**APLICACIÓN VETERINARIA**

* Servicios y el modo administrativo en .**NET Core**
* Front end de la aplicación en **Xamarin Form** con ayuda de **Prism** es un framework que facilita el desarrollo permite ligar view con la viewModel
* Rescribir la aplicación en **Xamarin Classic**

|  |  |  |  |
| --- | --- | --- | --- |
| **Funcionality** | **WEB** | | **APP** |
| **Admin** | **Customer** | **Customer** |
| Login | X | X | X |
| Register |  | X | X |
| Modify profile | X | X | X |
| Recover password | X | X | X |
| Admin managers | X |  |  |
| Admin Owners | X |  |  |
| Admin Pets | X | X | X |
| Admin pet types | X |  |  |
| Admin agenda | X |  |  |
| Assign/ unassign appoinments | X | X | X |
| Admin clinic history for Pets | X |  |  |
| See clinic history | X | X | X |
| See veterinary on map |  |  | X |
| Service “Pet love” |  |  | X |

**Proyectos**

1. **MyVet.Common(librería .NET Estándar)\*:** Transversal a toda la solución es decir todo código reutilizable tanto en web como en app se coloca acá

**Backend (web)**

1. **MyVet.Web(librería .NET Core)\*:** Tiene dos funcionalidades implícitas
2. **Front End Web(Admin):**
3. **API:** Servicios de comunicaciones porque se quiere acceder además de web vía app.

**Frontend (Mobile)**

1. **MyVet.Prism\*:** Se tiene todo el código compartido con limitaciones en interface de usuario
2. **MiVet.Prism.Android\*:** Proyecto en android de xamarin form
3. **MiVet.Prism.Ios\*:** Proyecto en ios de xamarin form
4. **MyVet.Cross.Android\***
5. **MyVet.Cross.iOS\***

El 4 y 5 Comparten código de la librería .NET Estándar no se tiene limitante en la interface de usuario

**GITHUB**

1. Crear repositorio

**Visual Studio**

1. Clonar repositorio en una ruta corta
2. Crear una solución en blanco llamada **MyVet**
3. Crear el primer proyecto es .NET Estándar una class library llamado **MyVet.Common** tendrá todo el código que se comparte
4. Crear el 2do proyecto el cual es será de tipo ASP.NET Core Web Application y se llamará **MyVet.Web** en **Core 2.1** y se escoge la opción **web Application(Model-View-Controller)** y se le habilita HTTPS para que los servicios se puedan configurar de forma segura.
5. Creación del modelo Owner

public class Owner

{

public int Id { get; set; }

[Required]

[MaxLength(30)]

[Display(Name ="Documento")]

public string Document { get; set; }

[Required]

[MaxLength(50)]

[Display(Name = "Nombres")]

public string FirstName { get; set; }

[Required]

[MaxLength(50)]

[Display(Name = "Apellidos")]

public string LastName { get; set; }

[MaxLength(20)]

[Display(Name = "Teléfono fijo")]

public string FixedPhone { get; set; }

[Required]

[MaxLength(20)]

[Display(Name = "Teléfono celular")]

public string CellPhone { get; set; }

[MaxLength(100)]

[Display(Name = "Dirección")]

public string Address { get; set; }

[Display(Name = "Propietario")

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

[Display(Name = "Propietario")]

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

public ICollection<Agenda> Agendas { get; set; }

public ICollection<Pet> Pets { get; set; }

}

1. Creación de la clase DataContext

public class DataContext : DbContext

{

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

{

}

public DbSet<Owner> Owners { get; set; }

}

1. En .NET Core hay 3 archivos importantes en la configuración del proyecto

-**appsettings.json:** Se define el string de conexión, pero 1ro se mira las instancias que se tienen instaladas o si la conexión se va hacer a una maquina remota. Para el caso será local en este archivo se adiciona el string

