GenericApp

**INDICE**

1 Matriz de funcionalidad 5

2 Arquitectura 5

3 Repositorio en GitHub 5

4 Crear Solución y Proyectos Common, Web y Prism 6

4.1 Solución en blanco 6

4.2 Proyecto Common 7

4.3 Proyecto Web (Net Core) 8

4.4 Proyectos Prism 10

5 Agregar Nuggets 11

6 Diagrama Entidad Relación 11

7 Creación de la Base de Datos 12

7.1 Entities 12

7.2 DataContext 14

7.3 Cadena de conexión 14

8 CRUD para Countries 15

8.1 Controlador 15

8.2 Mejorar CRUD Countries 21

9 Entities Department, City y Team 26

9.1 DepartmentEntity 26

9.2 CityEntity 27

9.3 CountryEntity 27

9.4 TeamEntity 28

9.5 Actualizamos DataContext 28

9.6 Actualizar Base de Datos 29

10 ViewModels para manejar imágenes y/o combos 29

10.1 CategoryViewModel 29

10.2 ProductViewModel 29

10.3 AddProductImageViewModel 29

10.4 CountryViewModel 30

10.5 TeamViewModel 30

11 Responses 30

11.1 Clase Response 30

11.2 Clase TokenResponse 30

11.3 Clase CountryResponse 31

11.4 Clase DepartmentResponse 31

11.5 Clase CityResponse 31

11.6 Clase TeamResponse 31

11.7 Clase UserResponse 32

11.8 Clase CategoryResponse 32

11.9 Clase ProductImageResponse 33

11.10 Clase ProductResponse 33

12 Requests 33

12.1 Clase ChangePasswordRequest 33

12.2 Clase EmailRequest 34

12.3 Clase TokenRequest 34

12.4 Clase UserRequest 34

13 ImageHelper 34

14 CombosHelper 36

15 ConverterHelper 38

16 IMailHelper 41

17 Maestro detalle MVC para Countries 42

17.1 Vista Index 42

17.2 Métodos Index y Details del controlador CountriesController: 42

17.3 Vista Details del controlador Countries: 43

17.4 Método AddDepartment 45

17.5 Método AddTeam 46

17.6 Método EditDepartment 48

17.7 Método EditTeam 49

17.8 Método DeleteDepartment 50

17.9 Método DeleteTeam 50

17.10 Borrado en cascada de Países que tengan Provincias y/o Equipos 51

17.11 Método DetailsDepartment 51

17.12 Método AddCity 52

17.13 Método EditCity 54

17.14 Método DeleteCity 55

18 Seeder 55

18.1 Clase SeedDb 55

18.2 Inyección de la Clase SeedDb 57

19 CRUD para Categories 58

19.1 Controlador 58

19.2 Vistas 60

20 CRUD para Products 62

20.1 Controlador 62

20.2 Vistas 66

21 Adición de usuarios y roles 70

21.1 Tipos de Usuario 70

21.2 Clase User 70

21.3 Modificar el DataContext 71

21.4 ViewModels para manejar Usuarios 71

21.5 UserHelper 74

21.6 Modificamos el método ConfigureServices del Startup: 76

21.7 Modificación del SeedDb 77

21.8 Confirmar Registro por Email 78

22 Implementación de Login/Logout 79

22.1 AccountController 79

22.2 Vista Login 84

22.3 Vista ConfirmEMail 85

22.4 Vista \_User 85

22.5 Vista \_User2 85

22.6 Vista Register 86

22.7 Vista ChangeUser 88

22.8 Vista ChangePasswordMVC 89

22.9 Vista RecoverPasswordMVC 90

22.10 Vista ResetPassword 90

22.11 Métodos y Vistas para manejo de errores 91

22.12 Modificamos el menú \_Layout: 92

23 Look & Feel 92

24 Administración de Usuarios 93

25 API sin seguridad 97

25.1 CountriesController 97

25.2 CategoriesController 97

25.3 ProductsController 98

25.4 AccountController 98

25.5 Agregamos la configuración en el Startup 102

PROYECTO PRISM 103

26 Inicializar el Xamarin.FFImageLoading.Forms 103

27 Iconos e imágenes 103

28 ApiService 103

29 App 108

29.1 App.xaml 108

29.2 App.xaml.cs 109

30 Icono & Splash 109

31 Settings 109

32 FilesHelper 110

33 RegexHelper 110

34 Models 111

34.1 Clase ElementsList 111

34.2 Clase Menu 111

35 CombosHelper 112

36 ItemViewModels 112

36.1 Clase MenuItemViewModel 112

36.2 Clase ProductItemViewModel 113

37 Recover Password 114

37.1 RecoverPasswordPage 114

37.2 RecoverPasswordPageViewModel 114

38 Registro de Usuario 116

38.1 MainActivity 116

38.2 AndroidManifest 116

38.3 Carpeta xml y archivo file\_paths.xml para sacar fotos 117

38.4 RegisterPage 117

38.1 RegisterPageViewModel 119

39 Cambiar Password 124

39.1 ChangePasswordPage 124

39.2 ChangePasswordPageViewModel 125

40 Modificar Usuario 126

40.1 ModifyUserPage 126

40.2 ModifyUserPageViewModel 128

41 Productos 133

41.1 ProductsPage 133

41.2 ProductsPageViewModel 134

42 MasterDetailPage 136

42.1 GenericAppMasterDetailPage 136

42.2 GenericAppMasterDetailViewModel 137

43 LoginPage 138

43.1 LoginPage 138

43.2 LoginPageViewModel 140

44 HomePage 142

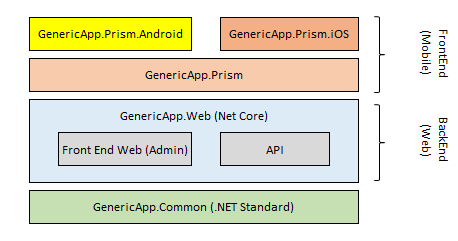
44.1 HomePage 142

44.2 HomePageViewModel 143

# Matriz de funcionalidad

|  |  |  |  |
| --- | --- | --- | --- |
| **Funcionalidad** | **Web** | | **App** |
| **Admin** | **User** | **User** |
| Login | X | X | X |
| Registrarse como usuario |  | X | X |
| Modificar el perfil | X | X | X |
| Recordar contraseña | X | X | X |
| Administrar administradores | X |  |  |

# Arquitectura



# Repositorio en GitHub

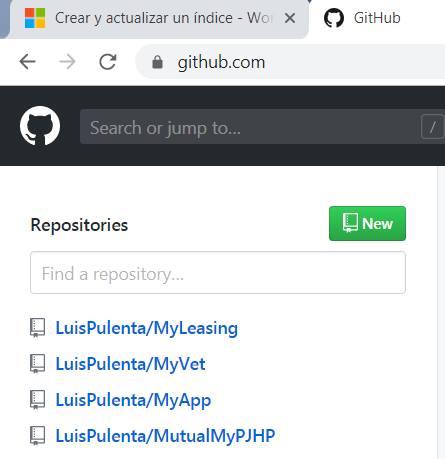
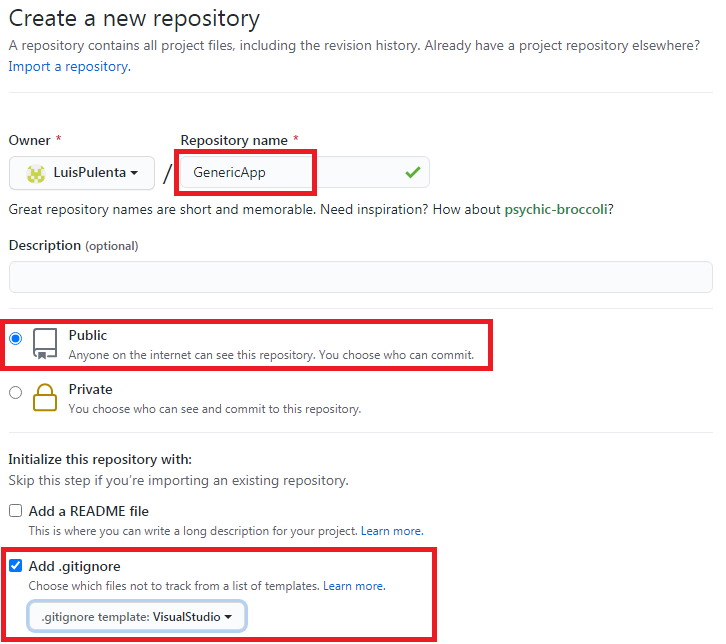
Empezamos haciendo el Repositorio en GitHub

Esta es la Web de GitHub, y mi usuario y contraseña:

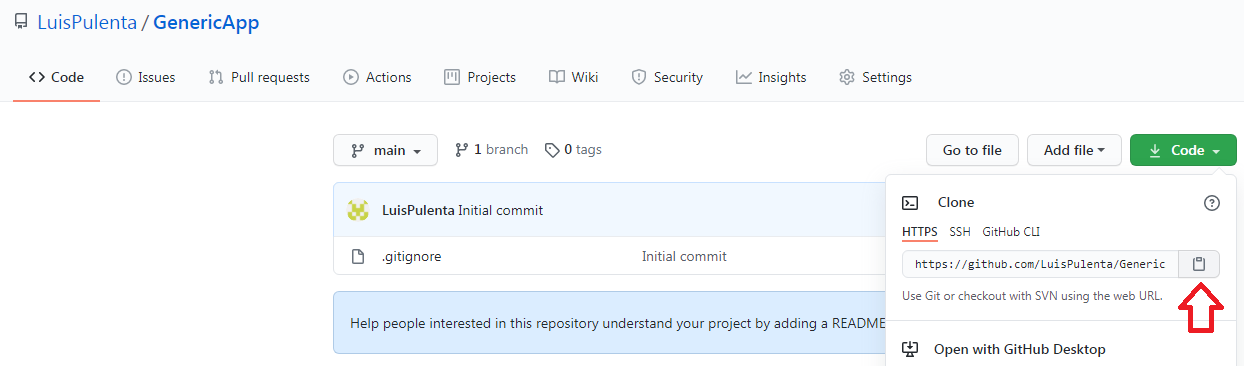
<https://github.com/> **Usuario:** LuisPulenta **Contraseña:** Talleres2306

Una vez dentro de GitHub vamos a “New” y ahí:

* Ponemos el Nombre
* Elegimos Public
* Agregamos un gitignore de tipo visualstudio

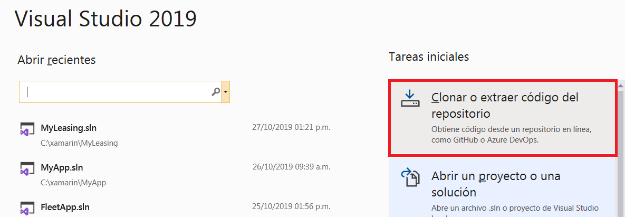
Luego copiamos la dirección para luego clonar en Visual Studio:



# Crear Solución y Proyectos Common, Web y Prism

## Solución en blanco

Abrimos Visual Studio y vamos a la opción Clonar

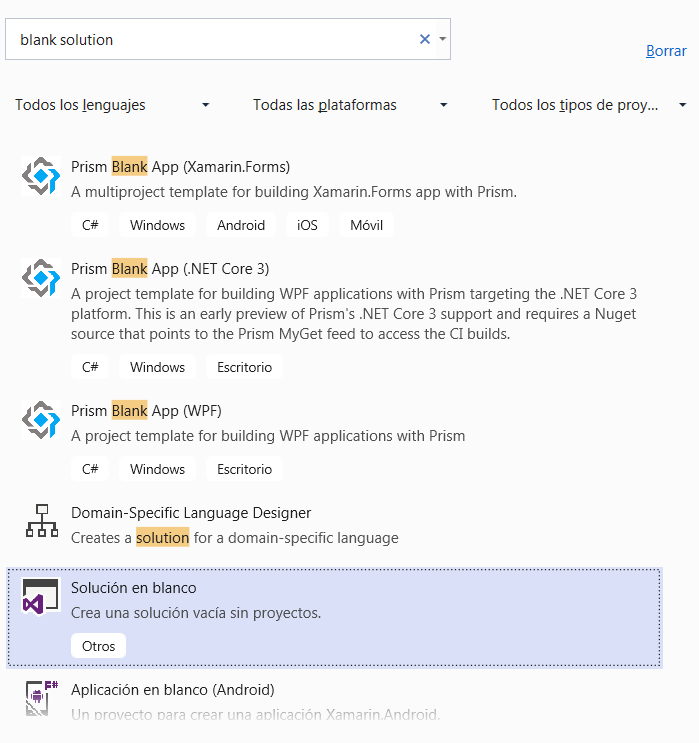
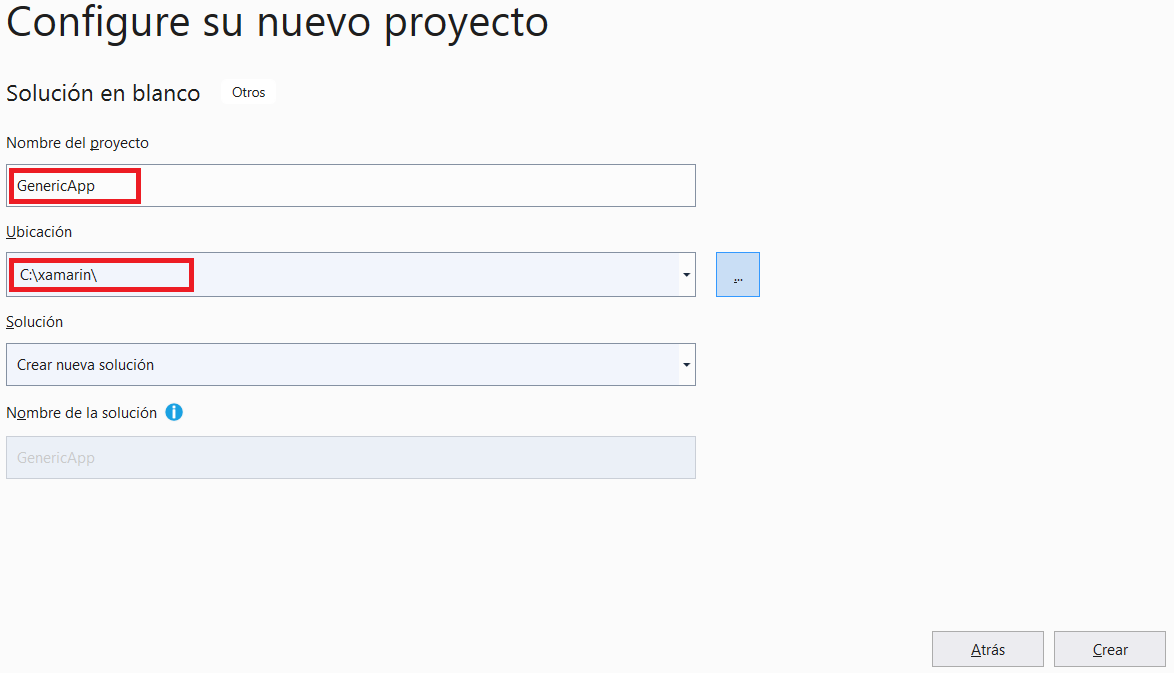




Que sea el mismo nombre del Repositorio. Debe ser una carpeta que no existe

Pegar

Luego hacemos **Archivo-Nuevo-Proyecto** y buscamos plantillas **blank solution** y elegimos Solución en blanco

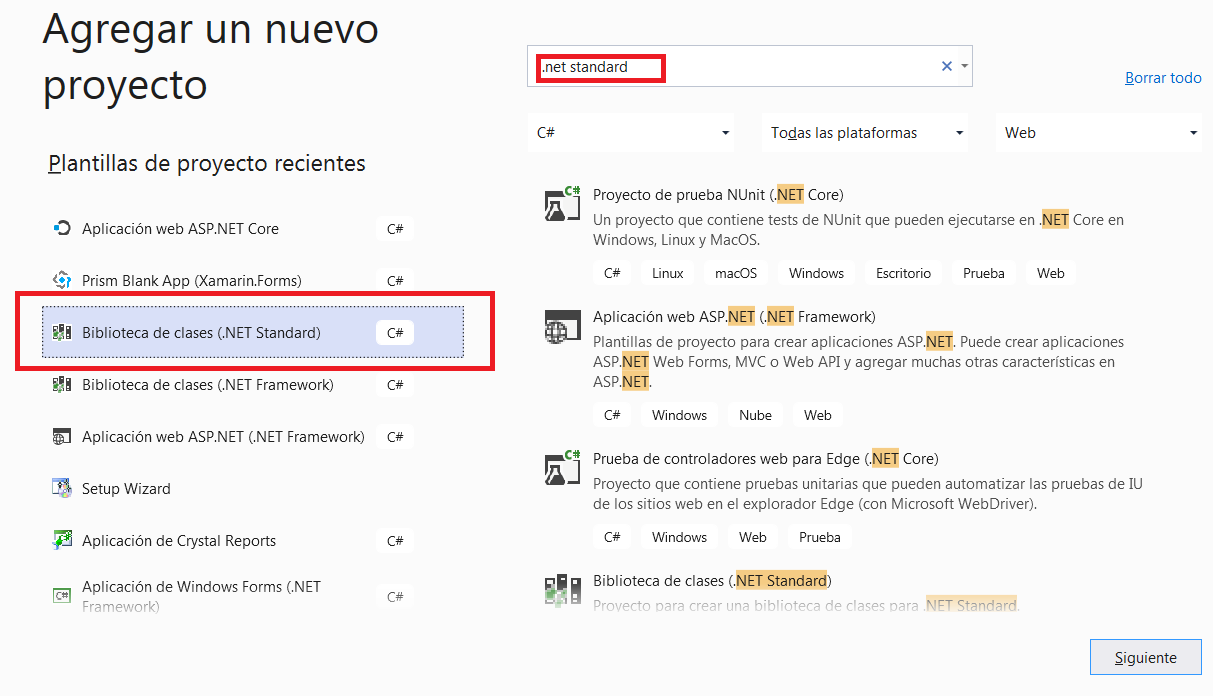
OJO!!! Debe ser la Carpeta CONTENEDORA

Que sea el mismo nombre del Repositorio

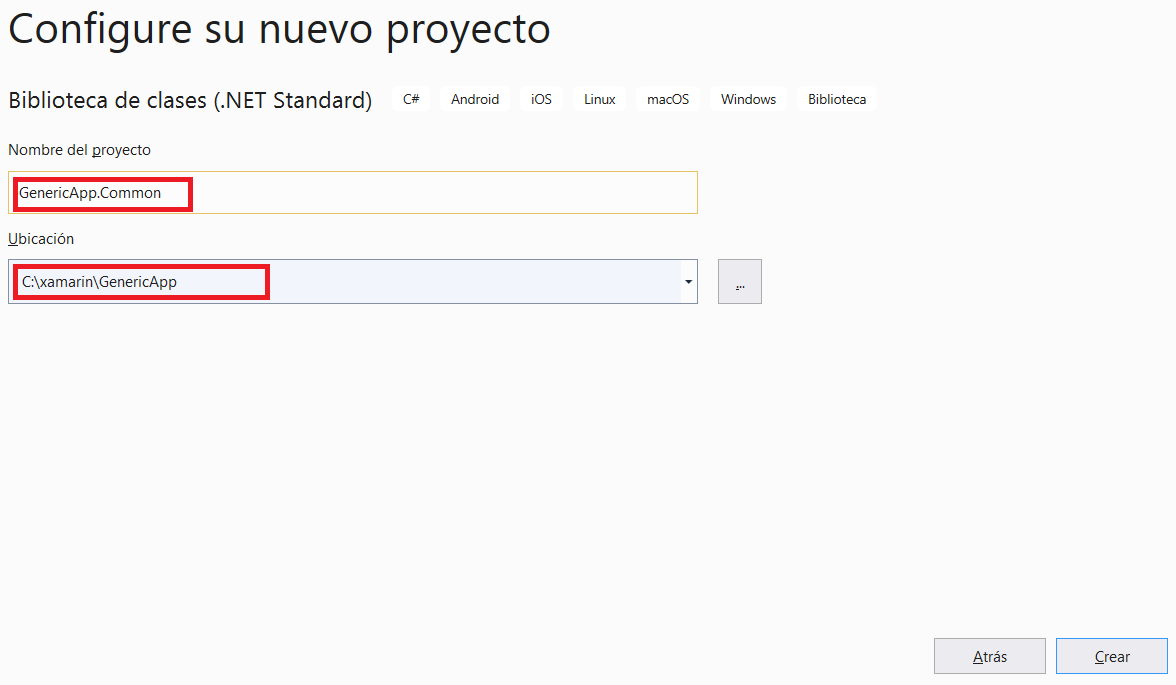
## Proyecto Common

Ahora hacemos clic derecho en la Solución y hacemos Agregar Nuevo Proyecto.

Ponemos **.net standard** en la plantilla y elegimos **Biblioteca de Clases c#**



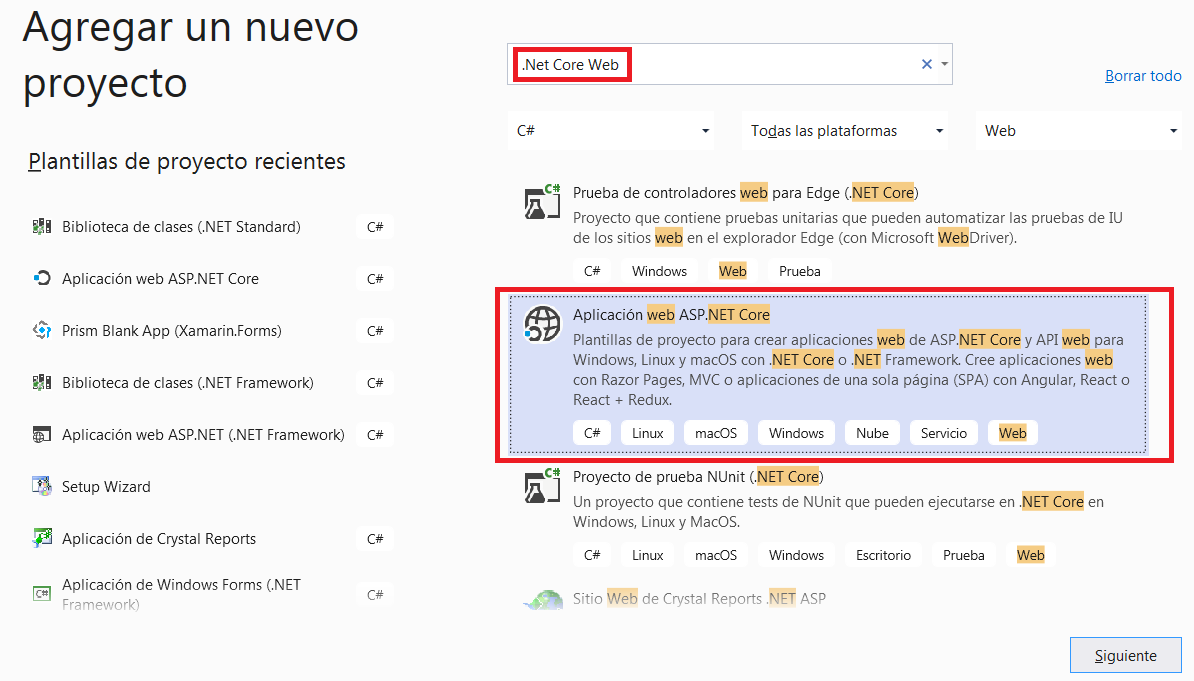
Y la llamamos **GenericApp.Common**

****

(La Clase Class1 que se crea la borramos)

## Proyecto Web (Net Core)

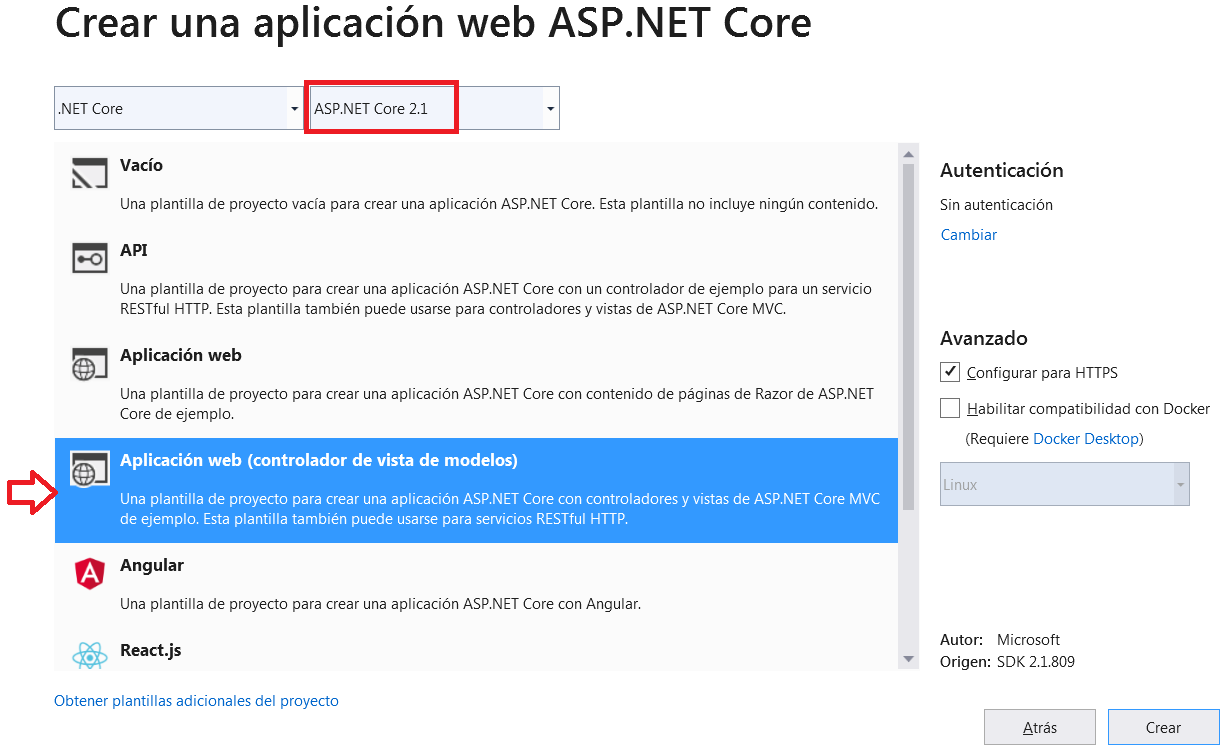
Volvemos a hacer clic derecho en la Solución y ahora ponemos **.Net Core Web** en la plantilla y elegimos **Aplicación Web.Net Core**



Como nombre le ponemos **GenericApp.Web**

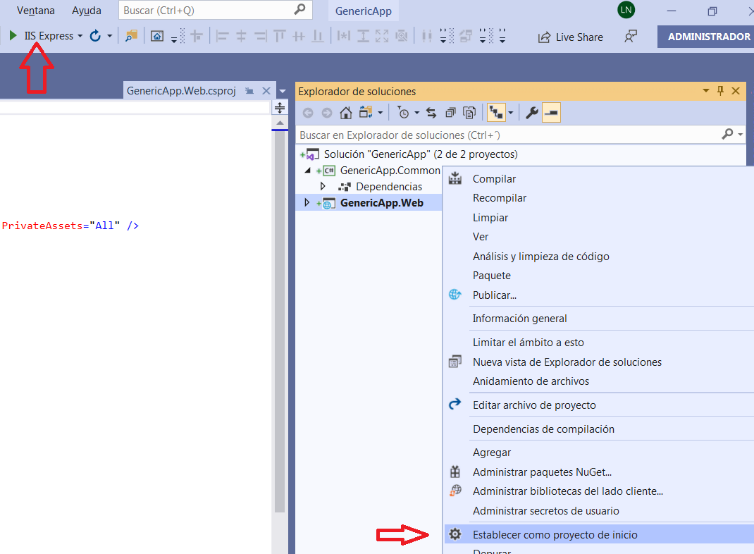
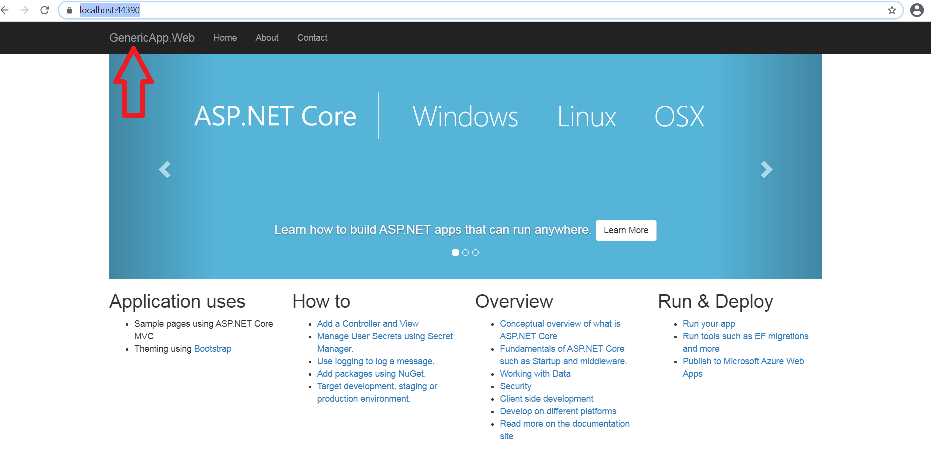


Debe ser Core 2.1 y elegimos la opción Aplicación web (controlador de vista de modelos)



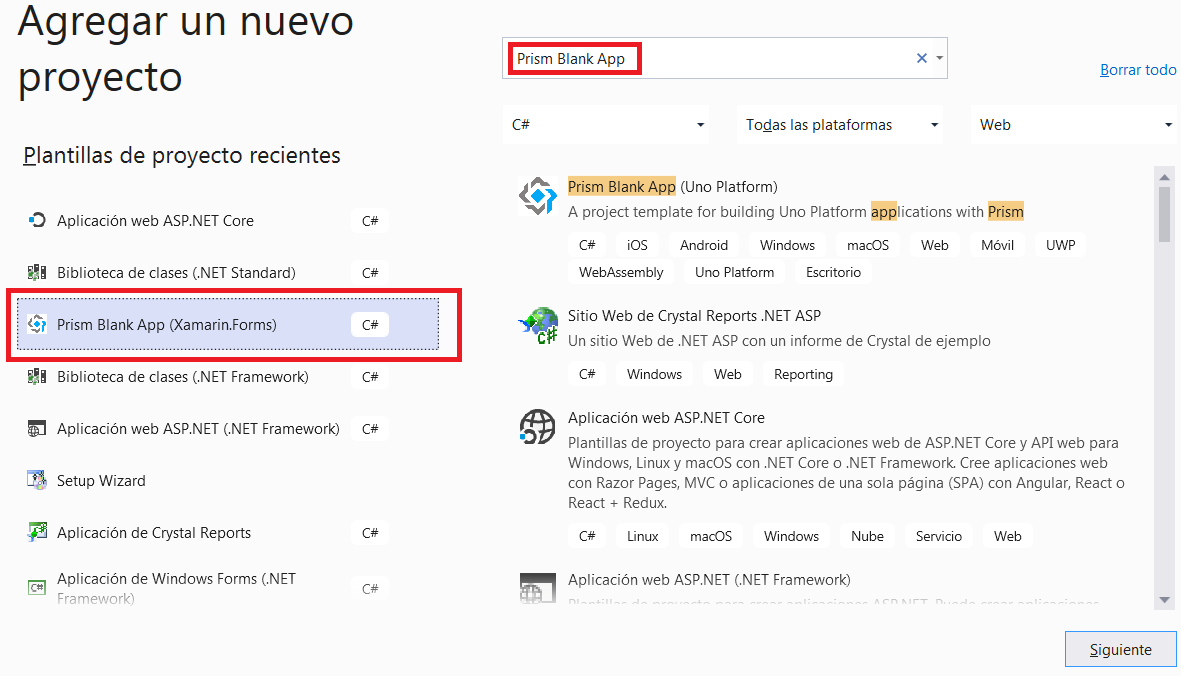
Este Proyecto ya debe arrrancar.

Hacemos clic derecho sobre el mismo y elegimos “Establecer como proyecto de inicio” y luego le damos Play:

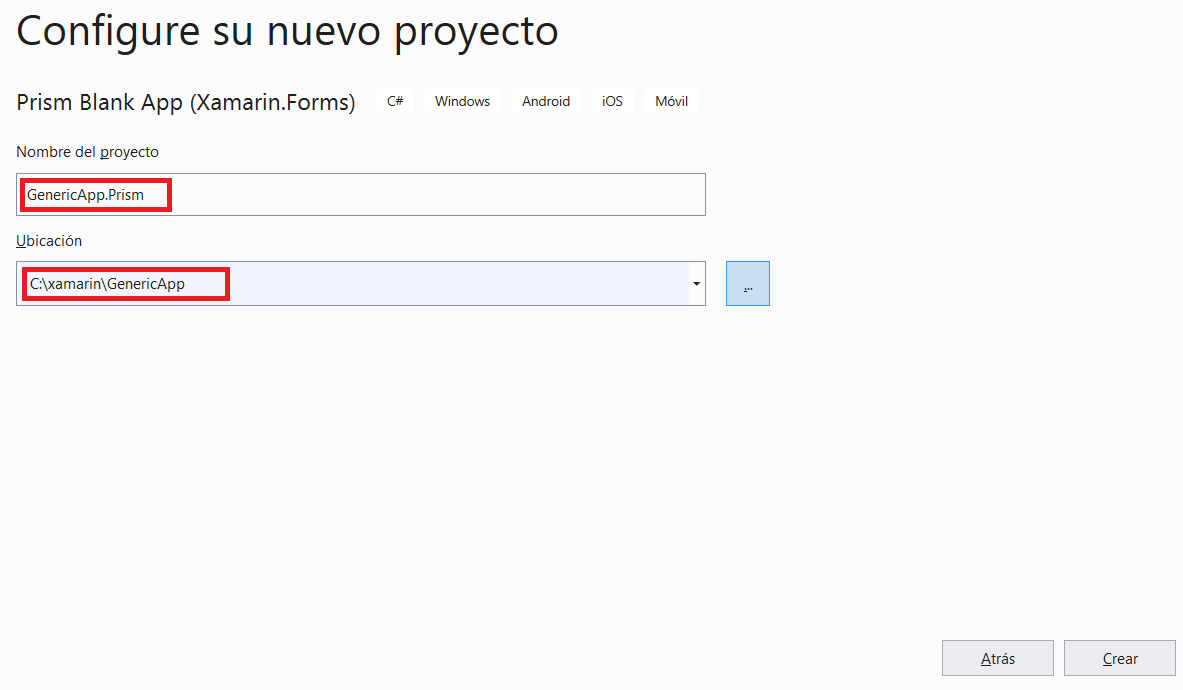
 

## Proyectos Prism

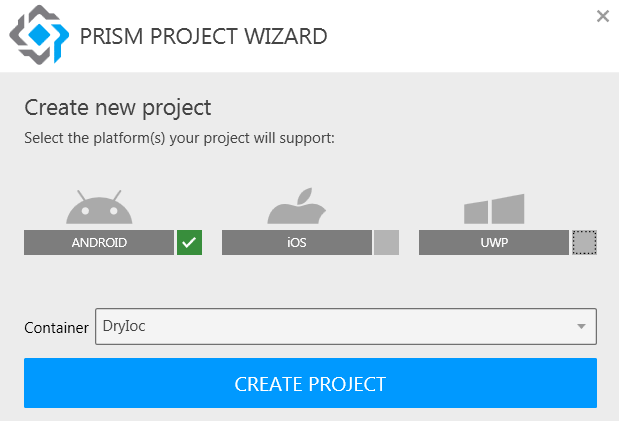
Volvemos a hacer clic derecho en la Solución y ahora ponemos **Prism Blank App** en la plantilla y elegimos **Prism Blank App (Xamarin.Forms)**

****

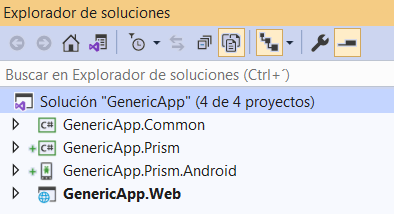
Como nombre le ponemos **GenericApp.Prism**



Elegimos **Container 🡪 DryIoc**



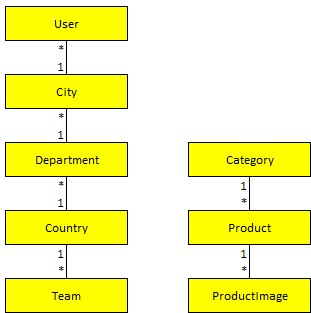
Una vez finalizado, así nos debe quedar la solución con todos sus Proyectos:



# Agregar Nuggets

|  |  |  |
| --- | --- | --- |
| **NUGGET** | **VERSION** | **AGREGAR A** |
| MailKit | 2.5.1 | Proyecto Web |
| Xamarin.FFImageLoading.Forms | 2.4.11.982 | Proyectos móviles |
| Syncfusion.Xamarin.Core | 17.3.0.9 beta | Proyectos móviles |
| Syncfusion.Xamarin.SfBusyIndicator | 17.3.0.9 beta | Proyectos móviles |
| Xam.Plugins.Settings | 4.1.0 beta | Proyecto Common |
| Xamarin.FFImageLoading.Transformations | 2.4.11.982 | Proyectos móviles |
| Xam.Plugin.Media | v.5.0.1 | Proyectos móviles |
| Plugin.Permissions | 6.0.1 | Proyecto Android |
| Plugin.CurrentActivity | 2.1.0.4 | Proyecto Android |
| Xamarin.Android.Support.v4 | 28.0.0.3 | Proyecto Android |
| Xamarin.Forms.Maps | 4.4.0.991265 | Proyectos móviles |
| Xam.Plugin.Geolocator | 4.6.2-beta | Proyectos móviles y Common |
| Plugin.Permissions | 3.0.0.12 | Proyecto Prism |
| Syncfusion.Xamarin.SfRotator | 17.3.0.9 beta | Proyectos móviles |

# Diagrama Entidad Relación



# Creación de la Base de Datos

## Entities

En el proyecto **Web** creamos la carpeta **Data**, dentro creamos la carpeta **Entities** y dentro de esta la clase **CountryEntity**:

|  |  |
| --- | --- |
| **CountryEntity** | **Comentarios** |
| using System.ComponentModel.DataAnnotations;  namespace GenericApp.Web.Data.Entities  {  public class CountryEntity  {  public int Id { get; set; }  [MaxLength(50, ErrorMessage = "El campo {0} debe contener menos de {1} caracteres")]  [Required(ErrorMessage = "El campo {0} es requerido")]  [Display(Name = "País")]  public string Name { get; set; }  [Display(Name = "Bandera")]  public string FlagImagePath { get; set; }  public string FlagImageFullPath => string.IsNullOrEmpty(FlagImagePath)  ? $"http://keypress.serveftp.net:88/GenericAppApi/images/Flags/noimage.png"  : $"http://keypress.serveftp.net:88/GenericAppApi{FlagImagePath.Substring(1)}";  }  } |  |

Dentro de **wwwroot/images** creamos la carpeta **Flags**

Dentro colocamos el archivo **noimage.png**

Creamos la Clase **CategoryEntity**

|  |  |
| --- | --- |
| **CategoryEntity** | **Comentarios** |
| using System.ComponentModel.DataAnnotations;  namespace GenericApp.Web.Data.Entities  {  public class CategoryEntity  {  public int Id { get; set; }  [MaxLength(50, ErrorMessage = "El campo {0} debe contener menos de {1} caracteres.")]  [Required(ErrorMessage = "El campo {0} es requerido")]  [Display(Name = "Categoría")]  public string Name { get; set; }  [Display(Name = "Imagen")]  public string ImagePath { get; set; }  [Display(Name = "Imagen")]  public string ImageFullPath => string.IsNullOrEmpty(ImagePath)  ? $"http://keypress.serveftp.net:88/GenericAppApi/images/Categories/noimage.png"  : $"http://keypress.serveftp.net:88/GenericAppApi{ImagePath.Substring(1)}";  }  } |  |

Dentro de **wwwroot/images** creamos la carpeta **Categories**

Dentro colocamos el archivo **noimage.png**

Creamos la Clase **StateEntity**

|  |  |
| --- | --- |
| **StateEntity** | **Comentarios** |
| using System.ComponentModel.DataAnnotations;  namespace GenericApp.Web.Data.Entities  {  public class StateEntity  {  public int Id { get; set; }  [MaxLength(50, ErrorMessage = "El campo {0} debe contener menos de {1} caracteres.")]  [Required(ErrorMessage = "El campo {0} es requerido")]  [Display(Name = "Estado")]  public string Name { get; set; }  }  } |  |

Creamos la Clase **ProductImageEntity**

|  |  |
| --- | --- |
| **ProductImageEntity** | **Comentarios** |
| using System.ComponentModel.DataAnnotations;  namespace GenericApp.Web.Data.Entities  {  public class ProductImageEntity  {  public int Id { get; set; }  [Display(Name = "Imagen")]  public string ImagePath { get; set; }  public ProductEntity Product { get; set; }  public string ImageFullPath => string.IsNullOrEmpty(ImagePath)  ? $"http://keypress.serveftp.net:88/GenericAppApi/images/Products/noimage.png"  : $"http://keypress.serveftp.net:88/GenericAppApi{ImagePath.Substring(1)}";  }  } |  |

Dentro de **wwwroot/images** creamos la carpeta **Products**

Dentro colocamos el archivo **noimage.png**

Creamos la Clase **ProductEntity**

|  |  |
| --- | --- |
| **ProductEntity** | **Comentarios** |
| using System.Collections.Generic;  using System.ComponentModel;  using System.ComponentModel.DataAnnotations;  using System.Linq;  namespace GenericApp.Web.Data.Entities  {  public class ProductEntity  {  public int Id { get; set; }  [MaxLength(50, ErrorMessage = "El campo {0} debe contener menos de {1} caracteres.")]  [Required(ErrorMessage = "El campo {0} es requerido")]  [Display(Name = "Producto")]  public string Name { get; set; }  [DataType(DataType.MultilineText)]  [Display(Name = "Descripción")]  public string Description { get; set; }  [DisplayFormat(DataFormatString = "{0:C2}")]  [Display(Name = "Precio")]  public decimal Price { get; set; }  [DisplayName("Activo")]  public bool IsActive { get; set; }  [DisplayName("Categoría")]  public CategoryEntity Category { get; set; }  [DisplayName("Latitud")]  [Required(ErrorMessage = "El campo {0} es requerido")]  [DisplayFormat(DataFormatString = "{0:N4}")]  public double Latitude { get; set; }    [DisplayName("Longitud")]  [Required(ErrorMessage = "El campo {0} es requerido")]  [DisplayFormat(DataFormatString = "{0:N4}")]  public double Longitude { get; set; }  [DisplayName("Estado")]  public StateEntity State { get; set; }  public ICollection<ProductImageEntity> ProductImages { get; set; }  [DisplayName("N° Imágenes")]  public int ProductImagesNumber => ProductImages == null ? 0 : ProductImages.Count;  [Display(Name = "Imagen")]  public string ImageFullPath => ProductImages == null || ProductImages.Count == 0  ? $"http://keypress.serveftp.net:88/GenericAppApi/images/Products/noimage.png"  : ProductImages.FirstOrDefault().ImageFullPath;  }  } |  |

## DataContext

Dentro de la carpeta **Web/Data** creamos la clase **DataContext**:

|  |  |
| --- | --- |
| **DataContext** | **Comentarios** |
| using GenericApp.Web.Data.Entities;  using Microsoft.EntityFrameworkCore;  namespace GenericApp.Web.Data  {  public class DataContext : DbContext  {  public DataContext(DbContextOptions<DataContext> options) : base(options)  {  }  public DbSet<CategoryEntity> Categories { get; set; }  public DbSet<CityEntity> Cities { get; set; }  public DbSet<CountryEntity> Countries { get; set; }  public DbSet<DepartmentEntity> Departments { get; set; }  public DbSet<ProductEntity> Products { get; set; }  public DbSet<StateEntity> States { get; set; }  public DbSet<ProductImageEntity> ProductImages { get; set; }  public DbSet<TeamEntity> Teams { get; set; }  protected override void OnModelCreating(ModelBuilder modelBuilder)  {  base.OnModelCreating(modelBuilder);  modelBuilder.Entity<CategoryEntity>()  .HasIndex(t => t.Name)  .IsUnique();  modelBuilder.Entity<CountryEntity>()  .HasIndex(t => t.Name)  .IsUnique();  modelBuilder.Entity<DepartmentEntity>(dep =>  {  dep.HasIndex("Name", "CountryId").IsUnique();  dep.HasOne(d => d.Country).WithMany(c => c.Departments).OnDelete(DeleteBehavior.Cascade);  });  modelBuilder.Entity<ProductEntity>()  .HasIndex(t => t.Name)  .IsUnique();  modelBuilder.Entity<CityEntity>(cit =>  {  cit.HasIndex("Name", "DepartmentId").IsUnique();  cit.HasOne(c => c.Department).WithMany(d => d.Cities).OnDelete(DeleteBehavior.Cascade);  });  modelBuilder.Entity<TeamEntity>(dep =>  {  dep.HasIndex("Name", "CountryId").IsUnique();  dep.HasOne(d => d.Country).WithMany(c => c.Teams).OnDelete(DeleteBehavior.Cascade);  });  }  }  } | Esto es un índice para que no se puedan repetirlos nombres de países |

Creamos la Clase **StateEntity**

|  |  |
| --- | --- |
| **StateEntity** | **Comentarios** |
| using System.ComponentModel.DataAnnotations;  namespace GenericApp.Web.Data.Entities  {  public class StateEntity  {  public int Id { get; set; }  [MaxLength(50, ErrorMessage = "El campo {0} debe contener menos de {1} caracteres.")]  [Required(ErrorMessage = "El campo {0} es requerido")]  [Display(Name = "Estado")]  public string Name { get; set; }  }  } |  |

## Cadena de conexión

Agregue una cadena de conexión al archivo **appsettings.json**:

|  |  |
| --- | --- |
| **appsettings.json** | **Comentarios** |
| {  "Logging": {  "LogLevel": {  "Default": "Warning"  }  },  "AllowedHosts": "\*",  "ConnectionStrings": { "DefaultConnection": "Server=keypress.serveftp.net;Database=LuisGenericApp;User Id=sa;password=sentey14$;Trusted\_Connection=False;MultipleActiveResultSets=true" },  "Tokens": {  "Key": "askñlakds8980234kjklfdosfuioJLJllksfjlk890()=jKLjouUOoiuKLiuioYDtDT#$fCjÑkKÑLkñjlkjlkJLkjlkj78G",  "Issuer": "localhost",  "Audience": "users"  },  "Mail": {  "From": "luissolflix@gmail.com",  "Smtp": "smtp.gmail.com",  "Port": 587,  "Password": "Solflix2306"  }  } | Acá va el nombre de la Base de Datos  Esto es para los Tokens  Esto es para el envío de mails |

Inyectamos la conexión a la base de datos en el archivo **Startup** en el método **ConfigureServices**:

|  |  |
| --- | --- |
| **Startup** | **Comentarios** |
| public void ConfigureServices(IServiceCollection services)  {  services.Configure<CookiePolicyOptions>(options =>  {  options.CheckConsentNeeded = context => true;  options.MinimumSameSitePolicy = SameSiteMode.None;  });  services.AddDbContext<DataContext>(cfg =>  {  cfg.UseSqlServer(Configuration.GetConnectionString("DefaultConnection"));  });  services.AddMvc().SetCompatibilityVersion(CompatibilityVersion.Version\_2\_1);  } |  |

En la Consola del Administrador de Paquetes, corremos los comandos para crear la base de datos de forma local:

PM> add-migration InitialDb

PM> update-database

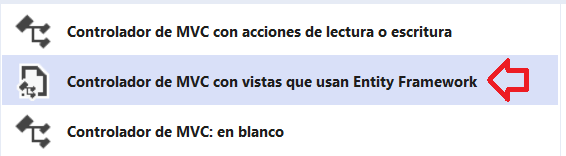
Verificamos en el SQL Server que la Base de Datos se haya creado.

