.NET CORE / XAMARIN FORMS / XAMARIN CLASSIC / MVVM CROSS

By Zulu

Medellín 2019

**Index**

[**Create the Solution 5**](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.gxlfce2c1vj6)

[**Create the Database 6**](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.3znysh7)

[**Modify DB 10**](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.2et92p0)

[**Seed the DB with initial data 12**](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.tyjcwt)

[**Implement the pattern repository 15**](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.3dy6vkm)

[**Add User Identities 21**](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.1t3h5sf)

[**Implement A Generic Repository & Some Fixes 27**](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.4d34og8)

[**Add API 41**](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.2s8eyo1)

[**Adding Images 44**](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.oz5yg1vh1h8)

[**Adding Other Methods To Generic Repository 55**](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.abagj621nacx)

[**Starting with Xamarin Forms 56**](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.keuc74vntav3)

[**Fix Bug to Don’t Replace Images 61**](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.iozd42yf2umd)

[**Consuming RestFull 62**](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.puok2kdj1ctb)

[**To Disable Cascade Delete Rule & Avoid Warnings in Update Database 75**](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.drcigckdodsy)

[**Implementing login and logout in Web 75**](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.rvvzh1dny0x)

[**Registering new users 81**](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.35nkun2)

[**Modifying users 86**](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.1ksv4uv)

[**Add Tokens Generation 93**](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.44sinio)

[**Add Font Awesome for Icons 96**](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.2jxsxqh)

[**Add Roles 97**](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.z337ya)

[**Redirect Pages 101**](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.aj0fqc8uxb93)

[Not Authorized 101](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.ghgz7haooxin)

[Handle Not Found Errors Gracefully 102](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.hi650tyy2260)

[Manage Not Found Pages 103](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.gjdgxs)

[**Orders Functionality 106**](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.iuig8g3elewc)

[**Add Modal Windows 131**](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.4i7ojhp)

[**Date Picker 137**](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.2xcytpi)

[**Cascade Drop Down List 143**](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.1ci93xb)

[**Confirm Email Registration 181**](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.3whwml4)

[**Password Recovery 188**](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.2bn6wsx)

[**Fix AwesomeFont and DatePicker on Published Site 195**](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.er2fy2w1ta67)

[**Improve the Seeder 200**](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.2fpn1sufdw4y)

[**Making A Little Easier The Modal Dialogs 205**](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.j1l9xw7tbphj)

[**Improve Index View 209**](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.8b6k2hep1w4n)

[**User Management 213**](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.11xkp3bboxau)

[**Login in Xamarin Forms 222**](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.l6rurqys43fe)

[**Master Detail in Xamarin Forms 232**](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.1pxezwc)

[**Completing the products API 242**](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.49x2ik5)

[**Completing the CRUD in Xamarin Forms 246**](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.2p2csry)

[**Implementing Settings in Xamarin Forms 269**](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.147n2zr)

[**Multi Language in Xamarin Forms 273**](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.3o7alnk)

[**Acceding To Camera and Gallery in Xamarin Forms 289**](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.23ckvvd)

[**Sending the Image to Backend 294**](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.ihv636)

[**Register Users From App in Xamarin Forms 298**](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.fvj8zqhg5nt8)

[**Recover Password From App in Xamarin Forms 321**](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.c7bwyq5zm41g)

[**Modify User From App in Xamarin Forms 330**](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.brkvnq7d4oki)

[**Modify Password From App in Xamarin Forms 353**](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.qpyf5uxa0zxi)

[**Add Icon & Splash to Xamarin Forms For Android 363**](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.ditq6h2uupkm)

[**Publish on Google Play 365**](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.lco5x66a5f39)

[**Starting With Xamarin Android Classic 385**](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.wubb068sks6)

[**Starting With Xamarin iOS Classic 397**](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.lz91aitkucc1)

[**Starting With MVVM Cross, Test Concept 410**](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.wry0sqfgg2ls)

[MVVM Cross Core Project (initial) 410](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.9y740ubu7onj)

[Forms Traditional Project - Way ONE 412](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.gwyiyvnb59hd)

[Xamarin Android Classic - Way TWO 418](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.5p2zogy78x4g)

[Xamarin iOS Classic - Way TWO 422](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.z18oy37v3tvj)

[MVVM Cross Core Project (definitive) 427](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.u4tfq4sqsii3)

[MVVM Cross Android Project  - Way THREE 430](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.3lp6gy1w9v3p)

[MVVM Cross iOS Project - Way THREE 435](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.dv0whsloo3gd)

[Forms Cross Project - Way FOUR 439](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.i9nc38j1asuq)

[**MVVM Cross Value Converters 445**](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.bnvtyw4eeomn)

[Core Project 445](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.gbf980kazwhy)

[Android Project 445](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.4aixqyiauzuc)

[iOS Project 446](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.9sorluallun2)

[**Making the Shop Project With MVVM Cross 446**](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.63drfiqfb65p)

[Core First Part (Login) 446](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.h7z1vm4fakxh)

[Android First Part (Login) 453](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.jgi7zfdbp5av)

[iOS First Part (Login) 459](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.uerm20i4l6vt)

[Core Second Part (Products List) 464](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.x6vzwx239ih3)

