Guía Paso a Paso: Sistema ASP.NET Core MVC - Productos y Categorías

# 1. Crear el proyecto

En Visual Studio:

- Nuevo proyecto → ASP.NET Core Web App (Model-View-Controller)

- Nombre del proyecto: SistemaProductosMVC

- Framework: .NET 8 o superior

- Desmarcar "Habilitar autenticación".

Esto generará una estructura con carpetas como Controllers, Models, Views, wwwroot, etc.

# 2. Instalar paquetes necesarios

Abre la consola del Administrador de Paquetes y ejecuta:

Install-Package Microsoft.EntityFrameworkCore.SqlServer

Install-Package Microsoft.EntityFrameworkCore.Tools

Install-Package Microsoft.VisualStudio.Web.CodeGeneration.Design

En una terminal (para scaffolding):

dotnet tool install --global dotnet-aspnet-codegenerator

# 3. Crear los modelos

📁 Carpeta: Models

Crea dos archivos: CategoriaProducto.cs y Producto.cs

📄 CategoriaProducto.cs:

------------------------

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

namespace SistemaProductosMVC.Models

{

public class CategoriaProducto

{

public int Id { get; set; }

[Required(ErrorMessage = "El nombre es obligatorio")]

public string Nombre { get; set; }

// Relación: una categoría tiene muchos productos

public ICollection<Producto> Productos { get; set; }

}

}

📄 Producto.cs:

---------------

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace SistemaProductosMVC.Models

{

public class Producto

{

public int Id { get; set; }

[Required(ErrorMessage = "El nombre es obligatorio")]

public string Nombre { get; set; }

[Range(0.01, 10000)]

public decimal Precio { get; set; }

[Display(Name = "Categoría")]

public int CategoriaProductoId { get; set; }

// Relación con la categoría

public CategoriaProducto CategoriaProducto { get; set; }

}

}

# 4. Crear el DbContext

📁 Carpeta: Data

Archivo: ApplicationDbContext.cs

📄 ApplicationDbContext.cs:

---------------------------

using Microsoft.EntityFrameworkCore;

using SistemaProductosMVC.Models;

namespace SistemaProductosMVC.Data

{

public class ApplicationDbContext : DbContext

{

public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options) : base(options) { }

// DbSet = Tabla en la base de datos

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

public DbSet<CategoriaProducto> CategoriasProducto { get; set; }

}

}

# 5. Configurar cadena de conexión

📄 Archivo: appsettings.json

"ConnectionStrings": {

"DefaultConnection": "Server=localhost;Database=SistemaProductosDB;Trusted\_Connection=True;TrustServerCertificate=True;"

}

# 6. Registrar el DbContext en Program.cs

📄 Archivo: Program.cs

// Agregar al inicio del archivo

using SistemaProductosMVC.Data;

// Registrar el contexto antes de app.Build()

builder.Services.AddDbContext<ApplicationDbContext>(options =>

options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection")));

# 7. Crear la base de datos

Desde la Consola del Administrador de Paquetes:

Add-Migration Inicial

Update-Database

Esto crea las tablas en la base de datos SistemaProductosDB.

# 8. Generar controladores y vistas

Desde la terminal en la carpeta del proyecto:

dotnet aspnet-codegenerator controller -name CategoriaProductoController -m CategoriaProducto -dc ApplicationDbContext --useDefaultLayout --referenceScriptLibraries

dotnet aspnet-codegenerator controller -name ProductoController -m Producto -dc ApplicationDbContext --useDefaultLayout --referenceScriptLibraries

# 9. Agregar enlaces al menú

📄 Archivo: Views/Shared/\_Layout.cshtml

<li class="nav-item">

<a class="nav-link text-dark" asp-controller="CategoriaProducto" asp-action="Index">Categorías</a>

</li>

<li class="nav-item">

<a class="nav-link text-dark" asp-controller="Producto" asp-action="Index">Productos</a>

</li>

# 10. Dropdown en formulario de productos

📄 En ProductoController.cs (Create/Edit):

ViewData["CategoriaProductoId"] = new SelectList(\_context.CategoriasProducto, "Id", "Nombre", producto?.CategoriaProductoId);

📄 En Create.cshtml y Edit.cshtml:

<div class="form-group">

<label asp-for="CategoriaProductoId" class="control-label"></label>

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

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

</div>

# ¡Sistema completo!

Ya puedes:

- Crear y editar categorías

- Asociar productos a cada categoría

- Mostrar datos dinámicamente en vistas

Controladores ASP.NET Core MVC: Producto y CategoriaProducto (Comentados)