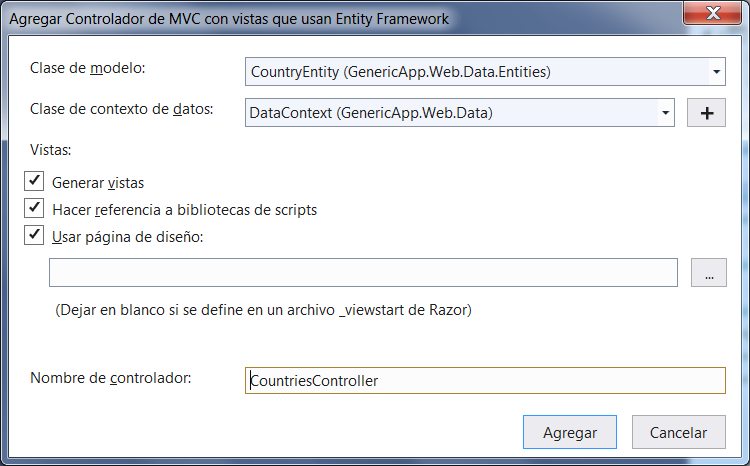
# CRUD para Countries

## Controlador

Creamos un controlador con el asistente para countries.

Clic derecho en Controllers, Agregar, Controlador, y elegimos Controlador de MVC con vistas que usan Entity Framework





Ponemos el nombre

Elegimos DataContext

Elegimos la Entity

|  |  |
| --- | --- |
| **CountriesController** | **Comentarios** |
| using System;  using System.Linq;  using System.Threading.Tasks;  using Microsoft.AspNetCore.Authorization;  using Microsoft.AspNetCore.Mvc;  using Microsoft.EntityFrameworkCore;  using OnSale.Common.Entities;  using OnSale.Web.Data;  namespace OnSale.Web.Controllers  {  [Authorize(Roles = "Admin")]  public class CountriesController : Controller  {  private readonly DataContext \_context;  public CountriesController(DataContext context)  {  \_context = context;  }  // GET: Countries  public async Task<IActionResult> Index()  {  return View(await \_context.Countries  .Include(c => c.Departments)  .ToListAsync());  }  // GET: Countries/Details/5  public async Task<IActionResult> Details(int? id)  {  if (id == null)  {  return NotFound();  }  var country = await \_context.Countries  .Include(c => c.Departments)  .ThenInclude(d=>d.Cities)  .FirstOrDefaultAsync(m => m.Id == id);  if (country == null)  {  return NotFound();  }  return View(country);  }  // GET: Countries/Create  public IActionResult Create()  {  return View(new Country());  }  // POST: Countries/Create  [HttpPost]  [ValidateAntiForgeryToken]  public async Task<IActionResult> Create(Country country)  {  if (ModelState.IsValid)  {  try  {  \_context.Add(country);  await \_context.SaveChangesAsync();  return RedirectToAction(nameof(Index));  }  catch (DbUpdateException dbUpdateException)  {  if (dbUpdateException.InnerException.Message.Contains("duplicate"))  {  ModelState.AddModelError(string.Empty, "There are a record with the same name.");  }  else  {  ModelState.AddModelError(string.Empty, dbUpdateException.InnerException.Message);  }  }  catch (Exception exception)  {  ModelState.AddModelError(string.Empty, exception.Message);  }  }  return View(country);  }  // GET: Countries/Edit/5  public async Task<IActionResult> Edit(int? id)  {  if (id == null)  {  return NotFound();  }  var country = await \_context.Countries.FindAsync(id);  if (country == null)  {  return NotFound();  }  return View(country);  }  // POST: Countries/Edit/5  [HttpPost]  [ValidateAntiForgeryToken]  public async Task<IActionResult> Edit(int id, Country country)  {  if (id != country.Id)  {  return NotFound();  }  if (ModelState.IsValid)  {  try  {  \_context.Update(country);  await \_context.SaveChangesAsync();  return RedirectToAction(nameof(Index));  }  catch (DbUpdateException dbUpdateException)  {  if (dbUpdateException.InnerException.Message.Contains("duplicate"))  {  ModelState.AddModelError(string.Empty, "There are a record with the same name.");  }  else  {  ModelState.AddModelError(string.Empty, dbUpdateException.InnerException.Message);  }  }  catch (Exception exception)  {  ModelState.AddModelError(string.Empty, exception.Message);  }  }  return View(country);  }  // GET: Countries/Delete/5  public async Task<IActionResult> Delete(int? id)  {  if (id == null)  {  return NotFound();  }  Country country = await \_context.Countries  .Include(c => c.Departments)  .ThenInclude(d => d.Cities)  .FirstOrDefaultAsync(m => m.Id == id);  if (country == null)  {  return NotFound();  }  \_context.Countries.Remove(country);  await \_context.SaveChangesAsync();  return RedirectToAction(nameof(Index));  }  private bool CountryExists(int id)  {  return \_context.Countries.Any(e => e.Id == id);  }  public async Task<IActionResult> AddDepartment(int? id)  {  if (id == null)  {  return NotFound();  }  Country country = await \_context.Countries.FindAsync(id);  if (country == null)  {  return NotFound();  }  Department model = new Department { IdCountry = country.Id };  return View(model);  }  [HttpPost]  [ValidateAntiForgeryToken]  public async Task<IActionResult> AddDepartment(Department department)  {  if (ModelState.IsValid)  {  Country country = await \_context.Countries  .Include(c => c.Departments)  .FirstOrDefaultAsync(c => c.Id == department.IdCountry);  if (country == null)  {  return NotFound();  }  try  {  department.Id = 0;  country.Departments.Add(department);  \_context.Update(country);  await \_context.SaveChangesAsync();  return RedirectToAction($"{nameof(Details)}/{country.Id}");  }  catch (DbUpdateException dbUpdateException)  {  if (dbUpdateException.InnerException.Message.Contains("duplicate"))  {  ModelState.AddModelError(string.Empty, "There are a record with the same name.");  }  else  {  ModelState.AddModelError(string.Empty, dbUpdateException.InnerException.Message);  }  }  catch (Exception exception)  {  ModelState.AddModelError(string.Empty, exception.Message);  }  }  return View(department);  }  public async Task<IActionResult> EditDepartment(int? id)  {  if (id == null)  {  return NotFound();  }  Department department = await \_context.Departments.FindAsync(id);  if (department == null)  {  return NotFound();  }  Country country = await \_context.Countries.FirstOrDefaultAsync(c => c.Departments.FirstOrDefault(d => d.Id == department.Id) != null);  department.IdCountry = country.Id;  return View(department);  }  [HttpPost]  [ValidateAntiForgeryToken]  public async Task<IActionResult> EditDepartment(Department department)  {  if (ModelState.IsValid)  {  try  {  \_context.Update(department);  await \_context.SaveChangesAsync();  return RedirectToAction($"{nameof(Details)}/{department.IdCountry}");  }  catch (DbUpdateException dbUpdateException)  {  if (dbUpdateException.InnerException.Message.Contains("duplicate"))  {  ModelState.AddModelError(string.Empty, "There are a record with the same name.");  }  else  {  ModelState.AddModelError(string.Empty, dbUpdateException.InnerException.Message);  }  }  catch (Exception exception)  {  ModelState.AddModelError(string.Empty, exception.Message);  }  }  return View(department);  }  public async Task<IActionResult> DeleteDepartment(int? id)  {  if (id == null)  {  return NotFound();  }  Department department = await \_context.Departments  .Include(d => d.Cities)  .FirstOrDefaultAsync(m => m.Id == id);  if (department == null)  {  return NotFound();  }  Country country = await \_context.Countries.FirstOrDefaultAsync(c => c.Departments.FirstOrDefault(d => d.Id == department.Id) != null);  \_context.Departments.Remove(department);  await \_context.SaveChangesAsync();  return RedirectToAction($"{nameof(Details)}/{country.Id}");  }  public async Task<IActionResult> DetailsDepartment(int? id)  {  if (id == null)  {  return NotFound();  }  Department department = await \_context.Departments  .Include(d => d.Cities)  .FirstOrDefaultAsync(m => m.Id == id);  if (department == null)  {  return NotFound();  }  Country country = await \_context.Countries.FirstOrDefaultAsync(c => c.Departments.FirstOrDefault(d => d.Id == department.Id) != null);  department.IdCountry = country.Id;  return View(department);  }  public async Task<IActionResult> AddCity(int? id)  {  if (id == null)  {  return NotFound();  }  Department department = await \_context.Departments.FindAsync(id);  if (department == null)  {  return NotFound();  }  City model = new City { IdDepartment = department.Id };  return View(model);  }  [HttpPost]  [ValidateAntiForgeryToken]  public async Task<IActionResult> AddCity(City city)  {  if (ModelState.IsValid)  {  Department department = await \_context.Departments  .Include(d => d.Cities)  .FirstOrDefaultAsync(c => c.Id == city.IdDepartment);  if (department == null)  {  return NotFound();  }  try  {  city.Id = 0;  department.Cities.Add(city);  \_context.Update(department);  await \_context.SaveChangesAsync();  return RedirectToAction($"{nameof(DetailsDepartment)}/{department.Id}");  }  catch (DbUpdateException dbUpdateException)  {  if (dbUpdateException.InnerException.Message.Contains("duplicate"))  {  ModelState.AddModelError(string.Empty, "There are a record with the same name.");  }  else  {  ModelState.AddModelError(string.Empty, dbUpdateException.InnerException.Message);  }  }  catch (Exception exception)  {  ModelState.AddModelError(string.Empty, exception.Message);  }  }  return View(city);  }  public async Task<IActionResult> EditCity(int? id)  {  if (id == null)  {  return NotFound();  }  City city = await \_context.Cities.FindAsync(id);  if (city == null)  {  return NotFound();  }  Department department = await \_context.Departments.FirstOrDefaultAsync(d => d.Cities.FirstOrDefault(c => c.Id == city.Id) != null);  city.IdDepartment = department.Id;  return View(city);  }  [HttpPost]  [ValidateAntiForgeryToken]  public async Task<IActionResult> EditCity(City city)  {  if (ModelState.IsValid)  {  try  {  \_context.Update(city);  await \_context.SaveChangesAsync();  return RedirectToAction($"{nameof(DetailsDepartment)}/{city.IdDepartment}");  }  catch (DbUpdateException dbUpdateException)  {  if (dbUpdateException.InnerException.Message.Contains("duplicate"))  {  ModelState.AddModelError(string.Empty, "There are a record with the same name.");  }  else  {  ModelState.AddModelError(string.Empty, dbUpdateException.InnerException.Message);  }  }  catch (Exception exception)  {  ModelState.AddModelError(string.Empty, exception.Message);  }  }  return View(city);  }  public async Task<IActionResult> DeleteCity(int? id)  {  if (id == null)  {  return NotFound();  }  City city = await \_context.Cities  .FirstOrDefaultAsync(m => m.Id == id);  if (city == null)  {  return NotFound();  }  Department department = await \_context.Departments.FirstOrDefaultAsync(d => d.Cities.FirstOrDefault(c => c.Id == city.Id) != null);  \_context.Cities.Remove(city);  await \_context.SaveChangesAsync();  return RedirectToAction($"{nameof(DetailsDepartment)}/{department.Id}");  }  }  } | LISTA DE PAISES  CREA UN PAIS (GET)  CREA UN PAIS (POST)  EDITA UN PAIS (GET)  EDITA UN PAIS (POST)  BORRA UN PAIS (GET)  BORRA UN PAIS (POST) |

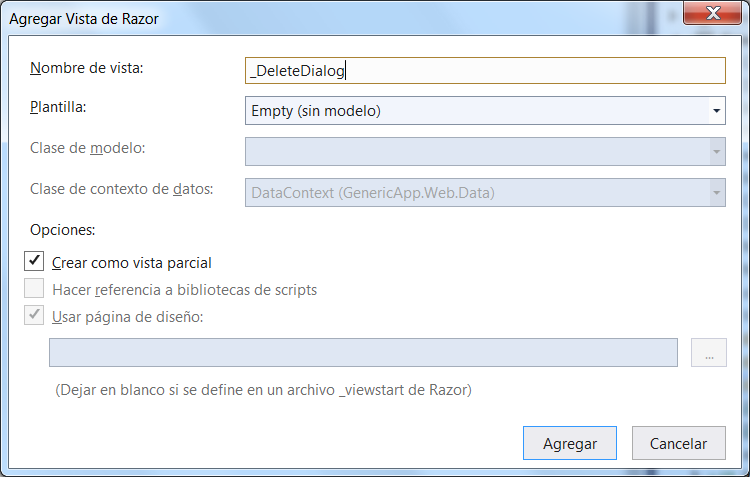
Modificamos el menú para poder probar lo que llevamos.

En el Proyecto **Web**, en **Views/Shared/\_Layout.cshtml** agregamos:

|  |  |
| --- | --- |
| **\_Layout.cshtml** | **Comentarios** |
| <div class="navbar-collapse collapse">  <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**="Countries" **asp-action**="Index">Países</a></li>  </ul>  </div> |  |

## Mejorar CRUD Countries

En **Views/Shared** hacemos clic derecho, Agregar, Vista, Vista de Razor, y la llamamos **\_DeleteDialog**:



Tildamos aquí

Ponemos el nombre

|  |  |
| --- | --- |
| **\_DeleteDialog** | **Comentarios** |
| <div class="modal fade" id="deleteDialog" tabindex="-1" role="dialog" aria-labelledby="exampleModalLabel" aria-hidden="true">  <div class="modal-dialog" role="document">  <div class="modal-content">  <div class="modal-header">  <h5 class="modal-title" id="exampleModalLabel">Borrar Registro</h5>  </div>  <div class="modal-body">  <p>Está seguro de borrar este registro?</p>  </div>  <div class="modal-footer">  <button type="button" class="btn btn-primary" data-dismiss="modal">No</button>  <button type="button" class="btn btn-danger" id="btnYesDelete">Si</button>  </div>  </div>  </div>  </div> |  |

En **wwwroot/js** hacemos clicl derecho, agregar, Nuevo elemento, Archivo JavaScript, y lo llamamos **deleteDialog.js**

|  |  |
| --- | --- |
| **deleteDialog.js** | **Comentarios** |
| (function (soccerDeleteDialog) {  var methods = {  "openModal": openModal,  "deleteItem": deleteItem  };  var item\_to\_delete;  /\*\*  \* Open a modal by class name or Id.  \*  \* @return string id item.  \*/  function openModal(modalName, classOrId, sourceEvent, deletePath, eventClassOrId) {  var textEvent;  if (classOrId) {  textEvent = "." + modalName;  } else {  textEvent = "#" + modalName;  }  $(textEvent).click((e) => {  item\_to\_delete = e.currentTarget.dataset.id;  deleteItem(sourceEvent, deletePath, eventClassOrId);  });  }  /\*\*  \* Path to delete an item.  \*  \* @return void.  \*/  function deleteItem(sourceEvent, deletePath, eventClassOrId) {  var textEvent;  if (eventClassOrId) {  textEvent = "." + sourceEvent;  } else {  textEvent = "#" + sourceEvent;  }  $(textEvent).click(function () {  window.location.href = deletePath + item\_to\_delete;  });  }  soccerDeleteDialog.sc\_deleteDialog = methods;  })(window); |  |

En el Controlador CountriesController, modificamos la acción **Delete.**

Borramos las acciones Delete Get y Delete Post y ponemos:

|  |  |
| --- | --- |
| **CountriesController** | **Comentarios** |
| // POST: Countries/Delete/5  public async Task<IActionResult> Delete(int? id)  {  if (id == null)  {  return NotFound();  }  CountryEntity country = await \_context.Countries  .FirstOrDefaultAsync(m => m.Id == id);  if (country == null)  {  return NotFound();  }  \_context.Countries.Remove(country);  await \_context.SaveChangesAsync();  return RedirectToAction(nameof(Index));  } |  |

Modificamos el resto de vistas colocando los botones con estilos.

Vista **Index**:

|  |  |
| --- | --- |
| **Index** | **Comentarios** |
| @model IEnumerable<GenericApp.Web.Data.Entities.CountryEntity>  @{  ViewData["Title"] = "Index";  }  <h2>  <img src="/images/Flags/countries.png" alt="Image" style="width:70;height:70px;max-height: 100%; width: auto;" />  Países  </h2>  <link rel="stylesheet" href="https://cdn.datatables.net/1.10.19/css/jquery.dataTables.min.css" />  <br />  <p>  <a **asp-action**="Create" class="btn btn-primary"><i class="glyphicon glyphicon-plus"></i> Agregar nuevo país</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.FlagImagePath)  </th>  <th>  @Html.DisplayNameFor(model => model.DepartmentsNumber)  </th>  <th>  @Html.DisplayNameFor(model => model.TeamsNumber)  </th>  <th></th>  </tr>  </thead>  <tbody>  @foreach (var item in Model)  {  <tr>  <td>  @Html.DisplayFor(modelItem => item.Name)  </td>  <td>  @if (!string.IsNullOrEmpty(item.FlagImagePath))  {  <img src="@Url.Content(item.FlagImagePath)" alt="Image" style="width:50px;height:50px;max-width: 100%; height: auto;" />  }  else  {  <img src="@Url.Content(item.FlagImagePath)" alt="Image" style="width:50px;height:50px;max-width: 100%; height: auto;" />  }  </td>  <td>  @Html.DisplayFor(modelItem => item.DepartmentsNumber)  </td>  <td>  @Html.DisplayFor(modelItem => item.TeamsNumber)  </td>  <td>  <a **asp-action**="Edit" **asp-route-id**="@item.Id" class="btn btn-warning"><i class="glyphicon glyphicon-pencil"></i></a>  <a **asp-action**="Details" **asp-route-id**="@item.Id" class="btn btn-info"><i class="glyphicon glyphicon-align-justify"></i></a>  <button data-id="@item.Id" class="btn btn-danger deleteItem" data-toggle="modal" data-target="#deleteDialog"><i class="glyphicon glyphicon-trash"></i></button>  </td>  </tr>  }  </tbody>  </table>  </div>  </div>  </div>  </div>  <**partial** **name**="\_DeleteDialog" />  @section Scripts {  @{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}  <script src="//cdn.datatables.net/1.10.19/js/jquery.dataTables.min.js"></script>  <script src="/js/deleteDialog.js"></script>  <script type="text/javascript">  $(document).ready(function () {  $('#MyTable').DataTable();  // Delete item  sc\_deleteDialog.openModal('deleteItem', true, 'btnYesDelete', '/Countries/Delete/', false);  });  </script>  } |  |

Adicionamos una validación al controlador para evitar errores de duplicados.

En el Controlador **CountriesController**, en la acción **Create Post**, hacemos:

|  |  |
| --- | --- |
| **CountriesController** | **Comentarios** |
| // POST: Countries/Create  [HttpPost]  [ValidateAntiForgeryToken]  public async Task<IActionResult> Create(CountryEntity country)  {  if (ModelState.IsValid)  {  try  {  \_context.Add(country);  await \_context.SaveChangesAsync();  return RedirectToAction(nameof(Index));  }  catch (DbUpdateException dbUpdateException)  {  if (dbUpdateException.InnerException.Message.Contains("duplicate"))  {  ModelState.AddModelError(string.Empty, "Hay un registro con el mismo nombre.");  }  else  {  ModelState.AddModelError(string.Empty, dbUpdateException.InnerException.Message);  }  }  catch (Exception exception)  {  ModelState.AddModelError(string.Empty, exception.Message);  }  }  return View(country);  } |  |

Hacemos algo similar en la acción **Edit Post**:

|  |  |
| --- | --- |
| **CountriesController** | **Comentarios** |
| // POST: Countries/Edit/5  [HttpPost]  [ValidateAntiForgeryToken]  public async Task<IActionResult> Edit(int id, CountryEntity country)  {  if (id != country.Id)  {  return NotFound();  }  if (ModelState.IsValid)  {  try  {  \_context.Update(country);  await \_context.SaveChangesAsync();  return RedirectToAction(nameof(Index));  }  catch (DbUpdateException dbUpdateException)  {  if (dbUpdateException.InnerException.Message.Contains("duplicate"))  {  ModelState.AddModelError(string.Empty, "Hay un registro con el mismo nombre.");  }  else  {  ModelState.AddModelError(string.Empty, dbUpdateException.InnerException.Message);  }  }  catch (Exception exception)  {  ModelState.AddModelError(string.Empty, exception.Message);  }  }  return View(country);  } |  |

Creamos la Vista Parcial **\_Country**

|  |  |
| --- | --- |
| **\_Country** | **Comentarios** |
| @model GenericApp.Web.Data.Entities.CountryEntity  <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> |  |

Modificamos la **Vista Create**:

|  |  |
| --- | --- |
| **Create** | **Comentarios** |
| @model GenericApp.Web.Models.CountryViewModel  @{  ViewData["Title"] = "Create";  }  <h2>  <img src="/images/Flags/addcountry.png" alt="Image" style="width:70;height:70px;max-height: 100%; width: auto;" />  Agregar nuevo País  </h2>  <h4></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>  <**partial** **name**="\_Country" />  <div class="form-group">  <input type="submit" value="Crear" class="btn btn-primary" />  <a **asp-action**="Index" class="btn btn-success">Regresar</a>  </div>  </form>  </div>  </div>  @section Scripts {  @{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}  } |  |

Modificamos la **Vista Edit**:

|  |  |
| --- | --- |
| **Edit** | **Comentarios** |
| @model GenericApp.Web.Data.Entities.CountryEntity  @{  ViewData["Title"] = "Edit";  }  <h2>Editar País</h2>  <h4></h4>  <hr />  <div class="row">  <div class="col-md-4">  <form **asp-action**="Edit">  <div **asp-validation-summary**="ModelOnly" class="text-danger"></div>  <input **type**="hidden" **asp-for**="Id" />  <**partial** **name**="\_Country" />  <div class="form-group">  <input type="submit" value="Grabar" class="btn btn-primary" />  <a **asp-action**="Index" class="btn btn-success">Regresar</a>  </div>  </form>  </div>  </div>  @section Scripts {  @{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}  } |  |

Eliminamos la **Vista Delete**

# Entities Department, City y Team

## DepartmentEntity

Creamos en **Web/Data/Entities** la Clase **DepartmentEntity**

|  |  |
| --- | --- |
| **DepartmentEntity** | **Comentarios** |
| using Newtonsoft.Json;  using System.Collections.Generic;  using System.ComponentModel;  using System.ComponentModel.DataAnnotations;  using System.ComponentModel.DataAnnotations.Schema;  namespace GenericApp.Web.Data.Entities  {  public class DepartmentEntity  {  public int Id { get; set; }  [MaxLength(50, ErrorMessage = "El campo {0} debe contener menos de {1} caracteres.")]  [Required(ErrorMessage = "El campo {0} es requerido")]  [Display(Name = "Provincia")]  public string Name { get; set; }  public ICollection<CityEntity> Cities { get; set; }  [DisplayName("N° Ciudades")]  public int CitiesNumber => Cities == null ? 0 : Cities.Count;  [JsonIgnore]  [NotMapped]  public int IdCountry { get; set; }  [JsonIgnore]  public CountryEntity Country { get; set; }  }  } |  |

## CityEntity

Creamos en **Web/Data/Entities** la Clase **CityEntity**

|  |  |
| --- | --- |
| **CityEntity** | **Comentarios** |
| using Newtonsoft.Json;  using System.ComponentModel.DataAnnotations;  using System.ComponentModel.DataAnnotations.Schema;  namespace GenericApp.Web.Data.Entities  {  public class CityEntity  {  public int Id { get; set; }  [MaxLength(50, ErrorMessage = "El campo {0} debe contener menos de {1} caracteres.")]  [Required(ErrorMessage = "El campo {0} es requerido")]  [Display(Name = "Ciudad")]  public string Name { get; set; }  [JsonIgnore]  [NotMapped]  public int IdDepartment { get; set; }  [JsonIgnore]  public DepartmentEntity Department { get; set; }  }  } |  |

## CountryEntity

Agregamos a **CountryEntity**

|  |  |
| --- | --- |
| **CountryEntity** | **Comentarios** |
| using System.Collections.Generic;  using System.ComponentModel;  using System.ComponentModel.DataAnnotations;  namespace GenericApp.Web.Data.Entities  {  public class CountryEntity  {  public int Id { get; set; }  [MaxLength(50, ErrorMessage = "El campo {0} debe contener menos de {1} caracteres")]  [Required(ErrorMessage = "El campo {0} es requerido")]  [Display(Name = "País")]  public string Name { get; set; }  [Display(Name = "Bandera")]  public string FlagImagePath { get; set; }  public ICollection<DepartmentEntity> Departments { get; set; }  public ICollection<TeamEntity> Teams { get; set; }  [DisplayName("N° Provincias")]  public int DepartmentsNumber => Departments == null ? 0 : Departments.Count;  [DisplayName("N° Equipos")]  public int TeamsNumber => Teams == null ? 0 : Teams.Count;  public string FlagImageFullPath => string.IsNullOrEmpty(FlagImagePath)  ? $"http://keypress.serveftp.net:88/GenericAppApi/images/Flags/noimage.png"  : $"http://keypress.serveftp.net:88/GenericAppApi{FlagImagePath.Substring(1)}";  }  } |  |

## TeamEntity

Creamos en **Web/Data/Entities** la Clase **TeamEntity**

|  |  |
| --- | --- |
| **TeamEntity** | **Comentarios** |
| using Newtonsoft.Json;  using System.ComponentModel.DataAnnotations;  using System.ComponentModel.DataAnnotations.Schema;  namespace GenericApp.Web.Data.Entities  {  public class TeamEntity  {  public int Id { get; set; }  [MaxLength(50, ErrorMessage = "El campo {0} debe contener menos de {1} caracteres.")]  [Required(ErrorMessage = "El campo {0} es requerido")]  [Display(Name = "Equipo")]  public string Name { get; set; }  [JsonIgnore]  [NotMapped]  public int IdCountry { get; set; }  [JsonIgnore]  public CountryEntity Country { get; set; }  [Display(Name = "Logo")]  public string LogoImagePath { get; set; }  public string LogoImageFullPath => string.IsNullOrEmpty(LogoImagePath)  ? $"http://keypress.serveftp.net:88/GenericAppApi/images/Logos/noimage.png"  : $"http://keypress.serveftp.net:88/GenericAppApi{LogoImagePath.Substring(1)}";  }  } |  |

## Actualizamos DataContext

|  |  |
| --- | --- |
| **DataContext** | **Comentarios** |
| using GenericApp.Web.Data.Entities;  using Microsoft.EntityFrameworkCore;  namespace GenericApp.Web.Data  {  public class DataContext : DbContext  {  public DataContext(DbContextOptions<DataContext> options) : base(options)  {  }  public DbSet<CityEntity> Cities { get; set; }  public DbSet<CountryEntity> Countries { get; set; }  public DbSet<DepartmentEntity> Departments { get; set; }  public DbSet<TeamEntity> Teams { get; set; }  protected override void OnModelCreating(ModelBuilder modelBuilder)  {  base.OnModelCreating(modelBuilder);  modelBuilder.Entity<CountryEntity>()  .HasIndex(t => t.Name)  .IsUnique();  modelBuilder.Entity<DepartmentEntity>(dep =>  {  dep.HasIndex("Name", "CountryId").IsUnique();  dep.HasOne(d => d.Country).WithMany(c => c.Departments).OnDelete(DeleteBehavior.Cascade);  });  modelBuilder.Entity<CityEntity>(cit =>  {  cit.HasIndex("Name", "DepartmentId").IsUnique();  cit.HasOne(c => c.Department).WithMany(d => d.Cities).OnDelete(DeleteBehavior.Cascade);  });  modelBuilder.Entity<TeamEntity>(dep =>  {  dep.HasIndex("Name", "CountryId").IsUnique();  dep.HasOne(d => d.Country).WithMany(c => c.Teams).OnDelete(DeleteBehavior.Cascade);  });  }  }  } |  |

## Actualizar Base de Datos

Guardamos los cambios y corremos los comandos para actualizar la base de datos:

PM> add-migration AddCityAndDepartment

PM> update-database

# ViewModels para manejar imágenes y/o combos

## CategoryViewModel

En el Proyecto **Web**, dentro de la carpeta **Models** creamos la Clase **CategoryViewModel**

|  |  |
| --- | --- |
| **CategoryViewModel** | **Comentarios** |
| using GenericApp.Web.Data.Entities;  using Microsoft.AspNetCore.Http;  using System.ComponentModel.DataAnnotations;  namespace GenericApp.Web.Models  {  public class CategoryViewModel:CategoryEntity  {  [Display(Name = "Imagen")]  public IFormFile ImageFile { get; set; }  }  } |  |

## ProductViewModel

En el Proyecto **Web**, dentro de la carpeta **Models** creamos la Clase **ProductViewModel**

|  |  |
| --- | --- |
| **ProductViewModel** | **Comentarios** |
| using Microsoft.AspNetCore.Http;  using Microsoft.AspNetCore.Mvc.Rendering;  using GenericApp.Web.Data.Entities;  using System.Collections.Generic;  using System.ComponentModel.DataAnnotations;  namespace GenericApp.Web.Models  {  public class ProductViewModel : ProductEntity  {  [Display(Name = "Categoría")]  [Range(1, int.MaxValue, ErrorMessage = "Debe seleccionar una categoría.")]  [Required(ErrorMessage = "El campo {0} es requerido")]  public int CategoryId { get; set; }  [Display(Name = "Estado")]  [Range(1, int.MaxValue, ErrorMessage = "Debe seleccionar un estado.")]  [Required(ErrorMessage = "El campo {0} es requerido")]  public int StateId { get; set; }  public IFormFile ImageFile { get; set; }  public IEnumerable<SelectListItem> Categories { get; set; }  public IEnumerable<SelectListItem> States { get; set; }  [Display(Name = "Latitud")]  [Required(ErrorMessage = "El campo {0} es requerido")]  [DisplayFormat(DataFormatString = "{0:N4}")]  public double Latitude { get; set; }  [Display(Name = "Longitud")]  [Required(ErrorMessage = "El campo {0} es requerido")]  [DisplayFormat(DataFormatString = "{0:N4}")]  public double Longitude { get; set; }  [Display(Name = "Precio")]  [MaxLength(12)]  [RegularExpression(@"^\d+([\.\,]?\d+)?$", ErrorMessage = "Use sólo números o . o , para poner decimales")]  [Required]  public string PriceString { get; set; }  }  } |  |

## AddProductImageViewModel

En el Proyecto **Web**, dentro de la carpeta **Models** creamos la Clase **AddProductImageViewModel**

|  |  |
| --- | --- |
| **AddProductImageViewModel** | **Comentarios** |
| using Microsoft.AspNetCore.Http;  using System.ComponentModel.DataAnnotations;  namespace GenericApp.Web.Models  {  public class AddProductImageViewModel  {  public int ProductId { get; set; }  [Display(Name = "Imagen")]  [Required(ErrorMessage = "El campo {0} es requerido")]  public IFormFile ImageFile { get; set; }  }  } |  |

## CountryViewModel

|  |  |
| --- | --- |
| **CountryViewModel** | **Comentarios** |
| using GenericApp.Web.Data.Entities;  using Microsoft.AspNetCore.Http;  using System.ComponentModel.DataAnnotations;  namespace GenericApp.Web.Models  {  public class CountryViewModel : CountryEntity  {  [Display(Name = "Imagen")]  public IFormFile ImageFile { get; set; }  }  } |  |

## TeamViewModel

|  |  |
| --- | --- |
| **TeamViewModel** | **Comentarios** |
| using GenericApp.Web.Data.Entities;  using Microsoft.AspNetCore.Http;  using System.ComponentModel.DataAnnotations;  namespace GenericApp.Web.Models  {  public class TeamViewModel : TeamEntity  {  public int CountryId { get; set; }  [Display(Name = "Imagen")]  public IFormFile ImageFile { get; set; }  }  } |  |

# Responses

En el Proyecto **Common** creamos la Carpeta **Responses**

## Clase Response

Dentro de la Carpeta **Responses** creamos la Clase **Response**

|  |  |
| --- | --- |
| **Response** | **Comentarios** |
| namespace GenericApp.Common.Responses  {  public class Response  {  public bool IsSuccess { get; set; }  public string Message { get; set; }  public object Result { get; set; }  }  } |  |

## Clase ResponseT

|  |  |
| --- | --- |
| **ResponseT** | **Comentarios** |
| namespace GenericApp.Common.Responses  {  public class ResponseT<T> where T : class  {  public bool IsSuccess { get; set; }  public string Message { get; set; }  public T Result { get; set; }  }  } |  |

## Clase TokenResponse

|  |  |
| --- | --- |
| **TokenResponse** | **Comentarios** |
| using System;  namespace GenericApp.Common.Responses  {  public class TokenResponse  {  public string Token { get; set; }  public UserResponse User { get; set; }  public DateTime Expiration { get; set; }  public DateTime ExpirationLocal => Expiration.ToLocalTime();  }  } |  |

## Clase CountryResponse

|  |  |
| --- | --- |
| **CountryResponse** | **Comentarios** |
| using System.Collections.Generic;  namespace GenericApp.Common.Responses  {  public class CountryResponse  {  public int Id { get; set; }  public string Name { get; set; }  public string FlagImagePath { get; set; }  public string FlagImageFullPath => string.IsNullOrEmpty(FlagImagePath)  ? $"http://keypress.serveftp.net:88/GenericAppApi/images/Flags/noimage.png"  : $"http://keypress.serveftp.net:88/GenericAppApi{FlagImagePath.Substring(1)}";  public ICollection<DepartmentResponse> Departments { get; set; }  public ICollection<TeamResponse> Teams { get; set; }  }  } |  |

## Clase DepartmentResponse

|  |  |
| --- | --- |
| **DepartmentResponse** | **Comentarios** |
| namespace GenericApp.Common.Responses  {  public class DepartmentResponse  {  public int Id { get; set; }  public string Name { get; set; }  public CountryResponse Country { get; set; }  public ICollection<CityResponse> Cities { get; set; }  }  } |  |

## Clase CityResponse

|  |  |
| --- | --- |
| **CityResponse** | **Comentarios** |
| namespace GenericApp.Common.Responses  {  public class CityResponse  {  public int Id { get; set; }  public string Name { get; set; }  public DepartmentResponse Department { get; set; }  }  } |  |

## Clase TeamResponse

|  |  |
| --- | --- |
| **TeamResponse** | **Comentarios** |
| namespace GenericApp.Common.Responses  {  public class TeamResponse  {  public int Id { get; set; }  public string Name { get; set; }  public CountryResponse Country { get; set; }  public string LogoImagePath { get; set; }  public string LogoImageFullPath => string.IsNullOrEmpty(LogoImagePath)  ? $"http://keypress.serveftp.net:88/GenericAppApi/images/Logos/noimage.png"  : $"http://keypress.serveftp.net:88/GenericAppApi{LogoImagePath.Substring(1)}";  }  } |  |

## Clase UserResponse

|  |  |
| --- | --- |
| **UserResponse** | **Comentarios** |
| using GenericApp.Common.Enums;  namespace GenericApp.Common.Responses  {  public class UserResponse  {  public string Id { get; set; }  public string Email { get; set; }  public string PhoneNumber { get; set; }  public string Document { get; set; }  public string FirstName { get; set; }  public string LastName { get; set; }  public string Address { get; set; }  public string PicturePath { get; set; }  public string PictureFullPath => string.IsNullOrEmpty(PicturePath)  ? $"http://keypress.serveftp.net:88/GenericAppiApi/images/Users/nouser.png"  : $"http://keypress.serveftp.net:88/GenericAppApi{PicturePath.Substring(1)}";  public UserType UserType { get; set; }  public CityResponse City { get; set; }  public TeamResponse FavoriteTeam { get; set; }  public string FullName => $"{FirstName} {LastName}";  public string FullNameWithDocument => $"{FirstName} {LastName} - {Document}";  }  } |  |

## Clase CategoryResponse

|  |  |
| --- | --- |
| **CategoryResponse** | **Comentarios** |
| namespace GenericApp.Common.Responses  {  public class CategoryResponse  {  public int Id { get; set; }  public string Name { get; set; }  public string ImagePath { get; set; }  public string ImageFullPath => string.IsNullOrEmpty(ImagePath)  ? $"http://keypress.serveftp.net:88/GenericAppApi/images/Categories/noimage.png"  : $"http://keypress.serveftp.net:88/GenericAppApi{ImagePath.Substring(1)}";  }  } |  |

## Clase ProductImageResponse

|  |  |
| --- | --- |
| **ProductImageResponse** | **Comentarios** |
| namespace GenericApp.Common.Responses  {  public class ProductImageResponse  {  public int Id { get; set; }  public string ImagePath { get; set; }  public int ProductId { get; set; }  public string ImageFullPath => string.IsNullOrEmpty(ImagePath)  ? $"http://keypress.serveftp.net:88/GenericAppApi/images/Products/noimage.png"  : $"http://keypress.serveftp.net:88/GenericAppApi{ImagePath.Substring(1)}";  }  } |  |

## Clase ProductImageResponse2

|  |  |
| --- | --- |
| **ProductImageResponse2** | **Comentarios** |
| namespace GenericApp.Common.Responses  {  public class ProductImageResponse2  {  public int Id { get; set; }  public string ImageUrl { get; set; }  }  } |  |