[Android Second Part (Products List) 469](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.lo6lur310j61)

[Core Third Part (Add Product) 472](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.n3j6oj5ebkir)

[Android Third Part (Add Product) 476](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.pjp8ds9rgg8)

[Core Fourth Part (Register User) 482](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.fpfhth31h3s1)

[Android Fourth Part (Register User) 488](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.11ufpdw8ate)

[Core Fifth Part (Product Details) 502](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.iwnjw4qf3jyb)

[Android Fifth Part (Product Details) 508](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.1oxdt8qqa10d)

[Core Sixth Part (Check internet connection) 515](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.uwjipqwj5rn6)

[Android Sixth Part (Check internet connection) 517](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.qty2qlp1bqtv)

[Android Seventh Part (Toolbar) 519](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.p3xw5utxvqy4)

[Core Eighth Part (Confirm Delete) 542](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.paul0n28qkjw)

[Android Eighth Part (Confirm Delete) 544](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.hs8r4o1hngup)

[Core Ninth Part (Change User, Change Password and Drawer Menu) 545](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.kl7ha4h07ebu)

[Android Ninth Part (Change User, Change Password and Drawer Menu) 561](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.e45nt1slw7vz)

[Core Tenth Part (Taking images from camera or gallery) 595](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.1a336b5cheza)

[Android Tenth Part (Taking images from camera or gallery) 601](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.5cguzetetidz)

[Android Eleventh Part (Maps and Geolocation) 620](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.s0ljr91zmfdk)

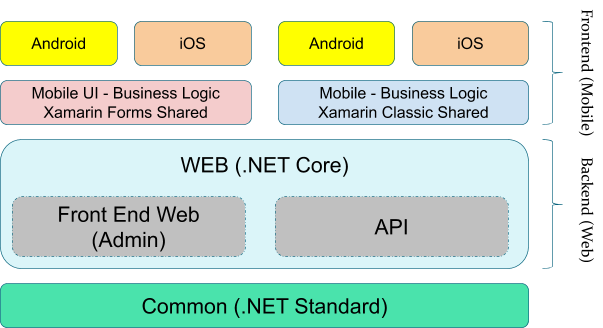
[Core Twelfth Part (Custom Dialogs) 637](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.u8w07tkfqh9a)

[Android Twelfth Part (Custom Dialogs) 639](https://docs.google.com/document/d/1EiqhDDbhQ4Rekh8rCAF8zAmIvaXHviyozO-QYIVwd50/edit#heading=h.1jzudnyzcksj)

# Create the Solution

**Note:** all the code are in: <https://github.com/Zulu55/Shop>

Create the following solution:



In Visual Studio, you must build something similar to:



Create the Database

**Note**: in this project we’ll work with entity framework code first, if you want to work with EF database first, I recommend this article: <https://docs.microsoft.com/en-us/ef/core/get-started/aspnetcore/existing-db>

1. Create the entities (in folder Web.Data.Entities):

using System;

using System.ComponentModel.DataAnnotations;

public class Product

{

public int Id { get; set; }

public string Name { get; set; }

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

public decimal Price { get; set; }

[Display(Name = "Image")]

public string ImageUrl { get; set; }

[Display(Name = "Last Purchase")]

public DateTime LastPurchase { get; set; }

[Display(Name = "Last Sale")]

public DateTime LastSale { get; set; }

[Display(Name = "Is Availabe?")]

public bool IsAvailabe { get; set; }

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

public double Stock { get; set; }

}

1. Create the context class (in folder Data):

using Common.Models;

using Microsoft.EntityFrameworkCore;

public class DataContext : DbContext

{

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

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

{

}

}

1. Add the connection string to the configuration json file (see the SQL Server Object Explorer):

{

  "Logging": {

"LogLevel": {

   "Default": "Warning"

}

  },

  "AllowedHosts": "\*",

  "ConnectionStrings": {

"DefaultConnection": "Server=(localdb)\\ProjectsV13;Database=Shop;Trusted\_Connection=True;MultipleActiveResultSets=true"

  }

}

**Note**: You must be sure of the servers names in your installation, you can check it out, by clicking in SQL Server Object Explorer:



In this case, there are three available servers: **(localdb)\MSSQLLocalDB**, **(localdb)\ProjectsV13** and **(localdb)\v11.0**. Or you can explore your server by clicking on “Add SQL Server” icon:



1. Add the database injection in **Startup** class:

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. Save changes and run those commands by command line in the same folder that is the web project:

dotnet ef database update

dotnet ef migrations add InitialDb

dotnet ef database update

Or you can run this commands in package manager console:

PM> update-database

PM> add-migration InitialDb

PM> update-database

1. Add the products controller.
2. Add the products menu and test the DB connection.

<ul class="nav navbar-nav">

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

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

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

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

</ul>

Modify DB

1. Modify the model product by:

using System;

using System.ComponentModel.DataAnnotations;

public class Product

{

public int Id { get; set; }

[MaxLength(50, ErrorMessage = "The field {0} only can contain a maximum {1} characters")]