*{*

*"Logging": {*

*"LogLevel": {*

*"Default": "Warning"*

*}*

*},*

*"AllowedHosts": "\*",*

*"ConnectionStrings": {*

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

*}*

*}*

- **Program.cs**

- **Starup.cs:** Se le dice al proyecto que use la bd.

public void ConfigureServices(IServiceCollection services)

{

services.Configure<CookiePolicyOptions>(options =>

{

// This lambda determines whether user consent for non-essential cookies is needed for a given request.

options.CheckConsentNeeded = context => true;

options.MinimumSameSitePolicy = SameSiteMode.None;

});

services.AddDbContext<DataContext>(cfg =>

{ cfg.UseSqlServer(Configuration.GetConnectionString("DefaultConnection"));

}); services.AddMvc().SetCompatibilityVersion(CompatibilityVersion.Version\_2\_1);

}

1. Se corren comandos en la consola nuget package manage asegurarse que este seleccionado el proyecto web y correr los siguientes comandos

Update-database

add-migration nombreMigracion

Update-database

1. Se crea controlador para owner
2. Base de datos inicial



1. Los otros 5 modelos
2. Agenda

*public class Agenda*

*{*

*public int Id { get; set; }*

*[Display(Name = "Fecha")]*

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

*[DataType(DataType.DateTime)]*

*[DisplayFormat(DataFormatString = "{0:yyyy/MM/dd HH:mm tt}", ApplyFormatInEditMode = true)]*

*public DateTime Date { get; set; }*

*[Display(Name = "Comentarios")]*

*public string Remarks { get; set; }*

*[Display(Name = "Disponible?")]*

*public bool IsAvailable { get; set; }*

*[Display(Name = "Fecha:")]*

*[DisplayFormat(DataFormatString = "{0:yyyy/MM/dd HH:mm}")]*

*public DateTime DateLocal => Date.ToLocalTime();*

*public Pet Pet { get; set; }*

*public Owner Owner { get; set; }*

*}*

1. *Pet*

*public class Pet*

*{*

*public int Id { get; set; }*

*[Display(Name = "Nombre")]*

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

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

*public string Name { get; set; }*

*[Display(Name = "Imagen")]*

*public string ImageUrl { get; set; }*

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

*[Display(Name = "Raza")]*

*public string Race { get; set; }*

*[Display(Name = "Fecha de nacimiento")]*

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

*[DataType(DataType.DateTime)]*

*[DisplayFormat(DataFormatString = "{0:yyyy/MM/dd}", ApplyFormatInEditMode = true)]*

*public DateTime Born { get; set; }*

*[Display(Name = "Comentarios")]*

*public string Remarks { get; set; }*

*//TODO: replace the correct URL for the image*

*public string ImageFullPath => string.IsNullOrEmpty(ImageUrl)*

*? null*

*: $"https://TBD.azurewebsites.net{ImageUrl.Substring(1)}";*

*[Display(Name = "Fecha de nacimiento")]*

*[DisplayFormat(DataFormatString = "{0:yyyy/MM/dd}", ApplyFormatInEditMode = true)]*

*public DateTime BornLocal => Born.ToLocalTime();*

*public PetType PetType { get; set; }*

*public ICollection<History> Histories { get; set; }*

*public ICollection<Agenda> Agendas { get; set; }*

*}*

1. *History*

*public class History*

*{*

*public int Id { get; set; }*

*[Display(Name = "Descripción")]*

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

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

*public string Description { get; set; }*

*[Display(Name = "Fecha")]*

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

*[DisplayFormat(DataFormatString = "{0:yyyy/MM/dd}", ApplyFormatInEditMode = true)]*

*public DateTime Date { get; set; }*

*[Display(Name = "Comentarios")]*

*public string Remarks { get; set; }*

*[Display(Name = "Fecha:")]*

*[DisplayFormat(DataFormatString = "{0:yyyy/MM/dd}", ApplyFormatInEditMode = true)]*