## Clase StateResponse

|  |  |
| --- | --- |
| **StateResponse** | **Comentarios** |
| namespace GenericApp.Common.Responses  {  public class StateResponse  {  public int Id { get; set; }  public string Name { get; set; }  }  } |  |

## Clase ProductResponse

|  |  |
| --- | --- |
| **ProductResponse** | **Comentarios** |
| using System.Collections.Generic;  using System.Linq;  namespace GenericApp.Common.Responses  {  public class ProductResponse  {  public int Id { get; set; }  public string Name { get; set; }  public string Description { get; set; }  public decimal Price { get; set; }  public bool IsActive { get; set; }  public CategoryResponse Category { get; set; }  public double Latitude { get; set; }  public double Longitude { get; set; }  public StateResponse State { get; set; }  public ICollection<ProductImageResponse> ProductImages { get; set; }  public int ProductImagesNumber => ProductImages == null ? 0 : ProductImages.Count;  public string ImageFullPath => ProductImages == null || ProductImages.Count == 0  ? $"http://keypress.serveftp.net:88/GenericAppApi/images/Products/noimage.png"  : ProductImages.FirstOrDefault().ImageFullPath;  }  } |  |

# Requests

En el Proyecto **Common** creamos la Carpeta **Requests**

## Clase ChangePasswordRequest

|  |  |
| --- | --- |
| **ChangePasswordRequest** | **Comentarios** |
| using System.ComponentModel.DataAnnotations;  namespace GenericApp.Common.Requests  {  public class ChangePasswordRequest  {  [Required]  [StringLength(20, MinimumLength = 6)]  public string OldPassword { get; set; }  [Required]  [StringLength(20, MinimumLength = 6)]  public string NewPassword { get; set; }  }  } |  |

## Clase EmailRequest

|  |  |
| --- | --- |
| **EmailRequest** | **Comentarios** |
| using System.ComponentModel.DataAnnotations;  namespace GenericApp.Common.Requests  {  public class EmailRequest  {  [EmailAddress]  [Required]  public string Email { get; set; }  }  } |  |

## Clase TokenRequest

|  |  |
| --- | --- |
| **TokenRequest** | **Comentarios** |
| namespace GenericApp.Common.Requests  {  public class TokenRequest  {  public string Username { get; set; }  public string Password { get; set; }  }  } |  |

## Clase UserRequest

|  |  |
| --- | --- |
| **UserRequest** | **Comentarios** |
| using System.ComponentModel.DataAnnotations;  namespace GenericApp.Common.Requests  {  public class UserRequest  {  [Required]  public string Document { get; set; }  [Required]  public string FirstName { get; set; }  [Required]  public string LastName { get; set; }  [Required]  public string Address { get; set; }  [Required]  public string Email { get; set; }  [Required]  public string Phone { get; set; }  [Required]  [StringLength(20, MinimumLength = 6)]  public string Password { get; set; }  [StringLength(20, MinimumLength = 6)]  public string PasswordConfirm { get; set; }  [Required]  public int CityId { get; set; }  public byte[] PictureArray { get; set; }  [Required]  public int FavoriteTeamId { get; set; }  }  } |  |

## Clase ProductRequest

|  |  |
| --- | --- |
| **ProductRequest** | **Comentarios** |
| using GenericApp.Common.Responses;  namespace GenericApp.Common.Requests  {  public class ProductRequest  {  public int Id { get; set; }  public string Name { get; set; }  public string Description { get; set; }  public decimal Price { get; set; }  public double Latitude { get; set; }  public double Longitude { get; set; }  public CategoryResponse Category { get; set; }  public StateResponse State { get; set; }  public byte[] PhotoArray { get; set; }  public string PhotoUrl { get; set; }  public int ProductId { get; set; }  }  } |  |

## Clase ProductImageRequest

|  |  |
| --- | --- |
| **ProductImageRequest** | **Comentarios** |
| using GenericApp.Common.Responses;  namespace GenericApp.Common.Requests  {  public class ProductImageRequest  {  public int Id { get; set; }  public byte[] ImageArray { get; set; }  public string ImageUrl { get; set; }  public ProductResponse Product { get; set; }  }  } |  |

# ImageHelper

En el Proyecto **Web** creamos una Carpeta llamada **Helpers**

Dentro creamos la Interfaz **IimageHelper**

|  |  |
| --- | --- |
| **IimageHelper** | **Comentarios** |
| using System.IO;  using System.Threading.Tasks;  using Microsoft.AspNetCore.Http;  namespace GenericApp.Web.Helpers  {  public interface IImageHelper  {  Task<string> UploadImageAsync(IFormFile imageFile, string folder);  string UploadImage(byte[] pictureArray, string folder);  Task<string> UploadImage2Async(Stream imageFile, string folder);  }  } |  |

Y creamos la implementación **ImageHelper**

|  |  |
| --- | --- |
| **ImageHelper** | **Comentarios** |
| using Microsoft.AspNetCore.Http;  using System;  using System.IO;  using System.Threading.Tasks;  namespace GenericApp.Web.Helpers  {  public class ImageHelper : IImageHelper  {  public async Task<string> UploadImageAsync(IFormFile imageFile, string folder)  {  string guid = Guid.NewGuid().ToString();  string file = $"{guid}.jpg";  string path = Path.Combine(  Directory.GetCurrentDirectory(),  $"wwwroot\\images\\{folder}",  file);  using (FileStream stream = new FileStream(path, FileMode.Create))  {  await imageFile.CopyToAsync(stream);  }  return $"~/images/{folder}/{file}";  }  public string UploadImage(byte[] pictureArray, string folder)  {  MemoryStream stream = new MemoryStream(pictureArray);  string guid = Guid.NewGuid().ToString();  string file = $"{guid}.jpg";  try  {  stream.Position = 0;  string path = Path.Combine(Directory.GetCurrentDirectory(), $"wwwroot\\images\\{folder}", file);  File.WriteAllBytes(path, stream.ToArray());  }  catch  {  return string.Empty;  }  return $"~/images/{folder}/{file}";  }  public async Task<string> UploadImage2Async(Stream imageFile, string folder)  {  string guid = Guid.NewGuid().ToString();  string file = $"{guid}.jpg";  string path = Path.Combine(  Directory.GetCurrentDirectory(),  $"wwwroot\\images\\{folder}",  file);  using (FileStream stream = new FileStream(path, FileMode.Create))  {  await imageFile.CopyToAsync(stream);  }  return $"~/images/{folder}/{file}";  }  }  } |  |

Agregamos la inyección en el archivo **Startup.cs**

|  |  |
| --- | --- |
| **Startup.cs** | **Comentarios** |
| public void ConfigureServices(IServiceCollection services)  {  services.Configure<CookiePolicyOptions>(options =>  {  options.CheckConsentNeeded = context => true;  options.MinimumSameSitePolicy = SameSiteMode.None;  });  services.AddDbContext<DataContext>(cfg =>  {  cfg.UseSqlServer(Configuration.GetConnectionString("DefaultConnection"));  });    services.AddTransient<SeedDb>();  services.AddScoped<IImageHelper, ImageHelper>();  services.AddMvc().SetCompatibilityVersion(CompatibilityVersion.Version\_2\_1);  } |  |

# CombosHelper

En el Proyecto **Web**, en la Carpeta llamada **Helpers,** creamos la Interfaz **ICombosHelper**

|  |  |
| --- | --- |
| **ICombosHelper** | **Comentarios** |
| using Microsoft.AspNetCore.Mvc.Rendering;  using System.Collections.Generic;  namespace GenericApp.Web.Helpers  {  public interface ICombosHelper  {  IEnumerable<SelectListItem> GetComboCategories();  IEnumerable<SelectListItem> GetComboCountries();  IEnumerable<SelectListItem> GetComboDepartments(int countryId);  IEnumerable<SelectListItem> GetComboCities(int departmentId);  IEnumerable<SelectListItem> GetComboTeams(int countryId);  IEnumerable<SelectListItem> GetComboUserTypes();  IEnumerable<SelectListItem> GetComboStates();  }  } |  |

Y creamos la implementación **CombosHelper**

|  |  |
| --- | --- |
| **CombosHelper** | **Comentarios** |
| using Microsoft.AspNetCore.Mvc.Rendering;  using Microsoft.EntityFrameworkCore;  using GenericApp.Web.Data.Entities;  using GenericApp.Web.Data;  using System.Collections.Generic;  using System.Linq;  namespace GenericApp.Web.Helpers  {  public class CombosHelper : ICombosHelper  {  private readonly DataContext \_context;  public CombosHelper(DataContext context)  {  \_context = context;  }  public IEnumerable<SelectListItem> GetComboCategories()  {  List<SelectListItem> list = \_context.Categories.Select(t => new SelectListItem  {  Text = t.Name,  Value = $"{t.Id}"  })  .OrderBy(t => t.Text)  .ToList();  list.Insert(0, new SelectListItem  {  Text = "[Seleccione una categoría...]",  Value = "0"  });  return list;  }  public IEnumerable<SelectListItem> GetComboCities(int departmentId)  {  List<SelectListItem> list = new List<SelectListItem>();  DepartmentEntity department = \_context.Departments  .Include(d => d.Cities)  .FirstOrDefault(d => d.Id == departmentId);  if (department != null)  {  list = department.Cities.Select(t => new SelectListItem  {  Text = t.Name,  Value = $"{t.Id}"  })  .OrderBy(t => t.Text)  .ToList();  }  list.Insert(0, new SelectListItem  {  Text = "[Seleccione una ciudad...]",  Value = "0"  });  return list;  }  public IEnumerable<SelectListItem> GetComboCountries()  {  List<SelectListItem> list = \_context.Countries.Select(t => new SelectListItem  {  Text = t.Name,  Value = $"{t.Id}"  })  .OrderBy(t => t.Text)  .ToList();  list.Insert(0, new SelectListItem  {  Text = "[Seleccione un país...]",  Value = "0"  });  return list;  }  public IEnumerable<SelectListItem> GetComboDepartments(int countryId)  {  List<SelectListItem> list = new List<SelectListItem>();  CountryEntity country = \_context.Countries  .Include(c => c.Departments)  .FirstOrDefault(c => c.Id == countryId);  if (country != null)  {  list = country.Departments.Select(t => new SelectListItem  {  Text = t.Name,  Value = $"{t.Id}"  })  .OrderBy(t => t.Text)  .ToList();  }  list.Insert(0, new SelectListItem  {  Text = "[Seleccione una provincia...]",  Value = "0"  });  return list;  }  public IEnumerable<SelectListItem> GetComboTeams(int countryId)  {  List<SelectListItem> list = new List<SelectListItem>();  CountryEntity country = \_context.Countries  .Include(c => c.Teams)  .FirstOrDefault(c => c.Id == countryId);  if (country != null)  {  list = country.Teams.Select(t => new SelectListItem  {  Text = t.Name,  Value = $"{t.Id}"  })  .OrderBy(t => t.Text)  .ToList();  }  list.Insert(0, new SelectListItem  {  Text = "[Seleccione un equipo...]",  Value = "0"  });  return list;  }  public IEnumerable<SelectListItem> GetComboUserTypes()  {  List<SelectListItem> list = new List<SelectListItem>();  list.Insert(0, new SelectListItem  {  Text = "[Seleccione un Tipo de Usuario...]",  Value = "0"  });  list.Insert(0, new SelectListItem  {  Text = "Admin",  Value = "1"  });  list.Insert(0, new SelectListItem  {  Text = "User",  Value = "2"  });  return list;  }  public IEnumerable<SelectListItem> GetComboStates()  {  List<SelectListItem> list = \_context.States.Select(t => new SelectListItem  {  Text = t.Name,  Value = $"{t.Id}"  })  .OrderBy(t => t.Value)  .ToList();  list.Insert(0, new SelectListItem  {  Text = "[Seleccione un estado...]",  Value = "0"  });  return list;  }  }  } |  |

Agregamos la inyección en el archivo **Startup.cs**

|  |  |
| --- | --- |
| **Startup.cs** | **Comentarios** |
| public void ConfigureServices(IServiceCollection services)  {  services.Configure<CookiePolicyOptions>(options =>  {  options.CheckConsentNeeded = context => true;  options.MinimumSameSitePolicy = SameSiteMode.None;  });  services.AddDbContext<DataContext>(cfg =>  {  cfg.UseSqlServer(Configuration.GetConnectionString("DefaultConnection"));  });    services.AddTransient<SeedDb>();  services.AddScoped<IImageHelper, ImageHelper>();  services.AddScoped<IConverterHelper, ConverterHelper>();  services.AddScoped<ICombosHelper, CombosHelper>();  services.AddMvc().SetCompatibilityVersion(CompatibilityVersion.Version\_2\_1);  } |  |

# ConverterHelper

En el Proyecto **Web** dentro de la Carpeta **Helpers,** creamos la Interfaz **IconverterHelper**

|  |  |
| --- | --- |
| **IconverterHelper** | **Comentarios** |
| using GenericApp.Common.Responses;  using GenericApp.Web.Data.Entities;  using GenericApp.Web.Models;  using System.Collections.Generic;  using System.Threading.Tasks;  namespace GenericApp.Web.Helpers  {  public interface IConverterHelper  {  CategoryEntity ToCategoryEntity(CategoryViewModel model, string path, bool isNew);  CategoryViewModel ToCategoryViewModel(CategoryEntity categoryEntity);  Task<ProductEntity> ToProductAsync(ProductViewModel model, bool isNew);  ProductViewModel ToProductViewModel(ProductEntity product);  CountryEntity ToCountryEntity(CountryViewModel model, string path, bool isNew);  CountryViewModel ToCountryViewModel(CountryEntity countryEntity);  TeamEntity ToTeamEntity(TeamViewModel model, string path,bool isNew);  TeamViewModel ToTeamViewModel(TeamEntity product);  ProductResponse ToProductResponse(ProductEntity productEntity);  List<ProductResponse> ToProductResponse(List<ProductEntity> productEntities);  }  } |  |

Creamos la implementación **ConverterHelper**

|  |  |
| --- | --- |
| **ConverterHelper** | **Comentarios** |
| using GenericApp.Common.Responses;  using GenericApp.Web.Data;  using GenericApp.Web.Data.Entities;  using GenericApp.Web.Models;  using System.Collections.Generic;  using System.Globalization;  using System.Linq;  using System.Threading.Tasks;  namespace GenericApp.Web.Helpers  {  public class ConverterHelper : IConverterHelper  {  private readonly DataContext \_context;  private readonly ICombosHelper \_combosHelper;  public ConverterHelper(DataContext context, ICombosHelper combosHelper)  {  \_context = context;  \_combosHelper = combosHelper;  }  public CategoryEntity ToCategoryEntity(CategoryViewModel model, string path, bool isNew)  {  return new CategoryEntity  {  Id = isNew ? 0 : model.Id,  ImagePath = path,  Name = model.Name  };  }  public CategoryViewModel ToCategoryViewModel(CategoryEntity categoryEntity)  {  return new CategoryViewModel  {  Id = categoryEntity.Id,  ImagePath = categoryEntity.ImagePath,  Name = categoryEntity.Name  };  }  public async Task<ProductEntity> ToProductAsync(ProductViewModel model, bool isNew)  {  return new ProductEntity  {  Category = await \_context.Categories.FindAsync(model.CategoryId),  Description = model.Description,  Id = isNew ? 0 : model.Id,  IsActive = model.IsActive,  Latitude = model.Latitude,  Longitude = model.Longitude,  Name = model.Name,  Price = ToPrice(model.PriceString),  ProductImages = model.ProductImages,  State = await \_context.States.FindAsync(model.StateId),  };  }  private decimal ToPrice(string priceString)  {  string nds = CultureInfo.CurrentCulture.NumberFormat.NumberDecimalSeparator;  if (nds == ".")  {  priceString = priceString.Replace(',', '.');  }  else  {  priceString = priceString.Replace('.', ',');  }  return decimal.Parse(priceString);  }  public ProductViewModel ToProductViewModel(ProductEntity product)  {  return new ProductViewModel  {  Categories = \_combosHelper.GetComboCategories(),  Category = product.Category,  CategoryId = product.Category.Id,  Description = product.Description,  Id = product.Id,  IsActive = product.IsActive,  Latitude = product.Latitude,  Longitude = product.Longitude,  Name = product.Name,  PriceString = $"{product.Price}",  ProductImages = product.ProductImages,  States = \_combosHelper.GetComboStates(),  State = product.State,  StateId = product.State.Id  };  }  public CountryEntity ToCountryEntity(CountryViewModel model, string path, bool isNew)  {  return new CountryEntity  {  Id = isNew ? 0 : model.Id,  FlagImagePath = path,  Name = model.Name  };  }  public CountryViewModel ToCountryViewModel(CountryEntity countryEntity)  {  return new CountryViewModel  {  Id = countryEntity.Id,  FlagImagePath = countryEntity.FlagImagePath,  Name = countryEntity.Name  };  }  public TeamEntity ToTeamEntity(TeamViewModel model, string path, bool isNew)  {  return new TeamEntity  {  Id = isNew ? 0 : model.Id,  LogoImagePath = path,  Name = model.Name,  IdCountry = model.IdCountry,  Country = model.Country,  };  }  public TeamViewModel ToTeamViewModel(TeamEntity team)  {  return new TeamViewModel  {  Countries = \_combosHelper.GetComboCountries(),  Country = team.Country,  CountryId = team.Country.Id,  Id = team.Id,  Name = team.Name,  LogoImagePath = team.LogoImagePath,  };  }  public ProductResponse ToProductResponse(ProductEntity productEntity)  {  return new ProductResponse  {  Id = productEntity.Id,  IsActive = productEntity.IsActive,  Description = productEntity.Description,  Name = productEntity.Name,  Price= productEntity.Price,  Latitude = productEntity.Latitude,  Longitude = productEntity.Longitude,  Category = new CategoryResponse  {  Id = productEntity.Category.Id,  Name = productEntity.Category.Name,  ImagePath = productEntity.Category.ImagePath,  },  State = new StateResponse {  Id = productEntity.State.Id,  Name = productEntity.State.Name,  },  ProductImages = productEntity.ProductImages?.Select(g => new ProductImageResponse  {  Id = g.Id,  ImagePath=g.ImagePath,  ProductId=g.Product.Id  }).ToList()  };  }  public List<ProductResponse> ToProductResponse(List<ProductEntity> productEntities)  {  List<ProductResponse> list = new List<ProductResponse>();  foreach (ProductEntity productEntity in productEntities)  {  list.Add(ToProductResponse(productEntity));  }  return list;  }  }  } |  |

Agregamos la inyacción en el archivo **Startup.cs**

|  |  |
| --- | --- |
| **Startup.cs** | **Comentarios** |
| public void ConfigureServices(IServiceCollection services)  {  services.Configure<CookiePolicyOptions>(options =>  {  options.CheckConsentNeeded = context => true;  options.MinimumSameSitePolicy = SameSiteMode.None;  });  services.AddDbContext<DataContext>(cfg =>  {  cfg.UseSqlServer(Configuration.GetConnectionString("DefaultConnection"));  });    services.AddTransient<SeedDb>();  services.AddScoped<IImageHelper, ImageHelper>();  services.AddScoped<IConverterHelper, ConverterHelper>();  services.AddMvc().SetCompatibilityVersion(CompatibilityVersion.Version\_2\_1);  } |  |

# IMailHelper

En el Proyecto **Web** dentro de la Carpeta **Helpers,** creamos la Interfaz **IMailHelper**

|  |  |
| --- | --- |
| **IMailHelper** | **Comentarios** |
| using GenericApp.Common.Responses;  namespace GenericApp.Web.Helpers  {  public interface IMailHelper  {  Response SendMail(string to, string subject, string body);  }  } |  |

Creamos la implementación **MailHelper**

|  |  |
| --- | --- |
| **MailHelper** | **Comentarios** |
| using MailKit.Net.Smtp;  using Microsoft.Extensions.Configuration;  using MimeKit;  using GenericApp.Common.Responses;  using System;  namespace GenericApp.Web.Helpers  {  public class MailHelper : IMailHelper  {  private readonly IConfiguration \_configuration;  public MailHelper(IConfiguration configuration)  {  \_configuration = configuration;  }  public Response SendMail(string to, string subject, string body)  {  try  {  string from = \_configuration["Mail:From"];  string smtp = \_configuration["Mail:Smtp"];  string port = \_configuration["Mail:Port"];  string password = \_configuration["Mail:Password"];  MimeMessage message = new MimeMessage();  message.From.Add(new MailboxAddress(from));  message.To.Add(new MailboxAddress(to));  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  };  }  }  }  } |  |

Agregamos la inyección en el archivo **Startup.cs**

|  |  |
| --- | --- |
| **Startup.cs** | **Comentarios** |
| services.AddTransient<SeedDb>();  services.AddScoped<IImageHelper, ImageHelper>();  services.AddScoped<IConverterHelper, ConverterHelper>();  services.AddScoped<ICombosHelper, CombosHelper>();  services.AddScoped<IUserHelper, UserHelper>();  services.AddScoped<IMailHelper, MailHelper>(); |  |

# Maestro detalle MVC para Countries

## Vista Index

Modificamos la vista **Index** de **CountriesController**:

|  |  |
| --- | --- |
| **Index** | **Comentarios** |
| <thead>  <tr>  <th>  @Html.DisplayNameFor(model => model.Name)  </th>  <th>  @Html.DisplayNameFor(model => model.DepartmentsNumber)  </th>  <th>  @Html.DisplayNameFor(model => model.TeamsNumber)  </th>  <th></th>  </tr>  </thead>  <tbody>  @foreach (var item in Model)  {  <tr>  <td>  @Html.DisplayFor(modelItem => item.Name)  </td>  <td>  @Html.DisplayFor(modelItem => item.DepartmentsNumber)  </td>  <td>  @Html.DisplayFor(modelItem => item. TeamsNumber)  </td>  <td>  <a **asp-action**="Edit" **asp-route-id**="@item.Id" class="btn btn-warning"><i class="glyphicon glyphicon-pencil"></i></a>  <a **asp-action**="Details" **asp-route-id**="@item.Id" class="btn btn-info"><i class="glyphicon glyphicon-align-justify"></i></a>  <button data-id="@item.Id" class="btn btn-danger deleteItem" data-toggle="modal" data-target="#deleteDialog"><i class="glyphicon glyphicon-trash"></i></button>  </td>  </tr>  }  </tbody> |  |

## Métodos Index y Details del controlador CountriesController:

Modificamos los métodos **Index** y **Details** del controlador **CountriesController**:

|  |  |
| --- | --- |
| **CountriesController** | **Comentarios** |
| // GET: Countries  public async Task<IActionResult> Index()  {  return View(await \_context.Countries  .Include(c => c.Departments)  .Include(t => t.Teams)  .ToListAsync());  }  // GET: Countries/Details/5  public async Task<IActionResult> Details(int? id)  {  if (id == null)  {  return NotFound();  }  var country = await \_context.Countries  .Include(c => c.Departments)  .ThenInclude(d => d.Cities)  .Include(t => t.Teams)  .FirstOrDefaultAsync(m => m.Id == id);  if (country == null)  {  return NotFound();  }  return View(country);  } |  |

## Vista Details del controlador Countries:

Modificamos la vista **Details** del controlador **CountriesController**:

|  |  |
| --- | --- |
| **Details** | **Comentarios** |
| @model GenericApp.Web.Data.Entities.CountryEntity  @{  ViewData["Title"] = "Details";  }  <link rel="stylesheet" href="https://cdn.datatables.net/1.10.19/css/jquery.dataTables.min.css" />  <h2>  <img src="@Url.Content(Model.FlagImagePath)" alt="Image" style="width:100;height:100px;max-height: 100%; width: auto;" />  @Html.DisplayFor(model => model.Name)  </h2>  <div>  <h4></h4>  <hr />  <div class="row">  <div class="col-md-4">  <dl class="dl-horizontal">  <dt>  @Html.DisplayNameFor(model => model.DepartmentsNumber)  </dt>  <dd>  @Html.DisplayFor(model => model.DepartmentsNumber)  </dd>  <dt>  @Html.DisplayNameFor(model => model.TeamsNumber)  </dt>  <dd>  @Html.DisplayFor(model => model.TeamsNumber)  </dd>  </dl>  </div>    <div>  <a **asp-action**="AddDepartment" **asp-route-id**="@Model.Id" class="btn btn-primary"><i class="glyphicon glyphicon-plus"></i> Provincia</a>  <a **asp-action**="AddTeam" **asp-route-id**="@Model.Id" class="btn btn-primary"><i class="glyphicon glyphicon-plus"></i> Equipo</a>  <a **asp-action**="Edit" **asp-route-id**="@Model.Id" class="btn btn-warning">Editar</a>  <a **asp-action**="Index" class="btn btn-success">Regresar</a>  </div>  <hr />  <div class="row">  <div class="col-md-6">  <div class="panel panel-default">  <div class="panel-heading">  <h3 class="panel-title">Provincias</h3>  </div>  <div class="panel-body">  <table class="table table-hover table-responsive table-striped" id="MyTableDepartments">  <thead>  <tr>  <th>  @Html.DisplayNameFor(model => model.Departments.FirstOrDefault().Name)  </th>  <th>  @Html.DisplayNameFor(model => model.Departments.FirstOrDefault().CitiesNumber)  </th>  <th></th>  </tr>  </thead>  <tbody>  @foreach (var item in Model.Departments)  {  <tr>  <td>  @Html.DisplayFor(modelItem => item.Name)  </td>  <td>  @Html.DisplayFor(modelItem => item.CitiesNumber)  </td>  <td>  <a **asp-action**="EditDepartment" **asp-route-id**="@item.Id" class="btn btn-warning"><i class="glyphicon glyphicon-pencil"></i></a>  <a **asp-action**="DetailsDepartment" **asp-route-id**="@item.Id" class="btn btn-info"><i class="glyphicon glyphicon-align-justify"></i></a>  <button data-id="@item.Id" class="btn btn-danger deleteDepartment" data-toggle="modal" data-target="#deleteDialog"><i class="glyphicon glyphicon-trash"></i></button>  </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">Equipos</h3>  </div>  <div class="panel-body">  <table class="table table-hover table-responsive table-striped" id="MyTableTeams">  <thead>  <tr>  <th>  @Html.DisplayNameFor(model => model.Teams.FirstOrDefault().Name)  </th>  <th>  @Html.DisplayNameFor(model => model.Teams.FirstOrDefault().LogoImagePath)  </th>  <th></th>  </tr>  </thead>  <tbody>  @foreach (var item in Model.Teams)  {  <tr>  <td>  @Html.DisplayFor(modelItem => item.Name)  </td>  <td>  @if (!string.IsNullOrEmpty(item.LogoImagePath))  {  <img src="@Url.Content(item.LogoImagePath)" alt="Image" style="width:50px;height:50px;max-width: 100%; height: auto;" />  }  else  {  <img src="@Url.Content(item.LogoImagePath)" alt="Image" style="width:50px;height:50px;max-width: 100%; height: auto;" />  }  </td>  <td>  <a **asp-action**="EditTeam" **asp-route-id**="@item.Id" class="btn btn-warning"><i class="glyphicon glyphicon-pencil"></i></a>  <button data-id="@item.Id" class="btn btn-danger deleteTeam" data-toggle="modal" data-target="#deleteDialog"><i class="glyphicon glyphicon-trash"></i></button>  </td>  </tr>  }  </tbody>  </table>  </div>  </div>  </div>  </div>  <**partial** **name**="\_DeleteDialog" />  @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 () {  $('#MyTableDepartments').DataTable();  $('#MyTableTeams').DataTable();  // Delete item  var item\_to\_delete;  var action\_to\_delete;  $('.deleteDepartment').click((e) => {  item\_to\_delete = e.currentTarget.dataset.id;  action\_to\_delete = 1;  });  $('.deleteTeam').click((e) => {  item\_to\_delete = e.currentTarget.dataset.id;  action\_to\_delete = 2;  });  $("#btnYesDelete").click(function () {  if (action\_to\_delete == 1) {  window.location.href = '/Countries/DeleteDepartment/' + item\_to\_delete;  } else {  window.location.href = '/Countries/DeleteTeam/' + item\_to\_delete;  }  });  });  </script>  } | Para que funcionen botones Borrar en 2 Tablas tener en cuenta….  <-Nombre de primer tabla  En class dice “deleteDepartment”  <-Nombre de segunda tabla  En class dice “deleteTeam”  Nombre primer Tabla  Nombre segunda Tabla  deleteDepartment  deleteTeam  Ruta método DeleteDepartment  Ruta método DeleteTeam |

## Método AddDepartment

Agregamos al controlador **CountriesController** el método **AddDepartment**

|  |  |
| --- | --- |
| **CountriesController** | **Comentarios** |
| public async Task<IActionResult> AddDepartment(int? id)  {  if (id == null)  {  return NotFound();  }  CountryEntity country = await \_context.Countries.FindAsync(id);  if (country == null)  {  return NotFound();  }  DepartmentEntity model = new DepartmentEntity { IdCountry = country.Id };  return View(model);  }  [HttpPost]  [ValidateAntiForgeryToken]  public async Task<IActionResult> AddDepartment(DepartmentEntity department)  {  if (ModelState.IsValid)  {  CountryEntity country = await \_context.Countries  .Include(c => c.Departments)  .FirstOrDefaultAsync(c => c.Id == department.IdCountry);  if (country == null)  {  return NotFound();  }  try  {  department.Id = 0;  country.Departments.Add(department);  \_context.Update(country);  await \_context.SaveChangesAsync();  return RedirectToAction($"{nameof(Details)}/{country.Id}");  }  catch (DbUpdateException dbUpdateException)  {  if (dbUpdateException.InnerException.Message.Contains("duplicate"))  {  ModelState.AddModelError(string.Empty, "Hay un registro con el mismo nombre.");  }  else  {  ModelState.AddModelError(string.Empty, dbUpdateException.InnerException.Message);  }  }  catch (Exception exception)  {  ModelState.AddModelError(string.Empty, exception.Message);  }  }  return View(department);  } |  |

Adicionamos la vista parcial **\_Department**:

|  |  |
| --- | --- |
| **\_Department** | **Comentarios** |
| @model GenericApp.Web.Data.Entities.DepartmentEntity  <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> |  |

Adicionamos la vista **AddDepartment**:

|  |  |
| --- | --- |
| **AddDepartment** | **Comentarios** |
| @model GenericApp.Web.Data.Entities.DepartmentEntity  @{  ViewData["Title"] = "Add Department";  }  <h2>Agregar nueva Provincia</h2>  <h4></h4>  <hr />  <div class="row">  <div class="col-md-4">  <form **asp-action**="AddDepartment">  <div **asp-validation-summary**="ModelOnly" class="text-danger"></div>  <input **type**="hidden" **asp-for**="IdCountry" />  <**partial** **name**="\_Department" />  <div class="form-group">  <input type="submit" value="Grabar" class="btn btn-primary" />  <a **asp-action**="Details" **asp-route-id**="@Model.IdCountry" class="btn btn-success">Regresar</a>  </div>  </form>  </div>  </div>  @section Scripts {  @{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}  } |  |

## Método AddTeam

Agregamos al controlador **CountriesController** el método **AddTeam**

|  |  |
| --- | --- |
| **CountriesController** | **Comentarios** |
| public async Task<IActionResult> AddTeam(int? id)  {  if (id == null)  {  return NotFound();  }  CountryEntity country = await \_context.Countries.FindAsync(id);  if (country == null)  {  return NotFound();  }  var model = new TeamViewModel  {  CountryId = country.Id,  IdCountry = country.Id,  };  return View(model);  }  [HttpPost]  [ValidateAntiForgeryToken]  public async Task<IActionResult> AddTeam(TeamViewModel model)  {  if (ModelState.IsValid)  {  var path = string.Empty;  if (model.ImageFile != null)  {  path = await \_imageHelper.UploadImageAsync(model.ImageFile, "Teams");  }  CountryEntity country = await \_context.Countries  .Include(c => c.Teams)  .FirstOrDefaultAsync(c => c.Id == model.CountryId);  if (country == null)  {  return NotFound();  }  TeamEntity team = \_converterHelper.ToTeamEntity(model, path, true);  team.Country = country;  \_context.Add(team);  try  {  await \_context.SaveChangesAsync();  return RedirectToAction($"{nameof(Details)}/{country.Id}");  }  catch (Exception ex)  {  if (ex.InnerException.Message.Contains("duplicate"))  {  ModelState.AddModelError(string.Empty, "Este Equipo ya existe");  }  else  {  ModelState.AddModelError(string.Empty, ex.InnerException.Message);  }  }  }  return View(model);  } |  |

Adicionamos la vista parcial **\_Team**:

|  |  |
| --- | --- |
| **\_Team** | **Comentarios** |
| @model GenericApp.Web.Models.TeamViewModel  <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**="ImageFile" class="control-label"></label>  <input **asp-for**="ImageFile" **type**="file" class="form-control" />  <span **asp-validation-for**="ImageFile" class="text-danger"></span>  </div> |  |

Adicionamos la vista **AddTeam**

|  |  |
| --- | --- |
| **AddTeam** | **Comentarios** |
| @model GenericApp.Web.Models.TeamViewModel  @{  ViewData["Title"] = "Add Team";  }  <h2>Agregar nuevo Equipo</h2>  <h4></h4>  <hr />  <div class="row">  <div class="col-md-4">    <form **asp-action**="AddTeam" enctype="multipart/form-data">  <div **asp-validation-summary**="ModelOnly" class="text-danger"></div>  <input **type**="hidden" **asp-for**="CountryId" />  <input **type**="hidden" **asp-for**="IdCountry" />  <**partial** **name**="\_Team" />  <div class="form-group">  <input type="submit" value="Grabar" class="btn btn-primary" />  <a **asp-action**="Details" **asp-route-id**="@Model.CountryId" class="btn btn-success">Regresar</a>  </div>  </form>  </div>  </div>  @section Scripts {  @{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}  } |  |

## Método EditDepartment

Agregamos al controlador **CountriesController** el método **EditDepartment**

|  |  |
| --- | --- |
| **EditDepartment** | **Comentarios** |
| public async Task<IActionResult> EditDepartment(int? id)  {  if (id == null)  {  return NotFound();  }  DepartmentEntity department = await \_context.Departments.FindAsync(id);  if (department == null)  {  return NotFound();  }  CountryEntity country = await \_context.Countries.FirstOrDefaultAsync(c => c.Departments.FirstOrDefault(d => d.Id == department.Id) != null);  department.IdCountry = country.Id;  return View(department);  }  [HttpPost]  [ValidateAntiForgeryToken]  public async Task<IActionResult> EditDepartment(DepartmentEntity department)  {  if (ModelState.IsValid)  {  try  {  \_context.Update(department);  await \_context.SaveChangesAsync();  return RedirectToAction($"{nameof(Details)}/{department.IdCountry}");  }  catch (DbUpdateException dbUpdateException)  {  if (dbUpdateException.InnerException.Message.Contains("duplicate"))  {  ModelState.AddModelError(string.Empty, "Hay un registro con el mismo nombre.");  }  else  {  ModelState.AddModelError(string.Empty, dbUpdateException.InnerException.Message);  }  }  catch (Exception exception)  {  ModelState.AddModelError(string.Empty, exception.Message);  }  }  return View(department);  } |  |

Adicionamos la vista **EditDepartment**:

|  |  |
| --- | --- |
| **EditDepartment** | **Comentarios** |
| @model GenericApp.Web.Data.Entities.DepartmentEntity  @{  ViewData["Title"] = "Edit";  }  <h2>Editar Provincia</h2>  <h4></h4>  <hr />  <div class="row">  <div class="col-md-4">  <form **asp-action**="EditDepartment">  <div **asp-validation-summary**="ModelOnly" class="text-danger"></div>  <input **type**="hidden" **asp-for**="Id" />  <input **type**="hidden" **asp-for**="IdCountry" />  <**partial** **name**="\_Department" />  <div class="form-group">  <input type="submit" value="Grabar" class="btn btn-primary" />  <a **asp-action**="Details" **asp-route-id**="@Model.IdCountry" class="btn btn-success">Regresar</a>  </div>  </form>  </div>  </div>  @section Scripts {  @{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}  } |  |

## Método EditTeam

Agregamos al controlador **CountriesController** el método **EditTeam**

|  |  |
| --- | --- |
| **EditTeam** | **Comentarios** |
| public async Task<IActionResult> EditTeam(int? id)  {  if (id == null)  {  return NotFound();  }  TeamEntity team = await \_context.Teams.FindAsync(id);  if (team == null)  {  return NotFound();  }  CountryEntity country = await \_context.Countries.FirstOrDefaultAsync(c => c.Teams.FirstOrDefault(d => d.Id == team.Id) != null);  team.IdCountry = country.Id;  TeamViewModel model = \_converterHelper.ToTeamViewModel(team);  return View(model);  }  [HttpPost]  [ValidateAntiForgeryToken]  public async Task<IActionResult> EditTeam(TeamViewModel model)  {  if (ModelState.IsValid)  {  var path = model.LogoImagePath;  if (model.ImageFile != null)  {  path = await \_imageHelper.UploadImageAsync(model.ImageFile, "Teams");  }  try  {  TeamEntity team = \_converterHelper.ToTeamEntity(model, path,false);  \_context.Update(team);  await \_context.SaveChangesAsync();  return RedirectToAction($"{nameof(Details)}/{model.CountryId}");  }  catch (DbUpdateException dbUpdateException)  {  if (dbUpdateException.InnerException.Message.Contains("duplicate"))  {  ModelState.AddModelError(string.Empty, "Hay un registro con el mismo nombre.");  }  else  {  ModelState.AddModelError(string.Empty, dbUpdateException.InnerException.Message);  }  }  catch (Exception exception)  {  ModelState.AddModelError(string.Empty, exception.Message);  }  }  return View(model);  } |  |

Adicionamos la vista **EditTeam**:

|  |  |
| --- | --- |
| **EditTeam** | **Comentarios** |
| @model GenericApp.Web.Models.TeamViewModel  @{  ViewData["Title"] = "Edit";  }  <h2>Editar Equipo</h2>  <h4></h4>  <hr />  <div class="row">  <div class="col-md-4">  <form **asp-action**="EditTeam" enctype="multipart/form-data">  <div **asp-validation-summary**="ModelOnly" class="text-danger"></div>  <input **type**="hidden" **asp-for**="Id" />  <input **type**="hidden" **asp-for**="CountryId" />  <**partial** **name**="\_Team" />  <div class="form-group">  <input type="submit" value="Grabar" class="btn btn-primary" />  <a **asp-action**="Details" **asp-route-id**="@Model.CountryId" class="btn btn-success">Regresar</a>  </div>  </form>  </div>  </div>  @section Scripts {  @{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}  } |  |

## Método DeleteDepartment

Agregamos al controlador **CountriesController** el método **DeleteDepartment**

|  |  |
| --- | --- |
| **DeleteDepartment** | **Comentarios** |
| public async Task<IActionResult> DeleteDepartment(int? id)  {  if (id == null)  {  return NotFound();  }  DepartmentEntity department = await \_context.Departments  .Include(d => d.Cities)  .FirstOrDefaultAsync(m => m.Id == id);  if (department == null)  {  return NotFound();  }  CountryEntity country = await \_context.Countries.FirstOrDefaultAsync(c => c.Departments.FirstOrDefault(d => d.Id == department.Id) != null);  \_context.Departments.Remove(department);  await \_context.SaveChangesAsync();  return RedirectToAction($"{nameof(Details)}/{country.Id}");  } |  |

## Método DeleteTeam

Agregamos al controlador **CountriesController** el método **DeleteTeam**

|  |  |
| --- | --- |
| **DeleteTeam** | **Comentarios** |
| public async Task<IActionResult> DeleteTeam(int? id)  {  if (id == null)  {  return NotFound();  }  TeamEntity team = await \_context.Teams  .FirstOrDefaultAsync(m => m.Id == id);  if (team == null)  {  return NotFound();  }  CountryEntity country = await \_context.Countries.FirstOrDefaultAsync(c => c.Teams.FirstOrDefault(d => d.Id == team.Id) != null);  \_context.Teams.Remove(team);  await \_context.SaveChangesAsync();  return RedirectToAction($"{nameof(Details)}/{country.Id}");  } |  |

## Borrado en cascada de Países que tengan Provincias y/o Equipos

Para borrar en cascada un país que tenga matriculados provincias y/o equipos, modificamos el método **Delete** en **CountriesController**:

|  |  |
| --- | --- |
| **CountriesController** | **Comentarios** |
| public async Task<IActionResult> Delete(int? id)  {  if (id == null)  {  return NotFound();  }  CountryEntity country = await \_context.Countries  .Include(c => c.Departments)  .ThenInclude(d => d.Cities)  .Include(t => t.Teams)  .FirstOrDefaultAsync(m => m.Id == id);  if (country == null)  {  return NotFound();  }  \_context.Countries.Remove(country);  await \_context.SaveChangesAsync();  return RedirectToAction(nameof(Index));  } |  |

## Método DetailsDepartment

Agregamos al controlador **CountriesController** el método **DetailsDepartments**

|  |  |
| --- | --- |
| **CountriesController** | **Comentarios** |
| public async Task<IActionResult> DetailsDepartment(int? id)  {  if (id == null)  {  return NotFound();  }  DepartmentEntity department = await \_context.Departments  .Include(d => d.Cities)  .FirstOrDefaultAsync(m => m.Id == id);  if (department == null)  {  return NotFound();  }  CountryEntity country = await \_context.Countries.FirstOrDefaultAsync(c => c.Departments.FirstOrDefault(d => d.Id == department.Id) != null);  department.IdCountry = country.Id;  return View(department);  } |  |