[Required]

public string Name { get; set; }

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

public decimal Price { get; set; }

[Display(Name = "Image")]

public string ImageUrl { get; set; }

[Display(Name = "Last Purchase")]

public DateTime? LastPurchase { get; set; }

[Display(Name = "Last Sale")]

public DateTime? LastSale { get; set; }

[Display(Name = "Is Availabe?")]

public bool IsAvailabe { get; set; }

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

public double Stock { get; set; }

}

1. Run this commands:

dotnet ef migrations add ModifyProducts

dotnet ef database update

Or you can run this commands in package manager console:

PM> add-migration ModifyProducts

PM> update-database

1. Test it.

Seed the DB with initial data

1. Create the seed class, with your population data logic:

using System;

using System.Linq;

using System.Threading.Tasks;

using Common.Models;

public class SeedDb

{

private readonly DataContext context;

private Random random;

public SeedDb(DataContext context)

{

     this.context = context;

     this.random = new Random();

}

public async Task SeedAsync()

{

     await this.context.Database.EnsureCreatedAsync();

     if (!this.context.Products.Any())

     {

         this.AddProduct("First Product");

         this.AddProduct("Second Product");

         this.AddProduct("Third Product");

         await this.context.SaveChangesAsync();

     }

}

private void AddProduct(string name)

{

     this.context.Products.Add(new Product

     {

         Name = name,

         Price = this.random.Next(100),

         IsAvailabe = true,

         Stock = this.random.Next(100)

     });

}

}

1. Modify the Program class by:

using Data;

using Microsoft.AspNetCore;

using Microsoft.AspNetCore.Hosting;

using Microsoft.Extensions.DependencyInjection;

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) =>

     WebHost.CreateDefaultBuilder(args)

         .UseStartup<Startup>();

}

1. Add the injection for the seeder in Startup class (before cookie policy options lines):

services.AddTransient<SeedDb>();

1. Test it.

Implement the pattern repository

1. Create the repository class:

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Common.Models;

public class Repository

{

private readonly DataContext context;

public Repository(DataContext context)

{

     this.context = context;

}

public IEnumerable<Product> GetProducts()

{

     return this.context.Products.OrderBy(p => p.Name);

}

public Product GetProduct(int id)

{

     return this.context.Products.Find(id);

}

public void AddProduct(Product product)

{

     this.context.Products.Add(product);

}

public void UpdateProduct(Product product)

{

     this.context.Update(product);

}

public void RemoveProduct(Product product)

{

     this.context.Products.Remove(product);

}

public async Task<bool> SaveAllAsync()

{

     return await this.context.SaveChangesAsync() > 0;

}

public bool ProductExists(int id)

{

     return this.context.Products.Any(p => p.Id == id);

}

}

1. Extract the interface for the repository class:

using System.Collections.Generic;

using System.Threading.Tasks;

using Common.Models;

public interface IRepository

{

void AddProduct(Product product);

Product GetProduct(int id);

IEnumerable<Product> GetProducts();

bool ProductExists(int id);

void RemoveProduct(Product product);

Task<bool> SaveAllAsync();

void UpdateProduct(Product product);

}

1. Replace the controller to uses the repository and not uses the database context:

using Data;

using Data.Entities;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using System.Threading.Tasks;

public class ProductsController : Controller

{

    private readonly IRepository repository;

    public ProductsController(IRepository repository)

    {

        this.repository = repository;

    }

    public IActionResult Index()

    {

        return View(this.repository.GetProducts());

    }

    public IActionResult Details(int? id)

    {

        if (id == null)

        {

            return NotFound();

        }

        var product = this.repository.GetProduct(id.Value);

        if (product == null)

        {

            return NotFound();

        }

        return View(product);

    }

    public IActionResult Create()

    {

        return View();

    }

    [HttpPost]

    [ValidateAntiForgeryToken]

    public async Task<IActionResult> Create(Product product)

    {

        if (ModelState.IsValid)

        {

            this.repository.AddProduct(product);

            await this.repository.SaveAllAsync();

            return RedirectToAction(nameof(Index));

        }

        return View(product);

    }

    public IActionResult Edit(int? id)

    {

        if (id == null)

        {

            return NotFound();

        }

        var product = this.repository.GetProduct(id.Value);

        if (product == null)

        {

            return NotFound();

        }

        return View(product);

    }

    [HttpPost]

    [ValidateAntiForgeryToken]

    public async Task<IActionResult> Edit(Product product)

    {

        if (ModelState.IsValid)

        {

            try

            {

                this.repository.UpdateProduct(product);

                await this.repository.SaveAllAsync();

            }

            catch (DbUpdateConcurrencyException)

            {

                if (!this.repository.ProductExists(product.Id))

                {

                    return NotFound();

                }

                else

                {

                    throw;

                }

            }

            return RedirectToAction(nameof(Index));

        }

        return View(product);

    }

    public IActionResult Delete(int? id)

    {

        if (id == null)

        {

            return NotFound();

        }

        var product = this.repository.GetProduct(id.Value);

        if (product == null)

        {

            return NotFound();

        }

        return View(product);

    }

    [HttpPost, ActionName("Delete")]

    [ValidateAntiForgeryToken]

    public async Task<IActionResult> DeleteConfirmed(int id)

