Notas de .NET 6

[**Crear la BD con EF 2**](https://docs.google.com/document/d/1vBwbqh_Ysb1jUYx28uq75EEWo3yhFdPZzd0eAfAP4Yw/edit#heading=h.kdsiprpbyl38)

[**Ejemplo del DataTable 2**](https://docs.google.com/document/d/1vBwbqh_Ysb1jUYx28uq75EEWo3yhFdPZzd0eAfAP4Yw/edit#heading=h.jjjw3vw0dxiv)

[**Validación de duplicidad de índice 4**](https://docs.google.com/document/d/1vBwbqh_Ysb1jUYx28uq75EEWo3yhFdPZzd0eAfAP4Yw/edit#heading=h.n2xyq8s2ru)

[**Cambios en caliente 4**](https://docs.google.com/document/d/1vBwbqh_Ysb1jUYx28uq75EEWo3yhFdPZzd0eAfAP4Yw/edit#heading=h.uchlkrsn9rld)

[**Relación uno a muchos e índice compuesto 5**](https://docs.google.com/document/d/1vBwbqh_Ysb1jUYx28uq75EEWo3yhFdPZzd0eAfAP4Yw/edit#heading=h.1uyz16hng9j)

[**Configuración del alimentador de la BD 6**](https://docs.google.com/document/d/1vBwbqh_Ysb1jUYx28uq75EEWo3yhFdPZzd0eAfAP4Yw/edit#heading=h.6jz8pzuxuof7)

[**Adición de entidades de usuarios 9**](https://docs.google.com/document/d/1vBwbqh_Ysb1jUYx28uq75EEWo3yhFdPZzd0eAfAP4Yw/edit#heading=h.3hlufs9rr447)

[**Implementando Login/Logout 13**](https://docs.google.com/document/d/1vBwbqh_Ysb1jUYx28uq75EEWo3yhFdPZzd0eAfAP4Yw/edit#heading=h.35nkun2)

[**Páginas de redirección 16**](https://docs.google.com/document/d/1vBwbqh_Ysb1jUYx28uq75EEWo3yhFdPZzd0eAfAP4Yw/edit#heading=h.b66kcggdktem)

[**Combos Helper 17**](https://docs.google.com/document/d/1vBwbqh_Ysb1jUYx28uq75EEWo3yhFdPZzd0eAfAP4Yw/edit#heading=h.lf3enmyi5sqc)

[**Blob Helper 19**](https://docs.google.com/document/d/1vBwbqh_Ysb1jUYx28uq75EEWo3yhFdPZzd0eAfAP4Yw/edit#heading=h.c2somaohphph)

[**Registro de usuarios 21**](https://docs.google.com/document/d/1vBwbqh_Ysb1jUYx28uq75EEWo3yhFdPZzd0eAfAP4Yw/edit#heading=h.8t4scxlseclc)

[**Crear administradores 28**](https://docs.google.com/document/d/1vBwbqh_Ysb1jUYx28uq75EEWo3yhFdPZzd0eAfAP4Yw/edit#heading=h.viy91o6h1kvv)

[**Modificado usuarios 35**](https://docs.google.com/document/d/1vBwbqh_Ysb1jUYx28uq75EEWo3yhFdPZzd0eAfAP4Yw/edit#heading=h.2xcytpi)

[**Cambiando Contraseña 38**](https://docs.google.com/document/d/1vBwbqh_Ysb1jUYx28uq75EEWo3yhFdPZzd0eAfAP4Yw/edit#heading=h.70elta1ilodl)

[**Pequeña corrección al momento de editar usuario 40**](https://docs.google.com/document/d/1vBwbqh_Ysb1jUYx28uq75EEWo3yhFdPZzd0eAfAP4Yw/edit#heading=h.vmynttfl3t7b)

[**Evitar Warnings por nulos 40**](https://docs.google.com/document/d/1vBwbqh_Ysb1jUYx28uq75EEWo3yhFdPZzd0eAfAP4Yw/edit#heading=h.lhv5jdq1xsj2)

[**Bloqueo de usuarios por intentos fallidos 40**](https://docs.google.com/document/d/1vBwbqh_Ysb1jUYx28uq75EEWo3yhFdPZzd0eAfAP4Yw/edit#heading=h.1ci93xb)

[**Confirmar el registro de usuarios 41**](https://docs.google.com/document/d/1vBwbqh_Ysb1jUYx28uq75EEWo3yhFdPZzd0eAfAP4Yw/edit#heading=h.15sdv0fclbbu)

[**Recuperación de contraseña 48**](https://docs.google.com/document/d/1vBwbqh_Ysb1jUYx28uq75EEWo3yhFdPZzd0eAfAP4Yw/edit#heading=h.3whwml4)

[**Adición de íconos y mejorar a la UI 52**](https://docs.google.com/document/d/1vBwbqh_Ysb1jUYx28uq75EEWo3yhFdPZzd0eAfAP4Yw/edit#heading=h.5anyvdrnzegf)

[**Productos 56**](https://docs.google.com/document/d/1vBwbqh_Ysb1jUYx28uq75EEWo3yhFdPZzd0eAfAP4Yw/edit#heading=h.a1055coywfd)

[**ACA VAMOS 64**](https://docs.google.com/document/d/1vBwbqh_Ysb1jUYx28uq75EEWo3yhFdPZzd0eAfAP4Yw/edit#heading=h.adrzab8akyir)

[**Reenvío de email de confirmación 81**](https://docs.google.com/document/d/1vBwbqh_Ysb1jUYx28uq75EEWo3yhFdPZzd0eAfAP4Yw/edit#heading=h.9laz8w9hz3q7)

[**Pantalla Home Básica 82**](https://docs.google.com/document/d/1vBwbqh_Ysb1jUYx28uq75EEWo3yhFdPZzd0eAfAP4Yw/edit#heading=h.746thlrg5lwe)

[**Agregando productos al carro de compras 86**](https://docs.google.com/document/d/1vBwbqh_Ysb1jUYx28uq75EEWo3yhFdPZzd0eAfAP4Yw/edit#heading=h.u2j0k8o5qt1d)

[**Detalle de productos usando un carrusel 89**](https://docs.google.com/document/d/1vBwbqh_Ysb1jUYx28uq75EEWo3yhFdPZzd0eAfAP4Yw/edit#heading=h.5pgcsip66jdd)

[**Mostrando y modificando el carro de compras 93**](https://docs.google.com/document/d/1vBwbqh_Ysb1jUYx28uq75EEWo3yhFdPZzd0eAfAP4Yw/edit#heading=h.4ezruvleg9jo)

[**Procesando el pedido 100**](https://docs.google.com/document/d/1vBwbqh_Ysb1jUYx28uq75EEWo3yhFdPZzd0eAfAP4Yw/edit#heading=h.aysl7yxb4s9h)

[**Administrando los pedidos 104**](https://docs.google.com/document/d/1vBwbqh_Ysb1jUYx28uq75EEWo3yhFdPZzd0eAfAP4Yw/edit#heading=h.z40h3q6sinik)

[**Colocar mensajes tipo Toast y cambiar el estado de los pedidos 110**](https://docs.google.com/document/d/1vBwbqh_Ysb1jUYx28uq75EEWo3yhFdPZzd0eAfAP4Yw/edit#heading=h.tyjcwt)

[**Ver el estado de “Mis” Pedidos 117**](https://docs.google.com/document/d/1vBwbqh_Ysb1jUYx28uq75EEWo3yhFdPZzd0eAfAP4Yw/edit#heading=h.oox34awsqj8z)

[**Filtrar registros por categoría 121**](https://docs.google.com/document/d/1vBwbqh_Ysb1jUYx28uq75EEWo3yhFdPZzd0eAfAP4Yw/edit#heading=h.o3gvtr6113sp)

[**Filtrar registros por nombre 124**](https://docs.google.com/document/d/1vBwbqh_Ysb1jUYx28uq75EEWo3yhFdPZzd0eAfAP4Yw/edit#heading=h.hwn50qmgv0r7)

[**Paginación por base de datos 125**](https://docs.google.com/document/d/1vBwbqh_Ysb1jUYx28uq75EEWo3yhFdPZzd0eAfAP4Yw/edit#heading=h.ios4qb3hf34k)

[**Política de seguridad 125**](https://docs.google.com/document/d/1vBwbqh_Ysb1jUYx28uq75EEWo3yhFdPZzd0eAfAP4Yw/edit#heading=h.92wgnrkz4s9w)

[**Publicación en Azure 125**](https://docs.google.com/document/d/1vBwbqh_Ysb1jUYx28uq75EEWo3yhFdPZzd0eAfAP4Yw/edit#heading=h.ex3aiup4fkkw)

[**Fin 125**](https://docs.google.com/document/d/1vBwbqh_Ysb1jUYx28uq75EEWo3yhFdPZzd0eAfAP4Yw/edit#heading=h.lhsr9wcpc5ld)

Crear la BD con EF

1. Crear la entidad
2. Crear el DbContext

    public class DataContext : DbContext

    {

        public DataContext(DbContextOptions<DataContext> options) : base(options)

        {

        }

        public DbSet<Country> Countries { get; set; }

        protected override void OnModelCreating(ModelBuilder modelBuilder)

        {

            base.OnModelCreating(modelBuilder);

            modelBuilder.Entity<Country>().HasIndex(c => c.Name).IsUnique();

        }

    }

1. Configurar el string de conexión:

  "ConnectionStrings": {

    "DefaultConnection": "Server=(localdb)\\MSSQLLocalDB;Database=Shopping;Trusted\_Connection=True;MultipleActiveResultSets=true"

  }

1. Agregar los paquetes:

Microsoft.EntityFrameworkCore.SqlServer

Microsoft.EntityFrameworkCore.Tools

1. Configurar la inyección del data context:

builder.Services.AddDbContext<DataContext>(o =>

{

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

});

1. Correr los comandos:

add-migration InitialDb

update-database

1. Crear el controlador y adicionar algunos registros.

Ejemplo del DataTable

@model IEnumerable<Shooping.Data.Entities.Country>

@{

    ViewData["Title"] = "Index";

}

<link rel="stylesheet" href="https://cdn.datatables.net/1.10.19/css/jquery.dataTables.min.css" />

<p>

    <a asp-action="Create" class="btn btn-outline-primary">Crear Nuevo</a>

</p>

<div class="row">

    <div class="col-md-12">

        <div class="panel panel-default">

            <div class="panel-heading">

                <h3 class="panel-title">Países</h3>

            </div>

            <div class="panel-body">

                <table class="table table-hover table-responsive table-striped" id="MyTable">

                    <thead>

                        <tr>

                            <th>

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

                            </th>

                            <th>

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

                            </th>

                            <th></th>

                        </tr>

                    </thead>

                    <tbody>

                        @foreach (var item in Model)

                        {

                            <tr>

                                <td>

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

                                </td>

                                <td>

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

                                </td>

                                <td>

                                    <a asp-action="Edit" asp-route-id="@item.Id" class="btn btn-outline-warning">Editar</a>

                                    <a asp-action="Details" asp-route-id="@item.Id" class="btn btn-outline-info">Detalles</a>

                                    <a asp-action="Delete" asp-route-id="@item.Id" class="btn btn-outline-danger">Borrar</a>

                                </td>

                            </tr>

                        }

                    </tbody>

                </table>

            </div>

        </div>

    </div>

</div>

@section Scripts {

    @{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

    <script src="//cdn.datatables.net/1.10.19/js/jquery.dataTables.min.js"></script>

    <script type="text/javascript">

        $(document).ready(function () {

            $('#MyTable').DataTable({

                "language": {

                    "url": "//cdn.datatables.net/plug-ins/9dcbecd42ad/i18n/Spanish.json"

                },

                "aLengthMenu": [

                    [25, 50, 100, 200, -1],

                    [25, 50, 100, 200, "Todos"]

                ]

            });

        });

    </script>

}

Validación de duplicidad de índice

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Create(Country country)

{

    if (ModelState.IsValid)

    {

        \_context.Add(country);

        try

        {

            await \_context.SaveChangesAsync();

            return RedirectToAction(nameof(Index));

        }

        catch (DbUpdateException dbUpdateException)

        {

            if (dbUpdateException.InnerException.Message.Contains("duplicate"))

            {

                ModelState.AddModelError(string.Empty, "Ya existe un país con el mismo nombre.");

            }

            else

            {

                ModelState.AddModelError(string.Empty, dbUpdateException.InnerException.Message);

            }

        }

        catch (Exception exception)

        {

            ModelState.AddModelError(string.Empty, exception.Message);

        }

    }

    return View(country);

}

Cambios en caliente

1. Agregar el paquete: **Microsoft.AspNetCore.Mvc.Razor.RuntimeCompilation**
2. Agregar esta línea en el **Program**: **builder.Services.AddRazorPages().AddRazorRuntimeCompilation();**

Relación uno a muchos e índice compuesto

* Clase **Country**:

using System.ComponentModel.DataAnnotations;

namespace Shooping.Data.Entities

{

    public class Country

    {

        public int Id { get; set; }

        [Display(Name = "País")]

        [MaxLength(50, ErrorMessage = "El campo {0} debe tener máximo {1} caractéres.")]

        [Required(ErrorMessage = "El campo {0} es obligatorio.")]

        public string Name { get; set; }

        public ICollection<State> States { get; set; }

        [Display(Name = "Estados / Departamentos")]

        public int StatesNumber =>  States == null ? 0: States.Count;

    }

}

* Clase **State**:

using System.ComponentModel.DataAnnotations;

namespace Shooping.Data.Entities

{

    public class State

    {

        public int Id { get; set; }

        [Display(Name = "Departamento/Estado")]

        [MaxLength(50, ErrorMessage = "El campo {0} debe tener máximo {1} caractéres.")]

        [Required(ErrorMessage = "El campo {0} es obligatorio.")]

        public string Name { get; set; }

        public Country Country { get; set; }

        public ICollection<City> Cities { get; set; }

        [Display(Name = "Ciudades")]

        public int CitiesNumber => Cities == null ? 0 : Cities.Count;

    }

}

* Clase **City**:

using System.ComponentModel.DataAnnotations;

namespace Shooping.Data.Entities

{

    public class City

    {

        public int Id { get; set; }

        [Display(Name = "Ciudad")]

        [MaxLength(50, ErrorMessage = "El campo {0} debe tener máximo {1} caractéres.")]

        [Required(ErrorMessage = "El campo {0} es obligatorio.")]

        public string Name { get; set; }

        public State State { get; set; }

    }

}

* Modificación al **DataContext**:

public DbSet<Category> Categories { get; set; }

public DbSet<City> Cities { get; set; }

public DbSet<Country> Countries { get; set; }

public DbSet<State> States { get; set; }

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

    base.OnModelCreating(modelBuilder);

    modelBuilder.Entity<Category>().HasIndex(c => c.Name).IsUnique();

    modelBuilder.Entity<City>().HasIndex("Name", "StateId").IsUnique();

    modelBuilder.Entity<Country>().HasIndex(c => c.Name).IsUnique();

    modelBuilder.Entity<State>().HasIndex("Name", "CountryId").IsUnique();

}

Configuración del alimentador de la BD

1. Agregamos la clase **SeedDb** dentro de la carpeta **Data**:

using Shooping.Data.Entities;

namespace Shooping.Data

{

    public class SeedDb

    {

        private readonly DataContext \_context;

        public SeedDb(DataContext context)

        {

            \_context = context;

        }

        public async Task SeedAsync()

        {

            await \_context.Database.EnsureCreatedAsync();

            await CheckCountriesAsync();

            await CheckCategoriesAsync();

        }

        private async Task CheckCategoriesAsync()

        {

            if (!\_context.Categories.Any())

            {

                \_context.Categories.Add(new Category { Name = "Tecnología" });

                \_context.Categories.Add(new Category { Name = "Ropa" });

                \_context.Categories.Add(new Category { Name = "Gamer" });

                \_context.Categories.Add(new Category { Name = "Belleza" });

                \_context.Categories.Add(new Category { Name = "Nutrición" });

            }

            await \_context.SaveChangesAsync();

        }

        private async Task CheckCountriesAsync()

        {

            if (!\_context.Countries.Any())

            {

                \_context.Countries.Add(new Country

                {

                    Name = "Colombia",

                    States = new List<State>()

                    {

                        new State()

                        {

                            Name = "Antioquia",

                            Cities = new List<City>() {

                                new City() { Name = "Medellín" },

                                new City() { Name = "Itagüí" },

                                new City() { Name = "Envigado" },

                                new City() { Name = "Bello" },

                                new City() { Name = "Rionegro" },

                            }

                        },

                        new State()

                        {

                            Name = "Bogotá",

                            Cities = new List<City>() {

                                new City() { Name = "Usaquen" },

                                new City() { Name = "Champinero" },

                                new City() { Name = "Santa fe" },

                                new City() { Name = "Useme" },

                                new City() { Name = "Bosa" },

                            }

                        },

                    }

                    });

                \_context.Countries.Add(new Country

                {

                    Name = "Estados Unidos",

                    States = new List<State>()

                    {

                        new State()

                        {

                            Name = "Florida",

                            Cities = new List<City>() {

                                new City() { Name = "Orlando" },

                                new City() { Name = "Miami" },

                                new City() { Name = "Tampa" },

                                new City() { Name = "Fort Lauderdale" },

                                new City() { Name = "Key West" },

                            }

                        },

                        new State()

                        {

                            Name = "Texas",

                            Cities = new List<City>() {

                                new City() { Name = "Houston" },

                                new City() { Name = "San Antonio" },

                                new City() { Name = "Dallas" },

                                new City() { Name = "Austin" },

                                new City() { Name = "El Paso" },

                            }

                        },

                    }

                });

            }

            await \_context.SaveChangesAsync();

        }

    }

}

1. Modificamos el **Program**:

builder.Services.AddTransient<SeedDb>();

WebApplication? app = builder.Build();

SeedData(app);

void SeedData(WebApplication app)

{

    IServiceScopeFactory? scopedFactory = app.Services.GetService<IServiceScopeFactory>();

    using (IServiceScope? scope = scopedFactory.CreateScope())

    {

        SeedDb? service = scope.ServiceProvider.GetService<SeedDb>();

        service.SeedAsync().Wait();

    }

}

1. Modificamos el **Index** de **Countries** para que muestre los estados.

Adición de entidades de usuarios

1. Como vamos a tener dos tipos de usuarios; administradores y usuarios. Vamos a crear una enumeración para diferenciarlos. Creamos la carpeta **Enums** en el proyecto **Common** y dentro de esta carpeta la enumeración **UserType**:

public enum UserType

{

    Admin,

    User

}

1. En el proyecto **Web** en la carpeta **Data**, crear la carpeta **Entities** y dentro de esta, crear la entidad **User**:

public class User : IdentityUser

{

    [Display(Name = "Documento")]

    [MaxLength(20, ErrorMessage = "El campo {0} debe tener máximo {1} caractéres.")]

    [Required(ErrorMessage = "El campo {0} es obligatorio.")]

    public string Document { get; set; }

    [Display(Name = "Nombres")]

    [MaxLength(50, ErrorMessage = "El campo {0} debe tener máximo {1} caractéres.")]

    [Required(ErrorMessage = "El campo {0} es obligatorio.")]

    public string FirstName { get; set; }

    [Display(Name = "Apellidos")]

    [MaxLength(50, ErrorMessage = "El campo {0} debe tener máximo {1} caractéres.")]

    [Required(ErrorMessage = "El campo {0} es obligatorio.")]

    public string LastName { get; set; }

    [Display(Name = "Dirección")]

    [MaxLength(200, ErrorMessage = "El campo {0} debe tener máximo {1} caractéres.")]

    [Required(ErrorMessage = "El campo {0} es obligatorio.")]

    public string Address { get; set; }

    [Display(Name = "Foto")]

    public Guid ImageId { get; set; }

    //TODO: Pending to put the correct paths

    [Display(Name = "Foto")]

    public string ImageFullPath => ImageId == Guid.Empty

        ? $"https://localhost:7057/images/noimage.png"

        : $"https://shoppingprep.blob.core.windows.net/users/{ImageId}";

    [Display(Name = "Tipo de usuario")]

    public UserType UserType { get; set; }

    [Display(Name = "Ciudad")]

    public City City { get; set; }

    [Display(Name = "Usuario")]

    public string FullName => $"{FirstName} {LastName}";

    [Display(Name = "Usuario")]

    public string FullNameWithDocument => $"{FirstName} {LastName} - {Document}";

}

1. Modificar el **DataContext**:

public class DataContext : IdentityDbContext<User>

1. Crear la interfaz **IUserHelper**:

public interface IUserHelper

{

    Task<User> GetUserAsync(string email);

    Task<IdentityResult> AddUserAsync(User user, string password);

    Task CheckRoleAsync(string roleName);

    Task AddUserToRoleAsync(User user, string roleName);

    Task<bool> IsUserInRoleAsync(User user, string roleName);

}

1. Creamos la implementación de la interfaz **UserHelper**:

public class UserHelper : IUserHelper

{

    private readonly DataContext \_context;

    private readonly UserManager<User> \_userManager;

    private readonly RoleManager<IdentityRole> \_roleManager;

    public UserHelper(DataContext context, UserManager<User> userManager, RoleManager<IdentityRole> roleManager)

    {

        \_context = context;

        \_userManager = userManager;

        \_roleManager = roleManager;

    }

    public async Task<IdentityResult> AddUserAsync(User user, string password)

    {

        return await \_userManager.CreateAsync(user, password);

    }

    public async Task AddUserToRoleAsync(User user, string roleName)

    {

        await \_userManager.AddToRoleAsync(user, roleName);

    }

    public async Task CheckRoleAsync(string roleName)

    {

        bool roleExists = await \_roleManager.RoleExistsAsync(roleName);

        if (!roleExists)

        {

            await \_roleManager.CreateAsync(new IdentityRole

            {

                Name = roleName

            });

        }

    }

    public async Task<User> GetUserAsync(string email)

    {

        return await \_context.Users

            .Include(u => u.City)

            .FirstOrDefaultAsync(u => u.Email == email);

    }

    public async Task<bool> IsUserInRoleAsync(User user, string roleName)

    {

        return await \_userManager.IsInRoleAsync(user, roleName);

    }

}

1. Modificamos el **Program**:

builder.Services.AddDbContext<DataContext>(o =>

{

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

});

builder.Services.AddIdentity<User, IdentityRole>(cfg =>

{

    cfg.User.RequireUniqueEmail = true;

    cfg.Password.RequireDigit = false;

    cfg.Password.RequiredUniqueChars = 0;

    cfg.Password.RequireLowercase = false;

    cfg.Password.RequireNonAlphanumeric = false;

    cfg.Password.RequireUppercase = false;

}).AddEntityFrameworkStores<DataContext>();

builder.Services.AddTransient<SeedDb>();

builder.Services.AddScoped<IUserHelper, UserHelper>();

builder.Services.AddRazorPages().AddRazorRuntimeCompilation();

WebApplication? app = builder.Build();

SeedData(app);

void SeedData(WebApplication app)

{

    IServiceScopeFactory? scopedFactory = app.Services.GetService<IServiceScopeFactory>();

    using (IServiceScope? scope = scopedFactory.CreateScope())

    {

        SeedDb? service = scope.ServiceProvider.GetService<SeedDb>();

        service.SeedAsync().Wait();

    }

}

if (!app.Environment.IsDevelopment())

{

    app.UseExceptionHandler("/Home/Error");

    app.UseHsts();

}

app.UseHttpsRedirection();

app.UseStaticFiles();

app.UseRouting();

app.UseAuthentication();

app.UseAuthorization();

1. Modificamos el **SeedDb**:

public async Task SeedAsync()

{

    await \_context.Database.EnsureCreatedAsync();

    await CheckCountriesAsync();

    await CheckCategoriesAsync();

    await CheckRolesAsync();

    await CheckUserAsync("1010", "Juan", "Zuluaga", "zulu@yopmail.com", "322 311 4620", "Calle Luna Calle Sol", UserType.Admin);

}

private async Task<User> CheckUserAsync(

    string document,

    string firstName,

    string lastName,

    string email,

    string phone,

    string address,

    UserType userType)

{

    User user = await \_userHelper.GetUserAsync(email);

    if (user == null)

    {

        user = new User

        {

            FirstName = firstName,

            LastName = lastName,

            Email = email,

            UserName = email,

            PhoneNumber = phone,

            Address = address,

            Document = document,

            City = \_context.Cities.FirstOrDefault(),

            UserType = userType,

        };

        await \_userHelper.AddUserAsync(user, "123456");

        await \_userHelper.AddUserToRoleAsync(user, userType.ToString());

    }

    return user;

}

private async Task CheckRolesAsync()

{

    await \_userHelper.CheckRoleAsync(UserType.Admin.ToString());

    await \_userHelper.CheckRoleAsync(UserType.User.ToString());

}

1. Corremos los siguientes comandos:

PM> drop-database

PM> add-migration Users

PM> update-database

Implementando Login/Logout

1. Creamos la **LoginViewModel**:

using System.ComponentModel.DataAnnotations;

namespace Shooping.Models

{

    public class LoginViewModel

    {

        [Display(Name = "Email")]

        [Required(ErrorMessage = "El campo {0} es obligatorio.")]

