sm-6 d-flex">
      <div class="input-group mb-3">
        <input type="text" class="form-control" placeholder="Buscar ..." aria-label="Buscar" aria-
describedby="button-addon2" />
        <button type="submit" class="btn btn-outline-primary">
          <i class="bi bi-search"></i>
        </button>
      </div>
    </div>
    <div class="row">
      @foreach (var producto in Model)
      {
        <div class="col-lg-3 col-md-6 col-sm-6 d-flex">
          <div class="card w-100 my-2">
            <img src="@DS.ImagenRuta@producto.ImagenUrl" class="card-img-top" style="aspect-ratio: 1 /
1" />
            <div class="card-body d-flex flex-column">
              <h5 class="card-title">@producto.Descripcion</h5>
              <p class="card-text"> $ @String.Format("{0:#,##0.00}", producto.Precio) </p>
              <div class="card-footer d-flex align-items-end pt-3 px-0 pb-0 mt-auto bg-white">
                <a href="#" class="btn btn-outline-primary">
                  <i class="bi bi-tags-fill"></i> Detalle
                </a>
```

```
                </div>
              </div>
            </div>
          </div>
        }
      </div>
    </div>
</section>
```

Paginación

1. Crear Modelos para la Paginación

- En SistemaInventario.Modelos creamos una capeta llamada Especificaciones y dentro tres clases llamadas Parametros.cs, Metadata.cs y PagedList.cs

Parametros.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SistemaInventario.Modelos.Especificaciones
{
  public class Parametros
  {
    public int PageNumber { get; set; } = 1;
    public int PageSize { get; set; } = 4;
  }
}
```

Metadata.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SistemaInventario.Modelos.Especificaciones
{
  public class MetaData
  {
    public int TotalPages { get; set; }
    public int PageSize { get; set; }
    public int TotalCount { get; set; }
  }
}
```

PagedList.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SistemaInventario.Modelos.Especificaciones
{
   public class PagedList<T> : List<T>
   {
      public MetaData MetaData { get; set; }

      public PagedList(List<T> items, int count, int pageNumber, int pageSize)
      {
         MetaData = new MetaData
         {
            TotalCount = count,
            PageSize = pageSize,
            TotalPages = (int)Math.Ceiling(count / (double)pageSize)   // Por ejemplo 1.5 lo transforma a 2
         };
         AddRange(items);  // Agrega los elementos de la coleccion al final de la lista
      }

      public static PagedList<T> ToPagedList(IEnumerable<T> entidad, int pageNumber, int pageSize)
      {
         var count = entidad.Count();
         var items = entidad.Skip((pageNumber-1) * pageSize).Take(pageSize).ToList();
         return new PagedList<T>(items, count, pageNumber, pageSize);

      }

   }
}
```

- En SistemaInventario.AccesoDatos/Repositorio/IRepositorio/IRepositorio.cs agregamos método ObtenerTodosPaginado de tipo PagedList.

```
PagedList<T> ObtenerTodosPaginado(Parametros parametros, Expression<Func<T, bool>> filtro = null,
   Func<IQueryable<T>, IOrderedQueryable<T>> orderBy = null,
   string incluirPropiedades = null,
   bool isTracking = true);
```

- En SistemaInventario.AccesoDatos/Repositorio/IRepositorio/Repositorio.cs Corregimos el error implementando la interfaz


```
public PagedList<T> ObtenerTodosPaginado(Parametros parametros, Expression<Func<T, bool>> filtro = null,
Func<IQueryable<T>, IOrderedQueryable<T>> orderBy = null, string incluirPropiedades = null, bool isTracking
= true)
{
   IQueryable<T> query = dbSet;
   if (filtro != null)
   {
      query = query.Where(filtro);   //  select /* from where ....
   }
   if (incluirPropiedades != null)
   {
      foreach (var incluirProp in incluirPropiedades.Split(new char[] { ',' },
StringSplitOptions.RemoveEmptyEntries))
      {
         query = query.Include(incluirProp);    //  ejemplo "Categoria,Marca"
      }
   }
   if (orderBy != null)
   {
      query = orderBy(query);
   }
   if (!isTracking)
   {
      query = query.AsNoTracking();
   }
   return PagedList<T>.ToPagedList(query, parametros.PageNumber, parametros.PageSize);
}
```

2. Manejar Paginación en Producto Controller

- Modificamos SistemaInventario/Areas/Inventario/Controllers/HomeController.cs en el método index

```
public IActionResult Index(int pageNumber=1)
{
   if (pageNumber < 1) { pageNumber = 1; }
```

```
    Parametros parametros = new Parametros()
    {
        PageNumber = pageNumber,
        PageSize = 4
    };

    var resultado = _unidadTrabajo.Producto.ObtenerTodosPaginado(parametros);

    resultado = _unidadTrabajo.Producto.ObtenerTodosPaginado(parametros);

    ViewData["TotalPaginas"] = resultado.MetaData.TotalPages;
    ViewData["TotalRegistros"] = resultado.MetaData.TotalCount;
    ViewData["PageSize"] = resultado.MetaData.PageSize;
    ViewData["PageNumber"] = pageNumber;
    ViewData["Previo"] = "disabled";  // clase css para desactivar el boton
    ViewData["Siguiente"] = "";

    if (pageNumber > 1) { ViewData["Previo"] = ""; }
    if (resultado.MetaData.TotalPages <= pageNumber) { ViewData["Siguiente"] = "disabled"; }

    return View(resultado);
}
```

- Modificamos SistemaInventario/Areas/Inventario/Views/Home/index.cshtml.