    {

        var product = this.repository.GetProduct(id);

        this.repository.RemoveProduct(product);

        await this.repository.SaveAllAsync();

        return RedirectToAction(nameof(Index));

    }

}

1. Add the injection for the repository in Startup class (before cookie policy options lines):

services.AddScoped<IRepository, Repository>();

1. Test it.

Add User Identities

1. Create your own users class inherit from IdentityUser class (in Common.Models):

using Microsoft.AspNetCore.Identity;

public class User : IdentityUser

{

public string FirstName { get; set; }

public string LastName { get; set; }

}

1. Modify the data context class:

using Entities;

using Microsoft.AspNetCore.Identity.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore;

public class DataContext : IdentityDbContext<User>

{

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

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

    {

    }

}

1. Make the relations with other models:

public User User { get; set; }

1. Drop the database and add the new migrations with those commands:

dotnet ef database drop

dotnet ef migrations add Users

dotnet ef database update

Or you can run those commands in package manager console:

PM> drop-database

PM> add-migration Users

PM> update-database

1. Modify the seeder to add some user:

using System;

using System.Linq;

using System.Threading.Tasks;

using Common.Models;

using Microsoft.AspNetCore.Identity;

public class SeedDb

{

private readonly DataContext context;

private readonly UserManager<User> userManager;

private Random random;

public SeedDb(DataContext context, UserManager<User> userManager)

{

     this.context = context;

     this.userManager = userManager;

     this.random = new Random();

}

public async Task SeedAsync()

{

     await this.context.Database.EnsureCreatedAsync();

     var user = await this.userManager.FindByEmailAsync("jzuluaga55@gmail.com");

     if (user == null)

     {

         user = new User

         {

             FirstName = "Juan",

             LastName = "Zuluaga",

             Email = "jzuluaga55@gmail.com",

             UserName = "jzuluaga55@gmail.com"

         };

         var result = await this.userManager.CreateAsync(user, "123456");

         if (result != IdentityResult.Success)

         {

             throw new InvalidOperationException("Could not create the user in seeder");

         }

     }

     if (!this.context.Products.Any())

     {

         this.AddProduct("First Product", user);

         this.AddProduct("Second Product", user);

         this.AddProduct("Third Product", user);

         await this.context.SaveChangesAsync();

     }

}

private void AddProduct(string name, User user)

{

     this.context.Products.Add(new Product

     {

         Name = name,

         Price = this.random.Next(100),

         IsAvailabe = true,

         Stock = this.random.Next(100),

         User = user

     });

}

}

1. Modify the configuration to setup the new functionality:

public void ConfigureServices(IServiceCollection services)

{

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>();

services.AddDbContext<DataContext>(cfg =>

{

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

});

services.AddTransient<SeedDb>();

services.AddScoped<IRepository, Repository>();

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

}

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. Test it.

Implement A Generic Repository & Some Fixes

(Tanks to Fabian Camargo <https://www.youtube.com/user/fabiancv90>)

1. Create the folder **Helpers** and inside it add the interface **IUserHelper**:

using System.Threading.Tasks;

using Data.Entities;

using Microsoft.AspNetCore.Identity;

public interface IUserHelper

{

Task<User> GetUserByEmailAsync(string email);

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

}

1. In the same folder add the implementation (**UserHelper**):

using System.Threading.Tasks;

using Data.Entities;

using Microsoft.AspNetCore.Identity;

public class UserHelper : IUserHelper

{

private readonly UserManager<User> userManager;

public UserHelper(UserManager<User> userManager)

{

     this.userManager = userManager;

}

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

{

     return await this.userManager.CreateAsync(user, password);

}

public async Task<User> GetUserByEmailAsync(string email)

{

     var user = await this.userManager.FindByEmailAsync(email);

     return user;

}

}

1. In **Web.Data.Entities** add the interface **IEntity**:

public interface IEntity

{

int Id { get; set; }

}

1. Modify the **Products** entity:

public class Product : IEntity

1. In **Data** add the interfaz **IGenericRepository**:

using System.Linq;

using System.Threading.Tasks;

public interface IGenericRepository<T> where T : class

{

IQueryable<T> GetAll();

Task<T> GetByIdAsync(int id);

Task CreateAsync(T entity);

Task UpdateAsync(T entity);

Task DeleteAsync(T entity);

Task<bool> ExistAsync(int id);

}

1. In the same folder add the implementation (**GenericRepository**):

using System.Linq;

using System.Threading.Tasks;

using Entities;

using Microsoft.EntityFrameworkCore;

public class GenericRepository<T> : IGenericRepository<T> where T : class, IEntity

{

private readonly DataContext context;

public GenericRepository(DataContext context)

{

     this.context = context;

}

public IQueryable<T> GetAll()

{

     return this.context.Set<T>().AsNoTracking();

}

public async Task<T> GetByIdAsync(int id)

{

     return await this.context.Set<T>()

         .AsNoTracking()

         .FirstOrDefaultAsync(e => e.Id == id);

}

public async Task CreateAsync(T entity)

{

     await this.context.Set<T>().AddAsync(entity);

     await SaveAllAsync();

}

public async Task UpdateAsync(T entity)

{

     this.context.Set<T>().Update(entity);

     await SaveAllAsync();