        [EmailAddress(ErrorMessage = "Debes ingresar un correo válido.")]

        public string Username { get; set; }

        [Display(Name = "Contraseña")]

        [Required(ErrorMessage = "El campo {0} es obligatorio.")]

        [MinLength(6, ErrorMessage = "El campo {0} debe tener al menos {1} carácteres.")]

        public string Password { get; set; }

        [Display(Name = "Recordarme en este navegador")]

        public bool RememberMe { get; set; }

    }

}

1. Adicionamos estos métodos a la **IUserHelper**:

Task<SignInResult> LoginAsync(LoginViewModel model);

Task LogoutAsync();

1. Y agregamos su implementación en el **UserHelper:**

…

private readonly DataContext \_context;

private readonly UserManager<User> \_userManager;

private readonly RoleManager<IdentityRole> \_roleManager;

private readonly SignInManager<User> \_signInManager;

public UserHelper(DataContext context, UserManager<User> userManager, RoleManager<IdentityRole> roleManager, SignInManager<User> signInManager)

{

    \_context = context;

    \_userManager = userManager;

    \_roleManager = roleManager;

    \_signInManager = signInManager;

}

public async Task<SignInResult> LoginAsync(LoginViewModel model)

{

    return await \_signInManager.PasswordSignInAsync(

        model.Username,

        model.Password,

        model.RememberMe,

        false);

}

public async Task LogoutAsync()

{

    await \_signInManager.SignOutAsync();

}

…

1. Creamos el **AccountController**:

public class AccountController : Controller

{

    private readonly IUserHelper \_userHelper;

    public AccountController(IUserHelper userHelper)

    {

        \_userHelper = userHelper;

    }

    public IActionResult Login()

    {

        if (User.Identity.IsAuthenticated)

        {

            return RedirectToAction("Index", "Home");

        }

        return View(new LoginViewModel());

    }

    [HttpPost]

    public async Task<IActionResult> Login(LoginViewModel model)

    {

        if (ModelState.IsValid)

        {

            Microsoft.AspNetCore.Identity.SignInResult result = await \_userHelper.LoginAsync(model);

            if (result.Succeeded)

            {

                return RedirectToAction("Index", "Home");

            }

            ModelState.AddModelError(string.Empty, "Email o contraseña incorrectos.");

        }

        return View(model);

    }

    public async Task<IActionResult> Logout()

    {

        await \_userHelper.LogoutAsync();

        return RedirectToAction("Index", "Home");

    }

}

1. Adicionamos la vista **Login**:

@model Shooping.Models.LoginViewModel

@{

    ViewData["Title"] = "Login";

}

<div class="row">

    <div class="col-md-4">

    </div>

    <div class="col-md-4">

        <h3>Iniciar Sesión</h3>

        <form asp-action="Login">

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

            <div class="form-group">

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

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

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

            </div>

            <div class="form-group">

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

                <input asp-for="Password" type="password" class="form-control" />

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

            </div>

            <div class="form-group mt-2">

                <div class="form-check">

                    <input asp-for="RememberMe" type="checkbox" class="form-check-input" />

                    <label asp-for="RememberMe" class="form-check-label"></label>

                </div>

                <span asp-validation-for="RememberMe" class="text-warning"></span>

            </div>

            <div class="form-group mt-2">

                <input type="submit" value="Iniciar Sesión" class="btn btn-outline-primary" />

                <a asp-action="Register" class="btn btn-outline-secondary">Registrar Nuevo Usuario</a>

            </div>

        </form>

    </div>

    <div class="col-md-4">

    </div>

</div>

@section Scripts {

    @{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

}

1. Adicionamos la anotación authorize a los controladores previos:

[Authorize(Roles = "Admin")]

1. Modificamos nuestro menú **\_Layout**:

…

<div class="navbar-collapse collapse d-sm-inline-flex justify-content-between">

    <ul class="navbar-nav flex-grow-1">

        <li class="nav-item">

            <a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="Index">Inicio</a>

        </li>

        <li class="nav-item">

            <a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="Privacy">Políticas</a>

        </li>

        @if (User.Identity.IsAuthenticated && User.IsInRole("Admin"))

        {

            <li class="nav-item">

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

            </li>

            <li class="nav-item">

                <a class="nav-link text-dark" asp-area="" asp-controller="Countries" asp-action="Index">Países</a>

            </li>

            <li class="nav-item">

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

            </li>

        }

    </ul>

    <ul class="nav navbar-nav navbar-right">

        @if (User.Identity.IsAuthenticated)

        {

            <li class="nav-item">

                <a class="nav-link text-dark" asp-area="" asp-controller="Account" asp-action="ChangeUser">@User.Identity.Name</a>

            </li>

            <li class="nav-item">

                <a class="nav-link text-dark" asp-area="" asp-controller="Account" asp-action="Logout">Cerrar Sesión</a>

            </li>

        }

        else

        {

            <li class="nav-item">

                <a class="nav-link text-dark" asp-area="" asp-controller="Account" asp-action="Login">Iniciar Sesión</a>

            </li>

        }

    </ul>

</div>

…

1. Probamos.

Páginas de redirección

1. Adicionamos una imagen que usaremos para colocar en la página de no encontrada (gopher\_head-min.png).

1. Creamos el método **NotAuthorized** en el controlador **AccountController**:

public IActionResult NotAuthorized()

{

    return View();

}

1. Luego creamos la vista:

@{

ViewData["Title"] = "NotAuthorized";

}

<br />

<br />

<img src="~/images/gopher\_head-min.png" />

<h2>No estas autorizado para ejecutar esta acción!</h2>

1. Modificamos el **Program**:

builder.Services.ConfigureApplicationCookie(options =>

{

    options.LoginPath = "/Account/NotAuthorized";

    options.AccessDeniedPath = "/Account/NotAuthorized";

});

…

app.UseStatusCodePagesWithReExecute("/error/{0}");

1. Agregamos este método al **HomeController**:

[Route("error/404")]

public IActionResult Error404()

{

    return View();

}

1. Luego agregamos la vista:

@{

ViewData["Title"] = "Error404";

}

<br />

<br />

<img src="~/images/gopher\_head-min.png" />

<h2>Error, página no encontrada</h2>

Combos Helper

1. Creamos la interfaz:

using Microsoft.AspNetCore.Mvc.Rendering;

namespace Shooping.Helpers

{

    public interface ICombosHelper

    {

        Task<IEnumerable<SelectListItem>> GetComboCategoriesAsync();

        Task<IEnumerable<SelectListItem>> GetComboCountriesAsync();

        Task<IEnumerable<SelectListItem>> GetComboStatesAsync(int countryId);

        Task<IEnumerable<SelectListItem>> GetComboCitiesAsync(int stateId);

    }

}

1. Creamos la implementation:

using Microsoft.AspNetCore.Mvc.Rendering;

using Microsoft.EntityFrameworkCore;

using Shooping.Data;

namespace Shooping.Helpers

{

    public class CombosHelper : ICombosHelper

    {

        private readonly DataContext \_context;

        public CombosHelper(DataContext context)

        {

            \_context = context;

        }

        public async Task<IEnumerable<SelectListItem>> GetComboCategoriesAsync()

        {

            List<SelectListItem> list = await \_context.Categories.Select(x => new SelectListItem

            {

                Text = x.Name,

                Value = $"{x.Id}"

            })

                .OrderBy(x => x.Text)

                .ToListAsync();

            list.Insert(0, new SelectListItem

            {

                Text = "[Seleccione una categoría...]",

                Value = "0"

            });

            return list;

        }

        public async Task<IEnumerable<SelectListItem>> GetComboCitiesAsync(int stateId)

        {

            List<SelectListItem> list = await \_context.Cities

                .Where(x => x.State.Id == stateId)

                .Select(x => new SelectListItem

                {

                    Text = x.Name,

                    Value = $"{x.Id}"

                })

                .OrderBy(x => x.Text)

                .ToListAsync();

            list.Insert(0, new SelectListItem

            {

                Text = "[Seleccione una ciudad...]",

                Value = "0"

            });

            return list;

        }

        public async Task<IEnumerable<SelectListItem>> GetComboCountriesAsync()

        {

            List<SelectListItem> list = await \_context.Countries.Select(x => new SelectListItem

            {

                Text = x.Name,

                Value = $"{x.Id}"

            })

                .OrderBy(x => x.Text)

                .ToListAsync();

            list.Insert(0, new SelectListItem

            {

                Text = "[Seleccione un país...]",

                Value = "0"

            });

            return list;

        }

        public async Task<IEnumerable<SelectListItem>> GetComboStatesAsync(int countryId)

        {

            List<SelectListItem> list = await \_context.States

                .Where(x => x.Country.Id == countryId)

                .Select(x => new SelectListItem

                {

                    Text = x.Name,

                    Value = $"{x.Id}"

                })

                .OrderBy(x => x.Text)

                .ToListAsync();

            list.Insert(0, new SelectListItem

            {

                Text = "[Seleccione un departamento/estado...]",

                Value = "0"

            });

            return list;

        }

    }

}

1. Configuramos la inyección:

builder.Services.AddScoped<ICombosHelper, CombosHelper>();

Blob Helper

1. Creamos el blob en azure y agregamos valores al **appsettings**:

"Blob": {

  "ConnectionString": "DefaultEndpointsProtocol=https;AccountName=shoppingprep;AccountKey=9azHu2kSy5Lq199tvX9fOsdtacLhucwHYAt+xj+qKXIvzHNzfdV5e4IrJzRcnymnh2CTv8Xtl7w+VBc1PW72ng==;EndpointSuffix=core.windows.net"

}

1. Creamos la interfaz:

namespace Shooping.Helpers

{

    public interface IBlobHelper

    {

        Task<Guid> UploadBlobAsync(IFormFile file, string containerName);

        Task<Guid> UploadBlobAsync(byte[] file, string containerName);

        Task<Guid> UploadBlobAsync(string image, string containerName);

        Task DeleteBlobAsync(Guid id, string containerName);

    }

}

1. Creamos la implementation:

using Microsoft.WindowsAzure.Storage;

using Microsoft.WindowsAzure.Storage.Blob;

namespace Shooping.Helpers

{

    public class BlobHelper : IBlobHelper

    {

        private readonly CloudBlobClient \_blobClient;

        public BlobHelper(IConfiguration configuration)

        {

            string keys = configuration["Blob:ConnectionString"];

            CloudStorageAccount storageAccount = CloudStorageAccount.Parse(keys);

            \_blobClient = storageAccount.CreateCloudBlobClient();

        }

        public async Task DeleteBlobAsync(Guid id, string containerName)

        {

            CloudBlobContainer container = \_blobClient.GetContainerReference(containerName);

            CloudBlockBlob blockBlob = container.GetBlockBlobReference($"{id}");

            await blockBlob.DeleteAsync();

        }

        public async Task<Guid> UploadBlobAsync(IFormFile file, string containerName)

        {

            Stream stream = file.OpenReadStream();

            return await UploadBlobAsync(stream, containerName);

        }

        public async Task<Guid> UploadBlobAsync(byte[] file, string containerName)

        {

            MemoryStream stream = new MemoryStream(file);

            return await UploadBlobAsync(stream, containerName);

        }

        public async Task<Guid> UploadBlobAsync(string image, string containerName)

        {

            Stream stream = File.OpenRead(image);

            return await UploadBlobAsync(stream, containerName);

        }

        private async Task<Guid> UploadBlobAsync(Stream stream, string containerName)

        {

            Guid name = Guid.NewGuid();

            CloudBlobContainer container = \_blobClient.GetContainerReference(containerName);

            CloudBlockBlob blockBlob = container.GetBlockBlobReference($"{name}");

            await blockBlob.UploadFromStreamAsync(stream);

            return name;

        }

    }

}

1. Configuramos la inyección:

builder.Services.AddScoped<IBlobHelper, BlobHelper>();

Registro de usuarios

1. Adicionamos el **EditUserViewModel**:

public class EditUserViewModel

{

    public string Id { get; set; }

    [Display(Name = "Documento")]

    [MaxLength(20, ErrorMessage = "El campo {0} debe tener máximo {1} caractéres.")]

    [Required(ErrorMessage = "El campo {0} es obligatorio.")]

    public string Document { get; set; }

    [Display(Name = "Nombres")]

    [MaxLength(50, ErrorMessage = "El campo {0} debe tener máximo {1} caractéres.")]

    [Required(ErrorMessage = "El campo {0} es obligatorio.")]

    public string FirstName { get; set; }

    [Display(Name = "Apellidos")]

    [MaxLength(50, ErrorMessage = "El campo {0} debe tener máximo {1} caractéres.")]

    [Required(ErrorMessage = "El campo {0} es obligatorio.")]

    public string LastName { get; set; }

    [Display(Name = "Dirección")]

    [MaxLength(200, ErrorMessage = "El campo {0} debe tener máximo {1} caractéres.")]

    [Required(ErrorMessage = "El campo {0} es obligatorio.")]

    public string Address { get; set; }

    [Display(Name = "Teléfono")]

    [MaxLength(20, ErrorMessage = "El campo {0} debe tener máximo {1} caractéres.")]

    [Required(ErrorMessage = "El campo {0} es obligatorio.")]

    public string PhoneNumber { get; set; }

    [Display(Name = "Foto")]

    public Guid ImageId { get; set; }

    //TODO: Pending to put the correct paths

    [Display(Name = "Foto")]

    public string ImageFullPath => ImageId == Guid.Empty

        ? $"https://localhost:7057/images/noimage.png"

        : $"https://shoppingprep.blob.core.windows.net/users/{ImageId}";

    [Display(Name = "Image")]

    public IFormFile ImageFile { get; set; }

    [Display(Name = "País")]

    [Range(1, int.MaxValue, ErrorMessage = "Debes de seleccionar un país.")]

    [Required(ErrorMessage = "El campo {0} es obligatorio.")]

    public int CountryId { get; set; }

    public IEnumerable<SelectListItem> Countries { get; set; }

    [Display(Name = "Departmento/Estado")]

    [Range(1, int.MaxValue, ErrorMessage = "Debes de seleccionar un departamento/estado.")]

    [Required(ErrorMessage = "El campo {0} es obligatorio.")]

    public int StateId { get; set; }

    public IEnumerable<SelectListItem> States { get; set; }

    [Display(Name = "Ciuadad")]

    [Range(1, int.MaxValue, ErrorMessage = "Debes de seleccionar una ciudad.")]

    public int CityId { get; set; }

    public IEnumerable<SelectListItem> Cities { get; set; }

}

1. Adicionamos el **AddUserViewModel**:

public class AddUserViewModel : EditUserViewModel

{

    [Display(Name = "Email")]

    [EmailAddress(ErrorMessage = "Debes ingresar un correo válido.")]

    [MaxLength(100, ErrorMessage = "El campo {0} debe tener máximo {1} caractéres.")]

    [Required(ErrorMessage = "El campo {0} es obligatorio.")]

    public string Username { get; set; }

    [DataType(DataType.Password)]

    [Display(Name = "Contraseña")]

    [Required(ErrorMessage = "El campo {0} es obligatorio.")]

    [StringLength(20, MinimumLength = 6, ErrorMessage = "El campo {0} debe tener entre {2} y {1} carácteres.")]

    public string Password { get; set; }

    [Compare("Password", ErrorMessage = "La contraseña y la confirmación no son iguales.")]

    [Display(Name = "Confirmación de contraseña")]

    [DataType(DataType.Password)]

    [Required(ErrorMessage = "El campo {0} es obligatorio.")]

    [StringLength(20, MinimumLength = 6, ErrorMessage = "El campo {0} debe tener entre {2} y {1} carácteres.")]

    public string PasswordConfirm { get; set; }

    [Display(Name = "Tipo de usuario")]

    public UserType UserType { get; set; }

}

1. Adicionamos este método al **IUserHelper**:

Task<User> AddUserAsync(AddUserViewModel model);

1. Add this method to **UserHelper**:

public async Task<User> AddUserAsync(AddUserViewModel model, Guid imageId)

{

    User user = new User

    {

        Address = model.Address,

        Document = model.Document,

        Email = model.Username,

        FirstName = model.FirstName,

        LastName = model.LastName,

        ImageId = imageId,

        PhoneNumber = model.PhoneNumber,

        City = await \_context.Cities.FindAsync(model.CityId),

        UserName = model.Username,

        UserType = model.UserType

    };

    IdentityResult result = await \_userManager.CreateAsync(user, model.Password);

    if (result != IdentityResult.Success)

    {

        return null;

    }

    User newUser = await GetUserAsync(model.Username);

    await AddUserToRoleAsync(newUser, user.UserType.ToString());

    return newUser;

}

1. Modificamos el **AccountController**:

public class AccountController : Controller

{

    private readonly IUserHelper \_userHelper;

    private readonly DataContext \_context;

    private readonly ICombosHelper \_combosHelper;

    private readonly IBlobHelper \_blobHelper;

    public AccountController(IUserHelper userHelper, DataContext context, ICombosHelper combosHelper, IBlobHelper blobHelper)

    {

        \_userHelper = userHelper;

        \_context = context;

        \_combosHelper = combosHelper;

        \_blobHelper = blobHelper;

    }

public async Task<IActionResult> Register()

{

    AddUserViewModel model = new AddUserViewModel

    {

        Id = Guid.Empty.ToString(),

        Countries = await \_combosHelper.GetComboCountriesAsync(),

        States = await \_combosHelper.GetComboStatesAsync(0),

        Cities = await \_combosHelper.GetComboCitiesAsync(0),

        UserType = UserType.User,

    };

    return View(model);

}

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Register(AddUserViewModel model)

{

    if (ModelState.IsValid)

    {

        Guid imageId = Guid.Empty;

        if (model.ImageFile != null)

        {

            imageId = await \_blobHelper.UploadBlobAsync(model.ImageFile, "users");

        }

        User user = await \_userHelper.AddUserAsync(model, imageId);

        if (user == null)

        {

            ModelState.AddModelError(string.Empty, "Este correo ya está siendo usado.");

            return View(model);

        }

        LoginViewModel loginViewModel = new LoginViewModel

        {

            Password = model.Password,

            RememberMe = false,

            Username = model.Username

        };

        var result2 = await \_userHelper.LoginAsync(loginViewModel);

        if (result2.Succeeded)

        {

            return RedirectToAction("Index", "Home");

        }

    }

    return View(model);

}

public JsonResult GetStates(int countryId)

{

    Country country = \_context.Countries

        .Include(c => c.States)

        .FirstOrDefault(c => c.Id == countryId);

    if (country == null)

    {

        return null;

    }

    return Json(country.States.OrderBy(d => d.Name));

}

public JsonResult GetCities(int stateId)

{

    State state = \_context.States

        .Include(s => s.Cities)

        .FirstOrDefault(s => s.Id == stateId);

    if (state == null)

    {

        return null;

    }

    return Json(state.Cities.OrderBy(c => c.Name));

}

1. Adicionamos la vista parcial **\_User** en **\_Shared**:

@model Shooping.Models.EditUserViewModel

<div class="row">

    <div class="col-md-6">

        <div class="form-group">

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

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

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

        </div>

        <div class="form-group">

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

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

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

        </div>

        <div class="form-group">

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

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

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

        </div>

        <div class="form-group">

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

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

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

        </div>

        <div class="form-group">

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

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

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

        </div>

    </div>

    <div class="col-md-6">

        <div class="form-group">

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

            <input asp-for="ImageFile" class="form-control" type="file" />

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

        </div>

        <div class="form-group">

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

            <select asp-for="CountryId" asp-items="Model.Countries" class="form-control"></select>

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

        </div>

        <div class="form-group">

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

            <select asp-for="StateId" asp-items="Model.States" class="form-control"></select>

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

        </div>

        <div class="form-group">

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

            <select asp-for="CityId" asp-items="Model.Cities" class="form-control"></select>

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

        </div>

    </div>

</div>

1. Add the view **Register** on **AccountController**:

@model Shooping.Models.AddUserViewModel

@{

    ViewData["Title"] = "Register";

}

<h2>Registrar</h2>

<h4>Usuario</h4>

<hr />

<div class="row">

    <div class="col-md-12">

        <form asp-action="Register" enctype="multipart/form-data">

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

            <input type="hidden" asp-for="Id" />

            <input type="hidden" asp-for="UserType" />

            <input type="hidden" asp-for="Countries" />

            <input type="hidden" asp-for="States" />

            <input type="hidden" asp-for="Cities" />

            <div class="form-group">

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

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

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

            </div>

            <div class="row">

                <div class="col-md-6">

                    <div class="form-group">

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

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

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

                    </div>

                </div>

                <div class="col-md-6">

                    <div class="form-group">

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

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

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

                    </div>

                </div>

            </div>

            <partial name="\_User" />

            <div class="form-group mt-2">

                <input type="submit" value="Registrar" class="btn btn-outline-primary" />

            </div>

        </form>

    </div>

</div>

@section Scripts {

    @{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

    <script type="text/javascript">

        $(document).ready(function () {

            $("#CountryId").change(function () {

                debugger;

                $("#StateId").empty();

                $("#StateId").append('<option value="0">[Selecciona un departamento/estado...]</option>');

                $("#CityId").empty();

                $("#CityId").append('<option value="0">[Selecciona una ciudad...]</option>');

                $.ajax({

                    type: 'POST',

                    url: '@Url.Action("GetStates")',

                    dataType: 'json',

                    data: { countryId: $("#CountryId").val() },

                    success: function (states) {

                        $.each(states, function (i, state) {

                            $("#StateId").append('<option value="'

                                + state.id + '">'

                                + state.name + '</option>');

                        });

                    },

                    error: function (ex) {

                        alert('Failed to retrieve states.' + ex);

                    }

                });

                return false;

            })

            $("#StateId").change(function () {

                $("#CityId").empty();

                $("#CityId").append('<option value="0">[Selecciona una ciudad...]</option>');

                $.ajax({

                    type: 'POST',

                    url: '@Url.Action("GetCities")',

                    dataType: 'json',

                    data: { stateId: $("#StateId").val() },

                    success: function (cities) {

                        $.each(cities, function (i, city) {

                            debugger;

                            $("#CityId").append('<option value="'

                                + city.id + '">'

                                + city.name + '</option>');

                        });

                    },

                    error: function (ex) {

                        alert('Failed to retrieve cities.' + ex);

                    }

                });

                return false;