Adicionamos la vista **DetailsDepartments**:

|  |  |
| --- | --- |
| **DetailsDepartments** | **Comentarios** |
| @model GenericApp.Common.Entities.Department  @{  ViewData["Title"] = "Details";  }  <link rel="stylesheet" href="https://cdn.datatables.net/1.10.19/css/jquery.dataTables.min.css" />  <h2>Detalle Provincia</h2>  <div>  <h4> </h4>  <hr />  <dl class="dl-horizontal">  <dt>  @Html.DisplayNameFor(model => model.Name)  </dt>  <dd>  @Html.DisplayFor(model => model.Name)  </dd>  <dt>  @Html.DisplayNameFor(model => model.CitiesNumber)  </dt>  <dd>  @Html.DisplayFor(model => model.CitiesNumber)  </dd>  </dl>  </div>  <div>  <a **asp-action**="AddCity" **asp-route-id**="@Model.Id" class="btn btn-primary"><i class="glyphicon glyphicon-plus"></i> City</a>  <a **asp-action**="Edit" **asp-route-id**="@Model.Id" class="btn btn-warning">Edit</a>  <a **asp-action**="Details" **asp-route-id**="@Model.IdCountry" class="btn btn-success">Back to List</a>  </div>  <br />  <div class="row">  <div class="col-md-12">  <div class="panel panel-default">  <div class="panel-heading">  <h3 class="panel-title">Cities</h3>  </div>  <div class="panel-body">  <table class="table table-hover table-responsive table-striped" id="MyTable">  <thead>  <tr>  <th>  @Html.DisplayNameFor(model => model.Cities.FirstOrDefault().Name)  </th>  <th></th>  </tr>  </thead>  <tbody>  @foreach (var item in Model.Cities)  {  <tr>  <td>  @Html.DisplayFor(modelItem => item.Name)  </td>  <td>  <a **asp-action**="EditCity" **asp-route-id**="@item.Id" class="btn btn-warning"><i class="glyphicon glyphicon-pencil"></i></a>  <button data-id="@item.Id" class="btn btn-danger deleteItem" data-toggle="modal" data-target="#deleteDialog"><i class="glyphicon glyphicon-trash"></i></button>  </td>  </tr>  }  </tbody>  </table>  </div>  </div>  </div>  </div>  <**partial** **name**="\_DeleteDialog" />  @section Scripts {  @{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}  <script src="//cdn.datatables.net/1.10.19/js/jquery.dataTables.min.js"></script>  <script src="/js/deleteDialog.js"></script>  <script type="text/javascript">  $(document).ready(function () {  $('#MyTable').DataTable();  // Delete item  sc\_deleteDialog.openModal('deleteItem', true, 'btnYesDelete', '/Countries/DeleteCity/', false);  });  </script>  } |  |

## Método AddCity

Agregamos al controlador **CountriesController** el método **AddCity**

|  |  |
| --- | --- |
| **CountriesController** | **Comentarios** |
| public async Task<IActionResult> AddCity(int? id)  {  if (id == null)  {  return NotFound();  }  DepartmentEntity department = await \_context.Departments.FindAsync(id);  if (department == null)  {  return NotFound();  }  CityEntity model = new CityEntity { IdDepartment = department.Id };  return View(model);  }  [HttpPost]  [ValidateAntiForgeryToken]  public async Task<IActionResult> AddCity(CityEntity city)  {  if (ModelState.IsValid)  {  DepartmentEntity department = await \_context.Departments  .Include(d => d.Cities)  .FirstOrDefaultAsync(c => c.Id == city.IdDepartment);  if (department == null)  {  return NotFound();  }  try  {  city.Id = 0;  department.Cities.Add(city);  \_context.Update(department);  await \_context.SaveChangesAsync();  return RedirectToAction($"{nameof(DetailsDepartment)}/{department.Id}");  }  catch (DbUpdateException dbUpdateException)  {  if (dbUpdateException.InnerException.Message.Contains("duplicate"))  {  ModelState.AddModelError(string.Empty, "Hay un registro con el mismo nombre.");  }  else  {  ModelState.AddModelError(string.Empty, dbUpdateException.InnerException.Message);  }  }  catch (Exception exception)  {  ModelState.AddModelError(string.Empty, exception.Message);  }  }  return View(city);  } |  |

Adicionamos la vista parcial **\_City**:

|  |  |
| --- | --- |
| **\_\_City** | **Comentarios** |
| @model GenericApp.Web.Data.Entities.CityEntity  <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> |  |

Adicionamos la vista **AddCity**:

|  |  |
| --- | --- |
| **AddCity** | **Comentarios** |
| @model GenericApp.Web.Data.Entities.CityEntity  @{  ViewData["Title"] = "Add City";  }  <h2>Agregar Ciudad</h2>  <h4></h4>  <hr />  <div class="row">  <div class="col-md-4">  <form **asp-action**="AddCity">  <div **asp-validation-summary**="ModelOnly" class="text-danger"></div>  <input **type**="hidden" **asp-for**="IdDepartment" />  <**partial** **name**="\_City" />  <div class="form-group">  <input type="submit" value="Grabar" class="btn btn-primary" />  <a **asp-action**="DetailsDepartment" **asp-route-id**="@Model.IdDepartment" class="btn btn-success">Regresar</a>  </div>  </form>  </div>  </div>  @section Scripts {  @{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}  } |  |

## Método EditCity

Agregamos al controlador **CountriesController** el método **EditCity**

|  |  |
| --- | --- |
| **EditCity** | **Comentarios** |
| public async Task<IActionResult> EditCity(int? id)  {  if (id == null)  {  return NotFound();  }  CityEntity city = await \_context.Cities.FindAsync(id);  if (city == null)  {  return NotFound();  }  DepartmentEntity department = await \_context.Departments.FirstOrDefaultAsync(d => d.Cities.FirstOrDefault(c => c.Id == city.Id) != null);  city.IdDepartment = department.Id;  return View(city);  }  [HttpPost]  [ValidateAntiForgeryToken]  public async Task<IActionResult> EditCity(CityEntity city)  {  if (ModelState.IsValid)  {  try  {  \_context.Update(city);  await \_context.SaveChangesAsync();  return RedirectToAction($"{nameof(DetailsDepartment)}/{city.IdDepartment}");  }  catch (DbUpdateException dbUpdateException)  {  if (dbUpdateException.InnerException.Message.Contains("duplicate"))  {  ModelState.AddModelError(string.Empty, "Hay un registro con el mismo nombre.");  }  else  {  ModelState.AddModelError(string.Empty, dbUpdateException.InnerException.Message);  }  }  catch (Exception exception)  {  ModelState.AddModelError(string.Empty, exception.Message);  }  }  return View(city);  } |  |

Adicionamos la vista **EditCity**:

|  |  |
| --- | --- |
| **EditCity** | **Comentarios** |
| @model GenericApp.Web.Data.Entities.CityEntity  @{  ViewData["Title"] = "Edit";  }  <h2>Editar Ciudad</h2>  <h4></h4>  <hr />  <div class="row">  <div class="col-md-4">  <form **asp-action**="EditCity">  <div **asp-validation-summary**="ModelOnly" class="text-danger"></div>  <input **type**="hidden" **asp-for**="Id" />  <input **type**="hidden" **asp-for**="IdDepartment" />  <**partial** **name**="\_City" />  <div class="form-group">  <input type="submit" value="Grabar" class="btn btn-primary" />  <a **asp-action**="DetailsDepartment" **asp-route-id**="@Model.IdDepartment" class="btn btn-success">Regresar</a>  </div>  </form>  </div>  </div>  @section Scripts {  @{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}  } |  |

## Método DeleteCity

Agregamos al controlador **CountriesController** el método **DeleteCity**

|  |  |
| --- | --- |
| **DeleteCity** | **Comentarios** |
| public async Task<IActionResult> DeleteCity(int? id)  {  if (id == null)  {  return NotFound();  }  CityEntity city = await \_context.Cities  .FirstOrDefaultAsync(m => m.Id == id);  if (city == null)  {  return NotFound();  }  DepartmentEntity department = await \_context.Departments.FirstOrDefaultAsync(d => d.Cities.FirstOrDefault(c => c.Id == city.Id) != null);  \_context.Cities.Remove(city);  await \_context.SaveChangesAsync();  return RedirectToAction($"{nameof(DetailsDepartment)}/{department.Id}");  } |  |

# Seeder

## Clase SeedDb

Dentro de la carpeta **Data** creamos la Clase **SeedDb**

|  |  |
| --- | --- |
| **SeedDb** | **Comentarios** |
| using GenericApp.Web.Data;  using GenericApp.Web.Data.Entities;  using System.Collections.Generic;  using System.Linq;  using System.Threading.Tasks;  namespace GenericApp.Web.Data  {  public class SeedDb  {  private readonly DataContext \_context;  public SeedDb(DataContext context)  {  \_context = context;  }  public async Task SeedAsync()  {  await \_context.Database.EnsureCreatedAsync();  await CheckCountriesAsync();  }  private async Task CheckCountriesAsync()  {  if (!\_context.Countries.Any())  {  \_context.Countries.Add(new CountryEntity  {  Name = "Argentina",  Departments = new List<DepartmentEntity>  {  new DepartmentEntity  {  Name = "Córdoba",  Cities = new List<CityEntity>  {  new CityEntity { Name = "Córdoba" },  new CityEntity { Name = "Río Cuarto" },  new CityEntity { Name = "Villa María" }  }  },  new DepartmentEntity  {  Name = "Buenos Aires",  Cities = new List<CityEntity>  {  new CityEntity { Name = "La Plata" },  new CityEntity { Name = "Mar del Plata" },  new CityEntity { Name = "Tandil" }  }  },  new DepartmentEntity  {  Name = "Santa Fe",  Cities = new List<CityEntity>  {  new CityEntity { Name = "Santa Fe" },  new CityEntity { Name = "Rosario" },  new CityEntity { Name = "Venado Tuerto" }  }  }  },  Teams = new List<TeamEntity>  {  new TeamEntity  {  Name = "Talleres",  },  new TeamEntity  {  Name = "Belgrano",  },  new TeamEntity  {  Name = "River Plate",  },  new TeamEntity  {  Name = "Boca Juniors",  },  }  });  \_context.Countries.Add(new CountryEntity  {  Name = "USA",  Departments = new List<DepartmentEntity>  {  new DepartmentEntity  {  Name = "California",  Cities = new List<CityEntity>  {  new CityEntity { Name = "Los Angeles" },  new CityEntity { Name = "San Diego" },  new CityEntity { Name = "San Francisco" }  }  },  new DepartmentEntity  {  Name = "Illinois",  Cities = new List<CityEntity>  {  new CityEntity { Name = "Chicago" },  new CityEntity { Name = "Springfield" }  }  },  new DepartmentEntity  {  Name = "Florida",  Cities = new List<CityEntity>  {  new CityEntity { Name = "Miami" },  new CityEntity{ Name = "Orlando" }  }  }  },  Teams = new List<TeamEntity>  {  new TeamEntity  {  Name = "San Antonio Spurs",  },  new TeamEntity  {  Name = "Los Angeles Lakers",  },  new TeamEntity  {  Name = "Miami Heats",  },  new TeamEntity  {  Name = "New York Knicks",  },  }  });  };  await \_context.SaveChangesAsync();  }  }  } |  |

## Inyección de la Clase SeedDb

Modificamos el **Startup** para inyectar esta clase:

|  |  |
| --- | --- |
| **Startup** | **Comentarios** |
| public void ConfigureServices(IServiceCollection services)  {  services.Configure<CookiePolicyOptions>(options =>  {  options.CheckConsentNeeded = context => true;  options.MinimumSameSitePolicy = SameSiteMode.None;  });  services.AddDbContext<DataContext>(cfg =>  {  cfg.UseSqlServer(Configuration.GetConnectionString("DefaultConnection"));  });  services.AddTransient<SeedDb>();  services.AddMvc().SetCompatibilityVersion(CompatibilityVersion.Version\_2\_1);  } |  |

Modificamos el **Program** para llamar el seeder cada vez que inicie nuestro sitio WEB:

|  |  |
| --- | --- |
| **Program** | **Comentarios** |
| using Microsoft.AspNetCore;  using Microsoft.AspNetCore.Hosting;  using Microsoft.Extensions.DependencyInjection;  using GenericApp.Web.Data;  namespace GenericApp.Web  {  public class Program  {  public static void Main(string[] args)  {  IWebHost host = CreateWebHostBuilder(args).Build();  RunSeeding(host);  host.Run();  }  private static void RunSeeding(IWebHost host)  {  IServiceScopeFactory scopeFactory = host.Services.GetService<IServiceScopeFactory>();  using (IServiceScope scope = scopeFactory.CreateScope())  {  SeedDb seeder = scope.ServiceProvider.GetService<SeedDb>();  seeder.SeedAsync().Wait();  }  }  public static IWebHostBuilder CreateWebHostBuilder(string[] args)  {  return WebHost.CreateDefaultBuilder(args).UseStartup<Startup>();  }  }  } |  |

Borramos la base de datos con el comando:

drop-database

Corremos el proyecto y probamos.

# CRUD para Categories

## Controlador

Clic derecho en Controllers, Agregar, Clase y ponemos como nombre **CategoriesController**

|  |  |
| --- | --- |
| **CategoriesController** | **Comentarios** |
| using GenericApp.Web.Data;  using GenericApp.Web.Data.Entities;  using GenericApp.Web.Helpers;  using GenericApp.Web.Models;  using Microsoft.AspNetCore.Authorization;  using Microsoft.AspNetCore.Mvc;  using Microsoft.EntityFrameworkCore;  using System;  using System.Threading.Tasks;  //using Vereyon.Web;  namespace GenericApp.Web.Controllers  {  [Authorize(Roles = "Admin")]  public class CategoriesController : Controller  {  private readonly DataContext \_context;  private readonly IImageHelper \_imageHelper;  private readonly IConverterHelper \_converterHelper;  // private readonly IFlashMessage \_flashMessage;  public CategoriesController(DataContext context,  IImageHelper imageHelper,  IConverterHelper converterHelper)  // IFlashMessage flashMessage    {  \_context = context;  \_imageHelper = imageHelper;  \_converterHelper = converterHelper;  // this.\_flashMessage = flashMessage;  }  // GET: Categories  public async Task<IActionResult> Index()  {  return View(await \_context.Categories.ToListAsync());  }  // GET: Categories/Create  public IActionResult Create()  {  return View();  }  // POST: Categories/Create  [HttpPost]  [ValidateAntiForgeryToken]  public async Task<IActionResult> Create(CategoryViewModel model)  {  if (ModelState.IsValid)  {  var path = string.Empty;  if (model.ImageFile != null)  {  path = await \_imageHelper.UploadImageAsync(model.ImageFile, "Categories");  }  var category = \_converterHelper.ToCategoryEntity(model, path, true);  \_context.Add(category);  try  {  await \_context.SaveChangesAsync();  return RedirectToAction(nameof(Index));  }  catch (Exception ex)  {  if (ex.InnerException.Message.Contains("duplicate"))  {  ModelState.AddModelError(string.Empty, "Esta Categoría ya existe");  }  else  {  ModelState.AddModelError(string.Empty, ex.InnerException.Message);  }  }  }  return View(model);  }  // GET: Categories/Edit/5  public async Task<IActionResult> Edit(int? id)  {  if (id == null)  {  return NotFound();  }  CategoryEntity Category = await \_context.Categories.FindAsync(id);  if (Category == null)  {  return NotFound();  }  CategoryViewModel model = \_converterHelper.ToCategoryViewModel(Category);  return View(model);  }  // POST: Categories/Edit/5  [HttpPost]  [ValidateAntiForgeryToken]  public async Task<IActionResult> Edit(CategoryViewModel model)  {  if (ModelState.IsValid)  {  if (ModelState.IsValid)  {  var path = model.ImagePath;  if (model.ImageFile != null)  {  path = await \_imageHelper.UploadImageAsync(model.ImageFile, "Categories");  }  CategoryEntity category = \_converterHelper.ToCategoryEntity(model, path, false);  \_context.Update(category);  try  {  await \_context.SaveChangesAsync();  return RedirectToAction(nameof(Index));  }  catch (Exception ex)  {  if (ex.InnerException.Message.Contains("duplicate"))  {  ModelState.AddModelError(string.Empty, "Esta Categoría ya existe");  }  else  {  ModelState.AddModelError(string.Empty, ex.InnerException.Message);  }  }  }  }  return View(model);  }  // POST: Categories/Delete/5  public async Task<IActionResult> Delete(int? id)  {  if (id == null)  {  return NotFound();  }  CategoryEntity category = await \_context.Categories  .FirstOrDefaultAsync(m => m.Id == id);  if (category == null)  {  return NotFound();  }  try  {  \_context.Categories.Remove(category);  await \_context.SaveChangesAsync();  // \_flashMessage.Confirmation("La categoría fue borrada.");  }  catch  {  // \_flashMessage.Danger("No se puede borrar la categoría porque tiene registros relacionados.");  }  return RedirectToAction(nameof(Index));  }  }  } |  |

Agregamos en **Shared/\_Layout.cshtml**

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

## Vistas

Creamos la Vista **Index**

|  |  |
| --- | --- |
| **Index** | **Comentarios** |
| @model IEnumerable<GenericApp.Web.Data.Entities.CategoryEntity>  @{  ViewData["Title"] = "Index";  }  <h2><img src="/images/Categories/categorias.jpg" alt="Image" style="width:70;height:70px;max-height: 100%; width: auto;" />  Categorías</h2>  <flash dismissable="true" />  <p>  <a **asp-action**="Create" class="btn btn-primary">Crear Nueva Categoría</a>  </p>  <table class="table">  <thead>  <tr>  <th>  @Html.DisplayNameFor(model => model.Name)  </th>  <th>  @Html.DisplayNameFor(model => model.ImagePath)  </th>  <th></th>  </tr>  </thead>  <tbody>  @foreach (var item in Model)  {  <tr>  <td>  @Html.DisplayFor(modelItem => item.Name)  </td>  <td>  @if (!string.IsNullOrEmpty(item.ImagePath))  {  <img src="@Url.Content(item.ImagePath)" alt="Image" style="width:50px;height:50px;max-width: 100%; height: auto;" />  }  else  {  <img src="@Url.Content(item.ImageFullPath)" alt="Image" style="width:50px;height:50px;max-width: 100%; height: auto;" />  }  </td>  <td>  <a **asp-action**="Edit" **asp-route-id**="@item.Id" class="btn btn-warning"><i class="glyphicon glyphicon-pencil"></i></a>  <button data-id="@item.Id" class="btn btn-danger deleteItem" data-toggle="modal" data-target="#deleteDialog"><i class="glyphicon glyphicon-trash"></i></button>  </td>  </tr>  }  </tbody>  </table>  <!--Delete Item-->  <div class="modal fade" id="deleteDialog" tabindex="-1" role="dialog" aria-labelledby="exampleModalLabel" aria-hidden="true">  <div class="modal-dialog" role="document">  <div class="modal-content">  <div class="modal-header">  <h5 class="modal-title" id="exampleModalLabel">Borrar Registro</h5>  <button type="button" class="close" data-dismiss="modal" aria-label="Close">  <span aria-hidden="true">&times;</span>  </button>  </div>  <div class="modal-body">  <p>¿Quiere borrar esta Categoría?</p>  </div>  <div class="modal-footer">  <button type="button" class="btn btn-primary" data-dismiss="modal">Cerrar</button>  <button type="button" class="btn btn-danger" id="btnYesDelete">Borrar</button>  </div>  </div>  </div>  </div>  @section Scripts {  @{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}  <script type="text/javascript">  $(document).ready(function () {  // Delete item  var item\_to\_delete;  $('.deleteItem').click((e) => {  item\_to\_delete = e.currentTarget.dataset.id;  });  $("#btnYesDelete").click(function () {  window.location.href = '/Categories/Delete/' + item\_to\_delete;  });  });  </script>  } |  |

Creamos la vista parcial **\_Category**

|  |  |
| --- | --- |
| **\_Category** | **Comentarios** |
| @model GenericApp.Web.Models.CategoryViewModel  <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**="ImageFile" class="control-label"></label>  <input **asp-for**="ImageFile" **type**="file" class="form-control" />  <span **asp-validation-for**="ImageFile" class="text-danger"></span>  </div> |  |

Creamos la vista **Create**

|  |  |
| --- | --- |
| **Create** | **Comentarios** |
| @model GenericApp.Web.Models.CategoryViewModel  @{  ViewData["Title"] = "Create";  }  <h2>  <img src="/images/Categories/addcategory.png" alt="Image" style="width:70;height:70px;max-height: 100%; width: auto;" />  Crear Nueva Categoría  </h2>  <h4></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>  <**partial** **name**="\_Category" />  <div class="form-group">  <input type="submit" value="Crear" class="btn btn-primary" />  <a **asp-action**="Index" class="btn btn-success">Regresar</a>  </div>  </form>  </div>  </div>  @section Scripts {  @{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}  } |  |

Creamos la vista **Edit**

|  |  |
| --- | --- |
| **Edit** | **Comentarios** |
| @model GenericApp.Web.Models.CategoryViewModel  @{  ViewData["Title"] = "Edit";  }  <h2>Editar Categoría</h2>  <h4></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**="ImagePath" />  <**partial** **name**="\_Category" />  <div class="form-group">  <input type="submit" value="Grabar" class="btn btn-primary" />  <a **asp-action**="Index" class="btn btn-success">Regresar</a>  </div>  </form>  </div>  <div class="col-md-4">  @if (!string.IsNullOrEmpty(Model.ImagePath))  {  <img src="@Url.Content(Model.ImagePath)" alt="Image" style="width:200px;height:200px;max-width: 100%; height: auto;" />  }  </div>  </div>  @section Scripts {  @{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}  } |  |

# CRUD para Products

## Controlador

Clic derecho en Controllers, Agregar, Clase y ponemos como nombre **ProductsController**

|  |  |
| --- | --- |
| **ProductsController** | **Comentarios** |
| using Microsoft.AspNetCore.Mvc;  using Microsoft.EntityFrameworkCore;  using GenericApp.Web.Data;  using GenericApp.Web.Helpers;  using System;  using System.Collections.Generic;  using System.Linq;  using System.Threading.Tasks;  using GenericApp.Web.Data.Entities;  using GenericApp.Web.Models;  using Microsoft.AspNetCore.Authorization;  namespace GenericApp.Web.Controllers  {  [Authorize(Roles = "Admin")]  public class ProductsController : Controller  {  private readonly DataContext \_context;  private readonly IImageHelper \_imageHelper;  private readonly ICombosHelper \_combosHelper;  private readonly IConverterHelper \_converterHelper;  public ProductsController(DataContext context, IImageHelper imageHelper, ICombosHelper combosHelper, IConverterHelper converterHelper)  {  \_context = context;  \_imageHelper = imageHelper;  \_combosHelper = combosHelper;  \_converterHelper = converterHelper;  }  public async Task<IActionResult> Index()  {  return View(await \_context.Products  .Include(c => c.Category)  .Include(p => p.ProductImages)  .Include(s=>s.State)  .ToListAsync());  }  public IActionResult Create()  {  ProductViewModel model = new ProductViewModel  {  Categories = \_combosHelper.GetComboCategories(),  States = \_combosHelper.GetComboStates(),  IsActive = true  };  return View(model);  }  [HttpPost]  [ValidateAntiForgeryToken]  public async Task<IActionResult> Create(ProductViewModel model)  {  if (ModelState.IsValid)  {  var path = string.Empty;  try  {  ProductEntity product = await \_converterHelper.ToProductAsync(model, true);  if (model.ImageFile != null)  {  path = await \_imageHelper.UploadImageAsync(model.ImageFile, "Products");  product.ProductImages = new List<ProductImageEntity>  {  new ProductImageEntity { ImagePath = path }  };  }  \_context.Add(product);  await \_context.SaveChangesAsync();  return RedirectToAction(nameof(Index));  }  catch (DbUpdateException dbUpdateException)  {  if (dbUpdateException.InnerException.Message.Contains("duplicate"))  {  ModelState.AddModelError(string.Empty, "Hay 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 = \_combosHelper.GetComboCategories();  model.States = \_combosHelper.GetComboStates();  return View(model);  }  public async Task<IActionResult> Edit(int? id)  {  if (id == null)  {  return NotFound();  }  ProductEntity product = await \_context.Products  .Include(c => c.Category)  .Include(p => p.ProductImages)  .Include(s => s.State)  .FirstOrDefaultAsync(p => p.Id == id);  if (product == null)  {  return NotFound();  }  ProductViewModel model = \_converterHelper.ToProductViewModel(product);  return View(model);  }  [HttpPost]  [ValidateAntiForgeryToken]  public async Task<IActionResult> Edit(ProductViewModel model)  {  var path = string.Empty;  if (ModelState.IsValid)  {  try  {  ProductEntity product = await \_converterHelper.ToProductAsync(model, false);  if (model.ImageFile != null)  {  path = await \_imageHelper.UploadImageAsync(model.ImageFile, "Products");  if (product.ProductImages == null)  {  product.ProductImages = new List<ProductImageEntity>();  }  product.ProductImages.Add(new ProductImageEntity { ImagePath = path });  }  \_context.Update(product);  await \_context.SaveChangesAsync();  return RedirectToAction(nameof(Index));  }  catch (DbUpdateException dbUpdateException)  {  if (dbUpdateException.InnerException.Message.Contains("duplicate"))  {  ModelState.AddModelError(string.Empty, "Este producto ya existe.");  }  else  {  ModelState.AddModelError(string.Empty, dbUpdateException.InnerException.Message);  }  }  catch (Exception exception)  {  ModelState.AddModelError(string.Empty, exception.Message);  }  }  model.Categories = \_combosHelper.GetComboCategories();  return View(model);  }  public async Task<IActionResult> Delete(int? id)  {  if (id == null)  {  return NotFound();  }  ProductEntity product = await \_context.Products  .Include(p => p.ProductImages)  .FirstOrDefaultAsync(p => p.Id == id);  if (product == null)  {  return NotFound();  }  try  {  \_context.Products.Remove(product);  await \_context.SaveChangesAsync();  }  catch (Exception ex)  {  ModelState.AddModelError(string.Empty, ex.Message);  }  return RedirectToAction(nameof(Index));  }  public async Task<IActionResult> Details(int? id)  {  if (id == null)  {  return NotFound();  }  ProductEntity product = await \_context.Products  .Include(c => c.Category)  .Include(p => p.ProductImages)  .Include(s => s.State)  .FirstOrDefaultAsync(m => m.Id == id);  if (product == null)  {  return NotFound();  }  return View(product);  }  public async Task<IActionResult> AddImage(int? id)  {  if (id == null)  {  return NotFound();  }  ProductEntity product = await \_context.Products.FindAsync(id);  if (product == null)  {  return NotFound();  }  AddProductImageViewModel model = new AddProductImageViewModel { ProductId = product.Id };  return View(model);  }  [HttpPost]  [ValidateAntiForgeryToken]  public async Task<IActionResult> AddImage(AddProductImageViewModel model)  {  if (ModelState.IsValid)  {  var path = string.Empty;  ProductEntity product = await \_context.Products  .Include(p => p.ProductImages)  .FirstOrDefaultAsync(p => p.Id == model.ProductId);  if (product == null)  {  return NotFound();  }  try  {  path = await \_imageHelper.UploadImageAsync(model.ImageFile, "Products");  if (product.ProductImages == null)  {  product.ProductImages = new List<ProductImageEntity>();  }  product.ProductImages.Add(new ProductImageEntity { ImagePath = path });  \_context.Update(product);  await \_context.SaveChangesAsync();  return RedirectToAction($"{nameof(Details)}/{product.Id}");  }  catch (Exception exception)  {  ModelState.AddModelError(string.Empty, exception.Message);  }  }  return View(model);  }  public async Task<IActionResult> DeleteImage(int? id)  {  if (id == null)  {  return NotFound();  }  ProductImageEntity productImage = await \_context.ProductImages  .FirstOrDefaultAsync(m => m.Id == id);  if (productImage == null)  {  return NotFound();  }  ProductEntity product = await \_context.Products.FirstOrDefaultAsync(p => p.ProductImages.FirstOrDefault(pi => pi.Id == productImage.Id) != null);  \_context.ProductImages.Remove(productImage);  await \_context.SaveChangesAsync();  return RedirectToAction($"{nameof(Details)}/{product.Id}");  }  }  } |  |

Agregamos en **Shared/\_Layout.cshtml**

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

## Vistas

Creamos la Vista **Index**

|  |  |
| --- | --- |
| **Index** | **Comentarios** |
| @model IEnumerable<GenericApp.Web.Data.Entities.ProductEntity>  @{  ViewData["Title"] = "Index";  }  <h2>  <img src="~/images/Products/productos.jpg" alt="Image" style="width:70;height:70px;max-height: 100%; width: auto;" />  Productos  </h2>  <link rel="stylesheet" href="https://cdn.datatables.net/1.10.19/css/jquery.dataTables.min.css" />  <br />  <p>  <a **asp-action**="Create" class="btn btn-primary"><i class="glyphicon glyphicon-plus"></i> Agregar Nuevo Producto</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.ImageFullPath)  </th>  <th>  @Html.DisplayNameFor(model => model.Price)  </th>  <th>  @Html.DisplayNameFor(model => model.IsActive)  </th>  <th>  Categoría  </th>  <th>  @Html.DisplayNameFor(model => model.ProductImagesNumber)  </th>  <th>  @Html.DisplayNameFor(model => model.Latitude)  </th>  <th>  @Html.DisplayNameFor(model => model.Longitude)  </th>  <th>  Estado  </th>  <th width="120px"></th>  </tr>  </thead>  <tbody>  @foreach (var item in Model)  {  <tr>  <td>  @Html.DisplayFor(modelItem => item.Name)  </td>  <td>  <img src="@item.ImageFullPath" style="width:70px;height:70px;max-width: 100%; height: auto;" />  </td>  <td>  @Html.DisplayFor(modelItem => item.Price)  </td>  <td>  @Html.DisplayFor(modelItem => item.IsActive)  </td>  <td>  @Html.DisplayFor(modelItem => item.Category.Name)  </td>  <td>  @Html.DisplayFor(modelItem => item.ProductImagesNumber)  </td>  <td>  @Html.DisplayFor(modelItem => item.Latitude)  </td>  <td>  @Html.DisplayFor(modelItem => item.Longitude)  </td>  <td>  @Html.DisplayFor(modelItem => item.State.Name)  </td>  <td>  <a **asp-action**="Edit" **asp-route-id**="@item.Id" class="btn btn-warning"><i class="glyphicon glyphicon-pencil"></i></a>  <a **asp-action**="Details" **asp-route-id**="@item.Id" class="btn btn-info"><i class="glyphicon glyphicon-align-justify"></i></a>  <button data-id="@item.Id" class="btn btn-danger deleteItem" data-toggle="modal" data-target="#deleteDialog"><i class="glyphicon glyphicon-trash"></i></button>  </td>  </tr>  }  </tbody>  </table>  </div>  </div>  </div>  </div>  <**partial** **name**="\_DeleteDialog" />  @section Scripts {  @{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}  <script src="//cdn.datatables.net/1.10.19/js/jquery.dataTables.min.js"></script>  <script src="/js/deleteDialog.js"></script>  <script type="text/javascript">  $(document).ready(function () {  $('#MyTable').DataTable();  // Delete item  sc\_deleteDialog.openModal('deleteItem', true, 'btnYesDelete', '/Products/Delete/', false);  });  </script>  } |  |

Creamos la vista parcial **\_Product**

|  |  |
| --- | --- |
| **\_Product** | **Comentarios** |
| @model GenericApp.Web.Models.ProductViewModel  <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>  <textarea **asp-for**="Description" class="form-control"></textarea>  <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 class="form-group">  <label **asp-for**="PriceString" class="control-label"></label>  <input **asp-for**="PriceString" class="form-control" />  <span **asp-validation-for**="PriceString" class="text-danger"></span>  </div>  <div class="form-group">  <label **asp-for**="ImageFile" class="control-label"></label>  <input **asp-for**="ImageFile" **type**="file" class="form-control" />  <span **asp-validation-for**="ImageFile" class="text-danger"></span>  </div>  <div class="form-group">  <label **asp-for**="Latitude" class="control-label"></label>  <textarea **asp-for**="Latitude" class="form-control"></textarea>  <span **asp-validation-for**="Latitude" class="text-danger"></span>  </div>  <div class="form-group">  <label **asp-for**="Longitude" class="control-label"></label>  <textarea **asp-for**="Longitude" class="form-control"></textarea>  <span **asp-validation-for**="Longitude" 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">  <div class="checkbox">  <label>  <input **asp-for**="IsActive" /> @Html.DisplayNameFor(model => model.IsActive)  </label>  </div>  </div> |  |

Creamos la vista **Create**

|  |  |
| --- | --- |
| **Create** | **Comentarios** |
| @model GenericApp.Web.Models.ProductViewModel  @{  ViewData["Title"] = "Create";  }  <h2>  <img src="/images/Products/addproduct.png" alt="Image" style="width:70;height:70px;max-height: 100%; width: auto;" />  Crear Producto  </h2>  <h4></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>  <**partial** **name**="\_Product" />  <div class="form-group">  <input type="submit" value="Crear" class="btn btn-primary" />  <a **asp-action**="Index" class="btn btn-success">Regresar</a>  </div>  </form>  </div>  </div>  @section Scripts {  @{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}  } |  |

Creamos la vista **Edit**

|  |  |
| --- | --- |
| **Edit** | **Comentarios** |
| @model GenericApp.Web.Models.ProductViewModel  @{  ViewData["Title"] = "Edit";  }  <h2>Editar Producto</h2>  <h4></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" />  <**partial** **name**="\_Product" />  <div class="form-group">  <input type="submit" value="Grabar" class="btn btn-primary" />  <a **asp-action**="Index" class="btn btn-success">Regresar</a>  </div>  </form>  </div>  <div class="col-md-4">  <img src="@Model.ImageFullPath" style="width:200px;height:200px;max-width: 100%; height: auto;" />  </div>  </div>  @section Scripts {  @{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}  } |  |

Creamos la vista **Details**

|  |  |
| --- | --- |
| **Details** | **Comentarios** |
| @model GenericApp.Web.Data.Entities.ProductEntity  @{  ViewData["Title"] = "Details";  }  <link rel="stylesheet" href="https://cdn.datatables.net/1.10.19/css/jquery.dataTables.min.css" />  <h2>Detalle Producto</h2>  <div>  <h4></h4>  <hr />  <dl class="dl-horizontal">  <dt>  @Html.DisplayNameFor(model => model.Name)  </dt>  <dd>  @Html.DisplayFor(model => model.Name)  </dd>  <dt>  @Html.DisplayNameFor(model => model.Description)  </dt>  <dd>  @Html.DisplayFor(model => model.Description)  </dd>  <dt>  Categoría  </dt>  <dd>  @Html.DisplayFor(model => model.Category.Name)  </dd>  <dt>  @Html.DisplayNameFor(model => model.Price)  </dt>  <dd>  @Html.DisplayFor(model => model.Price)  </dd>  <dt>  @Html.DisplayNameFor(model => model.IsActive)  </dt>  <dd>  @Html.DisplayFor(model => model.IsActive)  </dd>  <dt>  @Html.DisplayNameFor(model => model.ProductImagesNumber)  </dt>  <dd>  @Html.DisplayFor(model => model.ProductImagesNumber)  </dd>  <dt>  @Html.DisplayNameFor(model => model.Latitude)  </dt>  <dd>  @Html.DisplayFor(model => model.Latitude)  </dd>  <dt>  @Html.DisplayNameFor(model => model.Longitude)  </dt>  <dd>  @Html.DisplayFor(model => model.Longitude)  </dd>  <dt>  @Html.DisplayNameFor(model => model.State)  </dt>  <dd>  @Html.DisplayFor(model => model.State.Name)  </dd>  </dl>  </div>  <div>  <a **asp-action**="AddImage" **asp-route-id**="@Model.Id" class="btn btn-primary"><i class="glyphicon glyphicon-plus"></i> Imagen</a>  <a **asp-action**="Edit" **asp-route-id**="@Model.Id" class="btn btn-warning">Editar</a>  <a **asp-action**="Index" class="btn btn-success">Regresar</a>  </div>  <br />  <div class="row">  <div class="col-md-12">  <div class="panel panel-default">  <div class="panel-heading">  <h3 class="panel-title">Imágenes del Producto</h3>  </div>  <div class="panel-body">  <table class="table table-hover table-responsive table-striped" id="MyTableImages">  <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:70px;height:70px;max-width: 100%; height: auto;" />  </td>  <td>  <button data-id="@item.Id" class="btn btn-danger deleteItem" data-toggle="modal" data-target="#deleteDialog"><i class="glyphicon glyphicon-trash"></i></button>  </td>  </tr>  }  </tbody>  </table>  </div>  </div>  </div>  </div>  <**partial** **name**="\_DeleteDialog" />  @section Scripts {  @{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}  <script src="//cdn.datatables.net/1.10.19/js/jquery.dataTables.min.js"></script>  <script src="/js/deleteDialog.js"></script>  <script type="text/javascript">  $(document).ready(function () {  $('#MyTableImages').DataTable();  $('#MyTableQualifications').DataTable();  // Delete item  sc\_deleteDialog.openModal('deleteItem', true, 'btnYesDelete', '/Products/DeleteImage/', false);  });  </script>  } |  |

Creamos la vista **AddImage**

|  |  |
| --- | --- |
| **AddImage** | **Comentarios** |
| @model GenericApp.Web.Models.AddProductImageViewModel  @{  ViewData["Title"] = "Add Image";  }  <h2>Agregar Imagen</h2>  <h4></h4>  <hr />  <div class="row">  <div class="col-md-4">  <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" **type**="file" class="form-control" />  <span **asp-validation-for**="ImageFile" class="text-danger"></span>  </div>  <div class="form-group">  <input type="submit" value="Grabar" class="btn btn-primary" />  <a **asp-action**="Details" **asp-route-id**="@Model.ProductId" class="btn btn-success">Regresar</a>  </div>  </form>  </div>  </div>  @section Scripts {  @{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}  } |  |

# Adición de usuarios y roles

## Tipos de Usuario

Vamos a tener dos tipos de usuarios; administradores y usuarios.