}

public async Task DeleteAsync(T entity)

{

     this.context.Set<T>().Remove(entity);

     await SaveAllAsync();

}

public async Task<bool> ExistAsync(int id)

{

     return await this.context.Set<T>().AnyAsync(e => e.Id == id);

}

public async Task<bool> SaveAllAsync()

{

     return await this.context.SaveChangesAsync() > 0;

}

}

1. Add the **IProductRepository**:

using Entities;

public interface IProductRepository : IGenericRepository<Product>

{

}

1. Add the **ProductRepository**:

using Entities;

public class ProductRepository : GenericRepository<Product>, IProductRepository

{

public ProductRepository(DataContext context) : base(context)

{

}

}

1. Delete the previous repository (**Repository** and **IRepository**).
2. Modify the **Startup**:

services.AddScoped<IProductRepository, ProductRepository>();

1. Modify the **ProductsController**:

using System.Threading.Tasks;

using Data;

using Data.Entities;

using Helpers;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

public class ProductsController : Controller

{

private readonly IProductRepository productRepository;

private readonly IUserHelper userHelper;

public ProductsController(IProductRepository productRepository, IUserHelper userHelper)

{

     this.productRepository = productRepository;

     this.userHelper = userHelper;

}

// GET: Products

public IActionResult Index()

{

     return View(this.productRepository.GetAll());

}

// GET: Products/Details/5

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

{

     if (id == null)

     {

         return NotFound();

     }

     var product = await this.productRepository.GetByIdAsync(id.Value);

     if (product == null)

     {

         return NotFound();

     }

     return View(product);

}

// GET: Products/Create

public IActionResult Create()

{

     return View();

}

// POST: Products/Create

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Create(Product product)

{

     if (ModelState.IsValid)

     {

         // TODO: Pending to change to: this.User.Identity.Name

         product.User = await this.userHelper.GetUserByEmailAsync("jzuluaga55@gmail.com");

         await this.productRepository.CreateAsync(product);

         return RedirectToAction(nameof(Index));

     }

     return View(product);

}

// GET: Products/Edit/5

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

{

     if (id == null)

     {

         return NotFound();

     }

     var product = await this.productRepository.GetByIdAsync(id.Value);

     if (product == null)

     {

         return NotFound();

     }

     return View(product);

}

// POST: Products/Edit/5

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Edit(Product product)

{

     if (ModelState.IsValid)

     {

         try

         {

             // TODO: Pending to change to: this.User.Identity.Name

             product.User = await this.userHelper.GetUserByEmailAsync("jzuluaga55@gmail.com");

             await this.productRepository.UpdateAsync(product);

         }

         catch (DbUpdateConcurrencyException)

         {

             if (!await this.productRepository.ExistAsync(product.Id))

             {

                 return NotFound();

             }

             else

             {

                 throw;

             }

         }

         return RedirectToAction(nameof(Index));

     }

     return View(product);

}

// GET: Products/Delete/5

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

{

     if (id == null)

     {

         return NotFound();

     }

     var product = await this.productRepository.GetByIdAsync(id.Value);

     if (product == null)

     {

         return NotFound();

     }

     return View(product);

}

// POST: Products/Delete/5

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public async Task<IActionResult> DeleteConfirmed(int id)

{

     var product = await this.productRepository.GetByIdAsync(id);

     await this.productRepository.DeleteAsync(product);

     return RedirectToAction(nameof(Index));

}

}

1. Modify the **SeedDb**:

using System;

using System.Linq;

using System.Threading.Tasks;

using Entities;

using Microsoft.AspNetCore.Identity;

using Shop.Web.Helpers;

public class SeedDb

{

private readonly DataContext context;

private readonly IUserHelper userHelper;

private Random random;

public SeedDb(DataContext context, IUserHelper userHelper)

{

     this.context = context;

     this.userHelper = userHelper;

     this.random = new Random();

}

public async Task SeedAsync()

{

     await this.context.Database.EnsureCreatedAsync();

     // Add user

     var user = await this.userHelper.GetUserByEmail("jzuluaga55@gmail.com");

     if (user == null)

     {

         user = new User

         {

             FirstName = "Juan",

             LastName = "Zuluaga",

             Email = "jzuluaga55@gmail.com",

             UserName = "jzuluaga55@gmail.com",

             PhoneNumber = "3506342747"

         };

         var result = await this.userHelper.AddUser(user, "123456");

         if (result != IdentityResult.Success)

         {

             throw new InvalidOperationException("Could not create the user in seeder");

         }

     }

     // Add products

     if (!this.context.Products.Any())

     {

         this.AddProduct("iPhone X", user);

         this.AddProduct("Magic Mouse", user);

         this.AddProduct("iWatch Series 4", user);

         await this.context.SaveChangesAsync();

     }

}

private void AddProduct(string name, User user)

{

     this.context.Products.Add(new Product

     {

         Name = name,

         Price = this.random.Next(1000),

         IsAvailabe = true,

         Stock = this.random.Next(100),

         User = user

     });

}

}

1. Test it.
2. Now to take advance the this implementation, we’ll create another entity that we’ll use nearly. Add the entity **Country**:

using System.ComponentModel.DataAnnotations;

public class Country : IEntity

{

public int Id { get; set; }

[MaxLength(50, ErrorMessage = "The field {0} only can contain {1} characters length.")]