            })

        });

    </script>

}

1. Colocamos el **JsonIgnore** para que serialice bien los objetos de estado y ciudad:

[JsonIgnore]

public Country Country { get; set; }

…

[JsonIgnore]

public State State { get; set; }

Crear administradores

1. Crear el **UserController**:

[Authorize(Roles = "Admin")]

public class UsersController : Controller

{

    private readonly IUserHelper \_userHelper;

    private readonly DataContext \_context;

    private readonly ICombosHelper \_combosHelper;

    private readonly IBlobHelper \_blobHelper;

    public UsersController(IUserHelper userHelper, DataContext context, ICombosHelper combosHelper, IBlobHelper blobHelper)

    {

        \_userHelper = userHelper;

        \_context = context;

        \_combosHelper = combosHelper;

        \_blobHelper = blobHelper;

    }

    public async Task<IActionResult> Index()

    {

        return View(await \_context.Users

            .Include(u => u.City)

            .ThenInclude(c => c.State)

            .ThenInclude(s => s.Country)

            .ToListAsync());

    }

    public async Task<IActionResult> Create()

    {

        AddUserViewModel model = new AddUserViewModel

        {

            Id = Guid.Empty.ToString(),

            Countries = await \_combosHelper.GetComboCountriesAsync(),

            States = await \_combosHelper.GetComboStatesAsync(0),

            Cities = await \_combosHelper.GetComboCitiesAsync(0),

            UserType = UserType.Admin,

        };

        return View(model);

    }

    [HttpPost]

    [ValidateAntiForgeryToken]

    public async Task<IActionResult> Create(AddUserViewModel model)

    {

        if (ModelState.IsValid)

        {

            Guid imageId = Guid.Empty;

            if (model.ImageFile != null)

            {

                imageId = await \_blobHelper.UploadBlobAsync(model.ImageFile, "users");

            }

            User user = await \_userHelper.AddUserAsync(model, imageId);

            if (user == null)

            {

                ModelState.AddModelError(string.Empty, "Este correo ya está siendo usado.");

                return View(model);

            }

            return RedirectToAction(nameof(Index));

        }

        return View(model);

    }

    public JsonResult? GetStates(int countryId)

    {

        Country? country = \_context.Countries

            .Include(c => c.States)

            .FirstOrDefault(c => c.Id == countryId);

        if (country == null)

        {

            return null;

        }

        return Json(country.States.OrderBy(d => d.Name));

    }

    public JsonResult? GetCities(int stateId)

    {

        State? state = \_context.States

            .Include(s => s.Cities)

            .FirstOrDefault(s => s.Id == stateId);

        if (state == null)

        {

            return null;

        }

        return Json(state.Cities.OrderBy(c => c.Name));

    }

    public IActionResult Login()

    {

        if (User.Identity.IsAuthenticated)

        {

            return RedirectToAction("Index", "Home");

        }

        return View(new LoginViewModel());

    }

}

1. Crear la vista **Index**:

@model IEnumerable<Shooping.Data.Entities.User>

@{

    ViewData["Title"] = "Index";

}

<link rel="stylesheet" href="https://cdn.datatables.net/1.10.19/css/jquery.dataTables.min.css" />

<p>

    <a asp-action="Create" class="btn btn-outline-primary">Nuevo Administrador</a>

</p>

<div class="row">

    <div class="col-md-12">

        <div class="panel panel-default">

            <div class="panel-heading">

                <h3 class="panel-title">Usuarios</h3>

            </div>

            <div class="panel-body">

                <table class="table table-hover table-responsive table-striped" id="MyTable">

                    <thead>

                        <tr>

                            <th>

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

                            </th>

                            <th>

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

                            </th>

                            <th>

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

                            </th>

                            <th>

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

                            </th>

                            <th>

                                @Html.DisplayNameFor(model => model.City.State.Country.Name)

                            </th>

                            <th>

                                @Html.DisplayNameFor(model => model.City.State.Name)

                            </th>

                            <th>

                                @Html.DisplayNameFor(model => model.City.Name)

                            </th>

                            <th>

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

                            </th>

                            <th>

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

                            </th>

                            <th>

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

                            </th>

                            <th>

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

                            </th>

                        </tr>

                    </thead>

                    <tbody>

                        @foreach (var item in Model)

                        {

                            <tr>

                                <td>

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

                                </td>

                                <td>

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

                                </td>

                                <td>

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

                                </td>

                                <td>

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

                                </td>

                                <td>

                                    @Html.DisplayFor(modelItem => item.City.State.Country.Name)

                                </td>

                                <td>

                                    @Html.DisplayFor(modelItem => item.City.State.Name)

                                </td>

                                <td>

                                    @Html.DisplayFor(modelItem => item.City.Name)

                                </td>

                                <td>

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

                                </td>

                                <td>

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

                                </td>

                                <td>

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

                                </td>

                                <td>

                                    <img src="@item.ImageFullPath" style="width:100px; height:100px; border-radius:50px" />

                                </td>

                            </tr>

                        }

                    </tbody>

                </table>

            </div>

        </div>

    </div>

</div>

@section Scripts {

    @{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

    <script src="//cdn.datatables.net/1.10.19/js/jquery.dataTables.min.js"></script>

    <script type="text/javascript">

        $(document).ready(function () {

            $('#MyTable').DataTable({

                "language": {

                    "url": "//cdn.datatables.net/plug-ins/9dcbecd42ad/i18n/Spanish.json"

                },

                "aLengthMenu": [

                    [25, 50, 100, 200, -1],

                    [25, 50, 100, 200, "Todos"]

                ]

            });

        });

    </script>

}

1. Crear la vista **Create**:

@model Shooping.Models.AddUserViewModel

@{

    ViewData["Title"] = "Register";

}

<h2>Crear</h2>

<h4>Administrador</h4>

<hr />

<div class="row">

    <div class="col-md-12">

        <form asp-action="Create" enctype="multipart/form-data">

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

            <input type="hidden" asp-for="Id" />

            <input type="hidden" asp-for="UserType" />

            <input type="hidden" asp-for="Countries" />

            <input type="hidden" asp-for="States" />

            <input type="hidden" asp-for="Cities" />

            <div class="form-group">

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

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

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

            </div>

            <div class="row">

                <div class="col-md-6">

                    <div class="form-group">

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

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

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

                    </div>

                </div>

                <div class="col-md-6">

                    <div class="form-group">

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

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

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

                    </div>

                </div>

            </div>

            <partial name="\_User" />

            <div class="form-group mt-2">

                <input type="submit" value="Crear" class="btn btn-outline-primary" />

            </div>

        </form>

    </div>

</div>

@section Scripts {

    @{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

    <script type="text/javascript">

        $(document).ready(function () {

            $("#CountryId").change(function () {

                debugger;

                $("#StateId").empty();

                $("#StateId").append('<option value="0">[Selecciona un Departamento / Estado...]</option>');

                $("#CityId").empty();

                $("#CityId").append('<option value="0">[Selecciona una ciudad...]</option>');

                $.ajax({

                    type: 'POST',

                    url: '@Url.Action("GetStates")',

                    dataType: 'json',

                    data: { countryId: $("#CountryId").val() },

                    success: function (states) {

                        $.each(states, function (i, state) {

                            $("#StateId").append('<option value="'

                                + state.id + '">'

                                + state.name + '</option>');

                        });

                    },

                    error: function (ex) {

                        alert('Failed to retrieve states.' + ex);

                    }

                });

                return false;

            })

            $("#StateId").change(function () {

                $("#CityId").empty();

                $("#CityId").append('<option value="0">[Selecciona una ciudad...]</option>');

                $.ajax({

                    type: 'POST',

                    url: '@Url.Action("GetCities")',

                    dataType: 'json',

                    data: { stateId: $("#StateId").val() },

                    success: function (cities) {

                        $.each(cities, function (i, city) {

                            debugger;

                            $("#CityId").append('<option value="'

                                + city.id + '">'

                                + city.name + '</option>');

                        });

                    },

                    error: function (ex) {

                        alert('Failed to retrieve cities.' + ex);

                    }

                });

                return false;

            })

        });

    </script>

}

1. Llamar la nueva opción en el menú:

<li class="nav-item">

    <a class="nav-link text-dark" asp-area="" asp-controller="Users" asp-action="Index">Usuarios</a>

</li>

Modificado usuarios

1. Creamos el **ChangePasswordViewModel**:

public class ChangePasswordViewModel

{

    [DataType(DataType.Password)]

    [Display(Name = "Contraseña actual")]

    [StringLength(20, MinimumLength = 6, ErrorMessage = "El campo {0} debe tener entre {2} y {1} carácteres.")]

    [Required(ErrorMessage = "El campo {0} es obligatorio.")]

    public string OldPassword { get; set; }

    [DataType(DataType.Password)]

    [Display(Name = "Nueva contraseña")]

    [StringLength(20, MinimumLength = 6, ErrorMessage = "El campo {0} debe tener entre {2} y {1} carácteres.")]

    [Required(ErrorMessage = "El campo {0} es obligatorio.")]

    public string NewPassword { get; set; }

    [Compare("NewPassword", ErrorMessage = "La nueva contraseña y la confirmación no son iguales.")]

    [DataType(DataType.Password)]

    [Display(Name = "Confirmación nueva contraseña")]

    [StringLength(20, MinimumLength = 6, ErrorMessage = "El campo {0} debe tener entre {2} y {1} carácteres.")]

    [Required(ErrorMessage = "El campo {0} es obligatorio.")]

    public string Confirm { get; set; }

}

1. Adicionamos estos métodos al **IUserHelper**:

Task<IdentityResult> ChangePasswordAsync(User user, string oldPassword, string newPassword);

Task<IdentityResult> UpdateUserAsync(User user);

Task<User> GetUserAsync(Guid userId);

1. Adicionamos la implementación en **UserHelper**:

public async Task<IdentityResult> ChangePasswordAsync(User user, string oldPassword, string newPassword)

{

    return await \_userManager.ChangePasswordAsync(user, oldPassword, newPassword);

}

public async Task<IdentityResult> UpdateUserAsync(User user)

{

    return await \_userManager.UpdateAsync(user);

}

public async Task<User> GetUserAsync(Guid userId)

{

    return await \_context.Users

        .Include(u => u.City)

        .ThenInclude(c => c.State)

        .ThenInclude(s => s.Country)

        .FirstOrDefaultAsync(u => u.Id == userId.ToString());

}

1. Adicione estos métods al **AccountController**:

public async Task<IActionResult> ChangeUser()

{

    User user = await \_userHelper.GetUserAsync(User.Identity.Name);

    if (user == null)

    {

        return NotFound();

    }

    EditUserViewModel model = new()

    {

        Address = user.Address,

        FirstName = user.FirstName,

        LastName = user.LastName,

        PhoneNumber = user.PhoneNumber,

        ImageId = user.ImageId,

        Cities = await \_combosHelper.GetComboCitiesAsync(user.City.State.Id),

        CityId = user.City.Id,

        Countries = await \_combosHelper.GetComboCountriesAsync(),

        CountryId = user.City.State.Country.Id,

        StateId = user.City.State.Id,

        States = await \_combosHelper.GetComboStatesAsync(user.City.State.Country.Id),

        Id = user.Id,

        Document = user.Document

    };

    return View(model);

}

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> ChangeUser(EditUserViewModel model)

{

    if (ModelState.IsValid)

    {

        Guid imageId = model.ImageId;

        if (model.ImageFile != null)

        {

            imageId = await \_blobHelper.UploadBlobAsync(model.ImageFile, "users");

        }

        User user = await \_userHelper.GetUserAsync(User.Identity.Name);

        user.FirstName = model.FirstName;

        user.LastName = model.LastName;

        user.Address = model.Address;

        user.PhoneNumber = model.PhoneNumber;

        user.ImageId = imageId;

        user.City = await \_context.Cities.FindAsync(model.CityId);

        user.Document = model.Document;

        await \_userHelper.UpdateUserAsync(user);

        return RedirectToAction("Index", "Home");

    }

    return View(model);

}

1. Adicionamos la vista **ChangeUser** en el **AccountController**:

@model Shooping.Models.EditUserViewModel

@{

    ViewData["Title"] = "Edit";

}

<h2>Editar</h2>

<h4>Usuario</h4>

<hr />

<div class="row">

    <div class="col-md-8">

        <form asp-action="ChangeUser" enctype="multipart/form-data">

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

            <input type="hidden" asp-for="Id" />

            <input type="hidden" asp-for="ImageId" />

            <input type="hidden" asp-for="Countries" />

            <input type="hidden" asp-for="States" />

            <input type="hidden" asp-for="Cities" />

            <partial name="\_User" />

            <div class="form-group mt-2">

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

                <a asp-action="ChangePassword" class="btn btn-outline-secondary">Cambiar Contraseña</a>

            </div>

        </form>

    </div>

    <div class="col-md-4">

        <img src="@Model.ImageFullPath" style="width:200px;height:200px;border-radius:50%" />

    </div>

</div>

@section Scripts {

    @{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

    <script type="text/javascript">

        $(document).ready(function () {

            $("#CountryId").change(function () {

                debugger;

                $("#StateId").empty();

                $("#StateId").append('<option value="0">[Selecciona un Departamento / Estado...]</option>');

                $("#CityId").empty();

                $("#CityId").append('<option value="0">[Selecciona una ciudad...]</option>');

                $.ajax({

                    type: 'POST',

                    url: '@Url.Action("GetStates")',

                    dataType: 'json',

                    data: { countryId: $("#CountryId").val() },

                    success: function (states) {

                        $.each(states, function (i, state) {

                            $("#StateId").append('<option value="'

                                + state.id + '">'

                                + state.name + '</option>');

                        });

                    },

                    error: function (ex) {

                        alert('Failed to retrieve states.' + ex);

                    }

                });

                return false;

            })

            $("#StateId").change(function () {

                $("#CityId").empty();

                $("#CityId").append('<option value="0">[Selecciona una ciudad...]</option>');

                $.ajax({

                    type: 'POST',

                    url: '@Url.Action("GetCities")',

                    dataType: 'json',

                    data: { stateId: $("#StateId").val() },

                    success: function (cities) {

                        $.each(cities, function (i, city) {

                            debugger;

                            $("#CityId").append('<option value="'

                                + city.id + '">'

                                + city.name + '</option>');

                        });

                    },

                    error: function (ex) {

                        alert('Failed to retrieve cities.' + ex);

                    }

                });

                return false;

            })

        });

    </script>

}

Cambiando Contraseña

1. Adicione estos métodos al **AccountControlle**:

public IActionResult ChangePassword()

{

return View();

}

[HttpPost]

public async Task<IActionResult> ChangePassword(ChangePasswordViewModel model)

{

if (ModelState.IsValid)

{

     var user = await \_userHelper.GetUserAsync(User.Identity.Name);

     if (user != null)

     {

         var result = await \_userHelper.ChangePasswordAsync(user, model.OldPassword, model.NewPassword);

         if (result.Succeeded)

         {

             return RedirectToAction("ChangeUser");

         }

         else

         {

             ModelState.AddModelError(string.Empty, result.Errors.FirstOrDefault().Description);

         }

     }

     else

     {

         ModelState.AddModelError(string.Empty, "User no found.");

     }

}

return View(model);

}

1. Add the view **ChangePassword** to **AccountController** class:

@model Shooping.Models.ChangePasswordViewModel

@{

ViewData["Title"] = "Change Password";

}

<h2>Cambio de Contraseña</h2>

<div class="row">

<div class="col-md-4 offset-md-4">

     <form method="post">

         <div asp-validation-summary="ModelOnly"></div>

         <div class="form-group">

             <label asp-for="OldPassword"></label>

             <input asp-for="OldPassword" type="password" class="form-control" />

             <span asp-validation-for="OldPassword" class="text-warning"></span>

         </div>

         <div class="form-group">

             <label asp-for="NewPassword"></label>

             <input asp-for="NewPassword" type="password" class="form-control" />

             <span asp-validation-for="NewPassword" class="text-warning"></span>

         </div>

         <div class="form-group">

             <label asp-for="Confirm"></label>

             <input asp-for="Confirm" type="password" class="form-control" />

             <span asp-validation-for="Confirm" class="text-warning"></span>

         </div>

         <div class="form-group mt-2">

             <input type="submit" value="Cambiar Contraseña" class="btn btn-outline-primary" />

             <a asp-action="ChangeUser" class="btn btn-outline-success">Regresar</a>

         </div>

     </form>

</div>

</div>

@section Scripts {

@{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

}

Pequeña corrección al momento de editar usuario

En el **EditUserViewModel** colocar el signo de interrogación en **public IFormFile? ImageFile**, para hacerlo opcional y no obligarnos a ingresar la imagen.

Evitar Warnings por nulos

Para evitar que salgan tantos Warning por motivo de manejo de nulos podemos deshabilitar esto haciendo doble clic al proyecto y editando esto (Gracias a Jimmy Dávila):

  <PropertyGroup>

    <TargetFramework>net6.0</TargetFramework>

    <Nullable>disable</Nullable>

    <ImplicitUsings>enable</ImplicitUsings>

  </PropertyGroup>

Bloqueo de usuarios por intentos fallidos

1. Cambiamos la configuración de usuarios en el **Program**:

builder.Services.AddIdentity<User, IdentityRole>(cfg =>

{

    cfg.User.RequireUniqueEmail = true;

    cfg.Password.RequireDigit = false;

    cfg.Password.RequiredUniqueChars = 0;

    cfg.Password.RequireLowercase = false;

    cfg.Password.RequireNonAlphanumeric = false;

    cfg.Password.RequireUppercase = false;

    cfg.Lockout.DefaultLockoutTimeSpan = TimeSpan.FromMinutes(5);

    cfg.Lockout.MaxFailedAccessAttempts = 3;

    cfg.Lockout.AllowedForNewUsers = true;

})

    .AddEntityFrameworkStores<DataContext>();

1. Modificamos el **UserHelper**:

public async Task<SignInResult> LoginAsync(LoginViewModel model)

{

    return await \_signInManager.PasswordSignInAsync(

        model.Username,

        model.Password,

        model.RememberMe,

        true);

}

1. Modificamos el método **Login** en el **AccountController**:

[HttpPost]

public async Task<IActionResult> Login(LoginViewModel model)

{

    if (ModelState.IsValid)

    {

        Microsoft.AspNetCore.Identity.SignInResult result = await \_userHelper.LoginAsync(model);

        if (result.Succeeded)

        {

            if (Request.Query.Keys.Contains("ReturnUrl"))

            {

                return Redirect(Request.Query["ReturnUrl"].First());

            }

            return RedirectToAction("Index", "Home");

        }

        if (result.IsLockedOut)

        {

            ModelState.AddModelError(string.Empty, "Ha superado el máximo número de intentos, su cuenta está bloqueada, intente de nuevo en 5 minutos.");

        }

        else

        {

            ModelState.AddModelError(string.Empty, "Email o contraseña incorrectos.");

        }

    }

    return View(model);

}

1. Probamos.

Confirmar el registro de usuarios

1. Cambiamos la configuración de usuarios en el **Program**:

builder.Services.AddIdentity<User, IdentityRole>(cfg =>

{

    cfg.Tokens.AuthenticatorTokenProvider = TokenOptions.DefaultAuthenticatorProvider;

    cfg.SignIn.RequireConfirmedEmail = true;

    cfg.User.RequireUniqueEmail = true;

    cfg.Password.RequireDigit = false;

    cfg.Password.RequiredUniqueChars = 0;

    cfg.Password.RequireLowercase = false;

    cfg.Password.RequireNonAlphanumeric = false;

    cfg.Password.RequireUppercase = false;

    cfg.Lockout.DefaultLockoutTimeSpan = TimeSpan.FromMinutes(5);

    cfg.Lockout.MaxFailedAccessAttempts = 3;

    cfg.Lockout.AllowedForNewUsers = true;

})

    .AddDefaultTokenProviders()

    .AddEntityFrameworkStores<DataContext>();

1. Verificamos que tengamos los permisos abiertos en nuestra cuenta de google: <https://myaccount.google.com/lesssecureapps> y <https://accounts.google.com/DisplayUnlockCaptcha>.

1. Adicionamos estos parámetros a nuestro archivo de configuración:

"Mail": {

  "From": "onsalezulu@gmail.com",

  "Smtp": "smtp.gmail.com",

  "Port": 587,

  "Password": "Zulu1234."

}

1. Adicionamos el nuget “**Mailkit**”.

1. En los **Common** creamos **Response**:

public class Response

{

    public bool IsSuccess { get; set; }

    public string Message { get; set; }

    public object Result { get; set; }

}

1. En los **Helpers** adicionamos la interzar **IMailHelper**:

public interface IMailHelper

{

    Response SendMail(string toName, string toEmail, string subject, string body);

}

1. Luego agregamos la implementation **MailHelper**:

using MailKit.Net.Smtp;

using MimeKit;

using Shooping.Common;

namespace Shooping.Helpers

{

    public class MailHelper : IMailHelper

    {

        private readonly IConfiguration \_configuration;

        public MailHelper(IConfiguration configuration)

        {

            \_configuration = configuration;

        }

        public Response SendMail(string toName,string toEmail, string subject, string body)

        {

            try

            {

                string from = \_configuration["Mail:From"];

                string name = \_configuration["Mail:Name"];

                string smtp = \_configuration["Mail:Smtp"];

                string port = \_configuration["Mail:Port"];

                string password = \_configuration["Mail:Password"];

                MimeMessage message = new MimeMessage();

                message.From.Add(new MailboxAddress(name, from));

                message.To.Add(new MailboxAddress(toName, toEmail));

                message.Subject = subject;

                BodyBuilder bodyBuilder = new BodyBuilder

                {

                    HtmlBody = body

                };

                message.Body = bodyBuilder.ToMessageBody();

                using (SmtpClient client = new SmtpClient())

                {

                    client.Connect(smtp, int.Parse(port), false);

                    client.Authenticate(from, password);

                    client.Send(message);

                    client.Disconnect(true);

                }

                return new Response { IsSuccess = true };

            }

            catch (Exception ex)

            {

                return new Response

                {

                    IsSuccess = false,

                    Message = ex.Message,

                    Result = ex

                };

            }

        }

    }

}

1. Configuramos la inyección del servicio:

builder.Services.AddScoped<IMailHelper, MailHelper>();

1. Add those methods to **IUserHelper**:

Task<string> GenerateEmailConfirmationTokenAsync(User user);

Task<IdentityResult> ConfirmEmailAsync(User user, string token);

Y la implementación:

public async Task<IdentityResult> ConfirmEmailAsync(User user, string token)

{

return await \_userManager.ConfirmEmailAsync(user, token);

}

public async Task<string> GenerateEmailConfirmationTokenAsync(User user)

{

return await \_userManager.GenerateEmailConfirmationTokenAsync(user);

}

1. Modificamos el POST de registrar usuario (primero inyectamos el **IMailHelper** en el **AccountController**):

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Register(AddUserViewModel model)

{

    if (ModelState.IsValid)

    {

        Guid imageId = Guid.Empty;

        if (model.ImageFile != null)

        {

            imageId = await \_blobHelper.UploadBlobAsync(model.ImageFile, "users");

        }

        User user = await \_userHelper.AddUserAsync(model, imageId);

        if (user == null)

        {

            ModelState.AddModelError(string.Empty, "Este correo ya está siendo usado.");

            return View(model);

        }

        string myToken = await \_userHelper.GenerateEmailConfirmationTokenAsync(user);

        string tokenLink = Url.Action("ConfirmEmail", "Account", new

        {

            userid = user.Id,

            token = myToken

        }, protocol: HttpContext.Request.Scheme);

        Response response = \_mailHelper.SendMail(

            $"{model.FirstName} {model.LastName}",

            model.Username,

            "Shopping - Confirmación de Email",

            $"<h1>Shopping - Confirmación de Email</h1>" +

                $"Para habilitar el usuario por favor hacer clicn en el siguiente link:, " +

                $"<p><a href = \"{tokenLink}\">Confirmar Email</a></p>");

        if (response.IsSuccess)

        {

            ViewBag.Message = "Las instrucciones para habilitar el usuario han sido enviadas al correo.";

            return View(model);

        }

        ModelState.AddModelError(string.Empty, response.Message);

    }

    return View(model);

}

1. Adicione estas líneas a la vista de registrar usuario:

<div class="text-success">

    <p>

        @ViewBag.Message

    </p>

</div>

1. Crear el método para confirmas el emai en el **AccountController**:

public async Task<IActionResult> ConfirmEmail(string userId, string token)

{

    if (string.IsNullOrEmpty(userId) || string.IsNullOrEmpty(token))

    {

        return NotFound();

    }

    User user = await \_userHelper.GetUserAsync(new Guid(userId));

    if (user == null)

    {

        return NotFound();

    }

    IdentityResult result = await \_userHelper.ConfirmEmailAsync(user, token);

    if (!result.Succeeded)

    {

        return NotFound();

    }

    return View();

}

1. Creamos la vista:

@{

    ViewData["Title"] = "Confirmación de Email";

}

<h2>@ViewData["Title"]</h2>

<div>

    <p>

        Gracias por confirmar el email, ahora puder iniciar sesión en el sistema.