*public DateTime DateLocal => Date.ToLocalTime();*

*public ServiceType ServiceType { get; set; }*

*}*

1. *Pettype*

*public class PetType*

*{*

*public int Id { get; set; }*

*[Display(Name = "Tipo de mascota")]*

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

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

*public string Name { get; set; }*

*public Owner Owner { get; set; }*

*public ICollection<Pet> Pets { get; set; }*

*}*

1. *ServiceType*

public class ServiceType

*{*

*public int Id { get; set; }*

*[Display(Name = "Tipo de servicio")]*

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

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

*public string Name { get; set; }*

*public ICollection<History> Histories { get; set; }*

*}*

1. *Añadir las otras tablas al datacontext*

*public class DataContext : DbContext*

*{*

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

*{*

*}*

*public DbSet<Owner> Owners { get; set; }*

*public DbSet<Agenda> Agendas { get; set; }*

*public DbSet<History> Histories { get; set; }*

*public DbSet<Pet> Pets { get; set; }*

*public DbSet<PetType> PetTypes { get; set; }*

*public DbSet<ServiceType> ServiceTypes { get; set; }*

*}*

1. *Agregar otra migración*

*Add-migration CompleteDB*

*Update-database*

1. *Alimentador de base de datos registros de prueba*

*public class SeedDb*

*{*

*private readonly DataContext \_context;*

*public SeedDb(DataContext context)*

*{*

*\_context = context;*

*}*

*public async Task SeedAsync()*

*{*

*await \_context.Database.EnsureCreatedAsync();*

*await CheckPetTypesAsync();*

*await CheckServiceTypesAsync();*

*await CheckOwnersAsync();*

*await CheckPetsAsync();*

*await CheckAgendasAsync();*

*}*

*private async Task CheckPetsAsync()*

*{*

*var owner = \_context.Owners.FirstOrDefault();*

*var petType = \_context.PetTypes.FirstOrDefault();*

*if (!\_context.Pets.Any())*

*{*

*AddPet("Otto", owner, petType, "Shih tzu");*

*AddPet("Killer", owner, petType, "Dobermann");*

*await \_context.SaveChangesAsync();*

*}*

*}*

*private async Task CheckServiceTypesAsync()*

*{*

*if (!\_context.ServiceTypes.Any())*

*{*

*\_context.ServiceTypes.Add(new ServiceType { Name = "Consulta" });*

*\_context.ServiceTypes.Add(new ServiceType { Name = "Urgencia" });*

*\_context.ServiceTypes.Add(new ServiceType { Name = "Vacunación" });*

*await \_context.SaveChangesAsync();*

*}*

*}*

*private async Task CheckPetTypesAsync()*

*{*

*if (!\_context.PetTypes.Any())*

*{*

*\_context.PetTypes.Add(new PetType { Name = "Perro" });*

*\_context.PetTypes.Add(new PetType { Name = "Gato" });*

*await \_context.SaveChangesAsync();*

*}*

*}*

*private async Task CheckOwnersAsync()*

*{*

*if (!\_context.Owners.Any())*

*{*

*AddOwner("8989898", "Carolina", "Russi", "234 3232", "310 322 3221", "Calle Luna Calle Sol");*

*AddOwner("7655544", "Alfonso", "Mercado", "343 3226", "300 322 3221", "Calle 77 #22 21");*

*AddOwner("6565555", "Maria", "López", "450 4332", "350 322 3221", "Carrera 56 #22 21");*

*await \_context.SaveChangesAsync();*

*}*

*}*

*private void AddOwner(string document, string firstName, string lastName, string fixedPhone, string cellPhone, string address)*