[Required]

[Display(Name = "Country")]

public string Name { get; set; }

}

1. Add the interface for countries:

using Entities;

public interface ICountryRepository : IGenericRepository<Country>

{

}

1. And add the implementation:

using Entities;

public class CountryRepository : GenericRepository<Country>, ICountryRepository

{

public CountryRepository(DataContext context) : base(context)

{

}

}

1. Add the injection in **StartUp**:

services.AddScoped<ICountryRepository, CountryRepository>();

1. Add the property in the **DataContext**.

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

1. Save all and run those commands to update the database:

dotnet ef migrations add Countries

dotnet ef database update

Or you can run this commands in package manager console:

PM> add-migration Countries

PM> update-database

1. Run the App and test it.

Add API

1. Create the API controller, this is an example (in Web.Controllers.API):

using Data;

using Microsoft.AspNetCore.Mvc;

[Route("api/[Controller]")]

public class ProductsController : Controller

{

private readonly IProductRepository productRepository;

public ProductsController(IProductRepository productRepository)

{

     this.productRepository = productRepository;

}

[HttpGet]

public IActionResult GetProducts()

{

     return this.Ok(this.productRepository.GetAll());

}

}

1. Test it.
2. Publish the App in Azure.







Adding Images

1. In Web the folder **Models** and the class **MainViewModel**.

using System.ComponentModel.DataAnnotations;

using Data.Entities;

using Microsoft.AspNetCore.Http;

public class ProductViewModel : Product

{

[Display(Name = "Image")]

public IFormFile ImageFile { get; set; }

}

1. Modify the **Create** products view:

@model Shop.Web.Models.ProductViewModel

@{

ViewData["Title"] = "Create";

}

<h2>Create</h2>

<h4>Product</h4>

<hr />

<div class="row">

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

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

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

         <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="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="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">

…

1. Add the folder **Products** into **wwwroot/images**.
2. Modify the method **Create** POST and the class **ProductsController**:

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Create(ProductViewModel view)

{

if (ModelState.IsValid)

{

     var path = string.Empty;

     if (view.ImageFile != null && view.ImageFile.Length > 0)

     {

         path = Path.Combine(Directory.GetCurrentDirectory(), "wwwroot\\images\\Products", view.ImageFile.FileName);

         using (var stream = new FileStream(path, FileMode.Create))

         {

             await view.ImageFile.CopyToAsync(stream);

         }

         path = $"~/images/Products/{view.ImageFile.FileName}";

     }

     // TODO: Pending to change to: this.User.Identity.Name

     view.User = await this.userHelper.GetUserByEmail("jzuluaga55@gmail.com");

     var product = this.ToProduct(view, path);

     await this.productRepository.CreateAsync(product);

     return RedirectToAction(nameof(Index));

}

return View(view);

}

private Product ToProduct(ProductViewModel view, string path)

{

return new Product

{

     Id = view.Id,

     ImageUrl = path,

     IsAvailabe = view.IsAvailabe,

     LastPurchase = view.LastPurchase,

     LastSale = view.LastSale,

     Name = view.Name,

     Price = view.Price,

     Stock = view.Stock,

     User = view.User

};

}

1. Modify the products index view:

<td>

@if (!string.IsNullOrEmpty(item.ImageUrl))

{

     <img src="@Url.Content(item.ImageUrl)" alt="Image" style="width:100px;height:150px;max-width: 100%; height: auto;" />

}

</td>

1. Test it what we do until the moment.

7. Now modify the GET and POST Edit in **ProductsController**.

// GET: Products/Edit/5

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

{

if (id == null)

{

     return NotFound();

}

var product = await this.productRepository.GetByIdAsync(id.Value);

if (product == null)

{

     return NotFound();

}

var view = this.ToProducViewModel(product);

return View(view);

}

private ProductViewModel ToProducViewModel(Product product)

{

return new ProductViewModel

{

     Id = product.Id,

     ImageUrl = product.ImageUrl,

     IsAvailabe = product.IsAvailabe,

     LastPurchase = product.LastPurchase,

     LastSale = product.LastSale,

     Name = product.Name,

     Price = product.Price,

     Stock = product.Stock,

     User = product.User

};

}

// POST: Products/Edit/5

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Edit(ProductViewModel view)

{

if (ModelState.IsValid)

{

     try

     {

         var path = view.ImageUrl;

         if (view.ImageFile != null && view.ImageFile.Length > 0)

         {

             path = Path.Combine(Directory.GetCurrentDirectory(), "wwwroot\\images\\Products", view.ImageFile.FileName);

             using (var stream = new FileStream(path, FileMode.Create))

             {

                 await view.ImageFile.CopyToAsync(stream);

             }

             path = $"~/images/Products/{view.ImageFile.FileName}";

         }

         // TODO: Pending to change to: this.User.Identity.Name

         view.User = await this.userHelper.GetUserByEmail("jzuluaga55@gmail.com");

         var product = this.ToProduct(view, path);

         await this.productRepository.UpdateAsync(product);