    </p>

</div>

1. Borramos la BD con el comando **drop-database** para asegurarnos que todos los usuarios que creamos tengan  un correo confirmado.

1. Modificamos el alimentador de la base de datos:

private async Task<User> CheckUserAsync(

    string document,

    string firstName,

    string lastName,

    string email,

    string phone,

    string address,

    UserType userType)

{

    User user = await \_userHelper.GetUserAsync(email);

    if (user == null)

    {

        user = new User

        {

            FirstName = firstName,

            LastName = lastName,

            Email = email,

            UserName = email,

            PhoneNumber = phone,

            Address = address,

            Document = document,

            City = \_context.Cities.FirstOrDefault(),

            UserType = userType,

        };

        await \_userHelper.AddUserAsync(user, "123456");

        await \_userHelper.AddUserToRoleAsync(user, userType.ToString());

        string token = await \_userHelper.GenerateEmailConfirmationTokenAsync(user);

        await \_userHelper.ConfirmEmailAsync(user, token);

    }

    return user;

}

1. Modificamos el método **Login** en el **AccountController**:

[HttpPost]

public async Task<IActionResult> Login(LoginViewModel model)

{

    if (ModelState.IsValid)

    {

        Microsoft.AspNetCore.Identity.SignInResult result = await \_userHelper.LoginAsync(model);

        if (result.Succeeded)

        {

            if (Request.Query.Keys.Contains("ReturnUrl"))

            {

                return Redirect(Request.Query["ReturnUrl"].First());

            }

            return RedirectToAction("Index", "Home");

        }

        if (result.IsLockedOut)

        {

            ModelState.AddModelError(string.Empty, "Ha superado el máximo número de intentos, su cuenta está bloqueada, intente de nuevo en 5 minutos.");

        }

        else if(result.IsNotAllowed)

        {

            ModelState.AddModelError(string.Empty, "El usuario no ha sido habilitado, debes de seguir las instrucciones del correo enviado para poder habilitar el usuario.");

        }

        else

        {

            ModelState.AddModelError(string.Empty, "Email o contraseña incorrectos.");

        }

    }

    return View(model);

}

1. Probamos lo que llevamos hasta el momento.

1. Ahora hacemos lo mismo para la creación de administradores. Primero modificamos el POST de crear administrador:

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Create(AddUserViewModel model)

{

    if (ModelState.IsValid)

    {

        Guid imageId = Guid.Empty;

        if (model.ImageFile != null)

        {

            imageId = await \_blobHelper.UploadBlobAsync(model.ImageFile, "users");

        }

        User user = await \_userHelper.AddUserAsync(model, imageId);

        if (user == null)

        {

            ModelState.AddModelError(string.Empty, "Este correo ya está siendo usado.");

            return View(model);

        }

        string myToken = await \_userHelper.GenerateEmailConfirmationTokenAsync(user);

        string tokenLink = Url.Action("ConfirmEmail", "Account", new

        {

            userid = user.Id,

            token = myToken

        }, protocol: HttpContext.Request.Scheme);

        Response response = \_mailHelper.SendMail(

            $"{model.FirstName} {model.LastName}",

            model.Username,

            "Shopping - Confirmación de Email",

            $"<h1>Shopping - Confirmación de Email</h1>" +

                $"Para habilitar el usuario por favor hacer clicn en el siguiente link:, " +

                $"<p><a href = \"{tokenLink}\">Confirmar Email</a></p>");

        if (response.IsSuccess)

        {

            ViewBag.Message = "Las instrucciones para habilitar el usuario han sido enviadas al correo.";

            return View(model);

        }

        ModelState.AddModelError(string.Empty, response.Message);

    }

    return View(model);

}

1. Adicionamos esto a la vista de create:

<div class="text-success">

    <p>

        @ViewBag.Message

    </p>

</div>

1. Modificamos la vista **Index** del **UsersController** para agregar el campo **EmailConfirmed**.

1. Probamos.

Recuperación de contraseña

1. Modificamos la vista del login:

<div class="form-group mt-2">

    <input type="submit" value="Iniciar Sesión" class="btn btn-outline-primary" />

    <a asp-action="Register" class="btn btn-outline-secondary">Registrar Nuevo Usuario</a>

    <a asp-action="RecoverPassword" class="btn btn-link">¿Has olvidado tu contraseña?</a>

</div>

1. Adicionamos el modelo **RecoverPasswordViewModel**:

public class RecoverPasswordViewModel

{

    [Display(Name = "Email")]

    [Required(ErrorMessage = "El campo {0} es obligatorio.")]

    [EmailAddress(ErrorMessage = "Debes ingresar un correo válido.")]

    public string Email { get; set; }

}

1. Adicionamos el modelo **ResetPasswordViewModel**:

public class ResetPasswordViewModel

{

    [Display(Name = "Email")]

    [EmailAddress(ErrorMessage = "Debes ingresar un correo válido.")]

    [Required(ErrorMessage = "El campo {0} es obligatorio.")]

    public string UserName { get; set; }

    [DataType(DataType.Password)]

    [Required(ErrorMessage = "El campo {0} es obligatorio.")]

    [StringLength(20, MinimumLength = 6, ErrorMessage = "El campo {0} debe tener entre {2} y {1} carácteres.")]

    public string Password { get; set; }

    [Compare("Password", ErrorMessage = "La nueva contraseña y la confirmación no son iguales.")]

    [DataType(DataType.Password)]

    [Required(ErrorMessage = "El campo {0} es obligatorio.")]

    [StringLength(20, MinimumLength = 6, ErrorMessage = "El campo {0} debe tener entre {2} y {1} carácteres.")]

    public string ConfirmPassword { get; set; }

    [Required(ErrorMessage = "El campo {0} es obligatorio.")]

    public string Token { get; set; }

}

1. Adicionamos estos métodos al **IUserHelper**:

Task<string> GeneratePasswordResetTokenAsync(User user);

Task<IdentityResult> ResetPasswordAsync(User user, string token, string password);

Y la implementación:

public async Task<string> GeneratePasswordResetTokenAsync(User user)

{

return await \_userManager.GeneratePasswordResetTokenAsync(user);

}

public async Task<IdentityResult> ResetPasswordAsync(User user, string token, string password)

{

return await \_userManager.ResetPasswordAsync(user, token, password);

}

1. Adicionamos estos métodos al **AccountController**:

public IActionResult RecoverPassword()

{

    return View();

}

[HttpPost]

public async Task<IActionResult> RecoverPassword(RecoverPasswordViewModel model)

{

    if (ModelState.IsValid)

    {

        User user = await \_userHelper.GetUserAsync(model.Email);

        if (user == null)

        {

            ModelState.AddModelError(string.Empty, "El email no corresponde a ningún usuario registrado.");

            return View(model);

        }

        string myToken = await \_userHelper.GeneratePasswordResetTokenAsync(user);

        string link = Url.Action(

            "ResetPassword",

            "Account",

            new { token = myToken }, protocol: HttpContext.Request.Scheme);

        \_mailHelper.SendMail(

            $"{user.FullName}",

            model.Email,

            "Shopping - Recuperación de Contraseña",

            $"<h1>Shopping - Recuperación de Contraseña</h1>" +

            $"Para recuperar la contraseña haga click en el siguiente enlace:" +

            $"<p><a href = \"{link}\">Reset Password</a></p>");

        ViewBag.Message = "Las instrucciones para recuperar la contraseña han sido enviadas a su correo.";

        return View();

    }

    return View(model);

}

public IActionResult ResetPassword(string token)

{

    return View();

}

[HttpPost]

public async Task<IActionResult> ResetPassword(ResetPasswordViewModel model)

{

    User user = await \_userHelper.GetUserAsync(model.UserName);

    if (user != null)

    {

        IdentityResult result = await \_userHelper.ResetPasswordAsync(user, model.Token, model.Password);

        if (result.Succeeded)

        {

            ViewBag.Message = "Contraseña cambiada con éxito.";

            return View();

        }

        ViewBag.Message = "Error cambiando la contraseña.";

        return View(model);

    }

    ViewBag.Message = "Usuario no encontrado.";

    return View(model);

}

1. Adicionamos la vista de recuperar contraseña:

@model Shooping.Models.RecoverPasswordViewModel

@{

    ViewData["Title"] = "Recover Password";

}

<h2>Recuperación de Constraseña</h2>

<div class="row">

    <div class="col-md-4 offset-md-4">

        <form method="post">

            <div asp-validation-summary="ModelOnly"></div>

            <div class="form-group">

                <label asp-for="Email"></label>

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

                <span asp-validation-for="Email" class="text-warning"></span>

            </div>

            <div class="form-group mt-2">

                <input type="submit" value="Recuperación de Contraseña" class="btn btn-outline-primary" />

                <a asp-action="Login" class="btn btn-outline-success">Regresar</a>

            </div>

        </form>

        <div class="text-success">

            <p>

                @ViewBag.Message

            </p>

        </div>

    </div>

</div>

@section Scripts {

    @{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

}

1. Y adicionamos la vista para resetear la contraseña:

@model Shooping.Models.ResetPasswordViewModel

@{

    ViewData["Title"] = "Reset Password";

}

<h1>Resetea tu Contraseña</h1>

<div class="row">

    <div class="col-md-4 offset-md-4">

        <form method="post">

            <div asp-validation-summary="All"></div>

            <input type="hidden" asp-for="Token" />

            <div class="form-group">

                <label asp-for="UserName"></label>

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

                <span asp-validation-for="UserName" class="text-warning"></span>

            </div>

            <div class="form-group">

                <label asp-for="Password"></label>

                <input asp-for="Password" type="password" class="form-control" />

                <span asp-validation-for="Password" class="text-warning"></span>

            </div>

            <div class="form-group">

                <label asp-for="ConfirmPassword"></label>

                <input asp-for="ConfirmPassword" type="password" class="form-control" />

                <span asp-validation-for="ConfirmPassword" class="text-warning"></span>

            </div>

            <div class="form-group mt-2">

                <input type="submit" value="Resetear Contraseña" class="btn btn-outline-primary" />

            </div>

        </form>

        <div class="text-success">

            <p>

                @ViewBag.Message

            </p>

        </div>

    </div>

</div>

@section Scripts {

    @{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

}

Adición de íconos y mejorar a la UI

1. Adicionamos las librerias de **Font-Awesone** haciendo click derecho sobre el **wwwroot/lib** y adicionando un **Client Side Library**.

1. Adicionamos el CSS y los Script de **Font-Awesone** en el **\_Layout**:

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="utf-8" />

    <meta name="viewport" content="width=device-width, initial-scale=1.0" />

    <title>@ViewData["Title"] - Shopping</title>

    <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="~/Shooping.styles.css" asp-append-version="true" />

    <link rel="stylesheet" href="~/lib/font-awesome/css/all.min.css" />

</head>

<body>

    <header>

       <nav class="navbar navbar-expand-sm navbar-toggleable-sm navbar-light bg-primary border-bottom box-shadow mb-3">

            <div class="container-fluid">

                <a class="navbar-brand " asp-area="" asp-controller="Home" asp-action="Index"><i class="fa-solid fa-bag-shopping text-white"></i> Shopping</a>

                <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target=".navbar-collapse" aria-controls="navbarSupportedContent"

                        aria-expanded="false" aria-label="Toggle navigation">

                    <span class="navbar-toggler-icon"></span>

                </button>

                <div class="navbar-collapse collapse d-sm-inline-flex justify-content-between">

                    <ul class="navbar-nav flex-grow-1">

                        <li class="nav-item">

                            <a class="nav-link text-white" asp-area="" asp-controller="Home" asp-action="Index">Inicio</a>

                        </li>

                        <li class="nav-item">

                            <a class="nav-link text-white" asp-area="" asp-controller="Home" asp-action="Privacy">Políticas</a>

                        </li>

                        @if (User.Identity.IsAuthenticated && User.IsInRole("Admin"))

                        {

                            <li class="nav-item">

                                <a class="nav-link text-white" asp-area="" asp-controller="Categories" asp-action="Index">Categorías</a>

                            </li>

                            <li class="nav-item">

                                <a class="nav-link text-white" asp-area="" asp-controller="Countries" asp-action="Index">Países</a>

                            </li>

                            <li class="nav-item">

                                <a class="nav-link text-white" asp-area="" asp-controller="Orders" asp-action="Index">Pedidos</a>

                            </li>

                            <li class="nav-item">

                                <a class="nav-link text-white" asp-area="" asp-controller="Products" asp-action="Index">Productos</a>

                            </li>

                            <li class="nav-item">

                                <a class="nav-link text-white" asp-area="" asp-controller="Users" asp-action="Index">Usuarios</a>

                            </li>

                        }

                        @if (User.Identity.IsAuthenticated && User.IsInRole("User"))

                        {

                            <li class="nav-item">

                                <a class="nav-link text-white" asp-area="" asp-controller="Orders" asp-action="MyOrders">Mis Pedidos</a>

                            </li>

                        }

                    </ul>

                    <ul class="nav navbar-nav navbar-right">

                        @if (User.Identity.IsAuthenticated)

                        {

                            <li class="nav-item">

                                <a class="nav-link text-white" asp-area="" asp-controller="Account" asp-action="ChangeUser">@User.Identity.Name</a>

                            </li>

                            <li class="nav-item">

                                <a class="nav-link text-white" asp-area="" asp-controller="Account" asp-action="Logout">Cerrar Sesión</a>

                            </li>

                        }

                        else

                        {

                            <li class="nav-item">

                                <a class="nav-link text-white" asp-area="" asp-controller="Account" asp-action="Login">Iniciar Sesión</a>

                            </li>

                        }

                    </ul>

                </div>

            </div>

        </nav>

    </header>

    <div class="container">

        <main role="main" class="pb-3">

            @RenderBody()

        </main>

    </div>

    <footer class="border-top footer text-muted">

        <div class="container">

            &copy; 2022 - Shopping - <a asp-area="" asp-controller="Home" asp-action="Privacy">Políticas de Privacidad</a>

        </div>

    </footer>

    <script src="~/lib/jquery/dist/jquery.min.js"></script>

    <script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>

    <script src="~/js/site.js" asp-append-version="true"></script>

    <script src="~/lib/font-awesome/js/fontawesome.js"></script>

    @await RenderSectionAsync("Scripts", required: false)

</body>

</html>

1. Modificamos la vista **Login**:

@model Shooping.Models.LoginViewModel

@{

    ViewData["Title"] = "Login";

}

<div class="row">

    <div class="col-md-4  offset-md-4">

        <div class="card bg-light mb-3" style="width: 450px; height: 450px">

            <h5 class="card-header text-center"><i class="fa-solid fa-envelope text-navy"></i> Iniciar Sesión</h5>

            <div class="card-body bg-warning">

                <form asp-action="Login">

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

                    <div class="form-group">

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

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

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

                    </div>

                    <div class="form-group">

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

                        <input asp-for="Password" type="password" class="form-control" />

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

                    </div>

                    <div class="form-group mt-2">

                        <div class="form-check">

                            <input asp-for="RememberMe" type="checkbox" class="form-check-input" />

                            <label asp-for="RememberMe" class="form-check-label"></label>

                        </div>

                        <span asp-validation-for="RememberMe" class="text-warning"></span>

                    </div>

                    <div class="form-group mt-2">

                        <button input type="submit" value="Login" class="btn btn-primary btn-block"><i class="fa-solid fa-user"></i> Iniciar Sesión</button>

                        <a asp-action="Register" class="btn btn-secondary"><i class="fa-solid fa-circle-plus"></i> Registrar Nuevo Usuario</a>

                    </div>

                </form>

            </div>

            <div class="card-footer bg-transparent text-center">

                ¿Has olvidado tu contraseña?<a asp-action="RecoverPassword"> Recuperala</a>

            </div>

        </div>

    </div>

</div>

@section Scripts {

    @{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

}

1. Modificamos el **Index** de **Categories**:

<a asp-action="Edit" asp-route-id="@item.Id" class="btn btn-outline-warning"><i class="fa-solid fa-pen-to-square"></i></a>

<a asp-action="Details" asp-route-id="@item.Id" class="btn btn-outline-info"><i class="fa-solid fa-circle-info"></i></a>

<a asp-action="Delete" asp-route-id="@item.Id" class="btn btn-outline-danger"><i class="fa-solid fa-trash"></i></a>

1. Modificamos el **Index** de **Countries**:

<a asp-action="Edit" asp-route-id="@item.Id" class="btn btn-outline-warning"><i class="fa-solid fa-pen-to-square"></i></a>

<a asp-action="Details" asp-route-id="@item.Id" class="btn btn-outline-info"><i class="fa-solid fa-circle-info"></i></a>

<a asp-action="Delete" asp-route-id="@item.Id" class="btn btn-outline-danger"><i class="fa-solid fa-trash"></i></a>

1. Modificamos el **Details** de **Countries**:

<a asp-action="EditState" asp-route-id="@item.Id" class="btn btn-outline-warning"><i class="fa-solid fa-pen-to-square"></i></a>

<a asp-action="DetailsState" asp-route-id="@item.Id" class="btn btn-outline-info"><i class="fa-solid fa-circle-info"></i></a>

<a asp-action="DeleteState" asp-route-id="@item.Id" class="btn btn-outline-danger"><i class="fa-solid fa-trash"></i></a>

1. Modificamos el **DetailsState** de **Countries**:

<a asp-action="EditCity" asp-route-id="@item.Id" class="btn btn-outline-warning"><i class="fa-solid fa-pen-to-square"></i></a>

<a asp-action="DetailsCity" asp-route-id="@item.Id" class="btn btn-outline-info"><i class="fa-solid fa-circle-info"></i></a>

<a asp-action="DeleteCity" asp-route-id="@item.Id" class="btn btn-outline-danger"><i class="fa-solid fa-trash"></i></a>

1. Probamos.

Productos

1. Creamos el **Products** entity:

public class Product

{

    public int Id { get; set; }

    [Display(Name = "Nombre")]

    [MaxLength(50, ErrorMessage = "El campo {0} debe tener máximo {1} caractéres.")]

    [Required(ErrorMessage = "El campo {0} es obligatorio.")]

    public string Name { get; set; }

    [DataType(DataType.MultilineText)]

    [Display(Name = "Descripción")]

    [MaxLength(500, ErrorMessage = "El campo {0} debe tener máximo {1} caractéres.")]

    public string Description { get; set; }

    [Column(TypeName = "decimal(18,2)")]

    [DisplayFormat(DataFormatString = "{0:C2}")]

    [Display(Name = "Precio")]

    [Required(ErrorMessage = "El campo {0} es obligatorio.")]

    public decimal Price { get; set; }

    [DisplayFormat(DataFormatString = "{0:N2}")]

    [Display(Name = "Inventario")]

    [Required(ErrorMessage = "El campo {0} es obligatorio.")]

    public float Stock { get; set; }

    public ICollection<ProductCategory> ProductCategories { get; set; }

    [Display(Name = "Categorías")]

    public int CategoriesNumber => ProductCategories == null ? 0 : ProductCategories.Count;

    public ICollection<ProductImage> ProductImages { get; set; }

    [Display(Name = "Fotos")]

    public int ImagesNumber => ProductImages == null ? 0 : ProductImages.Count;

    //TODO: Pending to change to the correct path

    [Display(Name = "Foto")]

    public string ImageFullPath => ProductImages == null || ProductImages.Count == 0

        ? $"https://localhost:7057/images/noimage.png"

        : ProductImages.FirstOrDefault().ImageFullPath;

}

1. Creamos el **ProductCategory** entity:

public class ProductCategory

{

    public int Id { get; set; }

    public Product Product { get; set; }

    public Category Category { get; set; }

}

1. Creamos el **ProductImage** entity:

public class ProductImage

{

    public int Id { get; set; }

    public Product Product { get; set; }

    [Display(Name = "Foto")]

    public Guid ImageId { get; set; }

    //TODO: Pending to change to the correct path

    [Display(Name = "Foto")]

    public string ImageFullPath => ImageId == Guid.Empty

        ? $"https://localhost:7057/images/noimage.png"

        : $"https://shopping4.blob.core.windows.net/products/{ImageId}";

}

1. Modificamos **Category** entity:

public ICollection<ProductCategory> ProductCategories { get; set; }

1. Modificamos el **DataContext**:

public DbSet<Product> Products { get; set; }

public DbSet<ProductCategory> ProductCategories { get; set; }

public DbSet<ProductImage> ProductImages { get; set; }

…

modelBuilder.Entity<Product>().HasIndex(c => c.Name).IsUnique();

modelBuilder.Entity<ProductCategory>().HasIndex("ProductId", "CategoryId").IsUnique();

1. Agregrar la migración y actualizar la BD.

1. Creamos el **ProductsController**:

[Authorize(Roles = "Admin")]

public class ProductsController : Controller

{

    private readonly DataContext \_context;

    private readonly ICombosHelper \_combosHelper;

    private readonly IBlobHelper \_blobHelper;

    public ProductsController(DataContext context, ICombosHelper combosHelper, IBlobHelper blobHelper)

    {

        \_context = context;

        \_combosHelper = combosHelper;

        \_blobHelper = blobHelper;

    }

    public async Task<IActionResult> Index()

    {

        return View(await \_context.Products

            .Include(p => p.ProductImages)

            .Include(p => p.ProductCategories)

            .ThenInclude(pc => pc.Category)

            .ToListAsync());

    }

}

1. Creamos la vista **Index**:

@model IEnumerable<Shooping.Data.Entities.Product>

@{

    ViewData["Title"] = "Index";

}

<link rel="stylesheet" href="https://cdn.datatables.net/1.10.19/css/jquery.dataTables.min.css" />

<p>

    <a asp-action="Create" class="btn btn-outline-primary">Crear Nuevo</a>

</p>

<div class="row">

    <div class="col-md-12">

        <div class="panel panel-default">

            <div class="panel-heading">

                <h3 class="panel-title">Productos</h3>

            </div>

            <div class="panel-body">

                <table class="table table-hover table-responsive table-striped" id="MyTable">

                    <thead>

                        <tr>

                            <th>

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

                            </th>

                            <th>

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

                            </th>

                            <th>

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

                            </th>

                            <th>

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

                            </th>

                            <th>

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

                            </th>

                            <th>

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

                            </th>

                            <th>

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

                            </th>

                            <th></th>

                        </tr>

                    </thead>

                    <tbody>

                        @foreach (var item in Model)

                        {

                            <tr>

                                <td>

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

                                </td>

                                <td>

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

                                </td>

                                <td>

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

                                </td>

                                <td>

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

                                </td>

                                <td>

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

                                </td>

                                <td>

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

                                </td>

                                <td>

                                    <img src="@item.ImageFullPath" style="width:100px;" />

                                </td>

                                <td>

                                    <a asp-action="Edit" asp-route-id="@item.Id" class="btn btn-outline-warning"><i class="fa-solid fa-pen-to-square"></i></a>

                                    <a asp-action="Details" asp-route-id="@item.Id" class="btn btn-outline-info"><i class="fa-solid fa-circle-info"></i></a>

                                    <a asp-action="Delete" asp-route-id="@item.Id" class="btn btn-outline-danger"><i class="fa-solid fa-trash"></i></a>

                                </td>

                            </tr>

                        }

                    </tbody>

                </table>

            </div>

        </div>

    </div>

</div>

@section Scripts {

    @{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

    <script src="//cdn.datatables.net/1.10.19/js/jquery.dataTables.min.js"></script>

    <script type="text/javascript">

        $(document).ready(function () {

            $('#MyTable').DataTable({

                "language": {

                    "url": "//cdn.datatables.net/plug-ins/9dcbecd42ad/i18n/Spanish.json"

                },

                "aLengthMenu": [

                    [25, 50, 100, 200, -1],

                    [25, 50, 100, 200, "Todos"]

                ]

            });

        });

    </script>

}

1. Agregamos la entrada en el menú y probamos.

1. Creamos el **EditProductViewModel**:

public class EditProductViewModel

{

    public int Id { get; set; }

    [Display(Name = "Nombre")]

    [MaxLength(50, ErrorMessage = "El campo {0} debe tener máximo {1} caractéres.")]