*{*

*\_context.Owners.Add(new Owner*

*{*

*Address = address,*

*CellPhone = cellPhone,*

*Document = document,*

*FirstName = firstName,*

*FixedPhone = fixedPhone,*

*LastName = lastName*

*});*

*}*

*private void AddPet(string name, Owner owner, PetType petType, string race)*

*{*

*\_context.Pets.Add(new Pet*

*{*

*Born = DateTime.Now.AddYears(-2),*

*Name = name,*

*Owner = owner,*

*PetType = petType,*

*Race = race*

*});*

*}*

*private async Task CheckAgendasAsync()*

*{*

*if (!\_context.Agendas.Any())*

*{*

*var initialDate = new DateTime(DateTime.Now.Year, DateTime.Now.Month, DateTime.Now.Day, 8, 0, 0);*

*var finalDate = initialDate.AddYears(1);*

*while (initialDate < finalDate)*

*{*

*if (initialDate.DayOfWeek != DayOfWeek.Sunday)*

*{*

*var finalDate2 = initialDate.AddHours(10);*

*while (initialDate < finalDate2)*

*{*

*\_context.Agendas.Add(new Agenda*

*{*

*Date = initialDate.ToUniversalTime(),*

*IsAvailable = true*

*});*

*initialDate = initialDate.AddMinutes(30);*

*}*

*initialDate = initialDate.AddHours(14);*

*}*

*else*

*{*

*initialDate = initialDate.AddDays(1);*

*}*

*}*

*await \_context.SaveChangesAsync();*

*}*

*}*

*}*

1. Se hace un cambio en la clase program inicialmente esta de esta forma

public class Program

{

public static void Main(string[] args)

{

CreateWebHostBuilder(args).Build().Run();

}

public static IWebHostBuilder CreateWebHostBuilder(string[] args) =>

WebHost.CreateDefaultBuilder(args)

.UseStartup<Startup>();

}

1. *Queda de esta forma*

public class Program

{

public static void Main(string[] args)

{

var host = CreateWebHostBuilder(args).Build();

RunSeeding(host);

host.Run();

}

private static void RunSeeding(IWebHost host)

{

var scopeFactory = host.Services.GetService<IServiceScopeFactory>();

using (var scope = scopeFactory.CreateScope())

{

var seeder = scope.ServiceProvider.GetService<SeedDb>();

seeder.SeedAsync().Wait();

}

}

public static IWebHostBuilder CreateWebHostBuilder(string[] args)

{

return WebHost.CreateDefaultBuilder(args).UseStartup<Startup>();

}

}

1. *Inyectar el seeder en el archivo de configuración en el starup*

*services.AddDbContext<DataContext>(cfg =>*

*{*

*cfg.UseSqlServer(Configuration.GetConnectionString("DefaultConnection"));*

*});*

*services.AddTransient<SeedDb>();*

*services.AddMvc().SetCompatibilityVersion(CompatibilityVersion.Version\_2\_1);*

1. *Crear controlador para agenda con respecto a la hora en la vista index en vez de mostrar el date se muestra el dateLocal*
2. *Adicionar al MER las tablas de usuario son del userEntity framework las 3 ms importantes son las mostradas y uma tabla manager que sirve para guardar los datos básicos de los administradores*



1. *Crear el modelo user el cual hereda del identityUser y se le adiciona los campos que se necesita que esta tabla no tiene para el caso de owner*
2. *En owner se reemplaza todo excepto la clave primaria y las relaciones por una relación con usuario*

*public class Owner*

*{*

*public int Id { get; set; }*

*public User User { get; set; }*

*public ICollection<Agenda> Agendas { get; set; }*

*public ICollection<Pet> Pets { get; set; }*

*}*

1. *Crea el modelo para manager que para integrarlo con el modelo de usuarios queda así*

*public class Manager*

*{*

*public int Id { get; set; }*

*public User User { get; set; }*

*}*

1. *En el datacontext ya no hereda del dbcontext así se le dice cual es la clase o modelo de usuario personalizado y tb se agrega la tabla manager queda así*