Creamos una enumeración para diferenciarlos. Creamos la carpeta **Enums** en el proyecto **Common** y dentro de esta carpeta la enumeración **UserType**:

|  |  |
| --- | --- |
| **UserType** | **Comentarios** |
| namespace GenericApp.Common.Enums  {  public enum UserType  {  Admin,  User  }  } |  |

## Clase User

En la Carpeta **Data/Entities** creamos la Clase **UserEntity**

|  |  |
| --- | --- |
| **User** | **Comentarios** |
| using GenericApp.Common.Enums;  using Microsoft.AspNetCore.Identity;  using System.ComponentModel.DataAnnotations;  namespace GenericApp.Web.Data.Entities  {  public class User : IdentityUser  {  [MaxLength(20, ErrorMessage = "El campo {0} debe contener menos de {1} caracteres")]  [Required(ErrorMessage = "El campo {0} es requerido")]  [Display(Name = "Documento")]  public string Document { get; set; }  [Display(Name = "Nombre")]  [MaxLength(50, ErrorMessage = "El campo {0} debe contener menos de {1} caracteres")]  [Required(ErrorMessage = "El campo {0} es requerido")]  public string FirstName { get; set; }  [Display(Name = "Apellido")]  [MaxLength(50, ErrorMessage = "El campo {0} debe contener menos de {1} caracteres")]  [Required(ErrorMessage = "El campo {0} es requerido")]  public string LastName { get; set; }  [Display(Name = "Dirección")]  [MaxLength(100, ErrorMessage = "El campo {0} debe contener menos de {1} caracteres")]  public string Address { get; set; }  [Display(Name = "Foto")]  public string PicturePath { get; set; }  [Display(Name = "Foto")]  public string PictureFullPath => string.IsNullOrEmpty(PicturePath)  ? $"http://keypress.serveftp.net:88/GenericAppApi/images/Users/nouser.png"  : $"http://keypress.serveftp.net:88/GenericAppApi{PicturePath.Substring(1)}";  [Display(Name = "Tipo de Usuario")]  public UserType UserType { get; set; }  [Display(Name = "Ciudad")]  public CityEntity City { get; set; }  [Display(Name = "Hincha de")]  public TeamEntity FavoriteTeam { get; set; }  [Display(Name = "Usuario")]  public string FullName => $"{FirstName} {LastName}";  [Display(Name = "Usuario")]  public string FullNameWithDocument => $"{FirstName} {LastName} - {Document}";  }  } |  |

## Modificar el DataContext

Modificamos el **DataContext**

|  |  |
| --- | --- |
| **DataContext** | **Comentarios** |
| public class DataContext : IdentityDbContext<User>  #region Constructor  public DataContext(DbContextOptions<DataContext> options) : base(options)  {  }  #endregion  … |  |

## ViewModels para manejar Usuarios

En la Carpeta **Models** creamos la Clase **EditUserViewModel**

|  |  |
| --- | --- |
| **EditUserViewModel** | **Comentarios** |
| using Microsoft.AspNetCore.Http;  using Microsoft.AspNetCore.Mvc.Rendering;  using System.Collections.Generic;  using System.ComponentModel.DataAnnotations;  namespace GenericApp.Web.Models  {  public class EditUserViewModel  {  public string Id { get; set; }  [Display(Name = "Documento")]  [MaxLength(20, ErrorMessage = "El campo {0} no puede tener más de {1} caracteres.")]  [Required(ErrorMessage = "El campo {0} es requerido.")]  public string Document { get; set; }  [Display(Name = "Nombre")]  [MaxLength(50, ErrorMessage = "El campo {0} no puede tener más de {1} caracteres.")]  [Required(ErrorMessage = "El campo {0} es requerido.")]  public string FirstName { get; set; }  [Display(Name = "Apellido")]  [MaxLength(50, ErrorMessage = "El campo {0} no puede tener más de {1} caracteres.")]  [Required(ErrorMessage = "El campo {0} es requerido.")]  public string LastName { get; set; }  [Display(Name = "Dirección")]  [MaxLength(100, ErrorMessage = "El campo {0} no puede tener más de {1} caracteres.")]  public string Address { get; set; }  [Display(Name = "Teléfono")]  [MaxLength(50, ErrorMessage = "El campo {0} no puede tener más de {1} caracteres.")]  public string PhoneNumber { get; set; }  [Display(Name = "Foto")]  public string PicturePath { get; set; }  [Display(Name = "Foto")]  public string PictureFullPath => string.IsNullOrEmpty(PicturePath)  ? $"http://keypress.serveftp.net:88/GenericAppApi/images/Users/nouser.png"  : $"http://keypress.serveftp.net:88/GenericAppApi{PicturePath.Substring(1)}";  [Display(Name = "Image")]  public IFormFile ImageFile { get; set; }  [Required(ErrorMessage = "El campo {0} es requerido.")]  [Display(Name = "País")]  [Range(1, int.MaxValue, ErrorMessage = "Debe seleccionar un país")]  public int CountryId { get; set; }  public IEnumerable<SelectListItem> Countries { get; set; }  [Required(ErrorMessage = "El campo {0} es requerido.")]  [Display(Name = "Provincia")]  [Range(1, int.MaxValue, ErrorMessage = "Debe seleccionar una provincia")]  public int DepartmentId { get; set; }  public IEnumerable<SelectListItem> Departments { get; set; }  [Required(ErrorMessage = "El campo {0} es requerido.")]  [Display(Name = "País")]  [Range(1, int.MaxValue, ErrorMessage = "Debe seleccionar un país")]  public int CountryTeamId { get; set; }  public IEnumerable<SelectListItem> Teams { get; set; }  [Required(ErrorMessage = "El campo {0} es requerido.")]  [Display(Name = "Ciudad")]  [Range(1, int.MaxValue, ErrorMessage = "Debe seleccionar una ciudad")]  public int CityId { get; set; }  public IEnumerable<SelectListItem> Cities { get; set; }  [Required(ErrorMessage = "El campo {0} es requerido.")]  [Display(Name = "Equipo")]  [Range(1, int.MaxValue, ErrorMessage = "Debe seleccionar un equipo")]  public int TeamId { get; set; }  [Display(Name = "Tipo Usuario")]  Public string UserType { get; set; }  public IEnumerable<SelectListItem> UserTypes { get; set; }  }  } |  |

En la Carpeta **Models** creamos la Clase **AddUserViewModel**

|  |  |
| --- | --- |
| **AddUserViewModel** | **Comentarios** |
| using System.ComponentModel.DataAnnotations;  namespace GenericApp.Web.Models  {  public class AddUserViewModel : EditUserViewModel  {  [Display(Name = "Email")]  [Required(ErrorMessage = "El campo {0} es requerido.")]  [MaxLength(100, ErrorMessage = "El campo {0} no puede tener más de {1} caracteres.")]  [EmailAddress]  public string Username { get; set; }  [Display(Name = "Password")]  [Required(ErrorMessage = "El campo {0} es requerido.")]  [DataType(DataType.Password)]  [StringLength(20, MinimumLength = 6, ErrorMessage = "El campo {0} debe tener entre {2} y {1} caracteres.")]  public string Password { get; set; }  [Display(Name = "Confirmación de Password")]  [Required(ErrorMessage = "El campo {0} es requerido.")]  [DataType(DataType.Password)]  [StringLength(20, MinimumLength = 6, ErrorMessage = "El campo {0} debe tener entre {2} y {1} caracteres.")]  [Compare("Password")]  public string PasswordConfirm { get; set; }  }  } |  |

En la Carpeta **Models** creamos la Clase **LoginViewModel**

|  |  |
| --- | --- |
| **LoginViewModel** | **Comentarios** |
| using System.ComponentModel.DataAnnotations;  namespace GenericApp.Web.Models  {  public class LoginViewModel  {  [Required(ErrorMessage = "El campo {0} es requerido.")]  [EmailAddress]  public string Username { get; set; }  [Required(ErrorMessage = "El campo {0} es requerido.")]  [MinLength(6, ErrorMessage = "El campo {0} debe tener al menos {1} caracteres.")]  public string Password { get; set; }  public bool RememberMe { get; set; }  }  } |  |

En la Carpeta **Models** creamos la Clase **ChangePasswordViewModel**

|  |  |
| --- | --- |
| **ChangePasswordViewModel** | **Comentarios** |
| using System.ComponentModel.DataAnnotations;  namespace GenericApp.Web.Models  {  public class ChangePasswordViewModel  {  [Display(Name = "Password actual")]  [Required(ErrorMessage = "El campo {0} es requerido.")]  [DataType(DataType.Password)]  [StringLength(20, MinimumLength = 6, ErrorMessage = "El campo {0} debe tener entre {2} y {1} caracteres.")]  public string OldPassword { get; set; }  [Display(Name = "Nuevo password")]  [Required(ErrorMessage = "El campo {0} es requerido.")]  [DataType(DataType.Password)]  [StringLength(20, MinimumLength = 6, ErrorMessage = "El campo {0} debe tener entre {2} y {1} caracteres.")]  public string NewPassword { get; set; }  [Display(Name = "Confirmar Password")]  [Required(ErrorMessage = "El campo {0} es requerido.")]  [DataType(DataType.Password)]  [StringLength(20, MinimumLength = 6, ErrorMessage = "El campo {0} debe tener entre {2} y {1} caracteres.")]  [Compare("NewPassword")]  public string Confirm { get; set; }  }  } |  |

En la Carpeta **Models** creamos la Clase **RecoverPasswordViewModel**

|  |  |
| --- | --- |
| **RecoverPasswordViewModel** | **Comentarios** |
| using System.ComponentModel.DataAnnotations;  namespace GenericApp.Web.Models  {  public class RecoverPasswordViewModel  {  [Required(ErrorMessage = "El campo {0} es requerido")]  [EmailAddress]  public string Email { get; set; }  }  } |  |

En la Carpeta **Models** creamos la Clase **ResetPasswordViewModel**

|  |  |
| --- | --- |
| **ResetPasswordViewModel** | **Comentarios** |
| using System.ComponentModel.DataAnnotations;  namespace GenericApp.Web.Models  {  public class ResetPasswordViewModel  {  [Required(ErrorMessage = "El campo {0} es requerido")]  [EmailAddress]  public string UserName { get; set; }  [Required(ErrorMessage = "El campo {0} es requerido")]  [StringLength(20, MinimumLength = 6, ErrorMessage = "El campo {0} debe tener entre {2} y {1} caracteres.")]  [DataType(DataType.Password)]  public string Password { get; set; }  [Required(ErrorMessage = "El campo {0} es requerido")]  [StringLength(20, MinimumLength = 6, ErrorMessage = "El campo {0} debe tener entre {2} y {1} caracteres.")]  [DataType(DataType.Password)]  [Compare("Password")]  public string ConfirmPassword { get; set; }  [Required]  public string Token { get; set; }  }  } |  |

## UserHelper

En la carpeta **Helpers** creamos la interfaz **IUserHelper**

|  |  |
| --- | --- |
| **IUserHelper** | **Comentarios** |
| using System;  using System.Threading.Tasks;  using Microsoft.AspNetCore.Identity;  using GenericApp.Web.Data.Entities;  using GenericApp.Web.Models;  using GenericApp.Common.Enums;  namespace GenericApp.Web.Helpers  {  public interface IUserHelper  {  Task<User> GetUserAsync(string email);  Task<User> GetUserAsync(Guid userId);  Task<IdentityResult> AddUserAsync(User user, string password);  Task<User> AddUserAsync(AddUserViewModel model, string path, UserType userType);  Task CheckRoleAsync(string roleName);  Task AddUserToRoleAsync(User user, string roleName);  Task<bool> IsUserInRoleAsync(User user, string roleName);  Task<SignInResult> LoginAsync(LoginViewModel model);  Task LogoutAsync();  Task<bool> DeleteUserAsync(string email);  Task<IdentityResult> UpdateUserAsync(User user);  Task<SignInResult> ValidatePasswordAsync(User user, string password);  Task<IdentityResult> ChangePasswordAsync(User user, string oldPassword, string newPassword);  Task<string> GenerateEmailConfirmationTokenAsync(User user);  Task<IdentityResult> ConfirmEmailAsync(User user, string token);  Task<User> GetUserByIdAsync(string userId);  Task<string> GeneratePasswordResetTokenAsync(User user);  Task<IdentityResult> ResetPasswordAsync(User user, string token, string password);  }  } |  |

Creamos la implementación **UserHelper**

|  |  |
| --- | --- |
| **UserHelper** | **Comentarios** |
| using Microsoft.AspNetCore.Identity;  using Microsoft.EntityFrameworkCore;  using GenericApp.Common.Enums;  using GenericApp.Web.Data;  using GenericApp.Web.Data.Entities;  using GenericApp.Web.Models;  using System;  using System.Threading.Tasks;  namespace GenericApp.Web.Helpers  {  public class UserHelper : IUserHelper  {  private readonly UserManager<User> \_userManager;  private readonly RoleManager<IdentityRole> \_roleManager;  private readonly SignInManager<User> \_signInManager;  private readonly DataContext \_context;  public UserHelper(  UserManager<User> userManager,  RoleManager<IdentityRole> roleManager,  SignInManager<User> signInManager,  DataContext context)  {  \_userManager = userManager;  \_roleManager = roleManager;  \_signInManager = signInManager;  \_context = context;  }  public async Task<IdentityResult> AddUserAsync(User user, string password)  {  return await \_userManager.CreateAsync(user, password);  }  public async Task<User> AddUserAsync(AddUserViewModel model, string path, UserType userType)  {  User user = new User  {  Address = model.Address,  Document = model.Document,  Email = model.Username,  FirstName = model.FirstName,  LastName = model.LastName,  PicturePath= path,  PhoneNumber = model.PhoneNumber,  UserName = model.Username,  City = await \_context.Cities.FindAsync(model.CityId),  FavoriteTeam = await \_context.Teams.FindAsync(model.TeamId),  UserType=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;  }  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> GetUserAsync(string email)  {  return await \_context.Users  .Include(u => u.FavoriteTeam)  .ThenInclude(l => l.Country)  .Include(c => c.City)  .ThenInclude(d => d.Department)  .ThenInclude(p => p.Country)  .FirstOrDefaultAsync(u => u.Email == email);  }  public async Task<User> GetUserAsync(Guid userId)  {  return await \_context.Users  .Include(u => u.FavoriteTeam)  .ThenInclude(l => l.Country)  .Include(c => c.City)  .ThenInclude(d => d.Department)  .ThenInclude(p => p.Country)  .FirstOrDefaultAsync(u => u.Id == userId.ToString());  }  public async Task<bool> IsUserInRoleAsync(User user, string roleName)  {  return await \_userManager.IsInRoleAsync(user, roleName);  }  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();  }  public async Task<bool> DeleteUserAsync(string email)  {  var user = await GetUserAsync(email);  if (user == null)  {  return true;  }  var response = await \_userManager.DeleteAsync(user);  return response.Succeeded;  }  public async Task<IdentityResult> UpdateUserAsync(User user)  {  return await \_userManager.UpdateAsync(user);  }  public async Task<SignInResult> ValidatePasswordAsync(User user, string password)  {  return await \_signInManager.CheckPasswordSignInAsync(  user,  password,  false);  }  public async Task<IdentityResult> ChangePasswordAsync(User user, string oldPassword, string newPassword)  {  return await \_userManager.ChangePasswordAsync(user, oldPassword, newPassword);  }  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);  }  public async Task<User> GetUserByIdAsync(string userId)  {  return await \_userManager.FindByIdAsync(userId);  }  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);  }  }  } |  |

## Modificamos el método ConfigureServices del Startup:

Modificamos el método **ConfigureServices** del **Startup**:

|  |  |
| --- | --- |
| **Startup** | **Comentarios** |
| public void ConfigureServices(IServiceCollection services)  {  services.Configure<CookiePolicyOptions>(options =>  {  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>();  services.AddDbContext<DataContext>(cfg =>  {  cfg.UseSqlServer(Configuration.GetConnectionString("DefaultConnection"));  });    services.AddTransient<SeedDb>();  services.AddScoped<IImageHelper, ImageHelper>();  services.AddScoped<IConverterHelper, ConverterHelper>();  services.AddScoped<ICombosHelper, CombosHelper>();  services.AddScoped<IUserHelper, UserHelper>();  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?}");  });  } | Acá se establecen los requisitos para el password |

## Modificación del SeedDb

Agregamos en la Clase **SeedDb**

|  |  |
| --- | --- |
| **SeedDb** | **Comentarios** |
| using GenericApp.Common.Enums;  using GenericApp.Web.Data.Entities;  using GenericApp.Web.Helpers;  using System.Collections.Generic;  using System.Linq;  using System.Threading.Tasks;  namespace GenericApp.Web.Data  {  public class SeedDb  {  private readonly DataContext \_context;  private readonly IUserHelper \_userHelper;  public SeedDb(DataContext context, IUserHelper userHelper)  {  \_context = context;  \_userHelper = userHelper;  }  public async Task SeedAsync()  {  await \_context.Database.EnsureCreatedAsync();  await CheckCountriesAsync();  await CheckRolesAsync();  await CheckUserAsync("17157729", "Luis", "Núñez", "luisalbertonu@gmail.com", "156 814 963", "Espora 2052", UserType.Admin);  }  private async Task CheckRolesAsync()  {  await \_userHelper.CheckRoleAsync(UserType.Admin.ToString());  await \_userHelper.CheckRoleAsync(UserType.User.ToString());  }  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(),  FavoriteTeam = \_context.Teams.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;  }  private async Task CheckCountriesAsync()  ……………… | Inyectamos UserHelper |

Borramos la BD y la creamos de nuevo:

PM> drop-database

## Confirmar Registro por Email

Agregamos en el **Startup.cs**

|  |  |
| --- | --- |
| **Startup.cs** | **Comentarios** |
| 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;  })  .AddDefaultTokenProviders()  .AddEntityFrameworkStores<DataContext>(); |  |

Agregar el nuget “**Mailkit**” al proyecto Web

# Implementación de Login/Logout

## AccountController

Creamos el Controlador **AccountController**

|  |  |
| --- | --- |
| **AccountController** | **Comentarios** |
| using Microsoft.AspNetCore.Authorization;  using Microsoft.AspNetCore.Identity;  using Microsoft.AspNetCore.Mvc;  using Microsoft.EntityFrameworkCore;  using GenericApp.Common.Enums;  using GenericApp.Common.Responses;  using GenericApp.Web.Data;  using GenericApp.Web.Data.Entities;  using GenericApp.Web.Helpers;  using GenericApp.Web.Models;  using System;  using System.Linq;  using System.Threading.Tasks;  namespace GenericApp.Web.Controllers  {  public class AccountController : Controller  {  private readonly DataContext \_context;  private readonly IUserHelper \_userHelper;  private readonly ICombosHelper \_combosHelper;  private readonly IImageHelper \_imageHelper;  private readonly IMailHelper \_mailHelper;  public AccountController(  DataContext context,  IUserHelper userHelper,  ICombosHelper combosHelper,  IImageHelper imageHelper,  IMailHelper mailHelper)  {  \_context = context;  \_userHelper = userHelper;  \_combosHelper = combosHelper;  \_imageHelper = imageHelper;  \_mailHelper = mailHelper;  }  [Authorize(Roles = "Admin")]  public async Task<IActionResult> Index()  {  return View(await \_context.Users  .Include(u => u.City)  .ToListAsync());  }  [Authorize(Roles = "Admin")]  [HttpGet]  public IActionResult Create()  {  AddUserViewModel model = new AddUserViewModel  {  Countries = \_combosHelper.GetComboCountries(),  Departments = \_combosHelper.GetComboDepartments(0),  Cities = \_combosHelper.GetComboCities(0),  Teams = \_combosHelper.GetComboTeams(0),  };  return View(model);  }  [HttpPost]  [ValidateAntiForgeryToken]  public async Task<IActionResult> Create(AddUserViewModel model)  {  if (ModelState.IsValid)  {  var imagePath = string.Empty;  if (model.ImageFile != null)  {  imagePath = await \_imageHelper.UploadImageAsync(model.ImageFile, "users");  }  User user = await \_userHelper.AddUserAsync(model, imagePath, UserType.Admin);  if (user == null)  {  ModelState.AddModelError(string.Empty, "Este mail ya está en uso.");  model.Countries = \_combosHelper.GetComboCountries();  model.Departments = \_combosHelper.GetComboDepartments(model.CountryId);  model.Cities = \_combosHelper.GetComboCities(model.DepartmentId);  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.Username, "Confirmación de Email", $"<h1>Confirmación de Email</h1>" +  $"Para habilitar el usuario, " +  $"por favor haga clic en este link: </br></br><a href = \"{tokenLink}\">Confirmación de Email</a>");  if (response.IsSuccess)  {  ViewBag.Message = "Las instrucciones para habilitar su usuario han sido enviadas a su email.";  return View(model);  }  ModelState.AddModelError(string.Empty, response.Message);  }  model.Countries = \_combosHelper.GetComboCountries();  model.Departments = \_combosHelper.GetComboDepartments(model.CountryId);  model.Cities = \_combosHelper.GetComboCities(model.DepartmentId);  return View(model);  }  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)  {  if (Request.Query.Keys.Contains("ReturnUrl"))  {  return Redirect(Request.Query["ReturnUrl"].First());  }  return RedirectToAction("Index", "Home");  }  ModelState.AddModelError(string.Empty, "Email o password incorrecto.");  }  return View(model);  }  public async Task<IActionResult> Logout()  {  await \_userHelper.LogoutAsync();  return RedirectToAction("Index", "Home");  }  public IActionResult NotAuthorized()  {  return View();  }  public IActionResult Register()  {  AddUserViewModel model = new AddUserViewModel  {  Countries = \_combosHelper.GetComboCountries(),  Departments = \_combosHelper.GetComboDepartments(0),  Cities = \_combosHelper.GetComboCities(0),  Teams = \_combosHelper.GetComboTeams(0),  };  return View(model);  }  [HttpPost]  [ValidateAntiForgeryToken]  public async Task<IActionResult> Register(AddUserViewModel model)  {  if (ModelState.IsValid)  {  var imagePath = string.Empty;  if (model.ImageFile != null)  {  imagePath = await \_imageHelper.UploadImageAsync(model.ImageFile, "Users");  }  User user = await \_userHelper.AddUserAsync(model, imagePath, UserType.User);  if (user == null)  {  ModelState.AddModelError(string.Empty, "Este mail ya está en uso.");  model.Countries = \_combosHelper.GetComboCountries();  model.Departments = \_combosHelper.GetComboDepartments(model.CountryId);  model.Cities = \_combosHelper.GetComboCities(model.DepartmentId);  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);  \_mailHelper.SendMail(model.Username, "Confirmación de Email", $"<h1>Confirmación de Email</h1>" +  $"Para habilitar el usuario, " +  $"por favor haga clic en este link: </br></br><a href = \"{tokenLink}\">Confirmación de Email</a>");  ViewBag.Message = "Las instrucciones para habilitar su usuario han sido enviadas a su email.";  return View(model);  }  model.Countries = \_combosHelper.GetComboCountries();  model.Departments = \_combosHelper.GetComboDepartments(model.CountryId);  model.Cities = \_combosHelper.GetComboCities(model.DepartmentId);  return View(model);  }  public JsonResult GetDepartments(int countryId)  {  CountryEntity country = \_context.Countries  .Include(c => c.Departments)  .FirstOrDefault(c => c.Id == countryId);  if (country == null)  {  return null;  }  return Json(country.Departments.OrderBy(d => d.Name));  }  public JsonResult GetTeams(int countryId)  {  CountryEntity country = \_context.Countries  .Include(c => c.Teams)  .FirstOrDefault(c => c.Id == countryId);  if (country == null)  {  return null;  }  return Json(country.Teams.OrderBy(d => d.Name));  }  public JsonResult GetCities(int departmentId)  {  DepartmentEntity department = \_context.Departments  .Include(d => d.Cities)  .FirstOrDefault(d => d.Id == departmentId);  if (department == null)  {  return null;  }  return Json(department.Cities.OrderBy(c => c.Name));  }  public async Task<IActionResult> ChangeUser()  {  User user = await \_userHelper.GetUserAsync(User.Identity.Name);  if (user == null)  {  return NotFound();  }  DepartmentEntity department = await \_context.Departments.FirstOrDefaultAsync(d => d.Cities.FirstOrDefault(c => c.Id == user.City.Id) != null);  if (department == null)  {  department = await \_context.Departments.FirstOrDefaultAsync();  }  TeamEntity team = await \_context.Teams.FirstOrDefaultAsync(c => c.Id == user.FavoriteTeam.Id);  if (team == null)  {  team = await \_context.Teams.FirstOrDefaultAsync();  }  CountryEntity country = await \_context.Countries.FirstOrDefaultAsync(c => c.Departments.FirstOrDefault(d => d.Id == department.Id) != null);  if (country == null)  {  country = await \_context.Countries.FirstOrDefaultAsync();  }  CountryEntity country2 = await \_context.Countries.FirstOrDefaultAsync(c => c.Teams.FirstOrDefault(d => d.Id == team.Id) != null);  if (country2 == null)  {  country2 = await \_context.Countries.FirstOrDefaultAsync();  }  EditUserViewModel model = new EditUserViewModel  {  Address = user.Address,  FirstName = user.FirstName,  LastName = user.LastName,  PhoneNumber = user.PhoneNumber,  PicturePath = user.PicturePath,  Cities = \_combosHelper.GetComboCities(department.Id),  CityId = user.City.Id,  Countries = \_combosHelper.GetComboCountries(),  CountryId = country.Id,  CountryTeamId= country2.Id,  DepartmentId = department.Id,  Departments = \_combosHelper.GetComboDepartments(country.Id),  Teams = \_combosHelper.GetComboTeams(country2.Id),  TeamId = team.Id,  Id = user.Id,  Document = user.Document,  UserTypes=\_combosHelper.GetComboUserTypes(),  UserType=user.UserType.ToString(),  };  return View(model);  }  [HttpPost]  [ValidateAntiForgeryToken]  public async Task<IActionResult> ChangeUser(EditUserViewModel model)  {  if (ModelState.IsValid)  {  var imagePath = string.Empty;  if (model.ImageFile != null)  {  imagePath = await \_imageHelper.UploadImageAsync(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.PicturePath = imagePath;  user.City = await \_context.Cities.FindAsync(model.CityId);  user.FavoriteTeam = await \_context.Teams.FindAsync(model.TeamId);  user.Document = model.Document;  await \_userHelper.UpdateUserAsync(user);  return RedirectToAction("Index", "Home");  }  model.Cities = \_combosHelper.GetComboCities(model.DepartmentId);  model.Countries = \_combosHelper.GetComboCountries();  model.Departments = \_combosHelper.GetComboDepartments(model.CityId);  model.Teams = \_combosHelper.GetComboTeams(model.CountryId);  return View(model);  }  public IActionResult ChangePasswordMVC()  {  return View();  }  [HttpPost]  public async Task<IActionResult> ChangePasswordMVC(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, "Usuario no encontrado.");  }  }  return View(model);  }  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();  }  public IActionResult RecoverPasswordMVC()  {  return View();  }  [HttpPost]  public async Task<IActionResult> RecoverPasswordMVC(RecoverPasswordViewModel model)  {  if (ModelState.IsValid)  {  User user = await \_userHelper.GetUserAsync(model.Email);  if (user == null)  {  ModelState.AddModelError(string.Empty, "Este mail no corresponde a un 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(model.Email, "Reseteo de Password", $"<h1>Reseteo de Password</h1>" +  $"Para resetear el password por favor haga clic en este link: </br></br>" +  $"<a href = \"{link}\">Reset Password</a>");  ViewBag.Message = "Las instrucciones para recuperar su password han sido enviadas a su email.";  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 = "Password reseteado con éxito.";  return View();  }  ViewBag.Message = "Error mientras se reseteaba el password.";  return View(model);  }  ViewBag.Message = "Usuario no encontrado.";  return View(model);  }  }  } |  |

Dentro de **wwwroot/images** creamos la carpeta **Users**

Dentro agregamos las imágenes **nouser.png, login.png, Reset.png, resetpassword.png**

## Vista Login

Agregamos la Vista **Login**

|  |  |
| --- | --- |
| **Login** | **Comentarios** |
| @model GenericApp.Web.Models.LoginViewModel  @{  ViewData["Title"] = "Login";  }  <h2><img src="/images/Users/login.png" alt="Image" style="width:70;height:70px;max-height: 100%; width: auto;" />  Login</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">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">  <div class="form-check">  <input **asp-for**="RememberMe" **type**="checkbox" class="form-check-input" />  <label **asp-for**="RememberMe" class="form-check-label">Recordarme en este equipo</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>  <a **asp-action**="RecoverPasswordMVC" class="btn btn-link">Olvidó su Password?</a>  </div>  </form>  </div>  </div>  @section Scripts {  @{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}  } |  |

## Vista ConfirmEMail

Agregamos la Vista **ConfirmEMail**

|  |  |
| --- | --- |
| **ConfirmEMail** | **Comentarios** |
| @{  ViewData["Title"] = "Confirm email";  }  <h2>@ViewData["Title"]</h2>  <div>  <p>  Gracias por confirmar su email. Ahora ya puede ingresar al Sistema.  </p>  </div> |  |

## Vista \_User

Agregamos la vista parcial **\_User**

|  |  |
| --- | --- |
| **\_User** | **Comentarios** |
| @model GenericApp.Web.Models.EditUserViewModel  <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 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> |  |

## Vista \_User2

Agregamos la vista parcial **\_User2**

|  |  |
| --- | --- |
| **\_User2** | **Comentarios** |
| @model GenericApp.Web.Models.EditUserViewModel  <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**="DepartmentId" class="control-label"></label>  <select **asp-for**="DepartmentId" **asp-items**="Model.Departments" class="form-control"></select>  <span **asp-validation-for**="DepartmentId" 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>  <h4>  EQUIPO  </h4>  <div class="form-group">  <label **asp-for**="CountryTeamId" class="control-label"></label>  <select **asp-for**="CountryTeamId" **asp-items**="Model.Countries" class="form-control"></select>  <span **asp-validation-for**="CountryTeamId" class="text-danger"></span>  </div>  <div class="form-group">  <label **asp-for**="TeamId" class="control-label"></label>  <select **asp-for**="TeamId" **asp-items**="Model.Teams" class="form-control"></select>  <span **asp-validation-for**="TeamId" class="text-danger"></span>  </div> |  |

## Vista Register

Agregamos la vista **Register**

|  |  |
| --- | --- |
| **Register** | **Comentarios** |
| @model GenericApp.Web.Models.AddUserViewModel  @{  ViewData["Title"] = "Register";  }  <h2>  <img src="/images/Users/nouser.png" alt="Image" style="width:70;height:70px;max-height: 100%; width: auto;" />  Registrar Nuevo Usuario  </h2>  <hr />  <div class="row">  <form **asp-action**="Register" enctype="multipart/form-data">  <div **asp-validation-summary**="ModelOnly" class="text-danger"></div>  <div class="form-group">  <div class="col-md-4">  <h4>  USUARIO Y PASSWORD  </h4>  <label **asp-for**="Username" class="control-label"></label>  <input **asp-for**="Username" class="form-control" />  <span **asp-validation-for**="Username" class="text-danger"></span>      <hr>  <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>  </hr>    <hr>  <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>  </hr>    </div>  <div class="col-md-4">  <div **asp-validation-summary**="ModelOnly" class="text-danger"></div>  <h4>  DATOS PERSONALES  </h4>  <**partial** **name**="\_User" />  </div>  <div class="col-md-4">  <div **asp-validation-summary**="ModelOnly" class="text-danger"></div>  <h4>  CIUDAD  </h4>  <**partial** **name**="\_User2" />  </div>  </form>  <div class="form-group">  <input type="submit" value="Registrar" class="btn btn-primary" />  </div>  </div>    <div class="text-success">  <p>  @ViewBag.Message  </p>  </div>  @section Scripts {  @{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}  <script type="text/javascript">  $(document).ready(function () {  $("#CountryId").change(function () {  $("#DepartmentId").empty();  $("#DepartmentId").append('<option value="0">[Seleccione una provincia...]</option>');  $("#CityId").empty();  $("#CityId").append('<option value="0">[Seleccione una ciudad...]</option>');  $.ajax({  type: 'POST',  url: '@Url.Action("GetDepartments")',  dataType: 'json',  data: { countryId: $("#CountryId").val() },  success: function (subcategories) {  $.each(subcategories, function (i, department) {  //debugger;  $("#DepartmentId").append('<option value="'  + department.id + '">'  + department.name + '</option>');  });  },  error: function (ex) {  alert('Error al buscar las provincias.' + ex);  }  });  return false;  })  $("#CountryTeamId").change(function () {  $("#TeamId").empty();  $("#TeamId").append('<option value="0">[Seleccione un Equipo...]</option>');  $.ajax({  type: 'POST',  url: '@Url.Action("GetTeams")',  dataType: 'json',  data: { countryId: $("#CountryTeamId").val() },  success: function (subcategories) {  $.each(subcategories, function (i, team) {  //debugger;  $("#TeamId").append('<option value="'  + team.id + '">'  + team.name + '</option>');  });  },  error: function (ex) {  alert('Error al buscar los Equipos.' + ex);  }  });  return false;  })  $("#DepartmentId").change(function () {  $("#CityId").empty();  $("#CityId").append('<option value="0">[Seleccione una ciudad...]</option>');  $.ajax({  type: 'POST',  url: '@Url.Action("GetCities")',  dataType: 'json',  data: { departmentId: $("#DepartmentId").val() },  success: function (cities) {  $.each(cities, function (i, city) {  //debugger;  $("#CityId").append('<option value="'  + city.id + '">'  + city.name + '</option>');  });  },  error: function (ex) {  alert('Error al buscar las ciudades.' + ex);  }  });  return false;  })  });  </script>  } |  |

## Vista ChangeUser

Agregamos la vista **ChangeUser**

|  |  |
| --- | --- |
| **ChangeUser** | **Comentarios** |
| @model GenericApp.Web.Models.EditUserViewModel  @{  ViewData["Title"] = "Edit";  }  <h2>  <img src="@Model.PictureFullPath" style="width:70px;height:70px;border-radius:50%" />  Editar Usuario  </h2>  <h4></h4>  <hr />  <div class="row">  <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**="PicturePath" />  <div class="form-group">  <div class="col-md-6">  <div **asp-validation-summary**="ModelOnly" class="text-danger"></div>  <h4>  DATOS PERSONALES  </h4>  <**partial** **name**="\_User" />  </div>  <div class="col-md-6">  <div **asp-validation-summary**="ModelOnly" class="text-danger"></div>  <h4>  CIUDAD  </h4>  <**partial** **name**="\_User2" />  </div>  </form>  <div class="form-group">  <input type="submit" value="Grabar" class="btn btn-primary" />  <a **asp-action**="ChangePasswordMVC" class="btn btn-warning">Cambiar Password</a>  </div>  </div>  </div>  @section Scripts {  @{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}  <script type="text/javascript">  $(document).ready(function () {  $("#CountryId").change(function () {  $("#DepartmentId").empty();  $("#DepartmentId").append('<option value="0">[Seleccione una provincia...]</option>');  $("#CityId").empty();  $("#CityId").append('<option value="0">[Seleccione una ciudad...]</option>');  $.ajax({  type: 'POST',  url: '@Url.Action("GetDepartments")',  dataType: 'json',  data: { countryId: $("#CountryId").val() },  success: function (subcategories) {  $.each(subcategories, function (i, department) {  //debugger;  $("#DepartmentId").append('<option value="'  + department.id + '">'  + department.name + '</option>');  });  },  error: function (ex) {  alert('Error al buscar las provincias.' + ex);  }  });  return false;  })  $("#CountryTeamId").change(function () {  $("#TeamId").empty();  $("#TeamId").append('<option value="0">[Seleccione un equipo...]</option>');  $.ajax({  type: 'POST',  url: '@Url.Action("GetTeams")',  dataType: 'json',  data: { countryId: $("#CountryTeamId").val() },  success: function (subcategories) {  $.each(subcategories, function (i, department) {  //debugger;  $("#TeamId").append('<option value="'  + department.id + '">'  + department.name + '</option>');  });  },  error: function (ex) {  alert('Error al buscar los equipos.' + ex);  }  });  return false;  })  $("#DepartmentId").change(function () {  $("#CityId").empty();  $("#CityId").append('<option value="0">[Seleccione una ciudad...]</option>');  $.ajax({  type: 'POST',  url: '@Url.Action("GetCities")',  dataType: 'json',  data: { departmentId: $("#DepartmentId").val() },  success: function (cities) {  $.each(cities, function (i, city) {  //debugger;  $("#CityId").append('<option value="'  + city.id + '">'  + city.name + '</option>');  });  },  error: function (ex) {  alert('Error al buscar las ciudades.' + ex);  }  });  return false;  })  });  </script>  } |  |

## Vista ChangePasswordMVC

Agregamos la vista **ChangePasswordMVC**

|  |  |
| --- | --- |
| **ChangePasswordMVC** | **Comentarios** |
| @model GenericApp.Web.Models.ChangePasswordViewModel  @{  ViewData["Title"] = "Register";  }  <h2><img src="/images/Users/resetpassword.png" alt="Image" style="width:70;height:70px;max-height: 100%; width: auto;" />  Cambiar Password</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">Password actual</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">Nuevo password</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">Confirmar password</label>  <input **asp-for**="Confirm" **type**="password" class="form-control" />  <span **asp-validation-for**="Confirm" class="text-warning"></span>  </div>  <div class="form-group">  <input type="submit" value="Cambiar password" class="btn btn-primary" />  <a **asp-action**="ChangeUser" class="btn btn-success">Regresar</a>  </div>  </form>  </div>  </div>  @section Scripts {  @{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}  } |  |

## Vista RecoverPasswordMVC

Agregamos la vista **RecoverPasswordMVC**

|  |  |
| --- | --- |
| **RecoverPasswordMVC** | **Comentarios** |
|  |  |

## Vista ResetPassword

Agregamos la vista **ResetPassword**

|  |  |
| --- | --- |
| **ResetPassword** | **Comentarios** |
| @model GenericApp.Web.Models.ResetPasswordViewModel  @{  ViewData["Title"] = "Reset Password";  }  <h1><img src="~/images/Users/resetpassword.png" alt="Image" style="width:70;height:70px;max-height: 100%; width: auto;" />  Resetear Password</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">Email</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">Nuevo 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">Confirmar password</label>  <input **asp-for**="ConfirmPassword" **type**="password" class="form-control" />  <span **asp-validation-for**="ConfirmPassword" class="text-warning"></span>  </div>  <div class="form-group">  <input type="submit" value="Resetear password" class="btn btn-primary" />  </div>  </form>  <div class="text-success">  <p>  @ViewBag.Message  </p>  </div>  </div>  </div>  @section Scripts {  @{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}  } |  |

## Métodos y Vistas para manejo de errores

Agregamos la vista **NotAuthorized**

|  |  |
| --- | --- |
| **NotAuthorized** | **Comentarios** |
| @{  ViewData["Title"] = "NotAuthorized";  }  <br />  <br />  <img src="~/images/error404.png" />  <h2>No está autorizado para esta acción!</h2> |  |

Dentro de **wwwroot/images** colocamos el archivo **error404.png**

Modificamos **Startup.cs** después de las líneas cookies:

|  |  |
| --- | --- |
| **Startup.cs** | **Comentarios** |
| public void ConfigureServices(IServiceCollection services)  {  services.Configure<CookiePolicyOptions>(options =>  {  options.CheckConsentNeeded = context => true;  options.MinimumSameSitePolicy = SameSiteMode.None;  });  services.ConfigureApplicationCookie(options =>  {  options.LoginPath = "/Account/NotAuthorized";  options.AccessDeniedPath = "/Account/NotAuthorized";  }); |  |

Y más abajo, también en **Startup.cs** agregamos

|  |  |
| --- | --- |
| **Startup.cs** | **Comentarios** |
| public void Configure(IApplicationBuilder app, IHostingEnvironment env)  {  if (env.IsDevelopment())  {  app.UseDeveloperExceptionPage();  }  else  {  app.UseExceptionHandler("/Home/Error");  app.UseHsts();  }  app.UseStatusCodePagesWithReExecute("/error/{0}");  app.UseHttpsRedirection();  app.UseStaticFiles();  app.UseAuthentication();  app.UseCookiePolicy(); |  |

En **HomeController** agregamos estos métodos:

|  |  |
| --- | --- |
| **HomeController** | **Comentarios** |
| [Route("error/404")]  public IActionResult Error404()  {  return View();  } |  |

Agregamos la **Vista Error404**

|  |  |
| --- | --- |
| **Error404** | **Comentarios** |
| @{  ViewData["Title"] = "Error404";  }  <br />  <br />  <img src="~/images/error404.png" />  <h2>Lo siento, página no encontrada.</h2> |  |

## Modificamos el menú \_Layout:

|  |  |
| --- | --- |
| **\_Layout** | **Comentarios** |
| <div class="navbar-collapse collapse">  <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>  @if (User.Identity.IsAuthenticated && User.IsInRole("Admin"))  {  <li><a **asp-area**="" **asp-controller**="Countries" **asp-action**="Index">Países</a></li>  <li><a **asp-area**="" **asp-controller**="Categories" **asp-action**="Index">Categorías</a></li>  <li><a **asp-area**="" **asp-controller**="Products" **asp-action**="Index">Productos</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>  }  </div> |  |

# Look & Feel

Modificamos en **Shared/\_Layout**

|  |  |
| --- | --- |
| **Shared/\_Layout** | **Comentarios** |
| <p>&copy; 2021 - GenericApp by Luis Núñez</p> |  |

Agregamos imágenes de 1**200x400** pixeles a la carpeta **wwwroot/images**

Cambiamos la Vista **Index** de **Home** por lo siguiente:

|  |  |
| --- | --- |
| **Index** | **Comentarios** |
| @{  ViewData["Title"] = "Home Page";  }  <div id="myCarousel" class="carousel slide" data-ride="carousel" data-interval="6000">  <ol class="carousel-indicators">  <li data-target="#myCarousel" data-slide-to="0" class="active"></li>  <li data-target="#myCarousel" data-slide-to="1"></li>  <li data-target="#myCarousel" data-slide-to="2"></li>  <li data-target="#myCarousel" data-slide-to="3"></li>  <li data-target="#myCarousel" data-slide-to="4"></li>  <li data-target="#myCarousel" data-slide-to="5"></li>  <li data-target="#myCarousel" data-slide-to="6"></li>  <li data-target="#myCarousel" data-slide-to="7"></li>  </ol>  <div class="carousel-inner" role="listbox">  <div class="item active">  <img src="~/images/v01.jpg" class="img-responsive" />  </div>  <div class="item">  <img src="~/images/v02.jpg" class="img-responsive" />  </div>  <div class="item">  <img src="~/images/v03.jpg" class="img-responsive" />  </div>  <div class="item">  <img src="~/images/v04.jpg" class="img-responsive" />  </div>  <div class="item">  <img src="~/images/v05.jpg" class="img-responsive" />  </div>  <div class="item">  <img src="~/images/v.jpg" class="img-responsive" />  </div>  </div>  <a class="left carousel-control" href="#myCarousel" role="button" data-slide="prev">  <span class="glyphicon glyphicon-chevron-left" aria-hidden="true"></span>  <span class="sr-only">Previous</span>  </a>  <a class="right carousel-control" href="#myCarousel" role="button" data-slide="next">  <span class="glyphicon glyphicon-chevron-right" aria-hidden="true"></span>  <span class="sr-only">Next</span>  </a>  </div>  <div class="row">  </div> |  |

En **\_Layout.cshtml** modificamos

|  |  |
| --- | --- |
| **\_Layout.cshtml** | **Comentarios** |
| <title>@ViewData["Title"] - GenericApp</title>  …….  <a **asp-area**="" **asp-controller**="Home" **asp-action**="Index" class="navbar-brand">GenericApp</a>  ……..  <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">Contacto</a></li> |  |

En **HomeController** modificamos

|  |  |
| --- | --- |
| **HomeController** | **Comentarios** |
| public IActionResult About()  {  ViewData["Message"] = "Esta es una App Genérica.";  return View();  }  public IActionResult Contact()  {  ViewData["Message"] = "Por consultas comunicarse con:";  return View();  } |  |

Vista **Contact**

|  |  |
| --- | --- |
| **Contact** | **Comentarios** |
| @{  ViewData["Title"] = "Contacto";  }  <h2>@ViewData["Title"]</h2>  <h3>@ViewData["Message"]</h3>  <address>  Luis Alberto Núñez<br />  Espora 2052 B° Rosedal - Córdoba<br />  <abr title="Phone">Teléfono:</abr>  351 - 4659552  </address>  <address>  <strong>Contacto:</strong> <a href="mailto:luisalbertonu@gmail.com">luisalbertonu@luisalbertonu.com</a><br />  <strong>Marketing:</strong> <a href="mailto:luisalbertonu@example.com">luisalbertonu@luisalbertonu.com</a>  </address> |  |

Vista **About**

|  |  |
| --- | --- |
| **About** | **Comentarios** |
| @{  ViewData["Title"] = "Acerca de";  }  <h2>@ViewData["Title"]</h2>  <h3>@ViewData["Message"]</h3>  <p>Toda información adicional puede agregarse aquí.</p> |  |

# Administración de Usuarios

Agregamos al método Index de **AccountController**

|  |  |
| --- | --- |
| **AccountController** | **Comentarios** |
| [Authorize(Roles = "Admin")]  public async Task<IActionResult> Index()  {  return View(await \_context.Users  .Include(u => u.City)  .Include(t => t.FavoriteTeam)  .ToListAsync());  } |  |

Agregamos la Vista **Index**

|  |  |
| --- | --- |
| **Index** | **Comentarios** |
| @model IEnumerable<GenericApp.Web.Data.Entities.User>  @{  ViewData["Title"] = "Index";  }  <link rel="stylesheet" href="https://cdn.datatables.net/1.10.19/css/jquery.dataTables.min.css" />  <br />  <p>  <a **asp-action**="Create" class="btn btn-primary"><i class="glyphicon glyphicon-plus"></i> 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">Users</h3>  </div>  <div class="panel-body">  <table class="table table-hover table-responsive table-striped" id="MyTable">  <thead>  <tr>  <th>  @Html.DisplayNameFor(model => model.FullName)  </th>  <th>  @Html.DisplayNameFor(model => model.UserType)  </th>  <th>  @Html.DisplayNameFor(model => model.Email)  </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.PictureFullPath)  </th>  <th>  @Html.DisplayNameFor(model => model.FavoriteTeam.LogoImagePath)  </th>  </tr>  </thead>  <tbody>  @foreach (var item in Model)  {  <tr>  <td>  @Html.DisplayFor(modelItem => item.FullName)  </td>  <td>  @Html.DisplayFor(modelItem => item.UserType)  </td>  <td>  @Html.DisplayFor(modelItem => item.Email)  </td>  <td>  @Html.DisplayFor(modelItem => item.City.Name)  </td>  <td>  @Html.DisplayFor(modelItem => item.Address)  </td>  <td>  @Html.DisplayFor(modelItem => item.PhoneNumber)  </td>  <td>  <img src="@item.PictureFullPath" style="width:80px;height:80px;border-radius:50%" />  </td>  <td>  <img src="@item.FavoriteTeam.LogoImagePath" style="width:80px;height:80px;border-radius:50%" />  </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 src="/js/deleteDialog.js"></script>  <script type="text/javascript">  $(document).ready(function () {  $('#MyTable').DataTable();  });  </script>  } |  |