    [Required(ErrorMessage = "El campo {0} es obligatorio.")]

    public string Name { get; set; }

    [Display(Name = "Descripción")]

    [MaxLength(500, ErrorMessage = "El campo {0} debe tener máximo {1} caractéres.")]

    [Required(ErrorMessage = "El campo {0} es obligatorio.")]

    public string Description { get; set; }

    [DisplayFormat(DataFormatString = "{0:C2}")]

    [Display(Name = "Precio")]

    [Required(ErrorMessage = "El campo {0} es obligatorio.")]

    public decimal Price { get; set; }

    [DisplayFormat(DataFormatString = "{0:N2}")]

    [Display(Name = "Inventario")]

    [Required(ErrorMessage = "El campo {0} es obligatorio.")]

    public float Stock { get; set; }

}

1. Creamos el **CreateProductViewModel**:

public class CreateProductViewModel : EditProductViewModel

{

    [Display(Name = "Categoría")]

    [Range(1, int.MaxValue, ErrorMessage = "Debes seleccionar una categoría.")]

    [Required(ErrorMessage = "El campo {0} es obligatorio.")]

    public int CategoryId { get; set; }

    public IEnumerable<SelectListItem> Categories { get; set; }

    [Display(Name = "Foto")]

    public IFormFile? ImageFile { get; set; }

}

1. Agredamos los métodos del **Create** en el **ProductsController**:

public async Task<IActionResult> Create()

{

    CreateProductViewModel model = new()

    {

        Categories = await \_combosHelper.GetComboCategoriesAsync(),

    };

    return View(model);

}

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Create(CreateProductViewModel model)

{

    if (ModelState.IsValid)

    {

        Guid imageId = Guid.Empty;

        if (model.ImageFile != null)

        {

            imageId = await \_blobHelper.UploadBlobAsync(model.ImageFile, "products");

        }

        Product product = new()

        {

            Description = model.Description,

            Name = model.Name,

            Price = model.Price,

            Stock = model.Stock,

        };

        product.ProductCategories = new List<ProductCategory>()

        {

            new ProductCategory

            {

                Category = await \_context.Categories.FindAsync(model.CategoryId)

            }

        };

        if (imageId != Guid.Empty)

        {

            product.ProductImages = new List<ProductImage>()

            {

                new ProductImage { ImageId = imageId }

            };

        }

        try

        {

            \_context.Add(product);

            await \_context.SaveChangesAsync();

            return RedirectToAction(nameof(Index));

        }

        catch (DbUpdateException dbUpdateException)

        {

            if (dbUpdateException.InnerException.Message.Contains("duplicate"))

            {

                ModelState.AddModelError(string.Empty, "Ya existe un producto con el mismo nombre.");

            }

            else

            {

                ModelState.AddModelError(string.Empty, dbUpdateException.InnerException.Message);

            }

        }

        catch (Exception exception)

        {

            ModelState.AddModelError(string.Empty, exception.Message);

        }

    }

    model.Categories = await \_combosHelper.GetComboCategoriesAsync();

    return View(model);

}

1. Agredamos la vista parcial **\_CreateProduct** en el **ProductsController**:

@model Shooping.Models.CreateProductViewModel

<div class="row">

    <div class="col-md-6">

        <div class="form-group">

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

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

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

        </div>

        <div class="form-group">

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

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

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

        </div>

        <div class="form-group">

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

            <select asp-for="CategoryId" asp-items="Model.Categories" class="form-control"></select>

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

        </div>

    </div>

    <div class="col-md-6">

        <div class="form-group">

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

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

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

        </div>

        <div class="form-group">

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

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

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

        </div>

        <div class="form-group">

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

            <input asp-for="ImageFile" class="form-control" type="file" />

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

        </div>

    </div>

</div>

1. Agredamos la vista **Create** en el **ProductsController**:

@model Shooping.Models.CreateProductViewModel

@{

    ViewData["Title"] = "Create";

}

<h1>Crear</h1>

<h4>Producto</h4>

<hr />

<div class="row">

    <div class="col-md-12">

        <form asp-action="Create" enctype="multipart/form-data">

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

            <input type="hidden" asp-for="Categories" />

            <partial name="\_CreateProduct"/>

            <div class="form-group mt-2">

                <input type="submit" value="Crear" class="btn btn-outline-primary" />

                <a asp-action="Index" class="btn btn-outline-success">Regresar</a>

            </div>

        </form>

    </div>

</div>

@section Scripts {

    @{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

}

1. Probemos lo que llevamos hasta el momento.

ACA VAMOS

1. Agredamos los métodos del **Edit** en el **ProductsController**:

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

{

    if (id == null)

    {

        return NotFound();

    }

    Product product = await \_context.Products.FindAsync(id);

    if (product == null)

    {

        return NotFound();

    }

    EditProductViewModel model = new()

    {

        Description = product.Description,

        Id = product.Id,

        Name = product.Name,

        Price = product.Price,

        Stock = product.Stock,

    };

    return View(model);

}

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Edit(int id, CreateProductViewModel model)

{

    if (id != model.Id)

    {

        return NotFound();

    }

    try

    {

        Product product = await \_context.Products.FindAsync(model.Id);

        product.Description = model.Description;

        product.Name = model.Name;

        product.Price = model.Price;

        product.Stock = model.Stock;

        \_context.Update(product);

        await \_context.SaveChangesAsync();

        return RedirectToAction(nameof(Index));

    }

    catch (DbUpdateException dbUpdateException)

    {

        if (dbUpdateException.InnerException.Message.Contains("duplicate"))

        {

            ModelState.AddModelError(string.Empty, "Ya existe un producto con el mismo nombre.");

        }

        else

        {

            ModelState.AddModelError(string.Empty, dbUpdateException.InnerException.Message);

        }

    }

    catch (Exception exception)

    {

        ModelState.AddModelError(string.Empty, exception.Message);

    }

    return View(model);

}

1. Agredamos la vista parcial **\_EditProduct** en el **ProductsController**:

@model Shooping.Models.EditProductViewModel

<div class="row">

    <div class="col-md-6">

        <div class="form-group">

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

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

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

        </div>

        <div class="form-group">

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

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

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

        </div>

    </div>

    <div class="col-md-6">

        <div class="form-group">

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

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

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

        </div>

        <div class="form-group">

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

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

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

        </div>

    </div>

</div>

1. Agredamos la vista **Edit** en el **ProductsController**:

@model Shooping.Models.EditProductViewModel

@{

    ViewData["Title"] = "Edit";

}

<h1>Editar</h1>

<h4>Producto</h4>

<hr />

<div class="row">

    <div class="col-md-12">

        <form asp-action="Edit">

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

            <input type="hidden" asp-for="Id" />

            <partial name="\_EditProduct"/>

            <div class="form-group mt-2">

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

                <a asp-action="Index" class="btn btn-outline-success">Regresar</a>

            </div>

        </form>

    </div>

</div>

@section Scripts {

    @{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

}

1. Creamos el método **Details**:

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

{

    if (id == null)

    {

        return NotFound();

    }

    Product product = await \_context.Products

        .Include(p => p.ProductImages)

        .Include(p => p.ProductCategories)

        .ThenInclude(pc => pc.Category)

        .FirstOrDefaultAsync(p => p.Id == id);

    if (product == null)

    {

        return NotFound();

    }

    return View(product);

}

1. Adicionamos la vista parcial **\_ProductDetails**:

@model Shooping.Data.Entities.Product

<h4>País</h4>

<hr />

<dl class="row">

    <dt class = "col-sm-2">

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

    </dt>

    <dd class = "col-sm-10">

        @Html.DisplayFor(model => model.Name)

    </dd>

    <dt class = "col-sm-2">

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

    </dt>

    <dd class = "col-sm-10">

        @Html.DisplayFor(model => model.Description)

    </dd>

    <dt class = "col-sm-2">

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

    </dt>

    <dd class = "col-sm-10">

        @Html.DisplayFor(model => model.Price)

    </dd>

    <dt class = "col-sm-2">

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

    </dt>

    <dd class = "col-sm-10">

        @Html.DisplayFor(model => model.Stock)

    </dd>

    <dt class = "col-sm-2">

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

    </dt>

    <dd class = "col-sm-10">

        @Html.DisplayFor(model => model.CategoriesNumber)

    </dd>

    <dt class = "col-sm-2">

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

    </dt>

    <dd class = "col-sm-10">

        @Html.DisplayFor(model => model.ImagesNumber)

    </dd>

</dl>

1. Adicionamos la vista **Details**:

@model Shooping.Data.Entities.Product

@{

    ViewData["Title"] = "Details";

}

<h1>Detalles</h1>

<div>

    <partial name="\_ProductDetails" />

</div>

<div>

    <a asp-action="AddImage" asp-route-id="@Model?.Id" class="btn btn-outline-primary">Adicionar Imagen</a>

    <a asp-action="AddCategory" asp-route-id="@Model?.Id" class="btn btn-outline-secondary">Adicionar Categoría</a>

    <a asp-action="Edit" asp-route-id="@Model?.Id" class="btn btn-outline-warning">Editar Producto</a>

    <a asp-action="Index" class="btn btn-outline-success">Regresar</a>

</div>

<link rel="stylesheet" href="https://cdn.datatables.net/1.10.19/css/jquery.dataTables.min.css" />

<hr />

<div class="row">

    <div class="col-md-6">

        <div class="panel panel-default">

            <div class="panel-heading">

                <h3 class="panel-title">Imágenes</h3>

            </div>

            <div class="panel-body">

                <table class="table table-hover table-responsive table-striped" id="ImagesTable">

                    <thead>

                        <tr>

                            <th>

                                @Html.DisplayNameFor(model => model.ProductImages.FirstOrDefault().ImageFullPath)

                            </th>

                            <th></th>

                        </tr>

                    </thead>

                    <tbody>

                        @foreach (var item in Model.ProductImages)

                        {

                            <tr>

                                <td>

                                    <img src="@item.ImageFullPath" style="width:150px;" />

                                </td>

                                <td>

                                    <a asp-action="DeleteImage" asp-route-id="@item.Id" class="btn btn-outline-danger"><i class="fa-solid fa-trash"></i></a>

                                </td>

                            </tr>

                        }

                    </tbody>

                </table>

            </div>

        </div>

    </div>

    <div class="col-md-6">

        <div class="panel panel-default">

            <div class="panel-heading">

                <h3 class="panel-title">Categorías</h3>

            </div>

            <div class="panel-body">

                <table class="table table-hover table-responsive table-striped" id="CategoriesTable">

                    <thead>

                        <tr>

                            <th>

                                @Html.DisplayNameFor(model => model.ProductCategories.FirstOrDefault().Category.Name)

                            </th>

                            <th></th>

                        </tr>

                    </thead>

                    <tbody>

                        @foreach (var item in Model.ProductCategories)

                        {

                            <tr>

                                <td>

                                    @Html.DisplayFor(modelItem => item.Category.Name)

                                </td>

                                <td>

                                    <a asp-action="DeleteCategory" asp-route-id="@item.Id" class="btn btn-outline-danger"><i class="fa-solid fa-trash"></i></a>

                                </td>

                            </tr>

                        }

                    </tbody>

                </table>

            </div>

        </div>

    </div>

</div>

@section Scripts {

    @{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

    <script src="//cdn.datatables.net/1.10.19/js/jquery.dataTables.min.js"></script>

    <script type="text/javascript">

        $(document).ready(function () {

            $('#ImagesTable').DataTable({

                "language": {

                    "url": "//cdn.datatables.net/plug-ins/9dcbecd42ad/i18n/Spanish.json"

                },

                "aLengthMenu": [

                    [25, 50, 100, 200, -1],

                    [25, 50, 100, 200, "Todos"]

                ]

            });

            $('#CategoriesTable').DataTable({

                "language": {

                    "url": "//cdn.datatables.net/plug-ins/9dcbecd42ad/i18n/Spanish.json"

                },

                "aLengthMenu": [

                    [25, 50, 100, 200, -1],

                    [25, 50, 100, 200, "Todos"]

                ]

            });

        });

    </script>

}

1. Adicionamos el modelo **AddProductImageViewModel**:

public class AddProductImageViewModel

{

    public int ProductId { get; set; }

    [Display(Name = "Foto")]

    [Required(ErrorMessage = "El campo {0} es obligatorio.")]

    public IFormFile ImageFile { get; set; }

}

1. Adicionamos los métodos **AddImage**:

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

{

    if (id == null)

    {

        return NotFound();

    }

    Product product = await \_context.Products.FindAsync(id);

    if (product == null)

    {

        return NotFound();

    }

    AddProductImageViewModel model = new()

    {

        ProductId = product.Id,

    };

    return View(model);

}

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> AddImage(AddProductImageViewModel model)

{

    if (ModelState.IsValid)

    {

        Guid imageId = Guid.Empty;

        if (model.ImageFile != null)

        {

            imageId = await \_blobHelper.UploadBlobAsync(model.ImageFile, "products");

        }

        Product product = await \_context.Products.FindAsync(model.ProductId);

        ProductImage productImage = new()

        {

            Product = product,

            ImageId = imageId,

        };

        try

        {

            \_context.Add(productImage);

            await \_context.SaveChangesAsync();

            return RedirectToAction(nameof(Details), new { Id = product.Id });

        }

        catch (Exception exception)

        {

            ModelState.AddModelError(string.Empty, exception.Message);

        }

    }

    return View(model);

}

1. Adicionamos la vista **AddImage**:

@model Shooping.Models.AddProductImageViewModel

@{

    ViewData["Title"] = "Create";

}

<h1>Crear</h1>

<h4>Imagen / Producto</h4>

<hr />

<div class="row">

    <div class="col-md-12">

        <form asp-action="AddImage" enctype="multipart/form-data">

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

            <input type="hidden" asp-for="ProductId" />

            <div class="form-group">

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

                <input asp-for="ImageFile" class="form-control" type="file" />

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

            </div>

            <div class="form-group mt-2">

                <input type="submit" value="Crear" class="btn btn-outline-primary" />

                <a asp-action="Details" asp-route-id="@Model.ProductId" class="btn btn-outline-success">Regresar</a>

            </div>

        </form>

    </div>

</div>

@section Scripts {

    @{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

}

1. Adicionamos el método **DeleteImage**:

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

{

    if (id == null)

    {

        return NotFound();

    }

    ProductImage productImage = await \_context.ProductImages

        .Include(pi => pi.Product)

        .FirstOrDefaultAsync(pi => pi.Id == id);

    if (productImage == null)

    {

        return NotFound();

    }

    await \_blobHelper.DeleteBlobAsync(productImage.ImageId, "products");

    \_context.ProductImages.Remove(productImage);

    await \_context.SaveChangesAsync();

    return RedirectToAction(nameof(Details), new { Id = productImage.Product.Id });

}

1. Adicionamos el modelo **AddCategoryProductViewModel**:

public class AddCategoryProductViewModel

{

    public int ProductId { get; set; }

    [Display(Name = "Categoría")]

    [Range(1, int.MaxValue, ErrorMessage = "Debes seleccionar una categoría.")]

    [Required(ErrorMessage = "El campo {0} es obligatorio.")]

    public int CategoryId { get; set; }

    public IEnumerable<SelectListItem> Categories { get; set; }

}

1. Adicionamos los métodos para **AddCategory**:

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

{

    if (id == null)

    {

        return NotFound();

    }

    Product product = await \_context.Products.FindAsync(id);

    if (product == null)

    {

        return NotFound();

    }

    AddCategoryProductViewModel model = new()

    {

        ProductId = product.Id,

        Categories = await \_combosHelper.GetComboCategoriesAsync(),

    };

    return View(model);

}

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> AddCategory(AddCategoryProductViewModel model)

{

    if (ModelState.IsValid)

    {

        Product product = await \_context.Products.FindAsync(model.ProductId);

        ProductCategory productCategory = new()

        {

            Category = await \_context.Categories.FindAsync(model.CategoryId),

            Product = product,

        };

        try

        {

            \_context.Add(productCategory);

            await \_context.SaveChangesAsync();

            return RedirectToAction(nameof(Details), new { Id = product.Id });

        }

        catch (Exception exception)

        {

            ModelState.AddModelError(string.Empty, exception.Message);

        }

    }

    return View(model);

}

1. Adicionamos la vista **AddCategory**:

@model Shooping.Models.AddCategoryProductViewModel

@{

    ViewData["Title"] = "Create";

}

<h1>Crear</h1>

<h4>Categoría / Producto</h4>

<hr />

<div class="row">

    <div class="col-md-12">

        <form asp-action="AddCategory" enctype="multipart/form-data">

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

            <input type="hidden" asp-for="ProductId" />

            <input type="hidden" asp-for="Categories" />

            <div class="form-group">

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

                <select asp-for="CategoryId" asp-items="Model.Categories" class="form-control"></select>

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

            </div>

            <div class="form-group mt-2">

                <input type="submit" value="Crear" class="btn btn-outline-primary" />

                <a asp-action="Details" asp-route-id="@Model.ProductId" class="btn btn-outline-success">Regresar</a>

            </div>

        </form>

    </div>

</div>

@section Scripts {

    @{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

}

1. Adicionamos el método para **DeleteCategory**:

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

{

    if (id == null)

    {

        return NotFound();

    }

    ProductCategory productCategory = await \_context.ProductCategories

        .Include(pc => pc.Product)

        .FirstOrDefaultAsync(pc => pc.Id == id);

    if (productCategory == null)

    {

        return NotFound();

    }

    \_context.ProductCategories.Remove(productCategory);

    await \_context.SaveChangesAsync();

    return RedirectToAction(nameof(Details), new { Id = productCategory.Product.Id });

}

1. Adicionamos los métodos para **Delete**:

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

{

    if (id == null)

    {

        return NotFound();

    }

    Product product = await \_context.Products

        .Include(p => p.ProductCategories)

        .Include(p => p.ProductImages)

        .FirstOrDefaultAsync(p => p.Id == id);

    if (product == null)

    {

        return NotFound();

    }

    return View(product);

}

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public async Task<IActionResult> DeleteConfirmed(int id)

{

    Product product = await \_context.Products

        .Include(p => p.ProductImages)

        .FirstOrDefaultAsync(p => p.Id == id);

    foreach (ProductImage productImage in product.ProductImages)

    {

        await \_blobHelper.DeleteBlobAsync(productImage.ImageId, "products");

    }

    \_context.Products.Remove(product);

    await \_context.SaveChangesAsync();

    return RedirectToAction(nameof(Index));

}

1. Adicionamos la vista **Delete**:

@model Shooping.Data.Entities.Product

@{

    ViewData["Title"] = "Delete";

}

<h1>Borrar</h1>

<h3>Esta seguro que querer borrar?</h3>

<div>

    <partial name="\_ProductDetails" />

    <form asp-action="Delete">

        <input type="hidden" asp-for="Id" />

        <input type="submit" value="Borrar" class="btn btn-outline-danger" />

        <a asp-action="Index" class="btn btn-outline-success">Regresar</a>

    </form>

</div>

1. Por último agreguemos algunos productos en el  **SeedDb**, y de paso le agregamos unas fotos a los usuarios. Primero creamos dentro de **root/images** las carpetas **products** y **users** y en estas agregamos las imágenes de las fotos que deseamos usar, de paso agregamos más ciudades y categoías. Empezamos inyectando el **BlobHelper** y hacemos estas modificaciones al **SeedBd**:

public async Task SeedAsync()

{

    await \_context.Database.EnsureCreatedAsync();

    await CheckCountriesAsync();

    await CheckCategoriesAsync();

    await CheckProductsAsync();

    await CheckRolesAsync();

    await CheckUserAsync("1010", "Juan", "Zuluaga", "zulu@yopmail.com", "322 311 4620", "Calle Luna Calle Sol", "JuanZuluaga.jpeg", UserType.Admin);

    await CheckUserAsync("2020", "Ledys", "Bedoya", "ledys@yopmail.com", "322 311 4620", "Calle Luna Calle Sol", "LedysBedoya.jpeg", UserType.User);

    await CheckUserAsync("3030", "Brad", "Pitt", "brad@yopmail.com", "322 311 4620", "Calle Luna Calle Sol", "Brad.jpg", UserType.User);

    await CheckUserAsync("4040", "Angelina", "Jolie", "angelina@yopmail.com", "322 311 4620", "Calle Luna Calle Sol", "Angelina.jpg", UserType.User);

}

private async Task CheckProductsAsync()

{

    if (!\_context.Products.Any())