*public class DataContext : IdentityDbContext<User>*

*{*

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

*{*

*}*

public DbSet<Manager> Managers { get; set; }

1. *Las tablas adicionales que están en otro color no se accede a ellas directamente, si no que se acceden por medio del userManager y roleManager*
2. *Crear carpeta llamada helper las cuales tienen interfaces y clases con utilidades*
3. *Se crea dentro de esta carpeta una interface llamada UserHelper que inicialmente tendrá 4 métodos*

*public interface IUserHelper*

*{*

*Task<User> GetUserByEmailAsync(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. *La clase userHelper*

*public class UserHelper : IUserHelper*

*{*

*private readonly UserManager<User> \_userManager;*

*private readonly RoleManager<IdentityRole> \_roleManager;*

*public UserHelper(*

*UserManager<User> userManager,*

*RoleManager<IdentityRole> roleManager)*

*{*

*\_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)*

*{*

*var roleExists = await \_roleManager.RoleExistsAsync(roleName);*

*if (!roleExists)*

*{*

*await \_roleManager.CreateAsync(new IdentityRole*

*{*

*Name = roleName*

*});*

*}*

*}*

*public async Task<User> GetUserByEmailAsync(string email)*

*{*

*var user = await \_userManager.FindByEmailAsync(email);*

*return user;*

*}*

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

*{*

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

*}*

*}*

1. *Configurar la nueva inyección en el Starup*

*services.AddDbContext<DataContext>(cfg =>*

*{*

*cfg.UseSqlServer(Configuration.GetConnectionString("DefaultConnection"));*

*})*

*services.AddTransient<SeedDb>();*

*services.AddScoped<IUserHelper, UserHelper>(); services.AddMvc().SetCompatibilityVersion(CompatibilityVersion.Version\_2\_1);*

1. *Modifica el seeder*

public class SeedDb

{

private readonly DataContext \_dataContext;

private readonly IUserHelper \_userHelper;

public SeedDb(

DataContext context,

IUserHelper userHelper)

{

\_dataContext = context;

\_userHelper = userHelper;

}

public async Task SeedAsync()

{

await \_dataContext.Database.EnsureCreatedAsync();

await CheckRoles();

var manager = await CheckUserAsync("1010", "Diana", "Russi", "diana.russiposada@gmail.com", "350 634 2747", "Calle Luna Calle Sol", "Admin");

var customer = await CheckUserAsync("2020", "Diana", "Russi", "dianarussi@yahoo.com", "350 634 2747", "Calle Luna Calle Sol", "Customer");

await CheckPetTypesAsync();

await CheckServiceTypesAsync();

await CheckOwnerAsync(customer);

await CheckManagerAsync(manager);

await CheckPetsAsync();

await CheckAgendasAsync();

}

private async Task CheckRoles()

{

await \_userHelper.CheckRoleAsync("Admin");

await \_userHelper.CheckRoleAsync("Customer");

}

private async Task<User> CheckUserAsync(string document, string firstName, string lastName, string email, string phone, string address, string role)

{

var user = await \_userHelper.GetUserByEmailAsync(email);

if (user == null)

{

user = new User

{

FirstName = firstName,

LastName = lastName,

Email = email,

UserName = email,

PhoneNumber = phone,

Address = address,

Document = document

};

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

await \_userHelper.AddUserToRoleAsync(user, role);

}

return user;

}

private async Task CheckPetsAsync()

{

if (!\_dataContext.Pets.Any())

{

var owner = \_dataContext.Owners.FirstOrDefault();

var petType = \_dataContext.PetTypes.FirstOrDefault();

AddPet("Otto", owner, petType, "Shih tzu");

AddPet("Killer", owner, petType, "Dobermann");

await \_dataContext.SaveChangesAsync();

}

}

private async Task CheckServiceTypesAsync()

{

if (!\_dataContext.ServiceTypes.Any())