**Controlador: Producto**

ProductoController.cs

---------------------

// Usamos los espacios de nombres necesarios

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Rendering;

using Microsoft.EntityFrameworkCore;

using SistemaProductosMVC.Data;

using SistemaProductosMVC.Models;

namespace SistemaProductosMVC.Controllers

{

public class ProductoController : Controller

{

private readonly ApplicationDbContext \_context;

public ProductoController(ApplicationDbContext context)

{

\_context = context;

}

// Mostrar lista de productos con su categoría

public async Task<IActionResult> Index()

{

var productos = \_context.Productos.Include(p => p.CategoriaProducto);

return View(await productos.ToListAsync());

}

// Mostrar formulario de creación

public IActionResult Create()

{

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

return View();

}

// Crear producto (POST)

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Create([Bind("Id,Nombre,Precio,CategoriaProductoId")] Producto producto)

{

if (ModelState.IsValid)

{

\_context.Add(producto);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

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

return View(producto);

}

// Mostrar formulario de edición

public async Task<IActionResult> Edit(int? id)

{

if (id == null) return NotFound();

var producto = await \_context.Productos.FindAsync(id);

if (producto == null) return NotFound();

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

return View(producto);

}

// Editar producto (POST)

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Edit(int id, [Bind("Id,Nombre,Precio,CategoriaProductoId")] Producto producto)

{

if (id != producto.Id) return NotFound();

if (ModelState.IsValid)

{

try

{

\_context.Update(producto);

await \_context.SaveChangesAsync();

}

catch (DbUpdateConcurrencyException)

{

if (!\_context.Productos.Any(e => e.Id == producto.Id)) return NotFound();

else throw;

}

return RedirectToAction(nameof(Index));

}

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

return View(producto);

}

// Confirmar eliminación

public async Task<IActionResult> Delete(int? id)

{

if (id == null) return NotFound();

var producto = await \_context.Productos.Include(p => p.CategoriaProducto).FirstOrDefaultAsync(m => m.Id == id);

if (producto == null) return NotFound();

return View(producto);

}

// Eliminar producto (POST)

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public async Task<IActionResult> DeleteConfirmed(int id)

{

var producto = await \_context.Productos.FindAsync(id);

\_context.Productos.Remove(producto);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

}

}

**Controlador: CategoriaProducto**

CategoriaProductoController.cs

------------------------------

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using SistemaProductosMVC.Data;

using SistemaProductosMVC.Models;

namespace SistemaProductosMVC.Controllers

{

public class CategoriaProductoController : Controller

{

private readonly ApplicationDbContext \_context;

public CategoriaProductoController(ApplicationDbContext context)

{

\_context = context;

}

// Listado de categorías

public async Task<IActionResult> Index()

{

return View(await \_context.CategoriasProducto.ToListAsync());

}

// Formulario para crear

public IActionResult Create()

{

return View();

}

// Crear categoría (POST)

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Create([Bind("Id,Nombre")] CategoriaProducto categoriaProducto)

{

if (ModelState.IsValid)

{

\_context.Add(categoriaProducto);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

return View(categoriaProducto);

}

// Formulario para editar

public async Task<IActionResult> Edit(int? id)

{

if (id == null) return NotFound();

var categoriaProducto = await \_context.CategoriasProducto.FindAsync(id);

if (categoriaProducto == null) return NotFound();

return View(categoriaProducto);

}

// Editar categoría (POST)

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Edit(int id, [Bind("Id,Nombre")] CategoriaProducto categoriaProducto)

{

if (id != categoriaProducto.Id) return NotFound();

if (ModelState.IsValid)

{

try

{

\_context.Update(categoriaProducto);

await \_context.SaveChangesAsync();

}

catch (DbUpdateConcurrencyException)

{

if (!\_context.CategoriasProducto.Any(e => e.Id == categoriaProducto.Id)) return NotFound();

else throw;

}

return RedirectToAction(nameof(Index));

}

return View(categoriaProducto);

}

// Confirmar eliminación

public async Task<IActionResult> Delete(int? id)

{

if (id == null) return NotFound();

var categoriaProducto = await \_context.CategoriasProducto.FirstOrDefaultAsync(m => m.Id == id);

if (categoriaProducto == null) return NotFound();

return View(categoriaProducto);

}

// Eliminar categoría (POST)

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public async Task<IActionResult> DeleteConfirmed(int id)

{

var categoriaProducto = await \_context.CategoriasProducto.FindAsync(id);

\_context.CategoriasProducto.Remove(categoriaProducto);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

}

}