    {

        await AddProductAsync("Adidas Barracuda", 270000M, 12F, new List<string>() { "Calzado", "Deportes" }, new List<string>() { "adidas\_barracuda.png" });

        await AddProductAsync("Adidas Superstar", 250000M, 12F, new List<string>() { "Calzado", "Deportes" }, new List<string>() { "Adidas\_superstar.png" });

        await AddProductAsync("AirPods", 1300000M, 12F, new List<string>() { "Tecnología", "Apple" }, new List<string>() { "airpos.png", "airpos2.png" });

        await AddProductAsync("Audifonos Bose", 870000M, 12F, new List<string>() { "Tecnología" }, new List<string>() { "audifonos\_bose.png" });

        await AddProductAsync("Bicicleta Ribble", 12000000M, 6F, new List<string>() { "Deportes" }, new List<string>() { "bicicleta\_ribble.png" });

        await AddProductAsync("Camisa Cuadros", 56000M, 24F, new List<string>() { "Ropa" }, new List<string>() { "camisa\_cuadros.png" });

        await AddProductAsync("Casco Bicicleta", 820000M, 12F, new List<string>() { "Deportes" }, new List<string>() { "casco\_bicicleta.png", "casco.png" });

        await AddProductAsync("iPad", 2300000M, 6F, new List<string>() { "Tecnología", "Apple" }, new List<string>() { "ipad.png" });

        await AddProductAsync("iPhone 13", 5200000M, 6F, new List<string>() { "Tecnología", "Apple" }, new List<string>() { "iphone13.png", "iphone13b.png", "iphone13c.png", "iphone13d.png" });

        await AddProductAsync("Mac Book Pro", 12100000M, 6F, new List<string>() { "Tecnología", "Apple" }, new List<string>() { "mac\_book\_pro.png" });

        await AddProductAsync("Mancuernas", 370000M, 12F, new List<string>() { "Deportes" }, new List<string>() { "mancuernas.png" });

        await AddProductAsync("Mascarilla Cara", 26000M, 100F, new List<string>() { "Belleza" }, new List<string>() { "mascarilla\_cara.png" });

        await AddProductAsync("New Balance 530", 180000M, 12F, new List<string>() { "Calzado", "Deportes" }, new List<string>() { "newbalance530.png" });

        await AddProductAsync("New Balance 565", 179000M, 12F, new List<string>() { "Calzado", "Deportes" }, new List<string>() { "newbalance565.png" });

        await AddProductAsync("Nike Air", 233000M, 12F, new List<string>() { "Calzado", "Deportes" }, new List<string>() { "nike\_air.png" });

        await AddProductAsync("Nike Zoom", 249900M, 12F, new List<string>() { "Calzado", "Deportes" }, new List<string>() { "nike\_zoom.png" });

        await AddProductAsync("Buso Adidas Mujer", 134000M, 12F, new List<string>() { "Ropa", "Deportes" }, new List<string>() { "buso\_adidas.png" });

        await AddProductAsync("Suplemento Boots Original", 15600M, 12F, new List<string>() { "Nutrición" }, new List<string>() { "Boost\_Original.png" });

        await AddProductAsync("Whey Protein", 252000M, 12F, new List<string>() { "Nutrición" }, new List<string>() { "whey\_protein.png" });

        await AddProductAsync("Arnes Mascota", 25000M, 12F, new List<string>() { "Mascotas" }, new List<string>() { "arnes\_mascota.png" });

        await AddProductAsync("Cama Mascota", 99000M, 12F, new List<string>() { "Mascotas" }, new List<string>() { "cama\_mascota.png" });

        await AddProductAsync("Teclado Gamer", 67000M, 12F, new List<string>() { "Gamer", "Tecnología" }, new List<string>() { "teclado\_gamer.png" });

        await AddProductAsync("Silla Gamer", 980000M, 12F, new List<string>() { "Gamer", "Tecnología" }, new List<string>() { "silla\_gamer.png" });

        await AddProductAsync("Mouse Gamer", 132000M, 12F, new List<string>() { "Gamer", "Tecnología" }, new List<string>() { "mouse\_gamer.png" });

        await \_context.SaveChangesAsync();

    }

}

private async Task CheckCategoriesAsync()

{

    if (!\_context.Categories.Any())

    {

        \_context.Categories.Add(new Category { Name = "Tecnología" });

        \_context.Categories.Add(new Category { Name = "Ropa" });

        \_context.Categories.Add(new Category { Name = "Gamer" });

        \_context.Categories.Add(new Category { Name = "Belleza" });

        \_context.Categories.Add(new Category { Name = "Nutrición" });

        \_context.Categories.Add(new Category { Name = "Calzado" });

        \_context.Categories.Add(new Category { Name = "Deportes" });

        \_context.Categories.Add(new Category { Name = "Mascotas" });

        \_context.Categories.Add(new Category { Name = "Apple" });

    }

    await \_context.SaveChangesAsync();

}

private async Task AddProductAsync(string name, decimal price, float stock, List<string> categories, List<string> images)

{

    Product prodcut = new()

    {

        Description = name,

        Name = name,

        Price = price,

        Stock = stock,

        ProductCategories = new List<ProductCategory>(),

        ProductImages = new List<ProductImage>()

    };

    foreach (string? category in categories)

    {

        prodcut.ProductCategories.Add(new ProductCategory { Category = await \_context.Categories.FirstOrDefaultAsync(c => c.Name == category) });

    }

    foreach (string? image in images)

    {

        Guid imageId = await \_blobHelper.UploadBlobAsync($"{Environment.CurrentDirectory}\\wwwroot\\images\\products\\{image}", "products");

        prodcut.ProductImages.Add(new ProductImage { ImageId = imageId });

    }

    \_context.Products.Add(prodcut);

}

private async Task<User> CheckUserAsync(

    string document,

    string firstName,

    string lastName,

    string email,

    string phone,

    string address,

    string image,

    UserType userType)

{

    User user = await \_userHelper.GetUserAsync(email);

    if (user == null)

    {

        Guid imageId = await \_blobHelper.UploadBlobAsync($"{Environment.CurrentDirectory}\\wwwroot\\images\\users\\{image}", "users");

        user = new User

        {

            FirstName = firstName,

            LastName = lastName,

            Email = email,

            UserName = email,

            PhoneNumber = phone,

            Address = address,

            Document = document,

            City = \_context.Cities.FirstOrDefault(),

            UserType = userType,

            ImageId = imageId

        };

        await \_userHelper.AddUserAsync(user, "123456");

        await \_userHelper.AddUserToRoleAsync(user, userType.ToString());

        string token = await \_userHelper.GenerateEmailConfirmationTokenAsync(user);

        await \_userHelper.ConfirmEmailAsync(user, token);

    }

    return user;

}

private async Task CheckCountriesAsync()

{

    if (!\_context.Countries.Any())

    {

        \_context.Countries.Add(new Country

        {

            Name = "Colombia",

            States = new List<State>()

            {

                new State()

                {

                    Name = "Antioquia",

                    Cities = new List<City>() {

                        new City() { Name = "Medellín" },

                        new City() { Name = "Itagüí" },

                        new City() { Name = "Envigado" },

                        new City() { Name = "Bello" },

                        new City() { Name = "Sabaneta" },

                        new City() { Name = "La Ceja" },

                        new City() { Name = "La Union" },

                        new City() { Name = "La Estrella" },

                        new City() { Name = "Copacabana" },

                    }

                },

                new State()

                {

                    Name = "Bogotá",

                    Cities = new List<City>() {

                        new City() { Name = "Usaquen" },

                        new City() { Name = "Champinero" },

                        new City() { Name = "Santa fe" },

                        new City() { Name = "Usme" },

                        new City() { Name = "Bosa" },

                    }

                },

                new State()

                {

                    Name = "Valle",

                    Cities = new List<City>() {

                        new City() { Name = "Calí" },

                        new City() { Name = "Jumbo" },

                        new City() { Name = "Jamundí" },

                        new City() { Name = "Chipichape" },

                        new City() { Name = "Buenaventura" },

                        new City() { Name = "Cartago" },

                        new City() { Name = "Buga" },

                        new City() { Name = "Palmira" },

                    }

                },

                new State()

                {

                    Name = "Santander",

                    Cities = new List<City>() {

                        new City() { Name = "Bucaramanga" },

                        new City() { Name = "Málaga" },

                        new City() { Name = "Barrancabermeja" },

                        new City() { Name = "Rionegro" },

                        new City() { Name = "Barichara" },

                        new City() { Name = "Zapatoca" },

                    }

                },

            }

        });

        \_context.Countries.Add(new Country

        {

            Name = "Estados Unidos",

            States = new List<State>()

            {

                new State()

                {

                    Name = "Florida",

                    Cities = new List<City>() {

                        new City() { Name = "Orlando" },

                        new City() { Name = "Miami" },

                        new City() { Name = "Tampa" },

                        new City() { Name = "Fort Lauderdale" },

                        new City() { Name = "Key West" },

                    }

                },

                new State()

                {

                    Name = "Texas",

                    Cities = new List<City>() {

                        new City() { Name = "Houston" },

                        new City() { Name = "San Antonio" },

                        new City() { Name = "Dallas" },

                        new City() { Name = "Austin" },

                        new City() { Name = "El Paso" },

                    }

                },

                new State()

                {

                    Name = "California",

                    Cities = new List<City>() {

                        new City() { Name = "Los Angeles" },

                        new City() { Name = "San Francisco" },

                        new City() { Name = "San Diego" },

                        new City() { Name = "San Bruno" },

                        new City() { Name = "Sacramento" },

                        new City() { Name = "Fresno" },

                    }

                },

            }

        });

        \_context.Countries.Add(new Country

        {

            Name = "Ecuador",

            States = new List<State>()

            {

                new State()

                {

                    Name = "Pichincha",

                    Cities = new List<City>() {

                        new City() { Name = "Quito" },

                    }

                },

                new State()

                {

                    Name = "Esmeraldas",

                    Cities = new List<City>() {

                        new City() { Name = "Esmeraldas" },

                    }

                },

            }

        });

    }

    await \_context.SaveChangesAsync();

}

Reenvío de email de confirmación

1. Creamos el **ResendTokenViewModel**:

public class ResendTokenViewModel

{

    [Display(Name = "Email")]

    [Required(ErrorMessage = "El campo {0} es obligatorio.")]

    [EmailAddress(ErrorMessage = "Debes ingresar un correo válido.")]

    public string Username { get; set; }

    public string FirstName { get; set; }

    public string LastName { get; set; }

}

1. Agregamos estos métodos al **AccountController**:

public IActionResult ResendToken()

{

    return View();

}

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> ResendToken(ResendTokenViewModel model)

{

    if (ModelState.IsValid)

    {

        User user = await \_userHelper.GetUserAsync(model.Username);

        string myToken = await \_userHelper.GenerateEmailConfirmationTokenAsync(user);

        string tokenLink = Url.Action("ConfirmEmail", "Account", new

        {

            userid = user.Id,

            token = myToken

        }, protocol: HttpContext.Request.Scheme);

        Response response = \_mailHelper.SendMail(

            $"{model.FirstName} {model.LastName}",

            model.Username,

            "Shopping - Confirmación de Email",

            $"<h1>Shopping - Confirmación de Email</h1>" +

                $"Para habilitar el usuario por favor hacer click en el siguiente link:, " +

                $"<p><a href = \"{tokenLink}\">Confirmar Email</a></p>");

        if (response.IsSuccess)

        {

            \_flashMessage.Info("Email Re-Envíado. Para poder ingresar al sistema, siga las instrucciones que han sido enviadas a su correo.");

            return RedirectToAction(nameof(Login));

        }

        \_flashMessage.Danger(response.Message);

    }

    return View(model);

}

1. Agregamos la vista **ResendToken** al **AccountController**:

@model Shooping.Models.ResendTokenViewModel

@{

    ViewData["Title"] = "Reenvío Email de Confirmación";

}

<flash dismissable="true" />

<div class="container">

    <div class="card">

        <h5 class="card-header"><i class="fa-solid fa-mail text-navy"></i> Reenvío de Email de Confirmación</h5>

        <div class="card-body">

            <div class="row">

                <div class="col-md-12">

                    <form method="post">

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

                        <div class="form-group">

                            <label asp-for="Username"></label>

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

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

                        </div>

                        <div class="form-group mt-2">

                            <button type="submit" class="btn btn-outline-primary"><i class="fa-solid fa-envelope"></i> Reenvíar Email de Confirmación</button>

                        </div>

                    </form>

                </div>

            </div>

        </div>

    </div>

</div>

1. Probamos.

Pantalla Home Básica

1. Agregar estos estilos CSS:

@charset "utf-8";

.card {

    display: flex;

    flex-direction: column;

    justify-content: space-between;

    width: 300px;

    height: 370px;

    border: 1px solid lightgray;

    box-shadow: 2px 2px 8px 4px #d3d3d3d1;

    border-radius: 15px;

    font-family: sans-serif;

    margin: 5px;

}

.card\_title {

    font-size: 24px;

    padding: 10px 10px 0 10px;

}

.card\_body {

    padding: 10px;

}

.card\_foot {

    background: #6699ff;

    border-radius: 0 0 15px 15px;

    padding: 10px;

    text-align: center;

}

.foot a {

    text-decoration: none;

    color: white;

}

.foot a:after {

    position: absolute;

    top: 0;

    right: 0;

    bottom: 0;

    left: 0;

    z-index: 1;

    content: ""

}

1. Creamos el model **ProductsHomeViewModel**:

public class ProductsHomeViewModel

{

    public Product Product1 { get; set; }

    public Product Product2 { get; set; }

    public Product Product3 { get; set; }

    public Product Product4 { get; set; }

}

1. En el **HomeController** inteyectar el DataContext y modificar el método **Index**:

        public async Task<IActionResult> Index()

        {

            List<Product>? products = await \_context.Products

                .Include(p => p.ProductImages)

                .Include(p => p.ProductCategories)

                .OrderBy(p => p.Description)

                .ToListAsync();

            List<ProductsHomeViewModel> productsHome = new() { new ProductsHomeViewModel() };

            int i = 1;

            foreach (Product? product in products)

            {

                if (i == 1)

                {

                    productsHome.LastOrDefault().Product1 = product;

                }

                if (i == 2)

                {

                    productsHome.LastOrDefault().Product2 = product;

                }

                if (i == 3)

                {

                    productsHome.LastOrDefault().Product3 = product;

                }

                if (i == 4)

                {

                    productsHome.LastOrDefault().Product4 = product;

                    productsHome.Add(new ProductsHomeViewModel());

                    i = 0;

                }

                i++;

            }

            return View(productsHome);

        }

1. Agregar la vista **Index** del **HomeController**:

@model IEnumerable<Shooping.Models.ProductsHomeViewModel>

@{

    ViewData["Title"] = "Index";

}

@foreach (var item in Model)

{

    <div class="row">

        @if (item.Product1 != null)

        {

            <div class="col-md-3">

                <div class="card">

                    <div class="card\_title">@item.Product1.Name</div>

                    <div class="card\_body">

                        <img src="@item.Product1.ImageFullPath" style="height:150px; max-width:280px;" />

                        <p class="mt-2">

                            @item.Product1.Description

                            <h4>@Html.DisplayFor(modelItem => item.Product1.Price)</h4>

                        </p>

                    </div>

                    <div class="card\_foot">

                        <a asp-action="Details" asp-route-id="@item.Product1.Id" class="btn btn-secondary">Detalles</a>

                        <a asp-action="Add" asp-route-id="@item.Product1.Id" class="btn btn-primary">Agregar</a>

                    </div>

                </div>

            </div>

        }

        @if (item.Product2 != null)

        {

            <div class="col-md-3">

                <div class="card">

                    <div class="card\_title">@item.Product2.Name</div>

                    <div class="card\_body">

                        <img src="@item.Product2.ImageFullPath" style="height:150px; max-width:280px;" />

                        <p class="mt-2">

                            @item.Product2.Description

                            <h4>@Html.DisplayFor(modelItem => item.Product2.Price)</h4>

                        </p>

                    </div>

                    <div class="card\_foot">

                        <a asp-action="Details" asp-route-id="@item.Product1.Id" class="btn btn-secondary">Detalles</a>

                        <a asp-action="Add" asp-route-id="@item.Product1.Id" class="btn btn-primary">Agregar</a>

                    </div>

                </div>

            </div>

        }

        @if (item.Product3 != null)

        {

            <div class="col-md-3">

                <div class="card">

                    <div class="card\_title">@item.Product3.Name</div>

                    <div class="card\_body">

                        <img src="@item.Product3.ImageFullPath" style="height:150px; max-width:280px;" />

                        <p class="mt-2">

                            @item.Product3.Description

                            <h4>@Html.DisplayFor(modelItem => item.Product3.Price)</h4>

                        </p>

                    </div>

                    <div class="card\_foot">

                        <a asp-action="Details" asp-route-id="@item.Product1.Id" class="btn btn-secondary">Detalles</a>

                        <a asp-action="Add" asp-route-id="@item.Product1.Id" class="btn btn-primary">Agregar</a>

                    </div>

                </div>

            </div>

        }

        @if (item.Product4 != null)

        {

            <div class="col-md-3">

                <div class="card">

                    <div class="card\_title">@item.Product4.Name</div>

                    <div class="card\_body">

                        <img src="@item.Product4.ImageFullPath" style="height:150px; max-width:280px;" />

                        <p class="mt-2">

                            @item.Product4.Description

                            <h4>@Html.DisplayFor(modelItem => item.Product4.Price)</h4>

                        </p>

                    </div>

                    <div class="card\_foot">

                        <a asp-action="Details" asp-route-id="@item.Product1.Id" class="btn btn-secondary">Detalles</a>

                        <a asp-action="Add" asp-route-id="@item.Product1.Id" class="btn btn-primary">Agregar</a>

                    </div>

                </div>

            </div>

        }

    </div>

}

Agregando productos al carro de compras

1. Creamos la entidad **TemporalSale**:

public class TemporalSale

{

    public int Id { get; set; }

    public User User { get; set; }

    public Product Product { get; set; }

    [DisplayFormat(DataFormatString = "{0:N2}")]

    [Display(Name = "Cantidad")]

    [Required(ErrorMessage = "El campo {0} es obligatorio.")]

    public float Quantity { get; set; }

    [DataType(DataType.MultilineText)]

    [Display(Name = "Comentarios")]

    public string? Remarks { get; set; }

}

1. La adicionamos en el **DataContext**:

public DbSet<TemporalSale> TemporalSales { get; set; }

1. Creamos la migración y actualizamos la base de datos.

1. Creamos el **HomeViewModel**:

public class HomeViewModel

{

    public ICollection<ProductsHomeViewModel> Products { get; set; }

    public float Quantity { get; set; }

}

1. Creamos el metodo **Add** en el **HomeController**:

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

{

    if (id == null)

    {

        return NotFound();

    }

    if (!User.Identity.IsAuthenticated)

    {

        return RedirectToAction("Login", "Account");

    }

    Product product = await \_context.Products.FindAsync(id);

    if (product == null)

    {

        return NotFound();

    }

    User user = await \_userHelper.GetUserAsync(User.Identity.Name);

    if (user == null)

    {

        return NotFound();

    }

    TemporalSale temporalSale = new()

    {

        Product = product,

        Quantity = 1,

        User = user

    };

    \_context.TemporalSales.Add(temporalSale);

    await \_context.SaveChangesAsync();

    return RedirectToAction(nameof(Index));

}

1. Inyectamos el **IUserHelper** en el **HomeController**.

1. Modificamos el método **Index** del **HomeController**:

public async Task<IActionResult> Index()

{

    List<Product>? products = await \_context.Products

        .Include(p => p.ProductImages)

        .Include(p => p.ProductCategories)

        .OrderBy(p => p.Description)

        .ToListAsync();

    List<ProductsHomeViewModel> productsHome = new() { new ProductsHomeViewModel() };

    int i = 1;

    foreach (Product? product in products)

    {

        if (i == 1)

        {

            productsHome.LastOrDefault().Product1 = product;

        }

        if (i == 2)

        {

            productsHome.LastOrDefault().Product2 = product;

        }

        if (i == 3)

        {

            productsHome.LastOrDefault().Product3 = product;

        }

        if (i == 4)

        {

            productsHome.LastOrDefault().Product4 = product;

            productsHome.Add(new ProductsHomeViewModel());

            i = 0;

        }

        i++;

    }

    HomeViewModel model = new() { Products = productsHome };

    User user = await \_userHelper.GetUserAsync(User.Identity.Name);

    if (user != null)

    {

        model.Quantity = await \_context.TemporalSales

            .Where(ts => ts.User.Id == user.Id)

            .SumAsync(ts => ts.Quantity);

    }

    return View(model);

}

1. Modificamos la vista **Index** del **HomeController**:

@model Shooping.Models.HomeViewModel

@{

    ViewData["Title"] = "Index";

}

@if(Model.Quantity > 0)

{

    <a asp-action="ShowCart" class="btn btn-outline-primary">Ver Carro de Compras (@Model.Quantity)</a>

}

@foreach (var item in Model.Products)

Detalle de productos usando un carrusel

1. Adicionamos el modelo **AddProductToCartViewModel**:

public class AddProductToCartViewModel

{

    public int Id { get; set; }

    [Display(Name = "Nombre")]

    [MaxLength(50, ErrorMessage = "El campo {0} debe tener máximo {1} caractéres.")]

    [Required(ErrorMessage = "El campo {0} es obligatorio.")]

    public string Name { get; set; }

    [DataType(DataType.MultilineText)]

    [Display(Name = "Descripción")]

    [MaxLength(500, ErrorMessage = "El campo {0} debe tener máximo {1} caractéres.")]

    public string Description { get; set; }

    [DisplayFormat(DataFormatString = "{0:C2}")]

    [Display(Name = "Precio")]

    [Required(ErrorMessage = "El campo {0} es obligatorio.")]

    public decimal Price { get; set; }

    [DisplayFormat(DataFormatString = "{0:N2}")]

    [Display(Name = "Inventario")]

    [Required(ErrorMessage = "El campo {0} es obligatorio.")]

    public float Stock { get; set; }

    [Display(Name = "Categorías")]

    public string Categories { get; set; }

    public ICollection<ProductImage> ProductImages { get; set; }

    [DisplayFormat(DataFormatString = "{0:N2}")]

    [Display(Name = "Cantidad")]

    [Range(0.0000001, float.MaxValue, ErrorMessage = "Debes de ingresar un valor mayor a cero en la cantidad.")]