{

\_dataContext.ServiceTypes.Add(new ServiceType { Name = "Consulta" });

\_dataContext.ServiceTypes.Add(new ServiceType { Name = "Urgencia" });

\_dataContext.ServiceTypes.Add(new ServiceType { Name = "Vacunación" });

await \_dataContext.SaveChangesAsync();

}

}

private async Task CheckPetTypesAsync()

{

if (!\_dataContext.PetTypes.Any())

{

\_dataContext.PetTypes.Add(new PetType { Name = "Perro" });

\_dataContext.PetTypes.Add(new PetType { Name = "Gato" });

await \_dataContext.SaveChangesAsync();

}

}

private async Task CheckOwnerAsync(User user)

{

if (!\_dataContext.Owners.Any())

{

\_dataContext.Owners.Add(new Owner { User = user });

await \_dataContext.SaveChangesAsync();

}

}

private async Task CheckManagerAsync(User user)

{

if (!\_dataContext.Managers.Any())

{

\_dataContext.Managers.Add(new Manager { User = user });

await \_dataContext.SaveChangesAsync();

}

}

private void AddPet(string name, Owner owner, PetType petType, string race)

{

\_dataContext.Pets.Add(new Pet

{

Born = DateTime.Now.AddYears(-2),

Name = name,

Owner = owner,

PetType = petType,

Race = race

});

}

private async Task CheckAgendasAsync()

{

if (!\_dataContext.Agendas.Any())

{

var initialDate = new DateTime(DateTime.Now.Year, DateTime.Now.Month, DateTime.Now.Day, 8, 0, 0);

var finalDate = initialDate.AddYears(1);

while (initialDate < finalDate)

{

if (initialDate.DayOfWeek != DayOfWeek.Sunday)

{

var finalDate2 = initialDate.AddHours(10);

while (initialDate < finalDate2)

{

\_dataContext.Agendas.Add(new Agenda

{

Date = initialDate,

IsAvailable = true

});

initialDate = initialDate.AddMinutes(30);

}

initialDate = initialDate.AddHours(14);

}

else

{

initialDate = initialDate.AddDays(1);

}

}

}

await \_dataContext.SaveChangesAsync();

}

}

}

1. *Borrar el controlador de owner y sus vistas y volverlo a crear y crear controlador para lo manger*
2. *Crear la opciones en el menú*
3. *Se ejecutan los siguientes comandos*

*drop-database*

*add-migration Users*

*update-database*

1. *Configurar las condiciones del password en el Starup*

services.Configure<CookiePolicyOptions>(options =>

{

// This lambda determines whether user consent for non-essential cookies is needed for a given request.

options.CheckConsentNeeded = context => true;

options.MinimumSameSitePolicy = SameSiteMode.None;

});

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>();

1. También se le dice que la aplicación va a utiliza autentificación por eso se agrega tb en ese mismo archivo esta liea de código

*public void Configure(IApplicationBuilder app, IHostingEnvironment env)*

*{*

*if (env.IsDevelopment())*

*{*

*app.UseDeveloperExceptionPage();*

*}*

*else*

*{*

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

*app.UseHsts();*

*}*

*app.UseHttpsRedirection();*

*app.UseStaticFiles();*

*app.UseAuthentication();*

*app.UseCookiePolicy();*

*app.UseMvc(routes =>*

*{*

*routes.MapRoute(*

*name: "default",*

*template: "{controller=Home}/{action=Index}/{id?}");*

*});*

*}*

1. *Ya se prueba ya maneja usuarios y la se puede hacer ogin y logout*
2. *Crear controlador MVC muchas veces el entity sirve de modelo en otras toca crearlo como en el login*
3. *Vamos al menú para hacer la parte de como se mostraría el login y logout*

*<div class="navbar-collapse collapse">*

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

*<li><a* ***asp-area****=""* ***asp-controller****="Home"* ***asp-action****="Index">Inicio</a></li>*