Agregamos la **Vista Create** para **Account**

|  |  |
| --- | --- |
| **Create** | **Comentarios** |
| @model GenericApp.Web.Models.AddUserViewModel  @{  ViewData["Title"] = "Register";  }  <h2>Agregar Administrador</h2>  <hr />  <div class="row">  <form **asp-action**="Create" enctype="multipart/form-data">  <div **asp-validation-summary**="ModelOnly" class="text-danger"></div>  <div class="form-group">  <div class="col-md-4">  <h4>  USUARIO Y PASSWORD  </h4>  <label **asp-for**="Username" class="control-label"></label>  <input **asp-for**="Username" class="form-control" />  <span **asp-validation-for**="Username" class="text-danger"></span>  <hr>  <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>  </hr>  <hr>  <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>  </hr>  </div>  <div class="col-md-4">  <div **asp-validation-summary**="ModelOnly" class="text-danger"></div>  <h4>  DATOS PERSONALES  </h4>  <**partial** **name**="\_User" />  </div>  <div class="col-md-4">  <div **asp-validation-summary**="ModelOnly" class="text-danger"></div>  <h4>  CIUDAD  </h4>  <**partial** **name**="\_User2" />  </div>  </form>  <div class="form-group">  <input type="submit" value="Registrar" class="btn btn-primary" />  </div>  </div>  <div class="text-success">  <p>  @ViewBag.Message  </p>  </div>  @section Scripts {  @{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}  <script type="text/javascript">  $(document).ready(function () {  $("#CountryId").change(function () {  $("#DepartmentId").empty();  $("#DepartmentId").append('<option value="0">[Seleccione una provincia...]</option>');  $("#CityId").empty();  $("#CityId").append('<option value="0">[Seleccione una ciudad...]</option>');  $.ajax({  type: 'POST',  url: '@Url.Action("GetDepartments")',  dataType: 'json',  data: { countryId: $("#CountryId").val() },  success: function (subcategories) {  $.each(subcategories, function (i, department) {  //debugger;  $("#DepartmentId").append('<option value="'  + department.id + '">'  + department.name + '</option>');  });  },  error: function (ex) {  alert('Error al buscar las provincias.' + ex);  }  });  return false;  })  $("#CountryTeamId").change(function () {  $("#TeamId").empty();  $("#TeamId").append('<option value="0">[Seleccione un Equipo...]</option>');  $.ajax({  type: 'POST',  url: '@Url.Action("GetTeams")',  dataType: 'json',  data: { countryId: $("#CountryTeamId").val() },  success: function (subcategories) {  $.each(subcategories, function (i, team) {  //debugger;  $("#TeamId").append('<option value="'  + team.id + '">'  + team.name + '</option>');  });  },  error: function (ex) {  alert('Error al buscar los Equipos.' + ex);  }  });  return false;  })  $("#DepartmentId").change(function () {  $("#CityId").empty();  $("#CityId").append('<option value="0">[Seleccione una ciudad...]</option>');  $.ajax({  type: 'POST',  url: '@Url.Action("GetCities")',  dataType: 'json',  data: { departmentId: $("#DepartmentId").val() },  success: function (cities) {  $.each(cities, function (i, city) {  //debugger;  $("#CityId").append('<option value="'  + city.id + '">'  + city.name + '</option>');  });  },  error: function (ex) {  alert('Error al buscar las ciudades.' + ex);  }  });  return false;  })  });  </script>  } |  |

En el Proyecto **Web**, en **Views/Shared/\_Layout.cshtml** agregamos:

|  |  |
| --- | --- |
| **\_Layout.cshtml** | **Comentarios** |
| @if (User.Identity.IsAuthenticated && User.IsInRole("Admin"))  {  <li><a **asp-area**="" **asp-controller**="Account" **asp-action**="Index">Usuarios</a></li>  <li><a **asp-area**="" **asp-controller**="Countries" **asp-action**="Index">Países</a></li>  <li><a **asp-area**="" **asp-controller**="Categories" **asp-action**="Index">Categorías</a></li>  <li><a **asp-area**="" **asp-controller**="Products" **asp-action**="Index">Productos</a></li>  } |  |

# API sin seguridad

## CountriesController

Creamos la carpeta **API** dentro de la carpeta Controllers, y dentro de API creamos el **CountriesController** con el siguiente código:

|  |  |
| --- | --- |
| **CountriesController** | **Comentarios** |
| using Microsoft.AspNetCore.Mvc;  using Microsoft.EntityFrameworkCore;  using GenericApp.Web.Data;  namespace GenericApp.Web.Controllers.API  {  [ApiController]  [Route("api/[controller]")]  public class CountriesController : ControllerBase  {  private readonly DataContext \_context;  public CountriesController(DataContext context)  {  \_context = context;  }  [HttpGet]  public IActionResult GetCountries()  {  return Ok(\_context.Countries  .Include(c => c.Departments)  .ThenInclude(d => d.Cities)  .Include(t => t.Teams));  }  }  } |  |

## CategoriesController

|  |  |
| --- | --- |
| **CategoriesController** | **Comentarios** |
| using Microsoft.AspNetCore.Mvc;  using GenericApp.Web.Data;  namespace GenericApp.Web.Controllers.API  {  [ApiController]  [Route("api/[controller]")]  public class CategoriesController : ControllerBase  {  private readonly DataContext \_context;  public CategoriesController(DataContext context)  {  \_context = context;  }  [HttpGet]  public IActionResult GetCategories()  {  return Ok(\_context.Categories);  }  }  } |  |

## StatesController

|  |  |
| --- | --- |
| **StatesController** | **Comentarios** |
| using Microsoft.AspNetCore.Mvc;  using GenericApp.Web.Data;  namespace GenericApp.Web.Controllers.API  {  [ApiController]  [Route("api/[controller]")]  public class StatesController : ControllerBase  {  private readonly DataContext \_context;  public StatesController(DataContext context)  {  \_context = context;  }  [HttpGet]  public IActionResult GetStates()  {  return Ok(\_context.States);  }  }  } |  |

## ProductsController

Creamos el **ProductsController**

|  |  |
| --- | --- |
| **ProductsController** | **Comentarios** |
| using Microsoft.AspNetCore.Mvc;  using Microsoft.EntityFrameworkCore;  using GenericApp.Web.Data;  using GenericApp.Web.Data.Entities;  using System.Collections.Generic;  using System.Linq;  using System.Threading.Tasks;  using GenericApp.Common.Requests;  using System.IO;  using System;  using GenericApp.Common.Helpers;  using GenericApp.Web.Helpers;  using GenericApp.Web.Models;  using GenericApp.Common.Responses;  namespace GenericApp.Web.Controllers.API  {  [ApiController]  [Route("api/[controller]")]  public class ProductsController : ControllerBase  {  private readonly DataContext \_context;  private readonly IConverterHelper \_converterHelper;  private readonly IImageHelper \_imageHelper;  private readonly IFilesHelper \_filesHelper;  public ProductsController(DataContext context, IConverterHelper converterHelper, IImageHelper imageHelper,IFilesHelper filesHelper)  {  \_context = context;  \_converterHelper = converterHelper;  \_imageHelper = imageHelper;  \_filesHelper = filesHelper;  }  [HttpGet]  public async Task<IActionResult> GetProducts()  {  List<ProductEntity> products = await \_context.Products  .Include(c => c.Category)  .Include(p => p.ProductImages)  .Include(s => s.State)  .ToListAsync();  return Ok(\_converterHelper.ToProductResponse(products));  }  // POST: api/Products  [HttpPost]  public async Task<IActionResult> PostProduct([FromBody] ProductRequest request)  {  if (!ModelState.IsValid)  {  return BadRequest(ModelState);  }    var imageUrl = string.Empty;  if (request.PhotoArray != null && request.PhotoArray.Length > 0)  {  var stream = new MemoryStream(request.PhotoArray);  var guid = Guid.NewGuid().ToString();  var file = $"{guid}.jpg";  var folder = "wwwroot\\images\\Products";  var fullPath = $"~/images/Products/{file}";  var response = \_filesHelper.UploadPhoto(stream, folder, file);  if (response)  {  imageUrl = fullPath;  }  }  var Product = new ProductEntity  {  Id = request.Id,  Name = request.Name,  Description = request.Description,  Category= \_context.Categories.FirstOrDefault(c=>c.Id==request.Category.Id),  IsActive=true,  Price=request.Price,  Latitude = request.Latitude,  Longitude = request.Longitude,  State = \_context.States.FirstOrDefault(c => c.Id == request.State.Id),  ProductImages = new List<ProductImageEntity>(),  };  Product.ProductImages.Add(new ProductImageEntity  {  ImagePath= imageUrl  });  \_context.Products.Add(Product);  await \_context.SaveChangesAsync();  ProductResponse newProduct= \_converterHelper.ToProductResponse(Product);  return Ok(newProduct);  }  // PUT: api/Products/5  [HttpPut("{id}")]  public async Task<IActionResult> PutProduct([FromRoute] int id, [FromBody] ProductRequest request)  {  if (!ModelState.IsValid)  {  return BadRequest(ModelState);  }  if (id != request.Id)  {  return BadRequest();  }  var oldProduct = await \_context.Products.FindAsync(request.Id);  if (oldProduct == null)  {  return BadRequest("La Luminaria no existe.");  }  var imageUrl = oldProduct.ImageFullPath;  if (request.PhotoArray != null && request.PhotoArray.Length > 0)  {  var stream = new MemoryStream(request.PhotoArray);  var guid = Guid.NewGuid().ToString();  var file = $"{guid}.jpg";  var folder = "wwwroot\\images\\Products";  var fullPath = $"~/images/Products/{file}";  var response = \_filesHelper.UploadPhoto(stream, folder, file);  if (response)  {  imageUrl = fullPath;  }  }  oldProduct.Id = request.Id;  oldProduct.Name = request.Name;  oldProduct.Description = request.Description;  oldProduct.Latitude = request.Latitude;  oldProduct.Longitude = request.Longitude;  oldProduct.Category = \_context.Categories.FirstOrDefault(c => c.Id == request.Category.Id);  oldProduct.IsActive = true;  oldProduct.Price = request.Price;  oldProduct.State = \_context.States.FirstOrDefault(c => c.Id == request.State.Id);  \_context.Products.Update(oldProduct);  await \_context.SaveChangesAsync();  return Ok();  }  // DELETE: api/Products/5  [HttpDelete("{id}")]  public async Task<IActionResult> DeleteProduct([FromRoute] int id)  {  if (!ModelState.IsValid)  {  return this.BadRequest(ModelState);  }  var Product = await \_context.Products  .FirstOrDefaultAsync(p => p.Id == id);  if (Product == null)  {  return this.NotFound();  }  \_context.Products.Remove(Product);  await \_context.SaveChangesAsync();  return Ok("Producto borrado");  }  }  } |  |

## ProductImagesController

Creamos el **ProductImagesController**

|  |  |
| --- | --- |
| **AccountController** | **Comentarios** |
| IConverter |  |

## AccountController

Creamos el **AccountController**

|  |  |
| --- | --- |
| **AccountController** | **Comentarios** |
| using Microsoft.AspNetCore.Authentication.JwtBearer;  using Microsoft.AspNetCore.Authorization;  using Microsoft.AspNetCore.Identity;  using Microsoft.AspNetCore.Mvc;  using Microsoft.EntityFrameworkCore;  using Microsoft.Extensions.Configuration;  using Microsoft.IdentityModel.Tokens;  using GenericApp.Common.Enums;  using GenericApp.Common.Requests;  using GenericApp.Common.Responses;  using GenericApp.Web.Data;  using GenericApp.Web.Data.Entities;  using GenericApp.Web.Helpers;  using GenericApp.Web.Models;  using System;  using System.Collections.Generic;  using System.IdentityModel.Tokens.Jwt;  using System.Linq;  using System.Security.Claims;  using System.Text;  using System.Threading.Tasks;  namespace GenericApp.Web.Controllers.API  {  [ApiController]  [Route("api/[controller]")]  public class AccountController : ControllerBase  {  private readonly IUserHelper \_userHelper;  private readonly IConfiguration \_configuration;  private readonly IImageHelper \_imageHelper;  private readonly IMailHelper \_mailHelper;  private readonly DataContext \_context;  public AccountController(IUserHelper userHelper, IConfiguration configuration, IImageHelper imageHelper,  IMailHelper mailHelper, DataContext context)  {  \_userHelper = userHelper;  \_configuration = configuration;  \_imageHelper = imageHelper;  \_mailHelper = mailHelper;  \_context = context;  }  [HttpPost]  [Route("CreateToken")]  public async Task<IActionResult> CreateToken([FromBody] LoginViewModel model)  {  if (ModelState.IsValid)  {  User user = await \_userHelper.GetUserAsync(model.Username);  if (user != null)  {  Microsoft.AspNetCore.Identity.SignInResult result = await \_userHelper.ValidatePasswordAsync(user, model.Password);  if (result.Succeeded)  {  Claim[] claims = new[]  {  new Claim(JwtRegisteredClaimNames.Sub, user.Email),  new Claim(JwtRegisteredClaimNames.Jti, Guid.NewGuid().ToString())  };  SymmetricSecurityKey key = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(\_configuration["Tokens:Key"]));  SigningCredentials credentials = new SigningCredentials(key, SecurityAlgorithms.HmacSha256);  JwtSecurityToken token = new JwtSecurityToken(  \_configuration["Tokens:Issuer"],  \_configuration["Tokens:Audience"],  claims,  expires: DateTime.UtcNow.AddDays(99),  signingCredentials: credentials);  var results = new  {  token = new JwtSecurityTokenHandler().WriteToken(token),  expiration = token.ValidTo,  user  };  return Created(string.Empty, results);  }  }  }  return BadRequest();  }  [Authorize(AuthenticationSchemes = JwtBearerDefaults.AuthenticationScheme)]  [HttpGet]  public async Task<IActionResult> GetUser()  {  if (!ModelState.IsValid)  {  return BadRequest();  }  string email = User.Claims.FirstOrDefault(c => c.Type == ClaimTypes.NameIdentifier).Value;  User user = await \_userHelper.GetUserAsync(email);  if (user == null)  {  return NotFound("Este Usuario no existe.");  }  return Ok(user);  }  [HttpPost]  [Route("Register")]  public async Task<IActionResult> PostUser([FromBody] UserRequest request)  {  if (!ModelState.IsValid)  {  return BadRequest(new Response  {  IsSuccess = false,  Message = "Requerimiento inválido",  Result = ModelState  });  }  User user = await \_userHelper.GetUserAsync(request.Email);  if (user != null)  {  return BadRequest(new Response  {  IsSuccess = false,  Message = "Este Usuario ya existe"  });  }  CityEntity city = await \_context.Cities.FindAsync(request.CityId);  if (city == null)  {  return BadRequest(new Response  {  IsSuccess = false,  Message = "Esta ciudad no existe"  });  }  TeamEntity team = await \_context.Teams.FindAsync(request.FavoriteTeamId);  if (team == null)  {  return BadRequest(new Response  {  IsSuccess = false,  Message = "Este Equipo no existe"  });  }  string picturePath = String.Empty;  if (request.PictureArray != null)  {  picturePath = \_imageHelper.UploadImage(request.PictureArray, "users");  }  user = new User  {  Address = request.Address,  Document = request.Document,  Email = request.Email,  FirstName = request.FirstName,  LastName = request.LastName,  PhoneNumber = request.Phone,  UserName = request.Email,  PicturePath = picturePath,  UserType = UserType.User,  City = city,  FavoriteTeam=team  };  IdentityResult result = await \_userHelper.AddUserAsync(user, request.Password);  if (result != IdentityResult.Success)  {  return BadRequest(result.Errors.FirstOrDefault().Description);  }  User userNew = await \_userHelper.GetUserAsync(request.Email);  await \_userHelper.AddUserToRoleAsync(userNew, user.UserType.ToString());  string myToken = await \_userHelper.GenerateEmailConfirmationTokenAsync(user);  string tokenLink = Url.Action("ConfirmEmail", "Account", new  {  userid = user.Id,  token = myToken  }, protocol: HttpContext.Request.Scheme);  \_mailHelper.SendMail(request.Email, "Confirmación de Email", $"<h1>Confirmación de Email</h1>" +  $"Para confirmar su Email por favor haga clic en este link <p><a href = \"{tokenLink}\">Confirmación de Email</a></p>");  return Ok(new Response { IsSuccess = true });  }  [HttpPost]  [Route("RecoverPassword")]  public async Task<IActionResult> RecoverPassword([FromBody] EmailRequest request)  {  if (!ModelState.IsValid)  {  return BadRequest(new Response  {  IsSuccess = false,  Message = "Requerimiento inválido"  });  }  User user = await \_userHelper.GetUserAsync(request.Email);  if (user == null)  {  return BadRequest(new Response  {  IsSuccess = false,  Message = "El Usuario no existe"  });  }  string myToken = await \_userHelper.GeneratePasswordResetTokenAsync(user);  string link = Url.Action("ResetPassword", "Account", new { token = myToken }, protocol: HttpContext.Request.Scheme);  \_mailHelper.SendMail(request.Email, "Recuperación de Password", $"<h1>Recuperación de Password</h1>" +  $"Haga clic en el siguiente link para cambiar su password: <p>" +  $"<a href = \"{link}\">Cambiar Password</a></p>");  return Ok(new Response  {  IsSuccess = true,  Message = "Se le envió un mail con instrucciones para resetear el Password."  });  }  [Authorize(AuthenticationSchemes = JwtBearerDefaults.AuthenticationScheme)]  [HttpPut]  public async Task<IActionResult> PutUser([FromBody] UserRequest request)  {  if (!ModelState.IsValid)  {  return BadRequest(ModelState);  }  string email = User.Claims.FirstOrDefault(c => c.Type == ClaimTypes.NameIdentifier).Value;  User user = await \_userHelper.GetUserAsync(email);  if (user == null)  {  return NotFound("Este Usuario no existe.");  }  CityEntity city = await \_context.Cities.FindAsync(request.CityId);  if (city == null)  {  return BadRequest(new Response  {  IsSuccess = false,  Message = "Este Ciudad no existe."  });  }  TeamEntity team = await \_context.Teams.FindAsync(request.FavoriteTeamId);  if (team == null)  {  return BadRequest(new Response  {  IsSuccess = false,  Message = "Este Equipo no existe."  });  }  string picturePath = user.PicturePath;  if (request.PictureArray != null && request.PictureArray.Length > 0)  {  picturePath = \_imageHelper.UploadImage(request.PictureArray, "Users");  }  user.FirstName = request.FirstName;  user.LastName = request.LastName;  user.Address = request.Address;  user.PhoneNumber = request.Phone;  user.Document = request.Document;  user.City = city;  user.PicturePath = picturePath;  user.FavoriteTeam = team;  IdentityResult respose = await \_userHelper.UpdateUserAsync(user);  if (!respose.Succeeded)  {  return BadRequest(respose.Errors.FirstOrDefault().Description);  }  User updatedUser = await \_userHelper.GetUserAsync(email);  return Ok(updatedUser);  }  [Authorize(AuthenticationSchemes = JwtBearerDefaults.AuthenticationScheme)]  [HttpPost]  [Route("ChangePassword")]  public async Task<IActionResult> ChangePassword([FromBody] ChangePasswordRequest request)  {  if (!ModelState.IsValid)  {  return BadRequest(new Response  {  IsSuccess = false,  Message = "Requerimiento inválido",  Result = ModelState  });  }  string email = User.Claims.FirstOrDefault(c => c.Type == ClaimTypes.NameIdentifier).Value;  User user = await \_userHelper.GetUserAsync(email);  if (user == null)  {  return NotFound("Este Usuario no existe.");  }  IdentityResult result = await \_userHelper.ChangePasswordAsync(user, request.OldPassword, request.NewPassword);  if (!result.Succeeded)  {  return BadRequest(new Response  {  IsSuccess = false,  Message = result.Errors.FirstOrDefault().Description  });  }  return Ok(new Response  {  IsSuccess = true,  Message = "El password fue cambiado con éxito."  });  }  [HttpGet]  [Route("GetUsers")]  public async Task<IActionResult> GetUsers()  {  List<User> users = await \_context.Users  .Include(u => u.City)  .Include(t => t.FavoriteTeam)  .ToListAsync();  return Ok(users);  }  }  } |  |

## Agregamos la configuración en el Startup

|  |  |
| --- | --- |
| **Startup** | **Comentarios** |
| ………….  .AddDefaultTokenProviders()  .AddEntityFrameworkStores<DataContext>();  services.AddAuthentication()  .AddCookie()  .AddJwtBearer(cfg =>  {  cfg.TokenValidationParameters = new TokenValidationParameters  {  ValidIssuer = Configuration["Tokens:Issuer"],  ValidAudience = Configuration["Tokens:Audience"],  IssuerSigningKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(Configuration["Tokens:Key"]))  };  });  services.AddDbContext<DataContext>(cfg =>  {  cfg.UseSqlServer(Configuration.GetConnectionString("DefaultConnection"));  });  …………. |  |

# PROYECTO PRISM

# Inicializar el Xamarin.FFImageLoading.Forms

En **MainActivity** ponemos:

|  |  |
| --- | --- |
| **MainActivity** | **Comentarios** |
| global::Xamarin.Forms.Forms.Init(this, savedInstanceState);  FFImageLoading.Forms.Platform.CachedImageRenderer.Init(true);  LoadApplication(new App(new AndroidInitializer())); |  |

# Iconos e imágenes

En el Proyecto **Prism.Android** agregamos en la carpeta **Resources/drawable/** los íconos:

* ic\_more\_vert.png
* ic\_card\_giftcard
* ic\_person
* ic\_exit\_to\_app
* ic\_home
* ic\_add\_circle\_outline
* ic\_map
* ic\_action\_looks\_two
* ic\_action\_satellite
* ic\_action\_streetview
* ic\_map\_black

y las imágenes:

* logo
* noimage
* nouser
* Reset
* resetpassword
* Splash
* pinamarillo.png
* pinazul.png
* pinceleste.png
* pinnaranja.png
* pinrojo.png
* pinrosa.png
* pinverde.png
* pinverdeclaro.png
* pinvioleta.png

# ApiService

En el proyecto **Common** creamos la carpeta **Services** y dentro de esta creamos la interfaz **IApiService**:

|  |  |
| --- | --- |
| **IApiService** | **Comentarios** |
| using GenericApp.Common.Requests;  using GenericApp.Common.Responses;  using System.IO;  using System.Threading.Tasks;  namespace GenericApp.Common.Services  {  public interface IApiService  {  Task<Response> GetListAsync<T>(string urlBase, string servicePrefix, string controller);  Task<Response> GetTokenAsync(string urlBase, string servicePrefix, string controller, TokenRequest request);  Task<Response> RegisterUserAsync(string urlBase, string servicePrefix, string controller, UserRequest userRequest);  Task<Response> RecoverPasswordAsync(string urlBase, string servicePrefix, string controller, EmailRequest emailRequest);  Task<Response> ModifyUserAsync(string urlBase, string servicePrefix, string controller, UserRequest userRequest, string token);  Task<Response> ChangePasswordAsync(string urlBase, string servicePrefix, string controller, ChangePasswordRequest changePasswordRequest, string token);  Task<Response> PostAsync<T>(string urlBase, string servicePrefix, string controller, T model, string token);  Task<Response> GetListAsync<T>(string urlBase, string servicePrefix, string controller, string token);  Task<Response> PutAsync<T>(string urlBase, string servicePrefix, string controller, T model, string token);  Task<Stream> GetPictureAsync(string urlBase, string servicePrefix);    Task<ResponseT<object>> PostAsync<T>(  string urlBase,  string servicePrefix,  string controller,  T model,  string tokenType,  string accessToken);  Task<ResponseT<object>> PutAsync<T>(  string urlBase,  string servicePrefix,  string controller,  int id,  T model,  string tokenType,  string accessToken);  Task<Response> DeleteAsync(  string urlBase,  string servicePrefix,  string controller,  int id,  string tokenType,  string accessToken);  }  } |  |

En **Common**/**Services** creamos la clase **ApiService**:

|  |  |
| --- | --- |
| **ApiService** | **Comentarios** |
| using Newtonsoft.Json;  using GenericApp.Common.Requests;  using GenericApp.Common.Responses;  using System;  using System.Collections.Generic;  using System.IO;  using System.Net.Http;  using System.Net.Http.Headers;  using System.Text;  using System.Threading.Tasks;  namespace GenericApp.Common.Services  {  public class ApiService : IApiService  {  public async Task<Response> GetListAsync<T>(  string urlBase,  string servicePrefix,  string controller)  {  try  {  HttpClient client = new HttpClient  {  BaseAddress = new Uri(urlBase),  };  string url = $"{servicePrefix}{controller}";  HttpResponseMessage response = await client.GetAsync(url);  string result = await response.Content.ReadAsStringAsync();  if (!response.IsSuccessStatusCode)  {  return new Response  {  IsSuccess = false,  Message = result,  };  }  List<T> list = JsonConvert.DeserializeObject<List<T>>(result);  return new Response  {  IsSuccess = true,  Result = list  };  }  catch (Exception ex)  {  return new Response  {  IsSuccess = false,  Message = ex.Message  };  }  }  public async Task<Response> GetTokenAsync(string urlBase, string servicePrefix, string controller, TokenRequest request)  {  try  {  string requestString = JsonConvert.SerializeObject(request);  StringContent content = new StringContent(requestString, Encoding.UTF8, "application/json");  HttpClient client = new HttpClient  {  BaseAddress = new Uri(urlBase)  };  string url = $"{servicePrefix}{controller}";  HttpResponseMessage response = await client.PostAsync(url, content);  string result = await response.Content.ReadAsStringAsync();  if (!response.IsSuccessStatusCode)  {  return new Response  {  IsSuccess = false,  Message = result,  };  }  TokenResponse token = JsonConvert.DeserializeObject<TokenResponse>(result);  return new Response  {  IsSuccess = true,  Result = token  };  }  catch (Exception ex)  {  return new Response  {  IsSuccess = false,  Message = ex.Message  };  }  }  public async Task<Response> RegisterUserAsync(string urlBase, string servicePrefix, string controller, UserRequest userRequest)  {  try  {  string request = JsonConvert.SerializeObject(userRequest);  StringContent content = new StringContent(request, Encoding.UTF8, "application/json");  HttpClient client = new HttpClient  {  BaseAddress = new Uri(urlBase)  };  string url = $"{servicePrefix}{controller}";  HttpResponseMessage response = await client.PostAsync(url, content);  string answer = await response.Content.ReadAsStringAsync();  Response obj = JsonConvert.DeserializeObject<Response>(answer);  return obj;  }  catch (Exception ex)  {  return new Response  {  IsSuccess = false,  Message = ex.Message  };  }  }  public async Task<Response> RecoverPasswordAsync(string urlBase, string servicePrefix, string controller, EmailRequest emailRequest)  {  try  {  string request = JsonConvert.SerializeObject(emailRequest);  StringContent content = new StringContent(request, Encoding.UTF8, "application/json");  HttpClient client = new HttpClient  {  BaseAddress = new Uri(urlBase)  };  string url = $"{servicePrefix}{controller}";  HttpResponseMessage response = await client.PostAsync(url, content);  string answer = await response.Content.ReadAsStringAsync();  Response obj = JsonConvert.DeserializeObject<Response>(answer);  return obj;  }  catch (Exception ex)  {  return new Response  {  IsSuccess = false,  Message = ex.Message,  };  }  }  public async Task<Response> ModifyUserAsync(string urlBase, string servicePrefix, string controller, UserRequest userRequest, string token)  {  try  {  string request = JsonConvert.SerializeObject(userRequest);  StringContent content = new StringContent(request, Encoding.UTF8, "application/json");  HttpClient client = new HttpClient  {  BaseAddress = new Uri(urlBase)  };  client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("bearer", token);  string url = $"{servicePrefix}{controller}";  HttpResponseMessage response = await client.PutAsync(url, content);  string answer = await response.Content.ReadAsStringAsync();  if (!response.IsSuccessStatusCode)  {  return JsonConvert.DeserializeObject<Response>(answer);  }  UserResponse user = JsonConvert.DeserializeObject<UserResponse>(answer);  return new Response  {  IsSuccess = true,  Result = user  };  }  catch (Exception ex)  {  return new Response  {  IsSuccess = false,  Message = ex.Message  };  }  }  public async Task<Response> ChangePasswordAsync(string urlBase, string servicePrefix, string controller, ChangePasswordRequest changePasswordRequest, string token)  {  try  {  string request = JsonConvert.SerializeObject(changePasswordRequest);  StringContent content = new StringContent(request, Encoding.UTF8, "application/json");  HttpClient client = new HttpClient  {  BaseAddress = new Uri(urlBase)  };  client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("bearer", token);  string url = $"{servicePrefix}{controller}";  HttpResponseMessage response = await client.PostAsync(url, content);  string answer = await response.Content.ReadAsStringAsync();  Response obj = JsonConvert.DeserializeObject<Response>(answer);  return obj;  }  catch (Exception ex)  {  return new Response  {  IsSuccess = false,  Message = ex.Message,  };  }  }  public async Task<Response> PostAsync<T>(string urlBase, string servicePrefix, string controller, T model, string token)  {  try  {  string request = JsonConvert.SerializeObject(model);  StringContent content = new StringContent(request, Encoding.UTF8, "application/json");  HttpClient client = new HttpClient  {  BaseAddress = new Uri(urlBase)  };  client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("bearer", token);  string url = $"{servicePrefix}{controller}";  HttpResponseMessage response = await client.PostAsync(url, content);  string result = await response.Content.ReadAsStringAsync();  if (!response.IsSuccessStatusCode)  {  return new Response  {  IsSuccess = false,  Message = result,  };  }  T item = JsonConvert.DeserializeObject<T>(result);  return new Response  {  IsSuccess = true,  Result = item  };  }  catch (Exception ex)  {  return new Response  {  IsSuccess = false,  Message = ex.Message  };  }  }  public async Task<Response> GetListAsync<T>(string urlBase, string servicePrefix, string controller, string token)  {  try  {  HttpClient client = new HttpClient  {  BaseAddress = new Uri(urlBase),  };  client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("bearer", token);  string url = $"{servicePrefix}{controller}";  HttpResponseMessage response = await client.GetAsync(url);  string result = await response.Content.ReadAsStringAsync();  if (!response.IsSuccessStatusCode)  {  return new Response  {  IsSuccess = false,  Message = result,  };  }  List<T> list = JsonConvert.DeserializeObject<List<T>>(result);  return new Response  {  IsSuccess = true,  Result = list  };  }  catch (Exception ex)  {  return new Response  {  IsSuccess = false,  Message = ex.Message  };  }  }  public async Task<Response> PutAsync<T>(string urlBase, string servicePrefix, string controller, T model, string token)  {  try  {  string request = JsonConvert.SerializeObject(model);  StringContent content = new StringContent(request, Encoding.UTF8, "application/json");  HttpClient client = new HttpClient  {  BaseAddress = new Uri(urlBase)  };  client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("bearer", token);  string url = $"{servicePrefix}{controller}";  HttpResponseMessage response = await client.PutAsync(url, content);  string result = await response.Content.ReadAsStringAsync();  if (!response.IsSuccessStatusCode)  {  return new Response  {  IsSuccess = false,  Message = result,  };  }  T item = JsonConvert.DeserializeObject<T>(result);  return new Response  {  IsSuccess = true,  Result = item  };  }  catch (Exception ex)  {  return new Response  {  IsSuccess = false,  Message = ex.Message  };  }  }  public async Task<Stream> GetPictureAsync(string urlBase, string servicePrefix)  {  try  {  HttpClient client = new HttpClient  {  BaseAddress = new Uri(urlBase),  };  string url = $"{servicePrefix}";  HttpResponseMessage response = await client.GetAsync(url);  Stream stream = await response.Content.ReadAsStreamAsync();  if (!response.IsSuccessStatusCode)  {  return null;  }  return stream;  }  catch  {  return null;  }  }  public async Task<ResponseT<object>> PostAsync<T>(  string urlBase,  string servicePrefix,  string controller,  T model,  string tokenType,  string accessToken)  {  try  {  var request = JsonConvert.SerializeObject(model);  var content = new StringContent(request, Encoding.UTF8, "application/json");  var client = new HttpClient  {  BaseAddress = new Uri(urlBase)  };  client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue(tokenType, accessToken);  var url = $"{servicePrefix}{controller}";  var response = await client.PostAsync(url, content);  var answer = await response.Content.ReadAsStringAsync();  if (!response.IsSuccessStatusCode)  {  return new ResponseT<object>  {  IsSuccess = false,  Message = answer,  };  }  var res = JsonConvert.DeserializeObject<T>(answer);  return new ResponseT<object>  {  IsSuccess = true,  Result = res  };  }  catch (Exception ex)  {  return new ResponseT<object>  {  IsSuccess = false,  Message = ex.Message,  };  }  }  public async Task<ResponseT<object>> PutAsync<T>(  string urlBase,  string servicePrefix,  string controller,  int id,  T model,  string tokenType,  string accessToken)  {  try  {  var request = JsonConvert.SerializeObject(model);  var content = new StringContent(request, Encoding.UTF8, "application/json");  var client = new HttpClient  {  BaseAddress = new Uri(urlBase)  };  client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue(tokenType, accessToken);  var url = $"{servicePrefix}{controller}/{id}";  var response = await client.PutAsync(url, content);  var answer = await response.Content.ReadAsStringAsync();  if (!response.IsSuccessStatusCode)  {  return new ResponseT<object>  {  IsSuccess = false,  Message = answer,  };  }  return new ResponseT<object>  {  IsSuccess = true,  };  }  catch (Exception ex)  {  return new ResponseT<object>  {  IsSuccess = false,  Message = ex.Message,  };  }  }  public async Task<Response> DeleteAsync(  string urlBase,  string servicePrefix,  string controller,  int id,  string tokenType,  string accessToken)  {  try  {  var client = new HttpClient  {  BaseAddress = new Uri(urlBase)  };  client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue(tokenType, accessToken);  var url = $"{servicePrefix}{controller}/{id}";  var response = await client.DeleteAsync(url);  var answer = await response.Content.ReadAsStringAsync();  if (!response.IsSuccessStatusCode)  {  return new Response  {  IsSuccess = false,  Message = answer,  };  }  return new Response  {  IsSuccess = true  };  }  catch (Exception ex)  {  return new Response  {  IsSuccess = false,  Message = ex.Message,  };  }  }  }  } |  |

Adicionamos la inyección del servicio creado en **App.xaml.cs**:

|  |  |
| --- | --- |
| **App.xaml.cs** | **Comentarios** |
| protected override void RegisterTypes(IContainerRegistry containerRegistry)  {  containerRegistry.RegisterSingleton<IAppInfo, AppInfoImplementation>();  containerRegistry.Register<IApiService, ApiService>();  containerRegistry.RegisterForNavigation<NavigationPage>();  containerRegistry.RegisterForNavigation<MainPage, MainPageViewModel>();  } |  |

# App

## App.xaml

En **App.xaml** definimos la dirección web del proyecto, los colores y los estilos que se aplicarán a los controles:

|  |  |
| --- | --- |
| **App.xaml** | **Comentarios** |
| <?xml version="1.0" encoding="utf-8" ?>  <prism:PrismApplication xmlns="http://xamarin.com/schemas/2014/forms"  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  xmlns:prism="http://prismlibrary.com"  x:Class="GenericApp.Prism.App">  <Application.Resources>  <!-- Parameters -->  <x:String x:Key="UrlAPI">http://keypress.serveftp.net:88/GenericAppApi/</x:String>  <x:String x:Key="UrlNoImage">noimage.png</x:String>  <x:String x:Key="UrlNoUser">nouser.png</x:String>  <!-- Colors -->  <Color x:Key="ColorBackground">#D9D9D9</Color>  <Color x:Key="ColorPrimary">#3E518C</Color>  <Color x:Key="ColorSecondary">#8C8C8C</Color>  <Color x:Key="ColorDanger">#73221A</Color>  <Color x:Key="ColorAccent">#73221A</Color>  <Color x:Key="ColorFont">#0D0D0D</Color>  <Color x:Key="ColorFontInverse">#D9D9D9</Color>  <!-- Styles -->  <Style TargetType="Button">  <Setter Property="BackgroundColor" Value="{StaticResource ColorPrimary}" />  <Setter Property="HorizontalOptions" Value="FillAndExpand" />  <Setter Property="TextColor" Value="{StaticResource ColorFontInverse}" />  </Style>  <Style TargetType="Label">  <Setter Property="TextColor" Value="{StaticResource ColorFont}" />  </Style>  <Style x:Key="SecondaryButton" TargetType="Button">  <Setter Property="BackgroundColor" Value="{StaticResource ColorSecondary}" />  </Style>  <Style x:Key="DangerButton" TargetType="Button">  <Setter Property="BackgroundColor" Value="{StaticResource ColorDanger}" />  </Style>  </Application.Resources>  </prism:PrismApplication> | Dirección web del proyecto  Colores  Estilos |

## App.xaml.cs

Modificamos la página de inicio:

|  |  |
| --- | --- |
| **App.xaml.cs** | **Comentarios** |
| if (Settings.IsLogin)  {  await NavigationService.NavigateAsync($"{nameof(GenericAppMasterDetailPage)}/NavigationPage/{nameof(HomePage)}");  }  else  {  await NavigationService.NavigateAsync($"{nameof(GenericAppMasterDetailPage)}/NavigationPage/{nameof(LoginPage)}");  } |  |

# Icono & Splash

Adicionamos una imagen para el Splash en la carpeta **drawable**, las dimensiones deben ser: 480 x 800 pixels o su equivalente y la llamamos **splash.png**.

Adicione estas líneas a **styles.xml**.

|  |  |
| --- | --- |
| **styles.xml** | **Comentarios** |
| <style name="Theme.Splash" parent="android:Theme">  <item name="android:windowBackground">@drawable/Splash</item>  <item name="android:windowNoTitle">true</item>  </style>  </resources> |  |

Para el ícono vamos al **Android** **Asset** **Studio** a la opción **Launcher** **icon** **Generator.** Bajamos el ZIP, lo descomprimimos e incorporamos los archivos **ic\_launcher.png** a las carpetas **mimaps**.