     }

     catch (DbUpdateConcurrencyException)

     {

         if (!await this.productRepository.ExistAsync(view.Id))

         {

             return NotFound();

         }

         else

         {

             throw;

         }

     }

     return RedirectToAction(nameof(Index));

}

return View(view);

}

8. Modify the edit product view model:

@model Shop.Web.Models.ProductViewModel

@{

ViewData["Title"] = "Edit";

}

<h2>Edit</h2>

<h4>Product</h4>

<hr />

<div class="row">

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

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

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

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

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

         <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="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="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="LastPurchase" class="control-label"></label>

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

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

         </div>

         <div class="form-group">

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

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

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

         </div>

         <div class="form-group">

             <div class="checkbox">

                 <label>

                     <input asp-for="IsAvailabe" /> @Html.DisplayNameFor(model => model.IsAvailabe)

                 </label>

             </div>

         </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">

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

             <a asp-action="Index" class="btn btn-success">Back to List</a>

         </div>

     </form>

</div>

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

     @if (!string.IsNullOrEmpty(Model.ImageUrl))

     {

         <img src="@Url.Content(Model.ImageUrl)" alt="Image" style="width:200px;height:300px;max-width: 100%; height: auto;" />

     }

</div>

</div>

@section Scripts {

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

}

* 1. Test it.

13. Finally add this property to Product entity:

public string ImageFullPath

{

get

{

     if (string.IsNullOrEmpty(this.ImageUrl))

     {

         return null;

     }

     return $"https://shopzulu.azurewebsites.net{this.ImageUrl.Substring(1)}";

}

}

14. Ant test the API and publish the Changes in Azure.

Adding Other Methods To Generic Repository

1. Modify the **IProductRepository**.

using Entities;

using System.Linq;

public interface IProductRepository : IGenericRepository<Product>

{

IQueryable GetAllWithUsers();

}

1. Modify the **ProductRepository**.

using System.Linq;

using Entities;

using Microsoft.EntityFrameworkCore;

public class ProductRepository : GenericRepository<Product>, IProductRepository

{

private readonly DataContext context;

public ProductRepository(DataContext context) : base(context)

{

     this.context = context;

}

public IQueryable GetAllWithUsers()

{

     return this.context.Products.Include(p => p.User).OrderBy(p => p.Name);

}

}

3.Modify the product API Controller.

[HttpGet]

public ActionResult GetProduct()

{

return Ok(this.productRepository.GetAllWithUsers());

}

4. Test it.

Starting with Xamarin Forms

1. Create the folder **ViewModels** and inside it add the class Main**ViewModel**.

public class MainViewModel

{

}

1. Create the folder **Infrastructure** and inside it add the class **InstanceLocator**.

public class InstanceLocator

{

public MainViewModel Main { get; set; }

public InstanceLocator()

{

     this.Main = new MainViewModel();

}

}

3. Modify the **App.xaml** to add an application dictionary:

<?xml version="1.0" encoding="utf-8" ?>

<Application xmlns="http://xamarin.com/schemas/2014/forms"

          xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

          xmlns:infra="clr-namespace:ShopPrep.UIForms.Infrastructure"

          x:Class="ShopPrep.UIForms.App">

<Application.Resources>

     <ResourceDictionary>

         <!-- Locator -->

         <infra:InstanceLocator x:Key="Locator"/>

     </ResourceDictionary>

</Application.Resources>

</Application>

4.Add the folder **Views** and inside it, create the **LoginPage**:

<?xml version="1.0" encoding="utf-8" ?>

<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"

          xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

          x:Class="Shop.UIForms.Views.LoginPage"

          BindingContext="{Binding Main, Source={StaticResource Locator}}"

          Title="Login">

<ContentPage.Content>

     <ScrollView

         BindingContext="{Binding Login}">

         <StackLayout

             Padding="5">

             <Label

                 Text="Email">

             </Label>

             <Entry

                 Keyboard="Email"

                 Placeholder="Enter your email..."

                 Text="{Binding Email}">

             </Entry>

             <Label

                 Text="Password">

             </Label>

             <Entry

                 IsPassword="True"

                 Placeholder="Enter your password..."

                 Text="{Binding Password}">

             </Entry>

             <Button

                 Command="{Binding LoginCommand}"

                 Text="Login">

             </Button>

         </StackLayout>

     </ScrollView>

</ContentPage.Content>

</ContentPage>

5. Add the NuGet **MvvmLigthLibsStd10**. (Plaase search as: **Mvvm Ligth Libs Std**)

* 1. In ViewModels add the class **LoginViewModel**:

using System.Windows.Input;

using GalaSoft.MvvmLight.Command;

using Xamarin.Forms;

public class LoginViewModel

{

public string Email { get; set; }

public string Password { get; set; }

public ICommand LoginCommand => new RelayCommand(this.Login);

private async void Login()

{

     if (string.IsNullOrEmpty(this.Email))

     {

         await Application.Current.MainPage.DisplayAlert("Error", "You must enter an email", "Accept");

         return;

     }

     if (string.IsNullOrEmpty(this.Password))

     {

         await Application.Current.MainPage.DisplayAlert("Error", "You must enter a password", "Accept");

         return;