*<li><a* ***asp-area****=""* ***asp-controller****="Home"* ***asp-action****="About">Acerca de</a></li>*

*<li><a* ***asp-area****=""* ***asp-controller****="Home"* ***asp-action****="Contact">Contactanos</a></li>*

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

*{*

*<li><a* ***asp-area****=""* ***asp-controller****="Owners"* ***asp-action****="Index">Propietarios</a></li>*

*<li><a* ***asp-area****=""* ***asp-controller****="Managers"* ***asp-action****="Index">Administrador</a></li>*

*<li><a* ***asp-area****=""* ***asp-controller****="Agenda"* ***asp-action****="Index">Agenda</a></li>*

*}*

*</ul>*

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

@if (User.Identity.IsAuthenticated)

{

<li><a **asp-area**="" **asp-controller**="Account" **asp-action**="ChangeUser">@User.Identity.Name</a></li>

<li><a **asp-area**="" **asp-controller**="Account" **asp-action**="Logout">Logout</a></li>

}

else

{

<li><a **asp-area**="" **asp-controller**="Account" **asp-action**="Login">Login</a></li>

}

</ul>

*</div>*

1. *Crear controlador accountController dentro del cual crear la acción login el get y la propiedad privada de solo lectura*

public class AccountController : Controller

private readonly IUserHelper \_userHelper;

public AccountController(IUserHelper userHelper)

{

\_userHelper = userHelper;

}

public IActionResult Login()

{

return View();

}

{

public IActionResult Login()

{

return View();

}

}

1. *Crear el modelo loginviewmodel*

*public class LoginViewModel*

*{*

*[Required]*

*[EmailAddress]*

*public string Username { get; set; }*

*[Required]*

*[MinLength(6)]*

*public string Password { get; set; }*

*public bool RememberMe { get; set; }*

*}*

1. *la vista login*

*@model MyVet.Web.Models.LoginViewModel*

*@{*

*ViewData["Title"] = "Login";*

*}*

*<h2>Inicio de sesión</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****="Username">Usuario</label>*

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

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

*</div>*

*<script src="~/lib/jquery-validation/dist/jquery.validate.js"></script>*

*<div class="form-group">*

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

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

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

*</div>*

*<div class="form-group">*

*<div class="form-check">*

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

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

*</div>*

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

*</div>*

*<div class="form-group">*

*<input type="submit" value="Login" class="btn btn-success" />*

*<a* ***asp-action****="Register" class="btn btn-primary">Registrar nuevo usuario</a>*

*</div>*

*</form>*

*</div>*

*</div>*

*@section Scripts {*

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

*}*

1. *En la interface de userHelper adicionar metodos para loguearse y desloguearse*

Task<SignInResult> LoginAsync(LoginViewModel model);

Task LogoutAsync();

1. *Implementar esos dos metodos en la clase de userHelper*

private readonly SignInManager<User> \_signInManager;

public UserHelper(

UserManager<User> userManager,

RoleManager<IdentityRole> roleManager,

SignInManager<User> signInManager)

{

\_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. Crear el post de la acción login

*[HttpPost]*

*public async Task<IActionResult> Login(LoginViewModel model)*

*{*

*if (ModelState.IsValid)*

*{*

*var result = await \_userHelper.LoginAsync(model);*

*if (result.Succeeded)*

*{*

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

*{*

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

*}*

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

*}*

*}*

*ModelState.AddModelError(string.Empty, "Failed to login.");*

*return View(model);*

*}*

1. *Acción logout en el controlador account*

*public async Task <IActionResult> Logout()*

*{*

*await \_userHelper.LogoutAsync();*

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

*}*

1. *Colocar en los controladores la anotación de authorice para que solo deje ingresar al usuario admin para el caso de owner, manager y agenda*

[Authorize(Roles ="Admin")]

***PARTE 9 LOGOUT AND PERMISIONS***