Por otro lado en el archivo Properties de Android, en la opción Manifiesto cambiamos el nombre de la App

En **drawable** hay un archivo llamado **splash\_screen.xml**. Le ponemos:

|  |  |
| --- | --- |
| **splash\_screen.xml** | **Comentarios** |
| <?xml version="1.0" encoding="utf-8" ?>  <layer-list xmlns:android="http://schemas.android.com/apk/res/android">  <item android:drawable="@color/launcher\_background"/>  <item>  <bitmap android:gravity="fill"  android:src="@drawable/splash" />  </item>  </layer-list> |  |

# Settings

En el Proyecto **Common** creamos la carpeta **Helpers** y dentro de esta la clase **Settings**:

|  |  |
| --- | --- |
| **Settings** | **Comentarios** |
| using Plugin.Settings;  using Plugin.Settings.Abstractions;  namespace GenericApp.Common.Helpers  {  public static class Settings  {  private const string \_token = "token";  private const string \_isLogin = "isLogin";  private const string \_product = "product";  private static readonly string \_stringDefault = string.Empty;  private static readonly bool \_boolDefault = false;  private static ISettings AppSettings => CrossSettings.Current;  public static string Token  {  get => AppSettings.GetValueOrDefault(\_token, \_stringDefault);  set => AppSettings.AddOrUpdateValue(\_token, value);  }  public static bool IsLogin  {  get => AppSettings.GetValueOrDefault(\_isLogin, \_boolDefault);  set => AppSettings.AddOrUpdateValue(\_isLogin, value);  }  public static string Product  {  get => AppSettings.GetValueOrDefault(\_product, \_stringDefault);  set => AppSettings.AddOrUpdateValue(\_product, value);  }  }  } |  |

# FilesHelper

Dentro de la carpeta **Common**/**Helpers** creamos la interfaz **IFilesHelper**:

|  |  |
| --- | --- |
| **IFilesHelper** | **Comentarios** |
| using System.IO;  namespace GenericApp.Common.Helpers  {  public interface IFilesHelper  {  byte[] ReadFully(Stream input);  bool UploadPhoto(MemoryStream stream, string folder, string name);  }  } |  |

Luego creamos la implementación **FilesHelper**

|  |  |
| --- | --- |
| **FilesHelper** | **Comentarios** |
| using System.IO;  namespace GenericApp.Common.Helpers  {  public class FilesHelper : IFilesHelper  {  public byte[] ReadFully(Stream input)  {  using (MemoryStream ms = new MemoryStream())  {  input.CopyTo(ms);  return ms.ToArray();  }  }  public bool UploadPhoto(MemoryStream stream, string folder, string name)  {  try  {  stream.Position = 0;  var path = Path.Combine(Directory.GetCurrentDirectory(), folder, name);  File.WriteAllBytes(path, stream.ToArray());  }  catch  {  return false;  }  return true;  }  }  } |  |

Adicionamos la inyección del servicio creado en **App.xaml.cs**:

|  |  |
| --- | --- |
| **App.xaml.cs** | **Comentarios** |
| protected override void RegisterTypes(IContainerRegistry containerRegistry)  {  containerRegistry.RegisterSingleton<IAppInfo, AppInfoImplementation>();  containerRegistry.Register<IApiService, ApiService>();  containerRegistry.Register<IFilesHelper, FilesHelper>();  …… |  |

# RegexHelper

En el Proyecto **Prism** creamos la carpeta **Helpers** y dentro de esta la clase **IRegexHelper**:

|  |  |
| --- | --- |
| **IRegexHelper** | **Comentarios** |
| namespace GenericApp.Prism.Helpers  {  public interface IRegexHelper  {  bool IsValidEmail(string emailaddress);  }  } |  |

Luego creamos la implementación **RegexHelper**

|  |  |
| --- | --- |
| **RegexHelper** | **Comentarios** |
| using System;  using System.Net.Mail;  namespace GenericApp.Prism.Helpers  {  public class RegexHelper : IRegexHelper  {  public bool IsValidEmail(string emailaddress)  {  try  {  new MailAddress(emailaddress);  return true;  }  catch (FormatException)  {  return false;  }  }  }  } |  |

Adicionamos la inyección del servicio creado en **App.xaml.cs**:

|  |  |
| --- | --- |
| **App.xaml.cs** | **Comentarios** |
| protected override void RegisterTypes(IContainerRegistry containerRegistry)  {  containerRegistry.RegisterSingleton<IAppInfo, AppInfoImplementation>();  containerRegistry.Register<IApiService, ApiService>();  containerRegistry.Register<IFilesHelper, FilesHelper>();  containerRegistry.Register<IRegexHelper, RegexHelper>();  …… |  |

# Geolocator Service

En el Proyecto **Common** creamos la interfaz **IGeolocatorService**:

|  |  |
| --- | --- |
| **IGeolocatorService** | **Comentarios** |
| using System.Threading.Tasks;  namespace GenericApp.Common.Services  {  public interface IGeolocatorService  {  double Latitude { get; set; }  double Longitude { get; set; }  Task GetLocationAsync();  }  } |  |

Luego creamos la implementación **GeolocatorService**

|  |  |
| --- | --- |
| **GeolocatorService** | **Comentarios** |
| using Plugin.Geolocator;  using System;  using System.Threading.Tasks;  namespace GenericApp.Common.Services  {  public class GeolocatorService : IGeolocatorService  {  public double Latitude { get; set; }  public double Longitude { get; set; }  public async Task GetLocationAsync()  {  try  {  var locator = CrossGeolocator.Current;  locator.DesiredAccuracy = 50;  var location = await locator.GetPositionAsync();  Latitude = location.Latitude;  Longitude = location.Longitude;  }  catch (Exception ex)  {  ex.ToString();  }  }  }  } |  |

Adicionamos la inyección del servicio creado en **App.xaml.cs**:

|  |  |
| --- | --- |
| **App.xaml.cs** | **Comentarios** |
| protected override void RegisterTypes(IContainerRegistry containerRegistry)  {  containerRegistry.RegisterSingleton<IAppInfo, AppInfoImplementation>();  containerRegistry.Register<IApiService, ApiService>();  containerRegistry.Register<IFilesHelper, FilesHelper>();  containerRegistry.Register<IRegexHelper, RegexHelper>();  containerRegistry.Register<ICombosHelper, CombosHelper>();  containerRegistry.Register<IGeolocatorService, GeolocatorService>();  …… |  |

# Models

## Clase ElementsList

En el Proyecto **Common** creamos la carpeta **Models** y dentro de esta la clase **ElementsList**:

|  |  |
| --- | --- |
| **ElementsList** | **Comentarios** |
| namespace GenericApp.Common.Models  {  public class ElementsList  {  public int Id { get; set; }  public string Name { get; set; }  }  } |  |

## Clase Menu

En la carpeta **Common/** **Models** creamos la clase **Menu**:

|  |  |
| --- | --- |
| **Menu** | **Comentarios** |
| namespace GenericApp.Common.Models  {  public class Menu  {  public string Icon { get; set; }  public string Title { get; set; }  public string PageName { get; set; }  public bool IsLoginRequired { get; set; }  }  } |  |

# CombosHelper

En el Proyecto **Prism** dentro de la carpeta **Helpers** creamos la clase **ICombosHelper**

|  |  |
| --- | --- |
| **ICombosHelper** | **Comentarios** |
| using GenericApp.Common.Models;  using System.Collections.Generic;  namespace GenericApp.Prism.Helpers  {  public interface ICombosHelper  {  IEnumerable<ElementsList> GetCountries();  }  } |  |

Luego creamos la implementación **CombosHelper**

|  |  |
| --- | --- |
| **CombosHelper** | **Comentarios** |
| using GenericApp.Common.Models;  using System.Collections.Generic;  namespace GenericApp.Prism.Helpers  {  public class CombosHelper : ICombosHelper  {  public IEnumerable<ElementsList> GetCountries()  {  List<ElementsList> paymentMethods = new List<ElementsList>  {  new ElementsList { Id = 1, Name = "Opción 1" },  new ElementsList { Id = 2, Name = "Opción 2" },  new ElementsList { Id = 3, Name = "Opción 3" }  };  return paymentMethods;  }  }  } |  |

Adicionamos la inyección del servicio creado en **App.xaml.cs**:

|  |  |
| --- | --- |
| **App.xaml.cs** | **Comentarios** |
| protected override void RegisterTypes(IContainerRegistry containerRegistry)  {  containerRegistry.RegisterSingleton<IAppInfo, AppInfoImplementation>();  containerRegistry.Register<IApiService, ApiService>();  containerRegistry.Register<IFilesHelper, FilesHelper>();  containerRegistry.Register<IRegexHelper, RegexHelper>();  containerRegistry.Register<ICombosHelper, CombosHelper>();  …… |  |

# ItemViewModels

En el Proyecto **Prism** creamos la carpeta **ItemVIewModels**

## Clase MenuItemViewModel

En la carpeta **ItemVIewModels** creamos la clase **MenuItemViewModel**:

|  |  |
| --- | --- |
| **MenuItemViewModel** | **Comentarios** |
| using GenericApp.Common.Helpers;  using GenericApp.Common.Models;  using GenericApp.Prism.Views;  using Prism.Commands;  using Prism.Navigation;  namespace GenericApp.Prism.ItemViewModels  {  public class MenuItemViewModel : Menu  {  private readonly INavigationService \_navigationService;  private DelegateCommand \_selectMenuCommand;  public MenuItemViewModel(INavigationService navigationService)  {  \_navigationService = navigationService;  }  public DelegateCommand SelectMenuCommand => \_selectMenuCommand ?? (\_selectMenuCommand = new DelegateCommand(SelectMenuAsync));  private async void SelectMenuAsync()  {  if (PageName == nameof(LoginPage) && Settings.IsLogin)  {  Settings.IsLogin = false;  Settings.Token = null;  }  if (IsLoginRequired && !Settings.IsLogin)  {  await App.Current.MainPage.DisplayAlert("Error", "Debe estar logueado", "Aceptar");  NavigationParameters parameters = new NavigationParameters  {  { "pageReturn", PageName }  };  await \_navigationService.NavigateAsync($"/{nameof(GenericAppMasterDetailPage)}/NavigationPage/{nameof(LoginPage)}", parameters);  }  else  {  await \_navigationService.NavigateAsync($"/{nameof(GenericAppMasterDetailPage)}/NavigationPage/{PageName}");  }  }  }  } |  |

## Clase ProductItemViewModel

En la carpeta **ItemVIewModels** creamos la clase **ProductItemViewModel**:

|  |  |
| --- | --- |
| **ProductItemViewModel** | **Comentarios** |
| using Newtonsoft.Json;  using GenericApp.Common.Helpers;  using GenericApp.Common.Responses;  using GenericApp.Prism.Views;  using Prism.Commands;  using Prism.Navigation;  namespace GenericApp.Prism.ItemViewModels  {  public class ProductItemViewModel : ProductResponse  {  private readonly INavigationService \_navigationService;  private DelegateCommand \_selectProductCommand;    public float Quantity { get; set; }  public string Remarks { get; set; }  public decimal Value => (decimal)Quantity \* Price;  public ProductItemViewModel(INavigationService navigationService)  {  \_navigationService = navigationService;  }  public DelegateCommand SelectProductCommand => \_selectProductCommand ?? (\_selectProductCommand = new DelegateCommand(SelectProductAsync));  private async void SelectProductAsync()  {  NavigationParameters parameters = new NavigationParameters  {  { "product", this }  };  Settings.Product = JsonConvert.SerializeObject(this);  await \_navigationService.NavigateAsync(nameof(ProductsPage), parameters);  }  }  } |  |

# Recover Password

## RecoverPasswordPage

Dentro de la carpeta **GenericApp.Prism /Views** creamos la **RecoverPasswordPage**

|  |  |
| --- | --- |
| **RecoverPasswordPage** | **Comentarios** |
| <?xml version="1.0" encoding="utf-8" ?>  <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  xmlns:prism="http://prismlibrary.com"  prism:ViewModelLocator.AutowireViewModel="True"  xmlns:ios="clr-namespace:Xamarin.Forms.PlatformConfiguration.iOSSpecific;assembly=Xamarin.Forms.Core"  ios:Page.UseSafeArea="true"  xmlns:busyindicator="clr-namespace:Syncfusion.SfBusyIndicator.XForms;assembly=Syncfusion.SfBusyIndicator.XForms"  xmlns:inputLayout="clr-namespace:Syncfusion.XForms.TextInputLayout;assembly=Syncfusion.Core.XForms"  x:Class="GenericApp.Prism.Views.RecoverPasswordPage"  BackgroundColor="{StaticResource ColorBackground}"  Title="{Binding Title}">  <AbsoluteLayout>  <StackLayout AbsoluteLayout.LayoutBounds="0,0,1,1"  AbsoluteLayout.LayoutFlags="All"  Padding="5">  <ScrollView>  <StackLayout Spacing="0">  <Image HeightRequest="150"  Margin="20"  Source="Reset"/>  <inputLayout:SfTextInputLayout Hint="Email"  ContainerType="Outlined">  <Entry Placeholder="Ingrese EMail..."  Keyboard="Email"  Text="{Binding Email}" />  </inputLayout:SfTextInputLayout>  </StackLayout>  </ScrollView>  <Button Command="{Binding RecoverCommand}"  IsEnabled="{Binding IsEnabled}"  Text="Recuperar Password"  VerticalOptions="EndAndExpand"/>  </StackLayout>  <busyindicator:SfBusyIndicator AnimationType="Gear"  AbsoluteLayout.LayoutBounds=".5,.5,.5,.5"  AbsoluteLayout.LayoutFlags="All"  BackgroundColor="Transparent"  HorizontalOptions="Center"  TextColor="Blue"  IsBusy="{Binding IsRunning}"  Title="Enviando mail..."  VerticalOptions="Center"  ViewBoxWidth="80"  ViewBoxHeight="80" />  </AbsoluteLayout>  </ContentPage> |  |

## RecoverPasswordPageViewModel

Modificamos la **RecoverPasswordPageViewModel**

|  |  |
| --- | --- |
| **RecoverPasswordPageViewModel** | **Comentarios** |
| using GenericApp.Common.Requests;  using GenericApp.Common.Responses;  using GenericApp.Common.Services;  using GenericApp.Prism.Helpers;  using Prism.Commands;  using Prism.Navigation;  using System.Threading.Tasks;  using Xamarin.Essentials;  namespace GenericApp.Prism.ViewModels  {  public class RecoverPasswordPageViewModel : ViewModelBase  {  private readonly INavigationService \_navigationService;  private readonly IApiService \_apiService;  private readonly IRegexHelper \_regexHelper;  private bool \_isRunning;  private bool \_isEnabled;  private DelegateCommand \_recoverCommand;  private string \_email;  public RecoverPasswordPageViewModel(  INavigationService navigationService,  IApiService apiService,  IRegexHelper regexHelper)  : base(navigationService)  {  \_navigationService = navigationService;  \_apiService = apiService;  \_regexHelper = regexHelper;  Title = "Recuperar Password";  IsEnabled = true;  }  public DelegateCommand RecoverCommand => \_recoverCommand ?? (\_recoverCommand = new DelegateCommand(RecoverAsync));  public string Email  {  get => \_email;  set => SetProperty(ref \_email, value);  }  public bool IsRunning  {  get => \_isRunning;  set => SetProperty(ref \_isRunning, value);  }  public bool IsEnabled  {  get => \_isEnabled;  set => SetProperty(ref \_isEnabled, value);  }  public override void OnNavigatedTo(INavigationParameters parameters)  {  base.OnNavigatedTo(parameters);  if (parameters.ContainsKey("email"))  {  Email = parameters.GetValue<string>("email");  }  }  private async void RecoverAsync()  {  bool isValid = await ValidateData();  if (!isValid)  {  return;  }  IsRunning = true;  IsEnabled = false;  if (Connectivity.NetworkAccess != NetworkAccess.Internet)  {  IsRunning = false;  IsEnabled = true;  await App.Current.MainPage.DisplayAlert("Error", "Error de connexión", "Aceptar");  return;  }  EmailRequest request = new EmailRequest { Email = Email };  string url = App.Current.Resources["UrlAPI"].ToString();  Response response = await \_apiService.RecoverPasswordAsync(url, "api", "/Account/RecoverPassword", request);  IsRunning = false;  IsEnabled = true;  if (!response.IsSuccess)  {  if (response.Message == "Error001")  {  await App.Current.MainPage.DisplayAlert("Error", "El Usuario no existe", "Aceptar");  }  else  {  await App.Current.MainPage.DisplayAlert("Error", response.Message, "Aceptar");  }  return;  }  await App.Current.MainPage.DisplayAlert("Ok", "Se le ha enviado un mail con instrucciones para resetear su password.", "Aceptar");  await \_navigationService.GoBackAsync();  }  private async Task<bool> ValidateData()  {  if (string.IsNullOrEmpty(Email) || !\_regexHelper.IsValidEmail(Email))  {  await App.Current.MainPage.DisplayAlert("Error", "Email incorrecto", "Aceptar");  return false;  }  return true;  }  }  } |  |

# Registro de Usuario

## MainActivity

Modificamos el **MainActivity**

|  |  |
| --- | --- |
| **MainActivity** | **Comentarios** |
| using Android.App;  using Android.Content.PM;  using Android.OS;  using Plugin.Permissions;  using Prism;  using Prism.Ioc;  using Syncfusion.SfBusyIndicator.XForms.Droid;  namespace GenericApp.Prism.Droid  {  [Activity(Theme = "@style/MainTheme",  ConfigurationChanges = ConfigChanges.ScreenSize | ConfigChanges.Orientation | ConfigChanges.UiMode | ConfigChanges.ScreenLayout | ConfigChanges.SmallestScreenSize)]  public class MainActivity : global::Xamarin.Forms.Platform.Android.FormsAppCompatActivity  {  protected override void OnCreate(Bundle savedInstanceState)  {  TabLayoutResource = Resource.Layout.Tabbar;  ToolbarResource = Resource.Layout.Toolbar;  base.OnCreate(savedInstanceState);  global::Xamarin.Forms.Forms.Init(this, savedInstanceState);  new SfBusyIndicatorRenderer();  FFImageLoading.Forms.Platform.CachedImageRenderer.Init(true);  LoadApplication(new App(new AndroidInitializer()));  }  public override void OnRequestPermissionsResult(int requestCode, string[] permissions, Android.Content.PM.Permission[] grantResults)  {  Xamarin.Essentials.Platform.OnRequestPermissionsResult(requestCode, permissions, grantResults);  PermissionsImplementation.Current.OnRequestPermissionsResult(requestCode, permissions, grantResults);  base.OnRequestPermissionsResult(requestCode, permissions, grantResults);  }  }  public class AndroidInitializer : IPlatformInitializer  {  public void RegisterTypes(IContainerRegistry containerRegistry)  {  // Register any platform specific implementations  }  }  } |  |

## AndroidManifest

Modificamos el **AndroidManifest**:

|  |  |
| --- | --- |
| **AndroidManifest** | **Comentarios** |
| <?xml version="1.0" encoding="utf-8"?>  <manifest xmlns:android="http://schemas.android.com/apk/res/android" android:versionCode="1" android:versionName="1.0" package="com.companyname.appname" android:installLocation="auto">  <uses-sdk android:minSdkVersion="21" android:targetSdkVersion="29" />  <uses-permission android:name="android.permission.INTERNET" />  <uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE" />  <uses-permission android:name="android.permission.ACCESS\_WIFI\_STATE" />  <uses-permission android:name="android.permission.CAMERA" />  <uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE" />  <uses-permission android:name="android.permission.READ\_EXTERNAL\_STORAGE" />  <uses-permission android:name="android.permission.ACCESS\_MOCK\_LOCATION" />  <uses-permission android:name="android.permission.ACCESS\_LOCATION\_EXTRA\_COMMANDS" />  <uses-permission android:name="android.permission.ACCESS\_FINE\_LOCATION" />  <uses-permission android:name="android.permission.ACCESS\_COARSE\_LOCATION" />  <application android:label="@string/app\_name" android:icon="@drawable/ic\_launcher">  <provider android:name="android.support.v4.content.FileProvider"  android:authorities="${applicationId}.fileprovider"  android:exported="false"  android:grantUriPermissions="true">  <meta-data android:name="android.support.FILE\_PROVIDER\_PATHS"  android:resource="@xml/file\_paths"></meta-data>  </provider>  <meta-data  android:name="com.google.android.maps.v2.API\_KEY"  android:value="AIzaSyAtxvXVhbzV9OTwZh8UxVsW2A58WYf-Btc" />  <uses-library android:name="org.apache.http.legacy" android:required="false" />  </application>  </manifest> |  |

## Carpeta xml y archivo file\_paths.xml para sacar fotos

Adicionamos la carpeta **xml** dentro de **Resources** y en esta, adicionamos el **file\_paths.xml**:

|  |  |
| --- | --- |
| **file\_paths.xml** | **Comentarios** |
| <?xml version="1.0" encoding="UTF-8" ?>  <paths xmlns:android="http://schemas.android.com/apk/res/android">  <external-files-path name="my\_images" path="Pictures" />  <external-files-path name="my\_movies" path="Movies" />  </paths> |  |

## RegisterPage

Dentro de la carpeta **GenericApp.Prism /Views** creamos la **RegisterPage**

|  |  |
| --- | --- |
| **RegisterPage** | **Comentarios** |
| <?xml version="1.0" encoding="utf-8" ?>  <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  xmlns:prism="http://prismlibrary.com"  prism:ViewModelLocator.AutowireViewModel="True"  xmlns:ios="clr-namespace:Xamarin.Forms.PlatformConfiguration.iOSSpecific;assembly=Xamarin.Forms.Core"  ios:Page.UseSafeArea="true"  xmlns:ffimageloading="clr-namespace:FFImageLoading.Forms;assembly=FFImageLoading.Forms"  xmlns:fftransformations="clr-namespace:FFImageLoading.Transformations;assembly=FFImageLoading.Transformations"  xmlns:inputLayout="clr-namespace:Syncfusion.XForms.TextInputLayout;assembly=Syncfusion.Core.XForms"  x:Class="GenericApp.Prism.Views.RegisterPage"  xmlns:busyindicator="clr-namespace:Syncfusion.SfBusyIndicator.XForms;assembly=Syncfusion.SfBusyIndicator.XForms"  BackgroundColor="{StaticResource ColorBackground}"  Title="{Binding Title}">  <AbsoluteLayout>  <StackLayout AbsoluteLayout.LayoutBounds="0,0,1,1"  AbsoluteLayout.LayoutFlags="All"  Padding="5">  <ScrollView>  <StackLayout Spacing="0">  <ffimageloading:CachedImage Aspect="AspectFit"  Source="{Binding Image}"  CacheDuration= "40"  HeightRequest="150"  Margin="5"  RetryCount= "3"  RetryDelay= "600">  <ffimageloading:CachedImage.Transformations>  <fftransformations:CircleTransformation />  </ffimageloading:CachedImage.Transformations>  <ffimageloading:CachedImage.GestureRecognizers>  <TapGestureRecognizer Command="{Binding ChangeImageCommand}"/>  </ffimageloading:CachedImage.GestureRecognizers>  </ffimageloading:CachedImage>  <Label Text="EMail:"  FontAttributes="Bold">  </Label>  <inputLayout:SfTextInputLayout Hint="Email"  ContainerType="Outlined">  <Entry Placeholder="Ingrese EMail..."  Keyboard="Email"  Text="{Binding User.Email}" />  </inputLayout:SfTextInputLayout>  <Label Text="Datos personales:"  FontAttributes="Bold">  </Label>  <inputLayout:SfTextInputLayout Hint="Documento"  ContainerType="Outlined">  <Entry Placeholder="Ingrese Documento..."  Text="{Binding User.Document}" />  </inputLayout:SfTextInputLayout>  <inputLayout:SfTextInputLayout Hint="Nombre"  ContainerType="Outlined">  <Entry Placeholder="Ingrese Nombre..."  Text="{Binding User.FirstName}" />  </inputLayout:SfTextInputLayout>  <inputLayout:SfTextInputLayout Hint="Apellido"  ContainerType="Outlined">  <Entry Placeholder="Ingrese Apellido..."  Text="{Binding User.LastName}" />  </inputLayout:SfTextInputLayout>    <inputLayout:SfTextInputLayout Hint="País"  ContainerType="Outlined">  <Picker ItemDisplayBinding="{Binding Name}"  ItemsSource="{Binding Countries}"  SelectedItem="{Binding Country}"  Title="Seleccione País..."/>  </inputLayout:SfTextInputLayout>  <inputLayout:SfTextInputLayout Hint="Provincia"  ContainerType="Outlined">  <Picker ItemDisplayBinding="{Binding Name}"  ItemsSource="{Binding Departments}"  SelectedItem="{Binding Department}"  Title="Seleccione Provincia...}"/>  </inputLayout:SfTextInputLayout>  <inputLayout:SfTextInputLayout Hint="Ciudad"  ContainerType="Outlined">  <Picker ItemDisplayBinding="{Binding Name}"  ItemsSource="{Binding Cities}"  SelectedItem="{Binding City}"  Title="Seleccione Ciudad..."/>  </inputLayout:SfTextInputLayout>  <inputLayout:SfTextInputLayout Hint="Dirección"  ContainerType="Outlined">  <Entry Placeholder="Ingrese Dirección..."  Text="{Binding User.Address}" />  </inputLayout:SfTextInputLayout>  <inputLayout:SfTextInputLayout Hint="Teléfono"  ContainerType="Outlined">  <Entry Placeholder="Ingrese Teléfono..."  Keyboard="Telephone"  Text="{Binding User.Phone}" />  </inputLayout:SfTextInputLayout>  <Label Text="Equipo Favorito:"  FontAttributes="Bold">  </Label>    <inputLayout:SfTextInputLayout Hint="País"  ContainerType="Outlined">  <Picker ItemDisplayBinding="{Binding Name}"  ItemsSource="{Binding Countries2}"  SelectedItem="{Binding Country2}"  Title="Seleccione País..."/>  </inputLayout:SfTextInputLayout>  <inputLayout:SfTextInputLayout Hint="Equipo"  ContainerType="Outlined">  <Picker ItemDisplayBinding="{Binding Name}"  ItemsSource="{Binding Teams2}"  SelectedItem="{Binding Team2}"  Title="Seleccione Equipo...}"/>  </inputLayout:SfTextInputLayout>  <Label Text="Password:"  FontAttributes="Bold">  </Label>  <inputLayout:SfTextInputLayout Hint="Password"  EnablePasswordVisibilityToggle="true"  ContainerType="Outlined">  <Entry Placeholder="Ingrese Password..."  IsPassword="True"  Text="{Binding User.Password}" />  </inputLayout:SfTextInputLayout>  <inputLayout:SfTextInputLayout Hint="Confirme el Password"  EnablePasswordVisibilityToggle="true"  ContainerType="Outlined">  <Entry Placeholder="Ingrese Conf. de Password..."  IsPassword="True"  Text="{Binding User.PasswordConfirm}" />  </inputLayout:SfTextInputLayout>  </StackLayout>  </ScrollView>  <Button Command="{Binding RegisterCommand}"  IsEnabled="{Binding IsEnabled}"  Text="REGISTRARSE"  VerticalOptions="EndAndExpand"/>  </StackLayout>  <busyindicator:SfBusyIndicator AnimationType="Gear"  AbsoluteLayout.LayoutBounds=".5,.5,.5,.5"  AbsoluteLayout.LayoutFlags="All"  BackgroundColor="Transparent"  HorizontalOptions="Center"  TextColor="Blue"  IsBusy="{Binding IsRunning}"  Title="Grabando..."  VerticalOptions="Center"  ViewBoxWidth="80"  ViewBoxHeight="80" />  </AbsoluteLayout>  </ContentPage> |  |

## RegisterPageViewModel

Modificamos la **RegisterPageViewModel**

|  |  |
| --- | --- |
| **RegisterPageViewModel** | **Comentarios** |
| using GenericApp.Common.Helpers;  using GenericApp.Common.Requests;  using GenericApp.Common.Responses;  using GenericApp.Common.Services;  using GenericApp.Prism.Helpers;  using Plugin.Media;  using Plugin.Media.Abstractions;  using Prism.Commands;  using Prism.Navigation;  using System.Collections.Generic;  using System.Collections.ObjectModel;  using System.Linq;  using System.Threading.Tasks;  using Xamarin.Essentials;  using Xamarin.Forms;  namespace GenericApp.Prism.ViewModels  {  public class RegisterPageViewModel : ViewModelBase  {  private readonly INavigationService \_navigationService;  private readonly IRegexHelper \_regexHelper;  private readonly IApiService \_apiService;  private readonly IFilesHelper \_filesHelper;  private bool \_isRunning;  public bool IsRunning  {  get => \_isRunning;  set => SetProperty(ref \_isRunning, value);  }    private bool \_isEnabled;  public bool IsEnabled  {  get => \_isEnabled;  set => SetProperty(ref \_isEnabled, value);  }  private MediaFile \_file;  private ImageSource \_image;  public ImageSource Image  {  get => \_image;  set => SetProperty(ref \_image, value);  }  private UserRequest \_user;  public UserRequest User  {  get => \_user;  set => SetProperty(ref \_user, value);  }  private CountryResponse \_country;  public CountryResponse Country  {  get => \_country;  set  {  Departments = value != null ? new ObservableCollection<DepartmentResponse>(value.Departments) : null;  Cities = new ObservableCollection<CityResponse>();  Department = null;  City = null;  SetProperty(ref \_country, value);  }  }  private ObservableCollection<CountryResponse> \_countries;  public ObservableCollection<CountryResponse> Countries  {  get => \_countries;  set => SetProperty(ref \_countries, value);  }  private DepartmentResponse \_department;  public DepartmentResponse Department  {  get => \_department;  set  {  Cities = value != null ? new ObservableCollection<CityResponse>(value.Cities) : null;  City = null;  SetProperty(ref \_department, value);  }  }  private ObservableCollection<DepartmentResponse> \_departments;  public ObservableCollection<DepartmentResponse> Departments  {  get => \_departments;  set => SetProperty(ref \_departments, value);  }  private CityResponse \_city;  public CityResponse City  {  get => \_city;  set => SetProperty(ref \_city, value);  }  private ObservableCollection<CityResponse> \_cities;  public ObservableCollection<CityResponse> Cities  {  get => \_cities;  set => SetProperty(ref \_cities, value);  }  private CountryResponse \_country2;  public CountryResponse Country2  {  get => \_country2;  set  {  Teams2 = value != null ? new ObservableCollection<TeamResponse>(value.Teams) : null;  Team2 = null;  SetProperty(ref \_country2, value);  }  }  private ObservableCollection<CountryResponse> \_countries2;  public ObservableCollection<CountryResponse> Countries2  {  get => \_countries2;  set => SetProperty(ref \_countries2, value);  }  private TeamResponse \_team2;  public TeamResponse Team2  {  get => \_team2;  set  {  SetProperty(ref \_team2, value);  }  }  private ObservableCollection<TeamResponse> \_teams2;  public ObservableCollection<TeamResponse> Teams2  {  get => \_teams2;  set => SetProperty(ref \_teams2, value);  }  private DelegateCommand \_registerCommand;  public DelegateCommand RegisterCommand => \_registerCommand ?? (\_registerCommand = new DelegateCommand(RegisterAsync));  private DelegateCommand \_changeImageCommand;  public DelegateCommand ChangeImageCommand => \_changeImageCommand ?? (\_changeImageCommand = new DelegateCommand(ChangeImageAsync));  public RegisterPageViewModel(  INavigationService navigationService,  IRegexHelper regexHelper,  IApiService apiService,  IFilesHelper filesHelper)  : base(navigationService)  {  \_navigationService = navigationService;  \_regexHelper = regexHelper;  \_apiService = apiService;  \_filesHelper = filesHelper;  Title = "Registrar Nuevo Usuario";  Image = App.Current.Resources["UrlNoUser"].ToString();  IsEnabled = true;  User = new UserRequest();  LoadCountriesAsync();  }  private async void LoadCountriesAsync()  {  IsRunning = true;  IsEnabled = false;  if (Connectivity.NetworkAccess != NetworkAccess.Internet)  {  IsRunning = false;  IsEnabled = true;  await App.Current.MainPage.DisplayAlert("Error", "Error de Conexión", "Aceptar");  return;  }  string url = App.Current.Resources["UrlAPI"].ToString();  Response response = await \_apiService.GetListAsync<CountryResponse>(url, "api", "/Countries");  IsRunning = false;  IsEnabled = true;  if (!response.IsSuccess)  {  await App.Current.MainPage.DisplayAlert("Error", response.Message, "Aceptar");  return;  }  List<CountryResponse> list = (List<CountryResponse>)response.Result;  Countries = new ObservableCollection<CountryResponse>(list.OrderBy(c => c.Name));  Countries2 = new ObservableCollection<CountryResponse>(list.OrderBy(c => c.Name));  }  private async void RegisterAsync()  {  bool isValid = await ValidateDataAsync();  if (!isValid)  {  return;  }  IsRunning = true;  IsEnabled = false;  if (Connectivity.NetworkAccess != NetworkAccess.Internet)  {  IsRunning = false;  IsEnabled = true;  await App.Current.MainPage.DisplayAlert("Error", "ConnectionError", "Aceptar");  return;  }  byte[] imageArray = null;  if (\_file != null)  {  imageArray = \_filesHelper.ReadFully(\_file.GetStream());  }  User.PictureArray = imageArray;  string url = App.Current.Resources["UrlAPI"].ToString();  User.CityId = City.Id;  User.FavoriteTeamId = Team2.Id;  Response response = await \_apiService.RegisterUserAsync(url, "api", "/Account/Register", User);  IsRunning = false;  IsEnabled = true;  if (!response.IsSuccess)  {  if (response.Message == "Error003")  {  await App.Current.MainPage.DisplayAlert("Error", "Este usuario ya existe", "Aceptar");  }  else if (response.Message == "Error004")  {  await App.Current.MainPage.DisplayAlert("Error", "La ciudad no es válida", "Aceptar");  }  else  {  await App.Current.MainPage.DisplayAlert("Error", response.Message, "Aceptar");  }  return;  }  await App.Current.MainPage.DisplayAlert("Ok", "El Registro fue correcto. Se le ha enviado un mail para confirmar el mismo.", "Aceptar");  await \_navigationService.GoBackAsync();  }  private async Task<bool> ValidateDataAsync()  {  if (string.IsNullOrEmpty(User.Email) || !\_regexHelper.IsValidEmail(User.Email))  {  await App.Current.MainPage.DisplayAlert("Error", "Ingrese un EMail", "Aceptar");  return false;  }  if (string.IsNullOrEmpty(User.Document))  {  await App.Current.MainPage.DisplayAlert("Error", "Ingrese Documento", "Aceptar");  return false;  }  if (string.IsNullOrEmpty(User.FirstName))  {  await App.Current.MainPage.DisplayAlert("Error", "Ingrese un Nombre", "Aceptar");  return false;  }  if (string.IsNullOrEmpty(User.LastName))  {  await App.Current.MainPage.DisplayAlert("Error", "Ingrese un Apellido", "Aceptar");  return false;  }  if (Country == null)  {  await App.Current.MainPage.DisplayAlert("Error", "Seleccione un País", "Aceptar");  return false;  }  if (Department == null)  {  await App.Current.MainPage.DisplayAlert("Error", "Seleccione una Provincia", "Aceptar");  return false;  }  if (City == null)  {  await App.Current.MainPage.DisplayAlert("Error", "Seleccione una Ciudad", "Aceptar");  return false;  }  if (string.IsNullOrEmpty(User.Address))  {  await App.Current.MainPage.DisplayAlert("Error", "Ingrese una Dirección", "Aceptar");  return false;  }  if (string.IsNullOrEmpty(User.Phone))  {  await App.Current.MainPage.DisplayAlert("Error", "Ingrese un Teléfono", "Aceptar");  return false;  }  if (Country2 == null)  {  await App.Current.MainPage.DisplayAlert("Error", "Seleccione un País", "Aceptar");  return false;  }  if (Team2 == null)  {  await App.Current.MainPage.DisplayAlert("Error", "Seleccione un Equipo", "Aceptar");  return false;  }  if (string.IsNullOrEmpty(User.Password) || User.Password?.Length < 6)  {  await App.Current.MainPage.DisplayAlert("Error", "Ingrese un Password", "Aceptar");  return false;  }  if (string.IsNullOrEmpty(User.PasswordConfirm))  {  await App.Current.MainPage.DisplayAlert("Error", "Ingrese una Conf. de Password", "Aceptar");  return false;  }  if (User.Password != User.PasswordConfirm)  {  await App.Current.MainPage.DisplayAlert("Error", "Password y su Confirmación deben ser iguales", "Aceptar");  return false;  }  return true;  }  private async void ChangeImageAsync()  {  await CrossMedia.Current.Initialize();  string source = await Application.Current.MainPage.DisplayActionSheet(  "De donde quiere tomar la foto?",  "Cancelar",  null,  "Galería",  "Cámara");  if (source == "Cancelar")  {  \_file = null;  return;  }  if (source == "Cámara")  {  if (!CrossMedia.Current.IsCameraAvailable)  {  await App.Current.MainPage.DisplayAlert("Error", "La cámara no está disponible", "Aceptar");  return;  }  \_file = await CrossMedia.Current.TakePhotoAsync(  new StoreCameraMediaOptions  {  Directory = "Sample",  Name = "test.jpg",  PhotoSize = PhotoSize.Small,  }  );  }  else  {  if (!CrossMedia.Current.IsPickPhotoSupported)  {  await App.Current.MainPage.DisplayAlert("Error", "La Galería no está disponible", "Aceptar");  return;  }  \_file = await CrossMedia.Current.PickPhotoAsync();  }  if (\_file != null)  {  Image = ImageSource.FromStream(() =>  {  System.IO.Stream stream = \_file.GetStream();  return stream;  });  }  }  }  } |  |

# Cambiar Password

## ChangePasswordPage

Dentro de la carpeta **GenericApp.Prism/Views** creamos la **ChangePasswordPage**

|  |  |
| --- | --- |
| **ChangePasswordPage** | **Comentarios** |
| <?xml version="1.0" encoding="utf-8" ?>  <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  xmlns:prism="http://prismlibrary.com"  prism:ViewModelLocator.AutowireViewModel="True"  xmlns:ios="clr-namespace:Xamarin.Forms.PlatformConfiguration.iOSSpecific;assembly=Xamarin.Forms.Core"  ios:Page.UseSafeArea="true"  xmlns:busyindicator="clr-namespace:Syncfusion.SfBusyIndicator.XForms;assembly=Syncfusion.SfBusyIndicator.XForms"  xmlns:inputLayout="clr-namespace:Syncfusion.XForms.TextInputLayout;assembly=Syncfusion.Core.XForms"  x:Class="GenericApp.Prism.Views.ChangePasswordPage"  BackgroundColor="{StaticResource ColorBackground}"  Title="{Binding Title}">  <AbsoluteLayout>  <StackLayout AbsoluteLayout.LayoutBounds="0,0,1,1"  AbsoluteLayout.LayoutFlags="All"  Padding="5">  <ScrollView>  <StackLayout Spacing="0">  <Image HeightRequest="150"  Margin="20"  Source="resetpassword"/>  <inputLayout:SfTextInputLayout Hint="Password actual"  EnablePasswordVisibilityToggle="true"  ContainerType="Outlined">  <Entry Placeholder="Password actual..."  IsPassword="True"  Text="{Binding CurrentPassword}" />  </inputLayout:SfTextInputLayout>  <inputLayout:SfTextInputLayout Hint="Nuevo Password"  EnablePasswordVisibilityToggle="true"  ContainerType="Outlined">  <Entry Placeholder="Nuevo Password..."  IsPassword="True"  Text="{Binding NewPassword}" />  </inputLayout:SfTextInputLayout>  <inputLayout:SfTextInputLayout Hint="Confirme Nuevo Password"  EnablePasswordVisibilityToggle="true"  ContainerType="Outlined">  <Entry Placeholder="Confirme Nuevo Password..."  IsPassword="True"  Text="{Binding PasswordConfirm}" />  </inputLayout:SfTextInputLayout>  </StackLayout>  </ScrollView>  <Button Command="{Binding ChangePasswordCommand}"  IsEnabled="{Binding IsEnabled}"  Text="Cambiar Password"  VerticalOptions="EndAndExpand"/>  </StackLayout>  <busyindicator:SfBusyIndicator AnimationType="Gear"  AbsoluteLayout.LayoutBounds=".5,.5,.5,.5"  AbsoluteLayout.LayoutFlags="All"  BackgroundColor="Transparent"  HorizontalOptions="Center"  TextColor="Blue"  IsBusy="{Binding IsRunning}"  Title="Grabando..."  VerticalOptions="Center"  ViewBoxWidth="80"  ViewBoxHeight="80" />  </AbsoluteLayout>  </ContentPage> |  |

## ChangePasswordPageViewModel

Modificamos la **ChangePasswordPageViewModel**

|  |  |
| --- | --- |
| **ChangePasswordPageViewModel** | **Comentarios** |
| using GenericApp.Common.Helpers;  using GenericApp.Common.Requests;  using GenericApp.Common.Responses;  using GenericApp.Common.Services;  using Newtonsoft.Json;  using Prism.Commands;  using Prism.Navigation;  using System.Threading.Tasks;  using Xamarin.Essentials;  namespace GenericApp.Prism.ViewModels  {  public class ChangePasswordPageViewModel : ViewModelBase  {  private readonly INavigationService \_navigationService;  private readonly IApiService \_apiService;  private bool \_isRunning;  private bool \_isEnabled;  private DelegateCommand \_changePasswordCommand;  public ChangePasswordPageViewModel(INavigationService navigationService, IApiService apiService)  : base(navigationService)  {  \_navigationService = navigationService;  \_apiService = apiService;  IsEnabled = true;  Title = "Cambiar Password";  }  public DelegateCommand ChangePasswordCommand => \_changePasswordCommand ?? (\_changePasswordCommand = new DelegateCommand(ChangePasswordAsync));  public string CurrentPassword { get; set; }  public string NewPassword { get; set; }  public string PasswordConfirm { get; set; }  public bool IsRunning  {  get => \_isRunning;  set => SetProperty(ref \_isRunning, value);  }  public bool IsEnabled  {  get => \_isEnabled;  set => SetProperty(ref \_isEnabled, value);  }  private async void ChangePasswordAsync()  {  var isValid = await ValidateDataAsync();  if (!isValid)  {  return;  }  IsRunning = true;  IsEnabled = false;  if (Connectivity.NetworkAccess != NetworkAccess.Internet)  {  IsRunning = false;  IsEnabled = true;  await App.Current.MainPage.DisplayAlert("Error", "Error de conexión", "Aceptar");  return;  }  ChangePasswordRequest request = new ChangePasswordRequest  {  NewPassword = NewPassword,  OldPassword = CurrentPassword,  };  TokenResponse token = JsonConvert.DeserializeObject<TokenResponse>(Settings.Token);  string url = App.Current.Resources["UrlAPI"].ToString();  Response response = await \_apiService.ChangePasswordAsync(url, "api", "/Account/ChangePassword", request, token.Token);  IsRunning = false;  IsEnabled = true;  if (!response.IsSuccess)  {  if (response.Message == "Error001")  {  await App.Current.MainPage.DisplayAlert("Error", "El usuario no existe", "Aceptar");  }  else if (response.Message == "Error005")  {  await App.Current.MainPage.DisplayAlert("Error", "El password actual es incorrecto", "Aceptar");  }  else  {  await App.Current.MainPage.DisplayAlert("Error", response.Message, "Aceptar");  }  return;  }  await App.Current.MainPage.DisplayAlert("Ok", "El Password fue cambiado con éxito!!", "Aceptar");  await \_navigationService.GoBackAsync();  }  private async Task<bool> ValidateDataAsync()  {  if (string.IsNullOrEmpty(CurrentPassword))  {  await App.Current.MainPage.DisplayAlert("Error", "Ingrese Password actual", "Aceptar");  return false;  }  if (string.IsNullOrEmpty(NewPassword) || NewPassword?.Length < 6)  {  await App.Current.MainPage.DisplayAlert("Error", "Ingrese Nuevo Password", "Aceptar");  return false;  }  if (string.IsNullOrEmpty(PasswordConfirm))  {  await App.Current.MainPage.DisplayAlert("Error", "Ingrese Confirmación de Password", "Aceptar");  return false;  }  if (NewPassword != PasswordConfirm)  {  await App.Current.MainPage.DisplayAlert("Error", "El Nuevo Password y su Confirmación no son iguales", "Aceptar");  return false;  }  return true;  }  }  } |  |

# Modificar Usuario

## ModifyUserPage

Dentro de la carpeta **GenericApp.Prism/Views** creamos la **ModifyUserPage**

|  |  |
| --- | --- |
| **ModifyUserPage** | **Comentarios** |
| <?xml version="1.0" encoding="utf-8" ?>  <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  xmlns:prism="http://prismlibrary.com"  prism:ViewModelLocator.AutowireViewModel="True"  xmlns:ios="clr-namespace:Xamarin.Forms.PlatformConfiguration.iOSSpecific;assembly=Xamarin.Forms.Core"  ios:Page.UseSafeArea="true"  xmlns:ffimageloading="clr-namespace:FFImageLoading.Forms;assembly=FFImageLoading.Forms"  xmlns:fftransformations="clr-namespace:FFImageLoading.Transformations;assembly=FFImageLoading.Transformations"  xmlns:inputLayout="clr-namespace:Syncfusion.XForms.TextInputLayout;assembly=Syncfusion.Core.XForms"  xmlns:busyindicator="clr-namespace:Syncfusion.SfBusyIndicator.XForms;assembly=Syncfusion.SfBusyIndicator.XForms"  x:Class="GenericApp.Prism.Views.ModifyUserPage"  BackgroundColor="{StaticResource ColorBackground}"  Title="{Binding Title}">  <AbsoluteLayout>  <StackLayout AbsoluteLayout.LayoutBounds="0,0,1,1"  AbsoluteLayout.LayoutFlags="All"  Padding="5">  <ScrollView>  <StackLayout Spacing="0">  <ffimageloading:CachedImage Aspect="AspectFit"  Source="{Binding Image}"  CacheDuration= "50"  HeightRequest="150"  Margin="5"  RetryCount= "3"  RetryDelay= "600">  <ffimageloading:CachedImage.Transformations>  <fftransformations:CircleTransformation />  </ffimageloading:CachedImage.Transformations>  <ffimageloading:CachedImage.GestureRecognizers>  <TapGestureRecognizer Command="{Binding ChangeImageCommand}"/>  </ffimageloading:CachedImage.GestureRecognizers>  </ffimageloading:CachedImage>  <Label Text="Datos personales:"  FontAttributes="Bold">  </Label>  <inputLayout:SfTextInputLayout Hint="Documento"  ContainerType="Outlined">  <Entry Placeholder="Ingrese Documento..."  Text="{Binding User.Document}" />  </inputLayout:SfTextInputLayout>  <inputLayout:SfTextInputLayout Hint="Nombre"  ContainerType="Outlined">  <Entry Placeholder="Ingrese Nombre..."  Text="{Binding User.FirstName}" />  </inputLayout:SfTextInputLayout>  <inputLayout:SfTextInputLayout Hint="Apellido"  ContainerType="Outlined">  <Entry Placeholder="Ingrese Apellido..."  Text="{Binding User.LastName}" />  </inputLayout:SfTextInputLayout>  <inputLayout:SfTextInputLayout Hint="País"  ContainerType="Outlined">  <Picker ItemDisplayBinding="{Binding Name}"  ItemsSource="{Binding Countries}"  SelectedItem="{Binding Country}"  Title="Seleccione País..."/>  </inputLayout:SfTextInputLayout>  <inputLayout:SfTextInputLayout Hint="Provincia"  ContainerType="Outlined">  <Picker ItemDisplayBinding="{Binding Name}"  ItemsSource="{Binding Departments}"  SelectedItem="{Binding Department}"  Title="Seleccione Provincia...}"/>  </inputLayout:SfTextInputLayout>  <inputLayout:SfTextInputLayout Hint="Ciudad"  ContainerType="Outlined">  <Picker ItemDisplayBinding="{Binding Name}"  ItemsSource="{Binding Cities}"  SelectedItem="{Binding City}"  Title="Seleccione Ciudad..."/>  </inputLayout:SfTextInputLayout>  <inputLayout:SfTextInputLayout Hint="Dirección"  ContainerType="Outlined">  <Entry Placeholder="Ingrese Dirección..."  Text="{Binding User.Address}" />  </inputLayout:SfTextInputLayout>  <inputLayout:SfTextInputLayout Hint="Teléfono"  ContainerType="Outlined">  <Entry Placeholder="Ingrese Teléfono..."  Keyboard="Telephone"  Text="{Binding User.PhoneNumber}" />  </inputLayout:SfTextInputLayout>  <Label Text="Equipo Favorito:"  FontAttributes="Bold">  </Label>  <inputLayout:SfTextInputLayout Hint="País"  ContainerType="Outlined">  <Picker ItemDisplayBinding="{Binding Name}"  ItemsSource="{Binding Countries2}"  SelectedItem="{Binding Country2}"  Title="Seleccione País..."/>  </inputLayout:SfTextInputLayout>  <inputLayout:SfTextInputLayout Hint="Equipo"  ContainerType="Outlined">  <Picker ItemDisplayBinding="{Binding Name}"  ItemsSource="{Binding Teams2}"  SelectedItem="{Binding Team2}"  Title="Seleccione Equipo...}"/>  </inputLayout:SfTextInputLayout>  </StackLayout>  </ScrollView>  <StackLayout Orientation="Horizontal"  VerticalOptions="EndAndExpand">  <Button Command="{Binding SaveCommand}"  IsEnabled="{Binding IsEnabled}"  Text="Grabar"/>  <Button Command="{Binding ChangePasswordCommand}"  IsEnabled="{Binding IsEnabled}"  Style="{StaticResource SecondaryButton}"  Text="Cambiar Password"/>  </StackLayout>  </StackLayout>  <busyindicator:SfBusyIndicator AnimationType="Gear"  AbsoluteLayout.LayoutBounds=".5,.5,.5,.5"  AbsoluteLayout.LayoutFlags="All"  BackgroundColor="Transparent"  HorizontalOptions="Center"  TextColor="Blue"  IsBusy="{Binding IsRunning}"  Title="Grabar..."  VerticalOptions="Center"  ViewBoxWidth="80"  ViewBoxHeight="80" />  </AbsoluteLayout>  </ContentPage> |  |