    [Required(ErrorMessage = "El campo {0} es obligatorio.")]

    public float Quantity { get; set; }

    [DataType(DataType.MultilineText)]

    [Display(Name = "Comentarios")]

    public string? Remarks { get; set; }

}

1. Adiciona estos estilos al CSS:

.carousel-inner {

    width: auto;

    height: 500px;

    max-height: 500px !important;

}

.carousel-content {

    color: black;

    display: flex;

    text-align: center;

}

1. Adicionamos los métodos de **Details** al **HomeController**:

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

{

    if (id == null)

    {

        return NotFound();

    }

    Product product = await \_context.Products

        .Include(p => p.ProductImages)

        .Include(p => p.ProductCategories)

        .ThenInclude(pc => pc.Category)

        .FirstOrDefaultAsync(p => p.Id == id);

    if (product == null)

    {

        return NotFound();

    }

    string categories = string.Empty;

    foreach (ProductCategory? category in product.ProductCategories)

    {

        categories += $"{category.Category.Name}, ";

    }

    categories = categories.Substring(0, categories.Length - 2);

    AddProductToCartViewModel model = new()

    {

        Categories = categories,

        Description = product.Description,

        Id = product.Id,

        Name = product.Name,

        Price = product.Price,

        ProductImages = product.ProductImages,

        Quantity = 1,

        Stock = product.Stock,

    };

    return View(model);

}

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Details(AddProductToCartViewModel model)

{

    if (!User.Identity.IsAuthenticated)

    {

        return RedirectToAction("Login", "Account");

    }

    Product product = await \_context.Products.FindAsync(model.Id);

    if (product == null)

    {

        return NotFound();

    }

    User user = await \_userHelper.GetUserAsync(User.Identity.Name);

    if (user == null)

    {

        return NotFound();

    }

    TemporalSale temporalSale = new()

    {

        Product = product,

        Quantity = model.Quantity,

        Remarks = model.Remarks,

        User = user

    };

    \_context.TemporalSales.Add(temporalSale);

    await \_context.SaveChangesAsync();

    return RedirectToAction(nameof(Index));

}

1. Adicionamos la viasta **Details** al **HomeController**:

@model Shooping.Models.AddProductToCartViewModel

@{

    ViewData["Title"] = "Details";

}

<h1>@Model?.Name</h1>

<div class="row">

    <div class="col-md-6">

        <div id="ImagesCarousel" class="carousel slide" data-ride="carousel">

            <div class="carousel-inner">

                @{

                    var first = true;

                }

                @foreach (var item in @Model?.ProductImages)

                {

                    <div class="carousel-item @(first?Html.Raw("active"):Html.Raw(""))">

                        <img class="d-block w-100" src="@item.ImageFullPath" alt="@item.Id">

                    </div>

                    first = false;

                }

            </div>

            <a id="ImagesCarouselPrev" class="carousel-control-prev" href="#ImagesCarousel" role="button"

               data-slide="prev">

                <span class="carousel-control-prev-icon" aria-hidden="true"></span>

                <span class="sr-only btn btn-secondary">Anterior</span>

            </a>

            <a id="ImagesCarouselNext" class="carousel-control-next" href="#ImagesCarousel" role="button"

               data-slide="next">

                <span class="carousel-control-next-icon" aria-hidden="true"></span>

                <span class="sr-only btn-primary btn">Siguiente</span>

            </a>

        </div>

    </div>

    <div class="col-md-6">

        <dl class="row">

            <dt class="col-sm-4">

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

            </dt>

            <dd class="col-sm-8">

                @Html.DisplayFor(model => model.Description)

            </dd>

            <dt class="col-sm-4">

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

            </dt>

            <dd class="col-sm-8">

                @Html.DisplayFor(model => model.Price)

            </dd>

            <dt class="col-sm-4">

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

            </dt>

            <dd class="col-sm-8">

                @Html.DisplayFor(model => model.Stock)

            </dd>

            <dt class="col-sm-4">

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

            </dt>

            <dd class="col-sm-8">

                @Html.DisplayFor(model => model.Categories)

            </dd>

            <div class="col-md-12">

                <form asp-action="Details">

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

                    <input type="hidden" asp-for="Id"/>

                    <div class="form-group">

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

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

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

                    </div>

                    <div class="form-group">

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

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

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

                    </div>

                    <div class="form-group mt-2">

                        <input type="submit" value="Agregar al Carro de Compras" class="btn btn-outline-primary" />

                        <a asp-action="Index" class="btn btn-outline-success">Regresar</a>

                    </div>

                </form>

            </div>

        </dl>

    </div>

</div>

@section Scripts {

    @{

    await Html.RenderPartialAsync("\_ValidationScriptsPartial");

}

<script type="text/javascript">

    $(document).ready(function () {

        $("#ImagesCarousel").carousel();

        $("#ImagesCarouselPrev").click(function(){

            $("#ImagesCarousel").carousel("prev");

        });

        $("#ImagesCarouselNext").click(function(){

            $("#ImagesCarousel").carousel("next");

        });

    });

</script>

}

1. Probamos.

Mostrando y modificando el carro de compras

1. Agregamos esta propiedad al **TemporalSale**:

[DisplayFormat(DataFormatString = "{0:C2}")]

[Display(Name = "Valor")]

public decimal Value => Product == null ? 0: (decimal)Quantity \* Product.Price;

1. Agregamos el modelo **ShowCartViewModel**:

public class ShowCartViewModel

{

    public User User { get; set; }

    [DataType(DataType.MultilineText)]

    [Display(Name = "Comentarios")]

    public string? Remarks { get; set; }

    public ICollection<TemporalSale> TemporalSales { get; set; }

    [DisplayFormat(DataFormatString = "{0:N2}")]

    [Display(Name = "Cantidad")]

    public float Quantity => TemporalSales == null ? 0 : TemporalSales.Sum(ts => ts.Quantity);

    [DisplayFormat(DataFormatString = "{0:C2}")]

    [Display(Name = "Valor")]

    public decimal Value => TemporalSales == null ? 0 : TemporalSales.Sum(ts => ts.Value);

}

1. Agregamos el método **ShowCart** al **HomeController**:

[Authorize]

public async Task<IActionResult> ShowCart()

{

    User user = await \_userHelper.GetUserAsync(User.Identity.Name);

    if (user == null)

    {

        return NotFound();

    }

    List<TemporalSale>? temporalSales = await \_context.TemporalSales

        .Include(ts => ts.Product)

        .ThenInclude(p => p.ProductImages)

        .Where(ts => ts.User.Id == user.Id)

        .ToListAsync();

    ShowCartViewModel model = new()

    {

        User = user,

        TemporalSales = temporalSales,

    };

    return View(model);

}

1. Agregamos la vista **ShowCart** al **HomeController**:

@model Shooping.Models.ShowCartViewModel

@{

    ViewData["Title"] = "Cart";

}

<link rel="stylesheet" href="https://cdn.datatables.net/1.10.19/css/jquery.dataTables.min.css" />

<div class="row">

    <div class="col-md-4">

        <dl class="row">

            <dt class="col-sm-4">

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

            </dt>

            <dd class="col-sm-8">

                <h3>@Html.DisplayFor(model => model.Quantity)</h3>

            </dd>

            <dt class="col-sm-4">

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

            </dt>

            <dd class="col-sm-8">

                <h3>@Html.DisplayFor(model => model.Value)</h3>

            </dd>

        </dl>

    </div>

    <div class="col-md-8">

            <div class="col-md-12">

                <form asp-action="ShowCart">

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

                    <div class="form-group">

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

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

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

                    </div>

                    <div class="form-group mt-2">

                        <input type="submit" value="Confirmar Pedido" class="btn btn-outline-primary" />

                        <a asp-action="Index" class="btn btn-outline-success">Regresar</a>

                    </div>

                </form>

            </div>

    </div>

</div>

<div class="row">

    <div class="col-md-12">

        <div class="panel panel-default">

            <div class="panel-heading">

                <h3 class="panel-title">Carro de Compras</h3>

            </div>

            <div class="panel-body">

                <table class="table table-hover table-responsive table-striped" id="MyTable">

                    <thead>

                        <tr>

                            <th>

                                @Html.DisplayNameFor(model => model.TemporalSales.FirstOrDefault().Product.ImageFullPath)

                            </th>

                            <th>

                                @Html.DisplayNameFor(model => model.TemporalSales.FirstOrDefault().Product.Name)

                            </th>

                            <th>

                                @Html.DisplayNameFor(model => model.TemporalSales.FirstOrDefault().Remarks)

                            </th>

                            <th>

                                @Html.DisplayNameFor(model => model.TemporalSales.FirstOrDefault().Product.Price)

                            </th>

                            <th></th>

                            <th>

                                @Html.DisplayNameFor(model => model.TemporalSales.FirstOrDefault().Quantity)

                            </th>

                            <th></th>

                            <th>

                                @Html.DisplayNameFor(model => model.TemporalSales.FirstOrDefault().Value)

                            </th>

                            <th></th>

                        </tr>

                    </thead>

                    <tbody>

                        @foreach (var item in Model.TemporalSales)

                        {

                            <tr>

                                <td>

                                    <img src="@item.Product.ImageFullPath" style="width:100px;" />

                                </td>

                                <td>

                                    @Html.DisplayFor(modelItem => item.Product.Name)

                                </td>

                                <td>

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

                                </td>

                                <td>

                                    @Html.DisplayFor(modelItem => item.Product.Price)

                                </td>

                                <td>

                                    <a asp-action="DecreaseQuantity" asp-route-id="@item.Id" class="btn btn-secondary">-</a>

                                </td>

                                <td>

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

                                </td>

                                <td>

                                    <a asp-action="IncreaseQuantity" asp-route-id="@item.Id" class="btn btn-primary">+</a>

                                </td>

                                <td>

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

                                </td>

                                <td>

                                    <a asp-action="Edit" asp-route-id="@item.Id" class="btn btn-outline-warning">Editar</a>

                                    <a asp-action="Delete" asp-route-id="@item.Id" class="btn btn-outline-danger">Borrar</a>

                                </td>

                            </tr>

                        }

                    </tbody>

                </table>

            </div>

        </div>

    </div>

</div>

@section Scripts {

    @{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

    <script src="//cdn.datatables.net/1.10.19/js/jquery.dataTables.min.js"></script>

    <script type="text/javascript">

        $(document).ready(function () {

            $('#MyTable').DataTable({

                "language": {

                    "url": "//cdn.datatables.net/plug-ins/9dcbecd42ad/i18n/Spanish.json"

                },

                "aLengthMenu": [

                    [25, 50, 100, 200, -1],

                    [25, 50, 100, 200, "Todos"]

                ]

            });

        });

    </script>

}

1. Agregamos estos métodos al **HomeController**:

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

{

    if (id == null)

    {

        return NotFound();

    }

    TemporalSale temporalSale = await \_context.TemporalSales.FindAsync(id);

    if (temporalSale == null)

    {

        return NotFound();

    }

    if (temporalSale.Quantity > 1)

    {

        temporalSale.Quantity--;

        \_context.TemporalSales.Update(temporalSale);

        await \_context.SaveChangesAsync();

    }

    return RedirectToAction(nameof(ShowCart));

}

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

{

    if (id == null)

    {

        return NotFound();

    }

    TemporalSale temporalSale = await \_context.TemporalSales.FindAsync(id);

    if (temporalSale == null)

    {

        return NotFound();

    }

    temporalSale.Quantity++;

    \_context.TemporalSales.Update(temporalSale);

    await \_context.SaveChangesAsync();

    return RedirectToAction(nameof(ShowCart));

}

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

{

    if (id == null)

    {

        return NotFound();

    }

    TemporalSale temporalSale = await \_context.TemporalSales.FindAsync(id);

    if (temporalSale == null)

    {

        return NotFound();

    }

    \_context.TemporalSales.Remove(temporalSale);

    await \_context.SaveChangesAsync();

    return RedirectToAction(nameof(ShowCart));

}

1. Agregamos el modelo **EditTemporalSale**:

public class EditTemporalSale

{

    public int Id { get; set; }

    [DataType(DataType.MultilineText)]

    [Display(Name = "Comentarios")]

    public string? Remarks { get; set; }

    [DisplayFormat(DataFormatString = "{0:N2}")]

    [Display(Name = "Cantidad")]

    [Range(0.0000001, float.MaxValue, ErrorMessage = "Debes de ingresar un valor mayor a cero en la cantidad.")]

    [Required(ErrorMessage = "El campo {0} es obligatorio.")]

    public float Quantity { get; set; }

}

1. Agregamos los métodos de **Edit** al **HomeController**:

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

{

    if (id == null)

    {

        return NotFound();

    }

    TemporalSale temporalSale = await \_context.TemporalSales.FindAsync(id);

    if (temporalSale == null)

    {

        return NotFound();

    }

    EditTemporalSale model = new()

    {

        Id = temporalSale.Id,

        Quantity = temporalSale.Quantity,

        Remarks = temporalSale.Remarks,

    };

    return View(model);

}

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Edit(int id, EditTemporalSale model)

{

    if (id != model.Id)

    {

        return NotFound();

    }

    if (ModelState.IsValid)

    {

        try

        {

            TemporalSale temporalSale = await \_context.TemporalSales.FindAsync(id);

            temporalSale.Quantity = model.Quantity;

            temporalSale.Remarks = model.Remarks;

            \_context.Update(temporalSale);

            await \_context.SaveChangesAsync();

        }

        catch (Exception exception)

        {

            ModelState.AddModelError(string.Empty, exception.Message);

            return View(model);

        }

        return RedirectToAction(nameof(ShowCart));

    }

    return View(model);

}

1. Agregamos la vista **Edit** al **HomeController**:

@model Shooping.Models.EditTemporalSale

@{

    ViewData["Title"] = "Edit";

}

<h1>Editar</h1>

<h4>Carro de Compras</h4>

<hr />

<div class="row">

    <div class="col-md-12">

        <form asp-action="Edit">

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

            <input type="hidden" asp-for="Id" />

            <div class="form-group">

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

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

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

            </div>

            <div class="form-group">

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

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

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

            </div>

            <div class="form-group mt-2">

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

                <a asp-action="ShowCart" class="btn btn-outline-success">Regresar</a>

            </div>

        </form>

    </div>

</div>

@section Scripts {

    @{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

}

1. Probamos.

Procesando el pedido

1. Agregamos la enumeración **OrderStatus**:

public enum OrderStatus

{

    Nuevo,

    Despachado,

    Enviado,

    Confirmado,

    Cancelado

}

1. Agregamos la entidad **Sale**:

public class Sale

{

    public int Id { get; set; }

    [DisplayFormat(DataFormatString = "{0:yyyy/MM/dd hh:mm tt}")]

    [Display(Name = "Inventario")]

    [Required(ErrorMessage = "El campo {0} es obligatorio.")]

    public DateTime Date { get; set; }

    public User User { get; set; }

    [DataType(DataType.MultilineText)]

    [Display(Name = "Comentarios")]

    public string? Remarks { get; set; }

    public OrderStatus OrderStatus { get; set; }

    public ICollection<SaleDetail> SaleDetails { get; set; }

    [DisplayFormat(DataFormatString = "{0:N0}")]

    [Display(Name = "Líneas")]

    public int Lines => SaleDetails == null ? 0 : SaleDetails.Count;

    [DisplayFormat(DataFormatString = "{0:N2}")]

    [Display(Name = "Cantidad")]

    public float Quantity => SaleDetails == null ? 0 : SaleDetails.Sum(sd => sd.Quantity);

    [DisplayFormat(DataFormatString = "{0:C2}")]

    [Display(Name = "Valor")]

    public decimal Value => SaleDetails == null ? 0 : SaleDetails.Sum(sd => sd.Value);

}

1. Agregamos la entidad **SaleDetail**:

public class SaleDetail

{

    public int Id { get; set; }

    public Sale Sale { get; set; }

    [DataType(DataType.MultilineText)]

    [Display(Name = "Comentarios")]

    public string? Remarks { get; set; }

    public Product Product { get; set; }

    [DisplayFormat(DataFormatString = "{0:N2}")]

    [Display(Name = "Cantidad")]

    [Required(ErrorMessage = "El campo {0} es obligatorio.")]

    public float Quantity { get; set; }

    [DisplayFormat(DataFormatString = "{0:C2}")]

    [Display(Name = "Valor")]

    public decimal Value => Product == null ? 0 : (decimal)Quantity \* Product.Price;

}

1. Modificamos la entidad **Product**:

public ICollection<SaleDetail> SaleDetails { get; set; }

1. Modificamos la entidad **User**:

public ICollection<Sale> Sales { get; set; }

1. Agregamos la migración y actualizamos la base de datos.

1. Creamos el **IOrdersHelper**:

public interface IOrdersHelper

{

    Task<Response> ProcessOrderAsync(ShowCartViewModel model);

}

1. Creamos el **OrdersHelper**:

public class OrdersHelper : IOrdersHelper

{

    private readonly DataContext \_context;

    public OrdersHelper(DataContext context)

    {

        \_context = context;

    }

    public async Task<Response> ProcessOrderAsync(ShowCartViewModel model)

    {

        Response response = await CheckInventoryAsync(model);

        if (!response.IsSuccess)

        {

            return response;

        }

        Sale sale = new()

        {

            Date = DateTime.UtcNow,

            User = model.User,

            Remarks = model.Remarks,

            SaleDetails = new List<SaleDetail>(),

            OrderStatus = OrderStatus.New

        };

        foreach (TemporalSale? item in model.TemporalSales)

        {

            sale.SaleDetails.Add(new SaleDetail

            {

                Product = item.Product,

                Quantity = item.Quantity,

                Remarks = item.Remarks,

            });

            Product product = await \_context.Products.FindAsync(item.Product.Id);

            if (product != null)

            {

                product.Stock -= item.Quantity;

                \_context.Products.Update(product);

            }

            \_context.TemporalSales.Remove(item);

        }

        \_context.Sales.Add(sale);

        await \_context.SaveChangesAsync();

        return response;

    }

    private async Task<Response> CheckInventoryAsync(ShowCartViewModel model)

    {

        Response response = new() { IsSuccess = true };

        foreach (TemporalSale? item in model.TemporalSales)

        {

            Product product = await \_context.Products.FindAsync(item.Product.Id);

            if (product == null)

            {

                response.IsSuccess = false;

                response.Message = $"El producto {item.Product.Name}, ya no está disponible";

                return response;

            }

            if (product.Stock < item.Quantity)

            {

                response.IsSuccess = false;

                response.Message = $"Lo sentimos no tenemos existencias suficientes del producto {item.Product.Name}, para tomar su pedido. Por favor disminuir la cantidad o sustituirlo por otro.";

                return response;

            }

        }

        return response;

    }

}

1. Lo inyectamos en el **Program**:

builder.Services.AddScoped<IOrdersHelper, OrdersHelper>();

1. Creamos el método **OrderSuccess** en el **HomeController**:

[Authorize]

public IActionResult OrderSuccess()

{

    return View();

}

1. Creamos la vista **OrderSuccess** en el **HomeController** (primero adicionamos la imagén a los recursos estáticos):

@{

    ViewData["Title"] = "Order Success";

}

<div class="row">

    <div class="col-md-4 offset-4">

        <img src="~/images/Shopping.png" style="width:400px;"/>

        <h2>¡Gracias!</h2>

        <h4>Su pedido fue registrado en nuestro sistema, pronto uno de nuestros asesores se comunicará con usted.</h4>

        <a asp-action="Index" class="mt-2 btn btn-outline-success">Inicio</a>

    </div>

</div>

1. Creamos el método POST **ShowCart** en el **HomeController**:

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> ShowCart(ShowCartViewModel model)

{

    User user = await \_userHelper.GetUserAsync(User.Identity.Name);

    if (user == null)

    {

        return NotFound();

    }

    model.User = user;

    model.TemporalSales = await \_context.TemporalSales

        .Include(ts => ts.Product)

        .ThenInclude(p => p.ProductImages)

        .Where(ts => ts.User.Id == user.Id)

        .ToListAsync();

    Response response = await \_ordersHelper.ProcessOrderAsync(model);

    if (response.IsSuccess)

    {

        return RedirectToAction(nameof(OrderSuccess));

    }

    ModelState.AddModelError(string.Empty, response.Message);

    return View(model);

}

1. Modificamos el **Index** del **HomeController** para que solo muestre los productos que tienen Stock disponible:

List<Product>? products = await \_context.Products

    .Include(p => p.ProductImages)

    .Include(p => p.ProductCategories)

    .Where(p => p.Stock > 0)

    .OrderBy(p => p.Description)

    .ToListAsync();

1. Probamos.

Administrando los pedidos

1. Creamos el **OrdersController**:

[Authorize(Roles = "Admin")]

public class OrdersController : Controller

{

    private readonly DataContext \_context;

    public OrdersController(DataContext context)

    {

        \_context = context;

    }

    public async Task<IActionResult> Index()

    {

        return View(await \_context.Sales

            .Include(s => s.User)

            .Include(s => s.SaleDetails)

            .ThenInclude(sd => sd.Product)

            .ToListAsync());

    }

}

1. Adicionamos la vista **Index** en el **OrdersController**:

@model IEnumerable<Shooping.Data.Entities.Sale>

@{

    ViewData["Title"] = "Index";

}

<link rel="stylesheet" href="https://cdn.datatables.net/1.10.19/css/jquery.dataTables.min.css" />

<div class="row">

    <div class="col-md-12">

        <div class="panel panel-default">

            <div class="panel-heading">

                <h3 class="panel-title">Pedidos</h3>

            </div>

            <div class="panel-body">

                <table class="table table-hover table-responsive table-striped" id="MyTable">

                    <thead>

                        <tr>

                            <th>

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

                            </th>

                            <th>

                                @Html.DisplayNameFor(model => model.User.FullName)

                            </th>

                            <th>

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

                            </th>

                            <th>

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

                            </th>

                            <th>

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

                            </th>

                            <th>

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

                            </th>

                            <th>

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

                            </th>

                            <th></th>

                        </tr>

                    </thead>

                    <tbody>

                        @foreach (var item in Model)

                        {

                            <tr>

                                <td>

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

                                </td>

                                <td>

                                    @Html.DisplayFor(modelItem => item.User.FullName)

                                </td>

                                <td>

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

                                </td>

                                <td>

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

                                </td>

                                <td>

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

                                </td>

                                <td>

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

                                </td>

                                <td>

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

                                </td>

                                <td>

                                    <a asp-action="Details" asp-route-id="@item.Id" class="btn btn-outline-info"><i class="fa-solid fa-circle-info"></i></a>

                                </td>

                            </tr>

                        }

                    </tbody>

                </table>

            </div>

        </div>

    </div>

</div>

@section Scripts {

    @{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

    <script src="//cdn.datatables.net/1.10.19/js/jquery.dataTables.min.js"></script>

    <script type="text/javascript">

        $(document).ready(function () {

            $('#MyTable').DataTable({

                "language": {

                    "url": "//cdn.datatables.net/plug-ins/9dcbecd42ad/i18n/Spanish.json"

                },

                "aLengthMenu": [

                    [25, 50, 100, 200, -1],

                    [25, 50, 100, 200, "Todos"]

                ]

            });

        });

    </script>

}

1. Adicionamos el método **Details** en el **OrdersController**:

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

{

    if (id == null)

    {

        return NotFound();

    }

    Sale sale = await \_context.Sales

        .Include(s => s.User)

        .Include(s => s.SaleDetails)

        .ThenInclude(sd => sd.Product)

        .ThenInclude(p => p.ProductImages)

        .FirstOrDefaultAsync(s => s.Id == id);

    if (sale == null)

    {

        return NotFound();

    }

    return View(sale);

}

1. Adicionamos la vista parcial **\_OrderDetails** en el **OrdersController**:

@model Shooping.Data.Entities.Sale

<h4>Pedido</h4>

<hr />

<dl class="row">

    <dt class = "col-sm-2">

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

    </dt>

    <dd class = "col-sm-10">

        @Html.DisplayFor(model => model.Date)

    </dd>

    <dt class = "col-sm-2">

        @Html.DisplayNameFor(model => model.User.FullName)

    </dt>

    <dd class = "col-sm-10">

        @Html.DisplayFor(model => model.User.FullName)

    </dd>

    <dt class = "col-sm-2">

        @Html.DisplayNameFor(model => model.User.Email)

    </dt>

    <dd class = "col-sm-10">

        @Html.DisplayFor(model => model.User.Email)

    </dd>

    <dt class = "col-sm-2">

        @Html.DisplayNameFor(model => model.User.PhoneNumber)

    </dt>

    <dd class = "col-sm-10">

        @Html.DisplayFor(model => model.User.PhoneNumber)

    </dd>

    <dt class = "col-sm-2">

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

    </dt>

    <dd class = "col-sm-10">

        @Html.DisplayFor(model => model.Remarks)

    </dd>

    <dt class = "col-sm-2">

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

    </dt>

    <dd class = "col-sm-10">

        @Html.DisplayFor(model => model.OrderStatus)

    </dd>

    <dt class = "col-sm-2">

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

    </dt>

    <dd class = "col-sm-10">

        @Html.DisplayFor(model => model.Lines)

    </dd>

    <dt class = "col-sm-2">

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

    </dt>

    <dd class = "col-sm-10">

        @Html.DisplayFor(model => model.Quantity)

    </dd>

    <dt class = "col-sm-2">

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

    </dt>

    <dd class = "col-sm-10">

        @Html.DisplayFor(model => model.Value)

    </dd>

</dl>

1. Adicionamos la vista **Details** en el **OrdersController**:

@model Shooping.Data.Entities.Sale

@{

    ViewData["Title"] = "Details";

}

<h1>Detalles</h1>

<div>

    <partial name="\_OrderDetails" />

</div>

<div>

    <a asp-action="Dispatch" asp-route-id="@Model?.Id" class="btn btn-outline-primary">Despachar</a>

    <a asp-action="Send" asp-route-id="@Model?.Id" class="btn btn-outline-secondary">Envíar</a>

    <a asp-action="Confirm" asp-route-id="@Model?.Id" class="btn btn-outline-warning">Confirmar</a>

    <a asp-action="Cancel" asp-route-id="@Model?.Id" class="btn btn-outline-danger">Cancelar</a>

    <a asp-action="Index" class="btn btn-outline-success">Regresar</a>

</div>

<link rel="stylesheet" href="https://cdn.datatables.net/1.10.19/css/jquery.dataTables.min.css" />

<hr />

<div class="row">

    <div class="col-md-12">

        <div class="panel panel-default">

            <div class="panel-heading">

                <h3 class="panel-title">Productos</h3>

            </div>

            <div class="panel-body">

                <table class="table table-hover table-responsive table-striped" id="MyTable">

                    <thead>

                        <tr>

                            <th>

                                @Html.DisplayNameFor(model => model.SaleDetails.FirstOrDefault().Product.Name)