Esto: @model IEnumerable<SistemaInventario.Modelos.Producto>

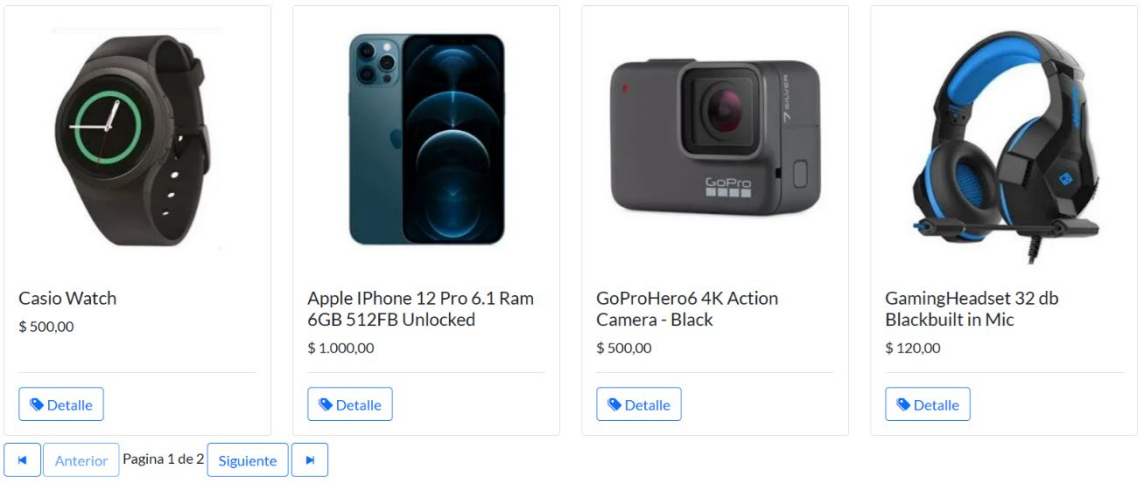
Por: @model SistemaInventario.Modelos.Especificaciones.PagedList<SistemaInventario.Modelos.Producto>

.
.
.

```
        </div>
        <a asp-action="Index"
          asp-route-pageNumber="@(1)" class="btn btn-outline-primary">
           <i class="bi bi-skip-start-fill"></i>
        </a>

        <a asp-action="Index"
          asp-route-pageNumber="@((int)ViewData["PageNumber"]- 1)" class="btn btn-outline-primary
@ViewData["Previo"]">
           Anterior
        </a>
        <span>Pagina @ViewData["PageNumber"] de @ViewData["TotalPaginas"]</span>
        <a asp-action="Index"
          asp-route-pageNumber="@((int)ViewData["PageNumber"] + 1)" class="btn btn-outline-primary
@ViewData["Siguiente"]">
           Siguiente
        </a>
        <a asp-action="Index"
          asp-route-pageNumber="@ViewData["TotalPaginas"]" class="btn btn-outline-primary">
           <i class="bi bi-skip-end-fill"></i>
```

```
    </a>
  </div>
</section>
```

- Ejecutamos la aplicación para verificar resultados



3. Filtro de Productos

- Modificamos SistemaInventario/Areas/Inventario/Controllers/HomeController.cs en el método index

```
public IActionResult Index(int pageNumber = 1, string busqueda = "", string busquedaActual = "")
{
   if (!String.IsNullOrEmpty(busqueda))
   {
      pageNumber = 1;
   }
   else
   {
      busqueda = busquedaActual;
   }
   ViewData["BusquedaActual"] = busqueda;
.
.
.
var resultado = _unidadTrabajo.Producto.ObtenerTodosPaginado(parametros);

if (!String.IsNullOrEmpty(busqueda))
{
   resultado = _unidadTrabajo.Producto.ObtenerTodosPaginado(parametros, p =>
p.Descripcion.Contains(busqueda));
}

ViewData["TotalPaginas"] = resultado.MetaData.TotalPages;
ViewData["TotalRegistros"] = resultado.MetaData.TotalCount;
```

.
.
.

- Modificamos SistemaInventario/Areas/Inventario/Views/Home/index.cshtml.

```
<section>
  <div class="container my-2">
    <form asp-action="Index" method="get">
      <div class="col-lg-6 col-md-6 col-sm-6 d-flex">
        <div class="input-group mb-3">
          <input type="text" class="form-control" placeholder="Buscar ..." aria-label="Buscar" aria-
describedby="button-addon2"
                 name="busqueda" value="@ViewData["BusquedaActual"]" />
          <button type="submit" class="btn btn-outline-primary">
            <i class="bi bi-search"></i>
          </button>
        </div>
      </div>
    </form>
.
.
.

<a asp-action="Index"
  asp-route-pageNumber="@(1)" class="btn btn-outline-primary"
  asp-route-busquedaActual="@ViewData["BusquedaActual"]"
  >
   <i class="bi bi-skip-start-fill"></i>
</a>

<a asp-action="Index"
  asp-route-pageNumber="@((int)ViewData["PageNumber"]-1)" class="btn btn-outline-primary
@ViewData["Previo"]"
  asp-route-busquedaActual="@ViewData["BusquedaActual"]">
   Anterior
</a>
<span>Pagina @ViewData["PageNumber"] de @ViewData["TotalPaginas"]</span>
<a asp-action="Index"
  asp-route-pageNumber="@((int)ViewData["PageNumber"] +1)" class="btn btn-outline-primary
@ViewData["Siguiente"]"
  asp-route-busquedaActual="@ViewData["BusquedaActual"]">
   Siguiente
</a>
<a asp-action="Index"
  asp-route-pageNumber="@ViewData["TotalPaginas"]" class="btn btn-outline-primary"
  asp-route-busquedaActual="@ViewData["BusquedaActual"]">
   <i class="bi bi-skip-end-fill"></i>
</a>
```

- Ejecutamos la aplicación para verificar resultados de busqueda

casio

Casio Watch

$ 500,00

Detalle