     }

     if (!this.Email.Equals("jzuluaga55@gmail.com") || !this.Password.Equals("123456"))

     {

         await Application.Current.MainPage.DisplayAlert("Error", "Incorrect user or password", "Accept");

         return;

     }

     await Application.Current.MainPage.DisplayAlert("Ok", "Fuck yeah!!!", "Accept");

}

}

* 1. Modify the **MainViewModel**:

public class MainViewModel

{

public LoginViewModel Login { get; set; }

public MainViewModel()

{

     this.Login = new LoginViewModel();

}

}

* 1. Modify the **App.xaml.cs**:

using Views;

using Xamarin.Forms;

public partial class App : Application

{

public App()

{

     InitializeComponent();

     this.MainPage = new NavigationPage(new LoginPage());

}

protected override void OnStart()

{

     // Handle when your app starts

}

protected override void OnSleep()

{

     // Handle when your app sleeps

}

protected override void OnResume()

{

     // Handle when your app resumes

}

}

* 1. Test it.

Fix Bug to Don’t Replace Images

1. Modify the MVC **ProductsController** in Create and Edit:

if (view.ImageFile != null && view.ImageFile.Length > 0)

{

var guid = Guid.NewGuid().ToString();

var file = $"{guid}.jpg";

path = Path.Combine(

     Directory.GetCurrentDirectory(),

     "wwwroot\\images\\Products",

     file);

using (var stream = new FileStream(path, FileMode.Create))

{

     await view.ImageFile.CopyToAsync(stream);

}

path = $"~/images/Products/{file}";

}

1. Test it.

# Consuming RestFull

1. Add the NuGet **Newtonsoft.Json** to project **Commond**.
2. Add the folder **Models** and inside it those classes (I recommend use the <http://json2csharp.com/> page):

using System;

using Newtonsoft.Json;

public class User

{

[JsonProperty("firstName")]

public string FirstName { get; set; }

[JsonProperty("lastName")]

public string LastName { get; set; }

[JsonProperty("id")]

public Guid Id { get; set; }

[JsonProperty("userName")]

public string UserName { get; set; }

[JsonProperty("normalizedUserName")]

public string NormalizedUserName { get; set; }

[JsonProperty("email")]

public string Email { get; set; }

[JsonProperty("normalizedEmail")]

public string NormalizedEmail { get; set; }

[JsonProperty("emailConfirmed")]

public bool EmailConfirmed { get; set; }

[JsonProperty("passwordHash")]

public string PasswordHash { get; set; }

[JsonProperty("securityStamp")]

public string SecurityStamp { get; set; }

[JsonProperty("concurrencyStamp")]

public Guid ConcurrencyStamp { get; set; }

[JsonProperty("phoneNumber")]

public string PhoneNumber { get; set; }

[JsonProperty("phoneNumberConfirmed")]

public bool PhoneNumberConfirmed { get; set; }

[JsonProperty("twoFactorEnabled")]

public bool TwoFactorEnabled { get; set; }

[JsonProperty("lockoutEnd")]

public object LockoutEnd { get; set; }

[JsonProperty("lockoutEnabled")]

public bool LockoutEnabled { get; set; }

[JsonProperty("accessFailedCount")]

public long AccessFailedCount { get; set; }

}

And:

using Newtonsoft.Json;

using System;

public class Product

{

[JsonProperty("id")]

public int Id { get; set; }

[JsonProperty("name")]

public string Name { get; set; }

[JsonProperty("price")]

public decimal Price { get; set; }

[JsonProperty("imageUrl")]

public string ImageUrl { get; set; }

[JsonProperty("lastPurchase")]

public DateTime LastPurchase { get; set; }

[JsonProperty("lastSale")]

public DateTime LastSale { get; set; }

[JsonProperty("isAvailabe")]

public bool IsAvailabe { get; set; }

[JsonProperty("stock")]

public double Stock { get; set; }

[JsonProperty("user")]

public User User { get; set; }

[JsonProperty("imageFullPath")]

public Uri ImageFullPath { get; set; }

}

1. Add the **Response** model.

public class Response

{

public bool IsSuccess { get; set; }

public string Message { get; set; }

public object Result { get; set; }

}

1. In Common project add the folder **Services** and inside it add the class **ApiService**.

using System;

using System.Collections.Generic;

using System.Net.Http;

using Models;

using Newtonsoft.Json;

using System.Threading.Tasks;

public class ApiService

{

public async Task<Response> GetListAsync<T>(string urlBase, string servicePrefix, string controller)

{

     try

     {

         var client = new HttpClient

         {

             BaseAddress = new Uri(urlBase)

         };

         var url = $"{servicePrefix}{controller}";

         var response = await client.GetAsync(url);

         var result = await response.Content.ReadAsStringAsync();

         if (!response.IsSuccessStatusCode)

         {

             return new Response

             {

                 IsSuccess = false,

                 Message = result,

             };

         }

         var list = JsonConvert.DeserializeObject<List<T>>(result);

         return new Response

         {

             IsSuccess = true,

             Result = list

         };

     }

     catch (Exception ex)

     {

         return new Response

         {

             IsSuccess = false,

             Message = ex.Message

         };

     }

}

}