                            </th>

                            <th>

                                @Html.DisplayNameFor(model => model.SaleDetails.FirstOrDefault().Remarks)

                            </th>

                            <th>

                                @Html.DisplayNameFor(model => model.SaleDetails.FirstOrDefault().Product.Price)

                            </th>

                            <th>

                                @Html.DisplayNameFor(model => model.SaleDetails.FirstOrDefault().Quantity)

                            </th>

                            <th>

                                @Html.DisplayNameFor(model => model.SaleDetails.FirstOrDefault().Value)

                            </th>

                            <th>

                                @Html.DisplayNameFor(model => model.SaleDetails.FirstOrDefault().Product.ImageFullPath)

                            </th>

                        </tr>

                    </thead>

                    <tbody>

                        @foreach (var item in Model.SaleDetails)

                        {

                            <tr>

                                <td>

                                    @Html.DisplayFor(modelItem => item.Product.Name)

                                </td>

                                <td>

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

                                </td>

                                <td>

                                    @Html.DisplayFor(modelItem => item.Product.Price)

                                </td>

                                <td>

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

                                </td>

                                <td>

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

                                </td>

                                <td>

                                    <img src="@item.Product.ImageFullPath" style="width:100px;" />

                                </td>

                            </tr>

                        }

                    </tbody>

                </table>

            </div>

        </div>

    </div>

</div>

@section Scripts {

    @{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

    <script src="//cdn.datatables.net/1.10.19/js/jquery.dataTables.min.js"></script>

    <script type="text/javascript">

        $(document).ready(function () {

            $('#MyTable').DataTable({

                "language": {

                    "url": "//cdn.datatables.net/plug-ins/9dcbecd42ad/i18n/Spanish.json"

                },

                "aLengthMenu": [

                    [25, 50, 100, 200, -1],

                    [25, 50, 100, 200, "Todos"]

                ]

            });

        });

    </script>

}

1. Probamos lo que llevamos hasta el momento.

Colocar mensajes tipo Toast y cambiar el estado de los pedidos

1. Instalamos el siguiente paquete:

PM> Install-Package Vereyon.Web.FlashMessage

1. Lo registramos en el **StartUp**:

services.AddFlashMessage();

1. Lo registramos en el **\_ViewImports**:

@addTagHelper \*, Vereyon.Web.FlashMessage

1. Lo inyectamos en el controlador donde queramos el mensaje, para el ejemplo en el **OrdersController**:

IFlashMessage flashMessage

1. Adicionamos el método **Dispatch** en el **OrdersController**:

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

{

    if (id == null)

    {

        return NotFound();

    }

    Sale sale = await \_context.Sales.FindAsync(id);

    if (sale == null)

    {

        return NotFound();

    }

    if (sale.OrderStatus != OrderStatus.Nuevo)

    {

        \_flashMessage.Danger("Solo se pueden despachar pedidos que estén en estado 'nuevo'.");

    }

    else

    {

        sale.OrderStatus = OrderStatus.Despachado;

        \_context.Sales.Update(sale);

        await \_context.SaveChangesAsync();

        \_flashMessage.Confirmation("El estado del pedido ha sido cambiado a 'despachado'.");

    }

    return RedirectToAction(nameof(Details), new { Id = sale.Id });

}

1. En la vista donde se mostrará el toast, en este caso la **Details** de **OrdersController**, adicionamos lo siguiente:

<flash dismissable="true" />

1. Probamos

Tomado de: <https://github.com/Vereyon/FlashMessage>

1. Completamos el resto de métodos para cambiar el estado de los pedidos, excepto cancelar que es un caso especial.

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

{

    if (id == null)

    {

        return NotFound();

    }

    Sale sale = await \_context.Sales.FindAsync(id);

    if (sale == null)

    {

        return NotFound();

    }

    if (sale.OrderStatus != OrderStatus.Despachado)

    {

        \_flashMessage.Danger("Solo se pueden enviar pedidos que estén en estado 'despachado'.");

    }

    else

    {

        sale.OrderStatus = OrderStatus.Enviado;

        \_context.Sales.Update(sale);

        await \_context.SaveChangesAsync();

        \_flashMessage.Confirmation("El estado del pedido ha sido cambiado a 'enviado'.");

    }

    return RedirectToAction(nameof(Details), new { Id = sale.Id });

}

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

{

    if (id == null)

    {

        return NotFound();

    }

    Sale sale = await \_context.Sales.FindAsync(id);

    if (sale == null)

    {

        return NotFound();

    }

    if (sale.OrderStatus != OrderStatus.Enviado)

    {

        \_flashMessage.Danger("Solo se pueden confirmar pedidos que estén en estado 'enviado'.");

    }

    else

    {

        sale.OrderStatus = OrderStatus.Confirmado;

        \_context.Sales.Update(sale);

        await \_context.SaveChangesAsync();

        \_flashMessage.Confirmation("El estado del pedido ha sido cambiado a 'confirmado'.");

    }

    return RedirectToAction(nameof(Details), new { Id = sale.Id });

}

1. Modificamos el **IOrdersHelper**:

Task<Response> CancelOrderAsync(int id);

1. Hacemos la implementación en el **OrdersHelper**:

public async Task<Response> CancelOrderAsync(int id)

{

    Sale sale = await \_context.Sales

        .Include(s => s.SaleDetails)

        .ThenInclude(sd => sd.Product)

        .FirstOrDefaultAsync(s => s.Id == id);

    foreach (SaleDetail saleDetail in sale.SaleDetails)

    {

        Product product = await \_context.Products.FindAsync(saleDetail.Product.Id);

        if (product != null)

        {

            product.Stock += saleDetail.Quantity;

        }

    }

    sale.OrderStatus = OrderStatus.Cancelado;

    await \_context.SaveChangesAsync();

    return new Response { IsSuccess = true };

}

1. Inyectamos el **IOrdersHelper** en el **OrdersController**.

1. Adicionamos el método **Cancel** en el **OrdersController**.

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

{

    if (id == null)

    {

        return NotFound();

    }

    Sale sale = await \_context.Sales.FindAsync(id);

    if (sale == null)

    {

        return NotFound();

    }

    if (sale.OrderStatus == OrderStatus.Cancelado)

    {

        \_flashMessage.Danger("No se puede cancelar un pedido que esté en estado 'cancelado'.");

    }

    else

    {

        await \_ordersHelper.CancelOrderAsync(sale.Id);

        \_flashMessage.Confirmation("El estado del pedido ha sido cambiado a 'cancelado'.");

    }

    return RedirectToAction(nameof(Details), new { Id = sale.Id });

}

1. Probamos.

1. Ahora colocamos algunos mensajes tipo toast para mejorar la experiencia del usuario. Empecemos cuando registramos el usuario. En el **AccountController** inyectamos el **IFlashMessage**, luego modificamos el POST del **Register**:

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Register(AddUserViewModel model)

{

    if (ModelState.IsValid)

    {

        Guid imageId = Guid.Empty;

        if (model.ImageFile != null)

        {

            imageId = await \_blobHelper.UploadBlobAsync(model.ImageFile, "users");

        }

        User user = await \_userHelper.AddUserAsync(model, imageId);

        if (user == null)

        {

            \_flashMessage.Danger("Este correo ya está siendo usado.");

            model.Countries = await \_combosHelper.GetComboCountriesAsync();

            model.States = await \_combosHelper.GetComboStatesAsync(model.CountryId);

            model.Cities = await \_combosHelper.GetComboCitiesAsync(model.StateId);

            return View(model);

        }

        string myToken = await \_userHelper.GenerateEmailConfirmationTokenAsync(user);

        string tokenLink = Url.Action("ConfirmEmail", "Account", new

        {

            userid = user.Id,

            token = myToken

        }, protocol: HttpContext.Request.Scheme);

        Response response = \_mailHelper.SendMail(

            $"{model.FirstName} {model.LastName}",

            model.Username,

            "Shopping - Confirmación de Email",

            $"<h1>Shopping - Confirmación de Email</h1>" +

                $"Para habilitar el usuario por favor hacer clicn en el siguiente link:, " +

                $"<p><a href = \"{tokenLink}\">Confirmar Email</a></p>");

            if (response.IsSuccess)

            {

                \_flashMessage.Info("Usuario registrado. Para poder ingresar al sistema, siga las instrucciones que han sido enviadas a su correo.");

                return RedirectToAction(nameof(Login));

            }

        ModelState.AddModelError(string.Empty, response.Message);

    }

    model.Countries = await \_combosHelper.GetComboCountriesAsync();

    model.States = await \_combosHelper.GetComboStatesAsync(model.CountryId);

    model.Cities = await \_combosHelper.GetComboCitiesAsync(model.StateId);

    return View(model);

}

1. Agregamos esta línea a la vista **Login** y la vista **Register**:

<flash dismissable="true" />

1. Ahora cuando el usuario recupera la contraseña. Modifiquemos el método **RecoverPassword**:

[HttpPost]

public async Task<IActionResult> RecoverPassword(RecoverPasswordViewModel model)

{

    if (ModelState.IsValid)

    {

        User user = await \_userHelper.GetUserAsync(model.Email);

        if (user == null)

        {

            \_flashMessage.Danger("El email no corresponde a ningún usuario registrado.");

            return View(model);

        }

        string myToken = await \_userHelper.GeneratePasswordResetTokenAsync(user);

        string link = Url.Action(

            "ResetPassword",

            "Account",

            new { token = myToken }, protocol: HttpContext.Request.Scheme);

        \_mailHelper.SendMail(

            $"{user.FullName}",

            model.Email,

            "Shopping - Recuperación de Contraseña",

            $"<h1>Shopping - Recuperación de Contraseña</h1>" +

            $"Para recuperar la contraseña haga click en el siguiente enlace:" +

            $"<p><a href = \"{link}\">Reset Password</a></p>");

        \_flashMessage.Info("Las instrucciones para recuperar la contraseña han sido enviadas a su correo.");

        return RedirectToAction(nameof(Login));

    }

    return View(model);

}

1. Agregamos esta línea a la vista **RecoverPassword** y de paso centramos el título:

<flash dismissable="true" />

<div class="row">

    <div class="col-md-4 offset-md-4">

        <h2>Recuperación de Constraseña</h2>

        <form method="post">

1. Modificamos el método **ResetPassword**:

[HttpPost]

public async Task<IActionResult> ResetPassword(ResetPasswordViewModel model)

{

    User user = await \_userHelper.GetUserAsync(model.UserName);

    if (user != null)

    {

        IdentityResult result = await \_userHelper.ResetPasswordAsync(user, model.Token, model.Password);

        if (result.Succeeded)

        {

            \_flashMessage.Info("Contraseña cambiada con éxito.");

            return RedirectToAction(nameof(Login));

        }

        \_flashMessage.Danger("Error cambiando la contraseña.");

        return View(model);

    }

    \_flashMessage.Danger("Usuario no encontrado.");

    return View(model);

}

1. Agregamos esta línea a la vista **ResetPassword** y de paso centramos el título:

<flash dismissable="true" />

<div class="row">

    <div class="col-md-4 offset-md-4">

        <h1>Resetea tu Contraseña</h1>

        <form method="post">

1. Probamos.

1. Modificamos el método **Login**:

[HttpPost]

public async Task<IActionResult> Login(LoginViewModel model)

{

    if (ModelState.IsValid)

    {

        SignInResult result = await \_userHelper.LoginAsync(model);

        if (result.Succeeded)

        {

            if (Request.Query.Keys.Contains("ReturnUrl"))

            {

                return Redirect(Request.Query["ReturnUrl"].First());

            }

            return RedirectToAction("Index", "Home");

        }

        if (result.IsLockedOut)

        {

            \_flashMessage.Danger("Ha superado el máximo número de intentos, su cuenta está bloqueada, intente de nuevo en 5 minutos.");

        }

        else if(result.IsNotAllowed)

        {

            \_flashMessage.Danger("El usuario no ha sido habilitado, debes de seguir las instrucciones enviadas al correo para poder habilitarlo.");

        }

        else

        {

            \_flashMessage.Danger("Email o contraseña incorrectos.");

        }

    }

    return View(model);

}

1. En resumen, busca en todo tu proyecto **ViewBag.Message** y **AddModelError** y reemplazalo por el **\_flashMessage**. No sobra colocar el **<flash dismissable="true" />** a todas las vistas.

1. Probamos.

Ver el estado de “Mis” Pedidos

1. Cambiamos los permisos del controlador **OrdersController** y los agregamos a nivel de método.

1. Agregamos el método **MyOrders** al controlador **OrdersController**:

[Authorize(Roles = "User")]

public async Task<IActionResult> MyOrders()

{

    return View(await \_context.Sales

        .Include(s => s.User)

        .Include(s => s.SaleDetails)

        .ThenInclude(sd => sd.Product)

        .Where(s => s.User.UserName == User.Identity.Name)

        .ToListAsync());

}

1. Adicionamos la vista **MyOrders** al controlador **OrdersController**:

@model IEnumerable<Shooping.Data.Entities.Sale>

@{

    ViewData["Title"] = "Index";

}

<link rel="stylesheet" href="https://cdn.datatables.net/1.10.19/css/jquery.dataTables.min.css" />

<div class="row">

    <div class="col-md-12">

        <div class="panel panel-default">

            <div class="panel-heading">

                <h3 class="panel-title">Mis Pedidos</h3>

            </div>

            <div class="panel-body">

                <table class="table table-hover table-responsive table-striped" id="MyTable">

                    <thead>

                        <tr>

                            <th>

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

                            </th>

                            <th>

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

                            </th>

                            <th>

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

                            </th>

                            <th>

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

                            </th>

                            <th>

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

                            </th>

                            <th>

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

                            </th>

                            <th></th>

                        </tr>

                    </thead>

                    <tbody>

                        @foreach (var item in Model)

                        {

                            <tr>

                                <td>

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

                                </td>

                                <td>

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

                                </td>

                                <td>

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

                                </td>

                                <td>

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

                                </td>

                                <td>

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

                                </td>

                                <td>

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

                                </td>

                                <td>

                                    <a asp-action="MyDetails" asp-route-id="@item.Id" class="btn btn-outline-info">Detalles</a>

                                </td>

                            </tr>

                        }

                    </tbody>

                </table>

            </div>

        </div>

    </div>

</div>

@section Scripts {

    @{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

    <script src="//cdn.datatables.net/1.10.19/js/jquery.dataTables.min.js"></script>

    <script type="text/javascript">

        $(document).ready(function () {

            $('#MyTable').DataTable({

                "language": {

                    "url": "//cdn.datatables.net/plug-ins/9dcbecd42ad/i18n/Spanish.json"

                },

                "aLengthMenu": [

                    [25, 50, 100, 200, -1],

                    [25, 50, 100, 200, "Todos"]

                ]

            });

        });

    </script>

}

1. Modificamos el menú:

@if (User.Identity.IsAuthenticated && User.IsInRole("User"))

{

    <li class="nav-item">

        <a class="nav-link text-dark" asp-area="" asp-controller="Orders" asp-action="MyOrders">Mis Pedidos</a>

    </li>

}

1. Adicionamos este método:

[Authorize(Roles = "User")]

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

{

    if (id == null)

    {

        return NotFound();

    }

    Sale sale = await \_context.Sales

        .Include(s => s.User)

        .Include(s => s.SaleDetails)

        .ThenInclude(sd => sd.Product)

        .ThenInclude(p => p.ProductImages)

        .FirstOrDefaultAsync(s => s.Id == id);

    if (sale == null)

    {

        return NotFound();

    }

    return View(sale);

}

1. Luego adicionamos la vista:

@model Shooping.Data.Entities.Sale

@{

    ViewData["Title"] = "Details";

}

<flash dismissable="true" />

<h1>Detalles</h1>

<div>

    <partial name="\_OrderDetails" />

</div>

<div>

    <a asp-action="MyOrders" class="btn btn-outline-success">Regresar</a>

</div>

<link rel="stylesheet" href="https://cdn.datatables.net/1.10.19/css/jquery.dataTables.min.css" />

<hr />

<div class="row">

    <div class="col-md-12">

        <div class="panel panel-default">

            <div class="panel-heading">

                <h3 class="panel-title">Productos</h3>

            </div>

            <div class="panel-body">

                <table class="table table-hover table-responsive table-striped" id="MyTable">

                    <thead>

                        <tr>

                            <th>

                                @Html.DisplayNameFor(model => model.SaleDetails.FirstOrDefault().Product.Name)

                            </th>

                            <th>

                                @Html.DisplayNameFor(model => model.SaleDetails.FirstOrDefault().Remarks)

                            </th>

                            <th>

                                @Html.DisplayNameFor(model => model.SaleDetails.FirstOrDefault().Product.Price)

                            </th>

                            <th>

                                @Html.DisplayNameFor(model => model.SaleDetails.FirstOrDefault().Quantity)

                            </th>

                            <th>

                                @Html.DisplayNameFor(model => model.SaleDetails.FirstOrDefault().Value)

                            </th>

                            <th>

                                @Html.DisplayNameFor(model => model.SaleDetails.FirstOrDefault().Product.ImageFullPath)

                            </th>

                        </tr>

                    </thead>

                    <tbody>

                        @foreach (var item in Model.SaleDetails)

                        {

                            <tr>

                                <td>

                                    @Html.DisplayFor(modelItem => item.Product.Name)

                                </td>

                                <td>

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

                                </td>

                                <td>

                                    @Html.DisplayFor(modelItem => item.Product.Price)

                                </td>

                                <td>

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

                                </td>

                                <td>

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

                                </td>

                                <td>

                                    <img src="@item.Product.ImageFullPath" style="width:100px;" />

                                </td>

                            </tr>

                        }

                    </tbody>

                </table>

            </div>

        </div>

    </div>

</div>

@section Scripts {

    @{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

    <script src="//cdn.datatables.net/1.10.19/js/jquery.dataTables.min.js"></script>

    <script type="text/javascript">

        $(document).ready(function () {

            $('#MyTable').DataTable({

                "language": {

                    "url": "//cdn.datatables.net/plug-ins/9dcbecd42ad/i18n/Spanish.json"

                },

                "aLengthMenu": [

                    [25, 50, 100, 200, -1],

                    [25, 50, 100, 200, "Todos"]

                ]

            });

        });

    </script>

}

1. Probamos.

Filtrar registros por categoría

1. Modificamos el **ICombosHelper**:

Task<IEnumerable<SelectListItem>> GetComboCategoriesAsync(bool withAll = false);

1. Modificamos el **CombosHelper**:

public async Task<IEnumerable<SelectListItem>> GetComboCategoriesAsync(bool withAll = false)

{

    List<SelectListItem> list = await \_context.Categories.Select(x => new SelectListItem

    {

        Text = x.Name,

        Value = $"{x.Id}"

    })

        .OrderBy(x => x.Text)

        .ToListAsync();

    list.Insert(0, new SelectListItem

    {

        Text = withAll ? "[Todas las categorías...]" : "[Seleccione una categoría...]",

        Value = "0"

    });

    return list;

}

1. Modificamos el **HomeViewModel**:

public class HomeViewModel

{

    public ICollection<ProductsHomeViewModel> Products { get; set; }

    public float Quantity { get; set; }

    [Display(Name = "Filtrar por Categoría")]

    public int CategoryId { get; set; }

    public IEnumerable<SelectListItem> Categories { get; set; }

}

1. Modificamos el método **Index** del **HomeController**:

@if (Model.Quantity > 0)

{

    <a asp-action="ShowCart" class="btn btn-outline-primary">Ver Carro de Compras (@Model.Quantity)</a>

}

<form asp-action="Index">

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

    <div class="form-group">

        <div class="row">

            <div class="col-md-11 mt-2 mb-2">

                <select asp-for="CategoryId" asp-items="Model.Categories" class="form-control"></select>

            </div>

            <div class="col-md-1 mt-2 mb-2">

                <input type="submit" value="Filtrar" class="btn btn-block btn-outline-primary" />

            </div>

        </div>

    </div>

</form>

@foreach (var item in Model.Products)

1. Modificamos la vista **Index** del **HomeController**, primero inyectamos el **ICombosHelper**:

[HttpGet]

[HttpPost]

public async Task<IActionResult> Index(HomeViewModel? model)

{

    List<Product>? products;

    if (model.CategoryId == 0)

    {

        products = await \_context.Products

            .Include(p => p.ProductImages)

            .Include(p => p.ProductCategories)

            .ThenInclude(pc => pc.Category)

            .Where(p => p.Stock > 0)

            .OrderBy(p => p.Description)

            .ToListAsync();

    }

    else

    {

        products = await \_context.Products

            .Include(p => p.ProductImages)

            .Include(p => p.ProductCategories)

            .ThenInclude(pc => pc.Category)

            .Where(p => p.Stock > 0 && p.ProductCategories.Any(pc => pc.Category.Id == model.CategoryId))

            .OrderBy(p => p.Description)

            .ToListAsync();

    }

    List<ProductsHomeViewModel> productsHome = new() { new ProductsHomeViewModel() };

    int i = 1;

    foreach (Product? product in products)

    {

        if (i == 1)

        {

            productsHome.LastOrDefault().Product1 = product;

        }

        if (i == 2)

        {

            productsHome.LastOrDefault().Product2 = product;

        }

        if (i == 3)

        {

            productsHome.LastOrDefault().Product3 = product;

        }

        if (i == 4)

        {

            productsHome.LastOrDefault().Product4 = product;

            productsHome.Add(new ProductsHomeViewModel());

            i = 0;

        }

        i++;

    }

    model.Products = productsHome;

    model.Categories = await \_combosHelper.GetComboCategoriesAsync(true);

    User user = await \_userHelper.GetUserAsync(User.Identity.Name);

    if (user != null)

    {

        model.Quantity = await \_context.TemporalSales

            .Where(ts => ts.User.Id == user.Id)

            .SumAsync(ts => ts.Quantity);

    }

    return View(model);

}

1. Probamos.

Filtrar registros por nombre

1. Agregar esta propiedad a la **HomeViewModel**:

[Display(Name = "Filtrar por Nombre")]

public string FilterName { get; set; }

1. Modificar el método **Index** del **HomeController**:

[HttpGet]

[HttpPost]

public async Task<IActionResult> Index(HomeViewModel? model)

{

List<Product>? products = await \_context.Products

        .Include(p => p.ProductImages)

        .Include(p => p.ProductCategories)

        .ThenInclude(pc => pc.Category)

        .Where(p => p.Stock > 0)

        .OrderBy(p => p.Description)

        .ToListAsync();

if (model.CategoryId != 0)

{

    products = products

        .Where(p => p.ProductCategories.Any(pc => pc.Category.Id == model.CategoryId))

        .ToList();

}

if (!string.IsNullOrEmpty(model.FilterName))

{

    products = products

        .Where(p => p.Name.ToLower().Contains(model.FilterName.ToLower()))

        .ToList();

}

1. Modificar la vista **Index** del **HomeController**:

<form asp-action="Index">

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

    <div class="form-group">

        <div class="row">

            <div class="col-md-1 mt-2 mb-2">

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

            </div>

            <div class="col-md-4 mt-2 mb-2">

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

            </div>

            <div class="col-md-1 mt-2 mb-2">

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

            </div>

            <div class="col-md-5 mt-2 mb-2">

                <select asp-for="CategoryId" asp-items="Model.Categories" class="form-control"></select>

            </div>

            <div class="col-md-1 mt-2 mb-2">

                <input type="submit" value="Filtrar" class="btn btn-block btn-outline-primary" />

            </div>

        </div>

    </div>

</form>

1. Probamos.

Paginación por base de datos

Política de seguridad

Publicación en Azure

Fin