## ModifyUserPageViewModel

Modificamos la **ModifyUserPageViewModel**

|  |  |
| --- | --- |
| **ModifyUserPageViewModel** | **Comentarios** |
| using Newtonsoft.Json;  using GenericApp.Common.Helpers;  using GenericApp.Common.Requests;  using GenericApp.Common.Responses;  using GenericApp.Common.Services;  using Plugin.Media;  using Plugin.Media.Abstractions;  using Prism.Commands;  using Prism.Navigation;  using System.Collections.Generic;  using System.Collections.ObjectModel;  using System.Linq;  using System.Threading.Tasks;  using Xamarin.Essentials;  using Xamarin.Forms;  using ImageSource = Xamarin.Forms.ImageSource;  using GenericApp.Prism.Views;  namespace GenericApp.Prism.ViewModels  {  public class ModifyUserPageViewModel : ViewModelBase  {  private readonly INavigationService \_navigationService;  private readonly IApiService \_apiService;  private readonly IFilesHelper \_filesHelper;  private bool \_isRunning;  public bool IsRunning  {  get => \_isRunning;  set => SetProperty(ref \_isRunning, value);  }  private bool \_isEnabled;  public bool IsEnabled  {  get => \_isEnabled;  set => SetProperty(ref \_isEnabled, value);  }  private MediaFile \_file;  private ImageSource \_image;  public ImageSource Image  {  get => \_image;  set => SetProperty(ref \_image, value);  }  private UserResponse \_user;  public UserResponse User  {  get => \_user;  set => SetProperty(ref \_user, value);  }  private CountryResponse \_country;  public CountryResponse Country  {  get => \_country;  set  {  Departments = value != null ? new ObservableCollection<DepartmentResponse>(value.Departments) : null;  Cities = new ObservableCollection<CityResponse>();  Department = null;  City = null;  SetProperty(ref \_country, value);  }  }  private ObservableCollection<CountryResponse> \_countries;  public ObservableCollection<CountryResponse> Countries  {  get => \_countries;  set => SetProperty(ref \_countries, value);  }  private DepartmentResponse \_department;  public DepartmentResponse Department  {  get => \_department;  set  {  Cities = value != null ? new ObservableCollection<CityResponse>(value.Cities) : null;  City = null;  SetProperty(ref \_department, value);  }  }  private ObservableCollection<DepartmentResponse> \_departments;  public ObservableCollection<DepartmentResponse> Departments  {  get => \_departments;  set => SetProperty(ref \_departments, value);  }  private CityResponse \_city;  public CityResponse City  {  get => \_city;  set => SetProperty(ref \_city, value);  }  private ObservableCollection<CityResponse> \_cities;  public ObservableCollection<CityResponse> Cities  {  get => \_cities;  set => SetProperty(ref \_cities, value);  }  private CountryResponse \_country2;  public CountryResponse Country2  {  get => \_country2;  set  {  Teams2 = value != null ? new ObservableCollection<TeamResponse>(value.Teams) : null;  Team2 = null;  SetProperty(ref \_country2, value);  }  }  private ObservableCollection<CountryResponse> \_countries2;  public ObservableCollection<CountryResponse> Countries2  {  get => \_countries2;  set => SetProperty(ref \_countries2, value);  }  private TeamResponse \_team2;  public TeamResponse Team2  {  get => \_team2;  set  {  SetProperty(ref \_team2, value);  }  }  private ObservableCollection<TeamResponse> \_teams2;  public ObservableCollection<TeamResponse> Teams2  {  get => \_teams2;  set => SetProperty(ref \_teams2, value);  }    private DelegateCommand \_changeImageCommand;  public DelegateCommand ChangeImageCommand => \_changeImageCommand ??  (\_changeImageCommand = new DelegateCommand(ChangeImageAsync));  private DelegateCommand \_saveCommand;  public DelegateCommand SaveCommand => \_saveCommand ??  (\_saveCommand = new DelegateCommand(SaveAsync));  private DelegateCommand \_changePasswordCommand;  public DelegateCommand ChangePasswordCommand => \_changePasswordCommand ??  (\_changePasswordCommand = new DelegateCommand(ChangePasswordAsync));  public ModifyUserPageViewModel(  INavigationService navigationService,  IApiService apiService,  IFilesHelper filesHelper)  : base(navigationService)  {  \_navigationService = navigationService;  \_apiService = apiService;  \_filesHelper = filesHelper;  Title = "Modificar Usuario";  IsEnabled = true;  TokenResponse token = JsonConvert.DeserializeObject<TokenResponse>(Settings.Token);  User = token.User;  Image = User.PictureFullPath;  LoadCountriesAsync();  }    public async void LoadCountriesAsync()  {  IsRunning = true;  IsEnabled = false;  if (Connectivity.NetworkAccess != NetworkAccess.Internet)  {  IsRunning = false;  IsEnabled = true;  await App.Current.MainPage.DisplayAlert(  "Error",  "Connection Error",  "Accept");  return;  }  string url = App.Current.Resources["UrlAPI"].ToString();  Response response = await \_apiService.GetListAsync<CountryResponse>(url, "api", "/Countries");  IsRunning = false;  IsEnabled = true;  if (!response.IsSuccess)  {  await App.Current.MainPage.DisplayAlert("Error", response.Message, "Aceptar");  return;  }  List<CountryResponse> list = (List<CountryResponse>)response.Result;  Countries = new ObservableCollection<CountryResponse>(list.OrderBy(c => c.Name));  Countries2 = new ObservableCollection<CountryResponse>(list.OrderBy(c => c.Name));  LoadCurrentCountryDepartmentCity();  LoadCurrentCountryTeam();  }  private void LoadCurrentCountryDepartmentCity()  {  Country = Countries.FirstOrDefault(c => c.Departments.FirstOrDefault(d => d.Cities.FirstOrDefault(ci => ci.Id == User.City.Id) != null) != null);  Department = Country.Departments.FirstOrDefault(d => d.Cities.FirstOrDefault(c => c.Id == User.City.Id) != null);  City = Department.Cities.FirstOrDefault(c => c.Id == User.City.Id);  }  private void LoadCurrentCountryTeam()  {  Country2 = Countries.FirstOrDefault(c => c.Teams.FirstOrDefault(ci => ci.Id == User.FavoriteTeam.Id) != null);  TokenResponse token = JsonConvert.DeserializeObject<TokenResponse>(Settings.Token);  User = token.User;  Team2 = Country2.Teams.FirstOrDefault(c => c.Id == User.FavoriteTeam.Id);  }  private async void ChangeImageAsync()  {  await CrossMedia.Current.Initialize();  string source = await Application.Current.MainPage.DisplayActionSheet(  "De donde quiere tomar la foto?",  "Cancelar",  null,  "Galería",  "Cámara");  if (source == "Cancelar")  {  \_file = null;  return;  }  if (source == "Cámara")  {  if (!CrossMedia.Current.IsCameraAvailable)  {  await App.Current.MainPage.DisplayAlert("Error", "La cámara no está disponible", "Aceptar");  return;  }  \_file = await CrossMedia.Current.TakePhotoAsync(  new StoreCameraMediaOptions  {  Directory = "Sample",  Name = "test.jpg",  PhotoSize = PhotoSize.Small,  }  );  }  else  {  if (!CrossMedia.Current.IsPickPhotoSupported)  {  await App.Current.MainPage.DisplayAlert("Error", "La Galería no está disponible", "Aceptar");  return;  }  \_file = await CrossMedia.Current.PickPhotoAsync();  }  if (\_file != null)  {  Image = Xamarin.Forms.ImageSource.FromStream(() =>  {  System.IO.Stream stream = \_file.GetStream();  return stream;  });  }  }  private async void SaveAsync()  {  bool isValid = await ValidateDataAsync();  if (!isValid)  {  return;  }  IsRunning = true;  IsEnabled = false;  if (Connectivity.NetworkAccess != NetworkAccess.Internet)  {  IsRunning = false;  IsEnabled = true;  await App.Current.MainPage.DisplayAlert("Error", "Connection Error", "Accept");  return;  }  byte[] imageArray = null;  if (\_file != null)  {  imageArray = \_filesHelper.ReadFully(\_file.GetStream());  }  UserRequest request = new UserRequest  {  Address = User.Address,  CityId = City.Id,  FavoriteTeamId = Team2.Id,  Document = User.Document,  Email = User.Email,  FirstName = User.FirstName,  PictureArray = imageArray,  LastName = User.LastName,  Password = "123456", // Doen't matter, it's only to pass the data annotation  Phone = User.PhoneNumber,  //FavoriteTeamId  };  TokenResponse token = JsonConvert.DeserializeObject<TokenResponse>(Settings.Token);  string url = App.Current.Resources["UrlAPI"].ToString();  Response response = await \_apiService.ModifyUserAsync(url, "api", "/Account", request, token.Token);  IsRunning = false;  IsEnabled = true;  if (!response.IsSuccess)  {  if (response.Message == "Error001")  {  await App.Current.MainPage.DisplayAlert("Error", "El Usuario no existe", "Aceptar");  }  else if (response.Message == "Error004")  {  await App.Current.MainPage.DisplayAlert("Error", "La ciudad no existe", "Aceptar");  }  else  {  await App.Current.MainPage.DisplayAlert("Error", response.Message, "Aceptar");  }  return;  }  UserResponse updatedUser = (UserResponse)response.Result;  token.User = updatedUser;  Settings.Token = JsonConvert.SerializeObject(token);  GenericAppMasterDetailPageViewModel.GetInstance().LoadUser();  await App.Current.MainPage.DisplayAlert("Ok", "El Usuario fue actualizado con éxito.", "Aceptar");  }  private async Task<bool> ValidateDataAsync()  {  if (string.IsNullOrEmpty(User.Document))  {  await App.Current.MainPage.DisplayAlert("Error", "Ingrese un Documento", "Aceptar");  return false;  }  if (string.IsNullOrEmpty(User.FirstName))  {  await App.Current.MainPage.DisplayAlert("Error", "Ingrese un Nombre", "Aceptar");  return false;  }  if (string.IsNullOrEmpty(User.LastName))  {  await App.Current.MainPage.DisplayAlert("Error", "Ingrese un Apellido", "Aceptar");  return false;  }  if (Country == null)  {  await App.Current.MainPage.DisplayAlert("Error", "Seleccione un país", "Accept");  return false;  }  if (Department == null)  {  await App.Current.MainPage.DisplayAlert("Error", "Seleccione un departamento", "Aceptar");  return false;  }  if (City == null)  {  await App.Current.MainPage.DisplayAlert("Error", "Seleccione una ciudad", "Aceptar");  return false;  }  if (string.IsNullOrEmpty(User.Address))  {  await App.Current.MainPage.DisplayAlert("Error", "Ingrese una Dirección", "Aceptar");  return false;  }  if (string.IsNullOrEmpty(User.PhoneNumber))  {  await App.Current.MainPage.DisplayAlert("Error", "Ingrese un Teléfono", "Aceptar");  return false;  }  if (Country2 == null)  {  await App.Current.MainPage.DisplayAlert("Error", "Seleccione un País", "Aceptar");  return false;  }  if (Team2 == null)  {  await App.Current.MainPage.DisplayAlert("Error", "Seleccione un Equipo", "Aceptar");  return false;  }  return true;  }  private async void ChangePasswordAsync()  {  await \_navigationService.NavigateAsync(nameof(ChangePasswordPage));  }  }  } |  |

# Productos

## ProductsPage

Dentro de la carpeta **GenericApp.Prism/Views** creamos la **ProductsPage**

|  |  |
| --- | --- |
| **ProductsPage** | **Comentarios** |
| <?xml version="1.0" encoding="utf-8" ?>  <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  xmlns:prism="http://prismlibrary.com"  prism:ViewModelLocator.AutowireViewModel="True"  xmlns:ffimageloading="clr-namespace:FFImageLoading.Forms;assembly=FFImageLoading.Forms"  xmlns:ios="clr-namespace:Xamarin.Forms.PlatformConfiguration.iOSSpecific;assembly=Xamarin.Forms.Core"  ios:Page.UseSafeArea="true"  xmlns:busyindicator="clr-namespace:Syncfusion.SfBusyIndicator.XForms;assembly=Syncfusion.SfBusyIndicator.XForms"  x:Class="GenericApp.Prism.Views.ProductsPage"  BackgroundColor="{StaticResource ColorBackground}"  Title="{Binding Title}">  <ContentPage.ToolbarItems>  <ToolbarItem Icon="ic\_add\_circle\_outline"  Command="{Binding AddProductCommand}" />  <ToolbarItem Icon="ic\_map"  Command="{Binding ProductsMapCommand}" />  </ContentPage.ToolbarItems>  <AbsoluteLayout>  <StackLayout AbsoluteLayout.LayoutBounds="0,0,1,1"  AbsoluteLayout.LayoutFlags="All"  Padding="5">  <SearchBar Placeholder="Buscar Producto..."  SearchCommand="{Binding SearchCommand}"  Text="{Binding Search}"/>  <CollectionView ItemsSource="{Binding Products}">  <CollectionView.ItemsLayout>  <GridItemsLayout Orientation="Vertical"/>  </CollectionView.ItemsLayout>  <CollectionView.ItemTemplate>  <DataTemplate>  <Grid>  <Grid.GestureRecognizers>  <TapGestureRecognizer Command="{Binding SelectProductCommand}"/>  </Grid.GestureRecognizers>  <Grid.ColumnDefinitions>  <ColumnDefinition Width="Auto" />  <ColumnDefinition Width="\*" />  <ColumnDefinition Width="Auto" />  </Grid.ColumnDefinitions>  <ffimageloading:CachedImage Grid.Column="0"  Aspect="AspectFill"  Source="{Binding ImageFullPath}"  CacheDuration= "50"  Margin="5"  RetryCount= "3"  RetryDelay= "600"  WidthRequest="100"/>  <StackLayout Grid.Column="1"  VerticalOptions="Center">  <Label Text="{Binding Name}"  FontAttributes="Bold"  FontSize="Medium"  LineBreakMode="TailTruncation" />    <Label Text="{Binding Price, StringFormat='{0:C2}'}"  LineBreakMode="TailTruncation"  FontAttributes="Italic"  VerticalOptions="End" />  <StackLayout Orientation="Horizontal">  <Label  Text="Lat:"  FontAttributes="Italic"  FontSize="Micro">  </Label>  <Label Text="{Binding Latitude}"  FontAttributes="Italic"  FontSize="Micro"  VerticalOptions="End" />  <Label  Text="Long:"  FontAttributes="Italic"  FontSize="Micro">  </Label>  <Label Text="{Binding Longitude}"  FontAttributes="Italic"  FontSize="Micro"  VerticalOptions="End" />  </StackLayout>  <Label Text="{Binding State.Name}"  FontAttributes="Italic"  VerticalOptions="End" />  </StackLayout>  <Image Grid.Column="2"  Source="ic\_more\_vert"/>  </Grid>  </DataTemplate>  </CollectionView.ItemTemplate>  </CollectionView>  </StackLayout>  <busyindicator:SfBusyIndicator AnimationType="Gear"  AbsoluteLayout.LayoutBounds=".5,.5,.5,.5"  AbsoluteLayout.LayoutFlags="All"  BackgroundColor="Transparent"  HorizontalOptions="Center"  TextColor="Blue"  IsBusy="{Binding IsRunning}"  Title="Cargando..."  VerticalOptions="Center"  ViewBoxWidth="80"  ViewBoxHeight="80" />  </AbsoluteLayout>  </ContentPage> |  |

## ProductsPageViewModel

Modificamos la **ProductsPageViewModel**

|  |  |
| --- | --- |
| **ProductsPageViewModel** | **Comentarios** |
| using GenericApp.Common.Responses;  using GenericApp.Common.Services;  using Prism.Commands;  using Prism.Navigation;  using System.Collections.Generic;  using System.Collections.ObjectModel;  using System.Linq;  using Xamarin.Essentials;  using GenericApp.Prism.ItemViewModels;  namespace GenericApp.Prism.ViewModels  {  public class ProductsPageViewModel : ViewModelBase  {  private readonly INavigationService \_navigationService;  private readonly IApiService \_apiService;    private bool \_isRunning;  public bool IsRunning  {  get => \_isRunning;  set => SetProperty(ref \_isRunning, value);  }  private bool \_isRefreshing;  public bool IsRefreshing { get => \_isRefreshing; set => SetProperty(ref \_isRefreshing, value); }  private List<ProductResponse> \_myProducts;    private ObservableCollection<ProductItemViewModel> \_products;  public ObservableCollection<ProductItemViewModel> Products  {  get => \_products;  set => SetProperty(ref \_products, value);  }  private string \_search;  public string Search  {  get => \_search;  set  {  SetProperty(ref \_search, value);  RefreshList();  }  }    private DelegateCommand \_addProductCommand;  public DelegateCommand AddProductCommand => \_addProductCommand ?? (\_addProductCommand = new DelegateCommand(AddProduct));  private DelegateCommand \_searchCommand;  public DelegateCommand SearchCommand => \_searchCommand ?? (\_searchCommand = new DelegateCommand(RefreshList));  private DelegateCommand \_productsMapCommand;  public DelegateCommand ProductsMapCommand => \_productsMapCommand ?? (\_productsMapCommand = new DelegateCommand(ProductsMap));  private static ProductsPageViewModel \_instance;  public static ProductsPageViewModel GetInstance()  {  return \_instance;  }  public ProductsPageViewModel(INavigationService navigationService, IApiService apiService) : base(navigationService)  {  \_navigationService = navigationService;  \_apiService = apiService;  \_instance = this;  Title = "Productos";  LoadProductsAsync();  }    public async void LoadProductsAsync()  {  if (Connectivity.NetworkAccess != NetworkAccess.Internet)  {  await App.Current.MainPage.DisplayAlert("Error", "Error de conexión.", "Aceptar");  return;  }  IsRunning = true;  string url = App.Current.Resources["UrlAPI"].ToString();  Response response = await \_apiService.GetListAsync<ProductResponse>(  url,  "api",  "/Products");  IsRunning = false;  if (!response.IsSuccess)  {  await App.Current.MainPage.DisplayAlert(  "Error",  response.Message,  "Aceptar");  return;  }  \_myProducts = (List<ProductResponse>)response.Result;  RefreshList();  }  public void RefreshList()  {  IsRefreshing = true;  if (string.IsNullOrEmpty(Search))  {  Products = new ObservableCollection<ProductItemViewModel>(\_myProducts.Select(p => new ProductItemViewModel(\_navigationService)  {  Category = p.Category,  Description = p.Description,  Id = p.Id,  IsActive = p.IsActive,  Latitude=p.Latitude,  Longitude=p.Longitude,  Name = p.Name,  Price = p.Price,  ProductImages = p.ProductImages,  State=p.State  })  .ToList());  }  else  {  Products = new ObservableCollection<ProductItemViewModel>(\_myProducts.Select(p => new ProductItemViewModel(\_navigationService)  {  Category = p.Category,  Description = p.Description,  Id = p.Id,  IsActive = p.IsActive,  Latitude = p.Latitude,  Longitude = p.Longitude,  Name = p.Name,  Price = p.Price,  ProductImages = p.ProductImages,  State = p.State  })  .Where(p =>  p.Name.ToLower().Contains(Search.ToLower())  ||  p.State.Name.ToLower().Contains(Search.ToLower())  )  .ToList());  }  }  private async void ProductsMap()  {  await \_navigationService.NavigateAsync("ProductsMapPage");  }  public async void CerrarMapa()  {  await \_navigationService.GoBackAsync();  }  private async void AddProduct()  {  await \_navigationService.NavigateAsync("AddProductPage");  }  }  } |  |

# MasterDetailPage

## GenericAppMasterDetailPage

Dentro de la carpeta **GenericApp.Prism /Views** creamos la **GenericAppMasterDetailPage**

|  |  |
| --- | --- |
| **GenericAppMasterDetailPage** | **Comentarios** |
| <?xml version="1.0" encoding="utf-8" ?>  <MasterDetailPage xmlns="http://xamarin.com/schemas/2014/forms"  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  xmlns:prism="http://prismlibrary.com"  xmlns:ffimageloading="clr-namespace:FFImageLoading.Forms;assembly=FFImageLoading.Forms"  xmlns:fftransformations="clr-namespace:FFImageLoading.Transformations;assembly=FFImageLoading.Transformations"  prism:ViewModelLocator.AutowireViewModel="True"    x:Class="GenericApp.Prism.Views.GenericAppMasterDetailPage">  <MasterDetailPage.Master>  <ContentPage BackgroundColor="{StaticResource ColorSecondary}"  IconImageSource="ic\_action\_menu"  Title="Menu">  <ContentPage.Padding>  <OnPlatform x:TypeArguments="Thickness">  <On Platform="Android, UWP">0</On>  <On Platform="iOS">0,20,0,0</On>  </OnPlatform>  </ContentPage.Padding>  <StackLayout Padding="20">  <RelativeLayout>  <RelativeLayout.GestureRecognizers>  <TapGestureRecognizer Command="{Binding ModifyUserCommand}"/>  </RelativeLayout.GestureRecognizers>  <Image x:Name="Logo" HorizontalOptions="Center"  RelativeLayout.XConstraint="{ConstraintExpression Type=RelativeToParent,Property=X,Factor=1,Constant=60}"  HeightRequest="150"  Source="logo"/>  <ffimageloading:CachedImage x:Name="Picture"  RelativeLayout.YConstraint="{ConstraintExpression Type=RelativeToView, ElementName=Logo,Property=Y,Factor=1,Constant=140}"  Aspect="AspectFit"  Source="{Binding User.PictureFullPath}"  CacheDuration= "50"  HeightRequest="100"  Margin="5"  RetryCount= "3"  RetryDelay= "600">  <ffimageloading:CachedImage.Transformations>  <fftransformations:CircleTransformation />  </ffimageloading:CachedImage.Transformations>  </ffimageloading:CachedImage>  <ffimageloading:CachedImage x:Name="Team"  RelativeLayout.YConstraint="{ConstraintExpression Type=RelativeToView, ElementName=Picture,Property=Y,Factor=1,Constant=50}"  RelativeLayout.XConstraint="{ConstraintExpression Type=RelativeToView, ElementName=Picture,Property=X,Factor=1,Constant=60}"  Aspect="AspectFit"  Source="{Binding User.FavoriteTeam.LogoImageFullPath}"  CacheDuration= "50"  HeightRequest="50"  Margin="5"  RetryCount= "3"  RetryDelay= "600"/>  <Label RelativeLayout.YConstraint="{ConstraintExpression Type=RelativeToView, ElementName=Picture,Property=Y,Factor=1,Constant=118}"  FontAttributes="Bold"  FontSize="Medium"  Text="{Binding Player.FullName}"  TextColor="Black"/>  <Label RelativeLayout.YConstraint="{ConstraintExpression Type=RelativeToView, ElementName=Picture,Property=Y,Factor=1,Constant=100}"  FontAttributes="Bold"  FontSize="Small"  Text="{Binding Player.NickName}"  TextColor="Blue"/>  </RelativeLayout>  <Label FontAttributes="Bold"  FontSize="Large"  Text="{Binding User.FullName}"/>  <ListView BackgroundColor="Transparent"  ItemsSource="{Binding Menus}"  HasUnevenRows="True"  SeparatorVisibility="None">  <ListView.ItemTemplate>  <DataTemplate>  <ViewCell>  <Grid>  <Grid.GestureRecognizers>  <TapGestureRecognizer Command="{Binding SelectMenuCommand}"/>  </Grid.GestureRecognizers>  <Grid.ColumnDefinitions>  <ColumnDefinition Width="Auto"></ColumnDefinition>  <ColumnDefinition Width="\*"></ColumnDefinition>  </Grid.ColumnDefinitions>  <Image Grid.Column="0"  HeightRequest="32"  Margin="5"  Source="{Binding Icon}"  WidthRequest="32"/>  <Label Grid.Column="1"  FontAttributes="Bold"  VerticalOptions="Center"  Text="{Binding Title}"/>  </Grid>  </ViewCell>  </DataTemplate>  </ListView.ItemTemplate>  </ListView>  </StackLayout>  </ContentPage>  </MasterDetailPage.Master>  </MasterDetailPage> |  |

## GenericAppMasterDetailViewModel

Modificamos la **GenericAppMasterDetailPageViewModel**

|  |  |
| --- | --- |
| **GenericAppMasterDetailPageViewModel** | **Comentarios** |
| using System.Collections.Generic;  using System.Collections.ObjectModel;  using System.Linq;  using Newtonsoft.Json;  using GenericApp.Common.Helpers;  using GenericApp.Common.Models;  using GenericApp.Common.Responses;  using GenericApp.Prism.ItemViewModels;  using GenericApp.Prism.Views;  using Prism.Navigation;  namespace GenericApp.Prism.ViewModels  {  public class GenericAppMasterDetailPageViewModel : ViewModelBase  {  private readonly INavigationService \_navigationService;  private static GenericAppMasterDetailPageViewModel \_instance;  public static GenericAppMasterDetailPageViewModel GetInstance()  {  return \_instance;  }  private UserResponse \_user;  public UserResponse User  {  get => \_user;  set => SetProperty(ref \_user, value);  }  public GenericAppMasterDetailPageViewModel(INavigationService navigationService) : base(navigationService)  {  \_instance = this;  \_navigationService = navigationService;  LoadMenus();  LoadUser();  }  public ObservableCollection<MenuItemViewModel> Menus { get; set; }  public void LoadUser()  {  if (Settings.IsLogin)  {  TokenResponse token = JsonConvert.DeserializeObject<TokenResponse>(Settings.Token);  User = token.User;  }  }  private void LoadMenus()  {  List<Menu> menus = new List<Menu>  {  new Menu  {  Icon = "ic\_home",  PageName = $"{nameof(HomePage)}",  Title = "Inicio"  },  new Menu  {  Icon = "ic\_card\_giftcard",  PageName = $"{nameof(ProductsPage)}",  Title = "Productos"  },    new Menu  {  Icon = "ic\_person",  PageName = $"{nameof(ModifyUserPage)}",  Title = "Modificar Usuario",  IsLoginRequired = true  },    new Menu  {  Icon = "ic\_exit\_to\_app",  PageName = $"{nameof(LoginPage)}",  Title = Settings.IsLogin ? "Cerrar Sesión" : "Login"  }  };  Menus = new ObservableCollection<MenuItemViewModel>(  menus.Select(m => new MenuItemViewModel(\_navigationService)  {  Icon = m.Icon,  PageName = m.PageName,  Title = m.Title,  IsLoginRequired = m.IsLoginRequired  }).ToList());  }  }  } |  |

# LoginPage

## LoginPage

Adicionamos la licencia de SyncFusion en **App.xaml.cs**:

|  |  |
| --- | --- |
| **CountryEntity** | **Comentarios** |
| protected override async void OnInitialized()  {  Syncfusion.Licensing.SyncfusionLicenseProvider.RegisterLicense("MTY2MzIyQDMxMzcyZTMzMmUzMFVnNW5KSnM2dTZmRDljWm1RYTduQXFwRmNKSzVPWk1lT1JGSFRySXZCUTA9");  InitializeComponent();  await NavigationService.NavigateAsync("NavigationPage/MainPage");  } |  |

Inicializamos en **MainActivity**:

|  |  |
| --- | --- |
| **MainActivity** | **Comentarios** |
| global::Xamarin.Forms.Forms.Init(this, savedInstanceState);  new SfBusyIndicatorRenderer();  FFImageLoading.Forms.Platform.CachedImageRenderer.Init(true);  LoadApplication(new App(new AndroidInitializer())); |  |

Dentro de la carpeta **GenericApp.Prism /Views** creamos la **LoginPage**

|  |  |
| --- | --- |
| **LoginPage** | **Comentarios** |
| <?xml version="1.0" encoding="utf-8" ?>  <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  xmlns:prism="http://prismlibrary.com"  prism:ViewModelLocator.AutowireViewModel="True"  xmlns:ios="clr-namespace:Xamarin.Forms.PlatformConfiguration.iOSSpecific;assembly=Xamarin.Forms.Core"  ios:Page.UseSafeArea="true"  xmlns:busyindicator="clr-namespace:Syncfusion.SfBusyIndicator.XForms;assembly=Syncfusion.SfBusyIndicator.XForms"  xmlns:inputLayout="clr-namespace:Syncfusion.XForms.TextInputLayout;assembly=Syncfusion.Core.XForms"  x:Class="GenericApp.Prism.Views.LoginPage"  BackgroundColor="{StaticResource ColorBackground}"  Title="{Binding Title}">  <AbsoluteLayout>  <StackLayout AbsoluteLayout.LayoutBounds="0,0,1,1"  AbsoluteLayout.LayoutFlags="All"  Padding="5">  <ScrollView>  <StackLayout>  <Image HeightRequest="150"  Margin="20"  Source="logo"/>  <StackLayout VerticalOptions="CenterAndExpand">  <inputLayout:SfTextInputLayout Hint="Email:"  ContainerType="Outlined">  <Entry Placeholder="Ingrese EMail..."  Keyboard="Email"  Text="{Binding Email}" />  </inputLayout:SfTextInputLayout>  <inputLayout:SfTextInputLayout Hint="Password:"  EnablePasswordVisibilityToggle="true"  ContainerType="Outlined">  <Entry Placeholder="Ingrese Password..."  IsPassword="True"  Text="{Binding Password}" />  </inputLayout:SfTextInputLayout>  </StackLayout>  <Label FontAttributes="Bold"  HorizontalOptions="Center"  Text="Olvidó su Password?"  TextColor="{StaticResource ColorAccent}"  VerticalOptions="CenterAndExpand">  <Label.GestureRecognizers>  <TapGestureRecognizer Command="{Binding ForgotPasswordCommand}"/>  </Label.GestureRecognizers>  </Label>  </StackLayout>  </ScrollView>  <StackLayout VerticalOptions="EndAndExpand">  <Button Command="{Binding LoginCommand}"  IsEnabled="{Binding IsEnabled}"  Text="Login"/>  <Button Command="{Binding RegisterCommand}"  IsEnabled="{Binding IsEnabled}"  Text="Registrarse"  Style="{StaticResource DangerButton}"/>  </StackLayout>  </StackLayout>  <busyindicator:SfBusyIndicator AnimationType="Gear"  AbsoluteLayout.LayoutBounds=".5,.5,.5,.5"  AbsoluteLayout.LayoutFlags="All"  BackgroundColor="Transparent"  HorizontalOptions="Center"  TextColor="Blue"  IsBusy="{Binding IsRunning}"  Title="Loading..."  VerticalOptions="Center"  ViewBoxWidth="80"  ViewBoxHeight="80" />  </AbsoluteLayout>  </ContentPage> |  |

## LoginPageViewModel

Modificamos la **LoginPageViewModel**

|  |  |
| --- | --- |
| **LoginPageViewModel** | **Comentarios** |
| using Newtonsoft.Json;  using GenericApp.Common.Requests;  using GenericApp.Common.Responses;  using GenericApp.Common.Services;  using GenericApp.Prism.Views;  using Prism.Commands;  using Prism.Navigation;  using Xamarin.Essentials;  using GenericApp.Common.Helpers;  namespace GenericApp.Prism.ViewModels  {  public class LoginPageViewModel : ViewModelBase  {  private bool \_isRunning;  private bool \_isEnabled;  private string \_password;  private DelegateCommand \_loginCommand;  private DelegateCommand \_registerCommand;  private DelegateCommand \_forgotPasswordCommand;  private readonly INavigationService \_navigationService;  private readonly IApiService \_apiService;  private string \_pageReturn;  public LoginPageViewModel(INavigationService navigationService, IApiService apiService) : base(navigationService)  {  \_navigationService = navigationService;  \_apiService = apiService;  Title = "Login";  IsEnabled = true;  }  public DelegateCommand LoginCommand => \_loginCommand ?? (\_loginCommand = new DelegateCommand(LoginAsync));  public DelegateCommand RegisterCommand => \_registerCommand ?? (\_registerCommand = new DelegateCommand(RegisterAsync));  public DelegateCommand ForgotPasswordCommand => \_forgotPasswordCommand ?? (\_forgotPasswordCommand = new DelegateCommand(ForgotPasswordAsync));  public bool IsRunning  {  get => \_isRunning;  set => SetProperty(ref \_isRunning, value);  }  public bool IsEnabled  {  get => \_isEnabled;  set => SetProperty(ref \_isEnabled, value);  }  public string Email { get; set; }  public string Password  {  get => \_password;  set => SetProperty(ref \_password, value);  }  public override void OnNavigatedTo(INavigationParameters parameters)  {  base.OnNavigatedTo(parameters);  if (parameters.ContainsKey("pageReturn"))  {  \_pageReturn = parameters.GetValue<string>("pageReturn");  }  }  private async void LoginAsync()  {  if (string.IsNullOrEmpty(Email))  {  await App.Current.MainPage.DisplayAlert(  "Error",  "Debe ingresar un EMail",  "Aceptar");  return;  }  if (string.IsNullOrEmpty(Password))  {  await App.Current.MainPage.DisplayAlert(  "Error",  "Debe ingresar unPassword",  "Aceptar");  return;  }  IsRunning = true;  IsEnabled = false;  if (Connectivity.NetworkAccess != NetworkAccess.Internet)  {  IsRunning = false;  IsEnabled = true;  await App.Current.MainPage.DisplayAlert(  "Error",  "Error de conexión",  "Aceptar");  return;  }  string url = App.Current.Resources["UrlAPI"].ToString();  TokenRequest request = new TokenRequest  {  Password = Password,  Username = Email  };  Response response = await \_apiService.GetTokenAsync(url, "api", "/Account/CreateToken", request);  IsRunning = false;  IsEnabled = true;  if (!response.IsSuccess)  {  await App.Current.MainPage.DisplayAlert(  "Error",  "Usuario o Password incorrectos",  "Aceptar");  Password = string.Empty;  return;  }  TokenResponse token = (TokenResponse)response.Result;  Settings.Token = JsonConvert.SerializeObject(token);  Settings.IsLogin = true;  IsRunning = false;  IsEnabled = true;  //await \_navigationService.NavigateAsync($"/{nameof(OnSaleMasterDetailPage)}/NavigationPage/{nameof(ProductsPage)}");  if (string.IsNullOrEmpty(\_pageReturn))  {  await \_navigationService.NavigateAsync($"/{nameof(GenericAppMasterDetailPage)}/NavigationPage/{nameof(HomePage)}");  }  else  {  await \_navigationService.NavigateAsync($"/{nameof(GenericAppMasterDetailPage)}/NavigationPage/{\_pageReturn}");  }  Password = string.Empty;  }  private async void ForgotPasswordAsync()  {  var parameters = new NavigationParameters();  parameters.Add("email", Email);  await \_navigationService.NavigateAsync(nameof(RecoverPasswordPage), parameters);  }  private async void RegisterAsync()  {  await \_navigationService.NavigateAsync(nameof(RegisterPage));  }  }  } |  |

# HomePage

## HomePage

Dentro de la carpeta **GenericApp.Prism /Views** creamos la **HomePage**

|  |  |
| --- | --- |
| **HomePage** | **Comentarios** |
| <?xml version="1.0" encoding="utf-8" ?>  <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  xmlns:prism="http://prismlibrary.com"  prism:ViewModelLocator.AutowireViewModel="True"  xmlns:ffimageloading="clr-namespace:FFImageLoading.Forms;assembly=FFImageLoading.Forms"  xmlns:fftransformations="clr-namespace:FFImageLoading.Transformations;assembly=FFImageLoading.Transformations"  xmlns:ios="clr-namespace:Xamarin.Forms.PlatformConfiguration.iOSSpecific;assembly=Xamarin.Forms.Core"  ios:Page.UseSafeArea="true"  x:Class="GenericApp.Prism.Views.HomePage"  BackgroundColor="{StaticResource ColorBackground}"  Title="{Binding Title}">    <ContentPage.Padding>  <OnPlatform x:TypeArguments="Thickness">  <On Platform="Android, UWP">0</On>  <On Platform="iOS">0,20,0,0</On>  </OnPlatform>  </ContentPage.Padding>  <StackLayout Padding="10">  <Image x:Name="Logo" HorizontalOptions="Center"  RelativeLayout.XConstraint="{ConstraintExpression Type=RelativeToParent,Property=X,Factor=1,Constant=60}"  HeightRequest="150"  Source="logo"/>  <BoxView  HeightRequest="5"  MinimumHeightRequest="1"  BackgroundColor="Blue">  </BoxView>    <Label  Text="BIENVENIDO/A"  FontAttributes="Bold"  FontSize="Large"  HorizontalOptions="CenterAndExpand">  </Label>  <Label FontAttributes="Bold"  FontSize="Large"  Text="{Binding User.FullName}"  HorizontalOptions="CenterAndExpand"/>  <ffimageloading:CachedImage x:Name="Picture"  Aspect="AspectFit"  Source="{Binding User.PictureFullPath}"  CacheDuration= "50"  HeightRequest="250"  Margin="5"  RetryCount= "3"  RetryDelay= "600">  <ffimageloading:CachedImage.Transformations>  <fftransformations:CircleTransformation />  </ffimageloading:CachedImage.Transformations>  </ffimageloading:CachedImage>  <RelativeLayout>  <ffimageloading:CachedImage x:Name="Team"  RelativeLayout.XConstraint="{ConstraintExpression Type=RelativeToParent, ElementName=Picture,Property=X,Factor=1,Constant=250}"  RelativeLayout.YConstraint="{ConstraintExpression Type=RelativeToParent, ElementName=Picture,Property=Y,Factor=1,Constant=-650}"    Aspect="AspectFit"  Source="{Binding User.FavoriteTeam.LogoImageFullPath}"  CacheDuration= "100"  HeightRequest="100"  Margin="5"  RetryCount= "3"  RetryDelay= "600"/>  </RelativeLayout>  </StackLayout>  </ContentPage> |  |

## HomePageViewModel

Modificamos la **HomePageViewModel**

|  |  |
| --- | --- |
| **HomePageViewModel** | **Comentarios** |
| using System.Collections.ObjectModel;  using Newtonsoft.Json;  using GenericApp.Common.Helpers;  using GenericApp.Common.Responses;  using GenericApp.Prism.ItemViewModels;  using Prism.Navigation;  namespace GenericApp.Prism.ViewModels  {  public class HomePageViewModel : ViewModelBase  {  private readonly INavigationService \_navigationService;  private UserResponse \_user;  public UserResponse User  {  get => \_user;  set => SetProperty(ref \_user, value);  }  public HomePageViewModel(INavigationService navigationService) : base(navigationService)  {  \_navigationService = navigationService;  Title = "Generic App";  LoadUser();  }  public ObservableCollection<MenuItemViewModel> Menus { get; set; }  public void LoadUser()  {  if (Settings.IsLogin)  {  TokenResponse token = JsonConvert.DeserializeObject<TokenResponse>(Settings.Token);  User = token.User;  }  }  }  } |  |

# Mapa

## Inicializar la librería para Android en MainActivity

|  |  |
| --- | --- |
| **MainActivity** | **Comentarios** |
| protected override void OnCreate(Bundle savedInstanceState)  {  TabLayoutResource = Resource.Layout.Tabbar;  ToolbarResource = Resource.Layout.Toolbar;  base.OnCreate(savedInstanceState);  global::Xamarin.Forms.Forms.Init(this, savedInstanceState);  new SfBusyIndicatorRenderer();  Xamarin.FormsMaps.Init(this, savedInstanceState);  FFImageLoading.Forms.Platform.CachedImageRenderer.Init(true);  LoadApplication(new App(new AndroidInitializer()));  } |  |

## Clave de Mapas

Agregar en el **AndroidManifest.xml**

|  |  |
| --- | --- |
| **AndroidManifest.xml** | **Comentarios** |
| <?xml version="1.0" encoding="utf-8"?>  <manifest xmlns:android="http://schemas.android.com/apk/res/android" android:versionCode="1" android:versionName="1.0" package="com.companyname.appname" android:installLocation="auto">  <uses-sdk android:minSdkVersion="21" android:targetSdkVersion="29" />  <uses-permission android:name="android.permission.INTERNET" />  <uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE" />  <uses-permission android:name="android.permission.ACCESS\_WIFI\_STATE" />  <uses-permission android:name="android.permission.CAMERA" />  <uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE" />  <uses-permission android:name="android.permission.READ\_EXTERNAL\_STORAGE" />  <uses-permission android:name="android.permission.ACCESS\_MOCK\_LOCATION" />  <uses-permission android:name="android.permission.ACCESS\_LOCATION\_EXTRA\_COMMANDS" />  <uses-permission android:name="android.permission.ACCESS\_FINE\_LOCATION" />  <uses-permission android:name="android.permission.ACCESS\_COARSE\_LOCATION" />  <application android:label="GenericApp" android:icon="@mipmap/ic\_launcher">  <provider android:name="android.support.v4.content.FileProvider" android:authorities="${applicationId}.fileprovider" android:exported="false" android:grantUriPermissions="true">  <meta-data android:name="android.support.FILE\_PROVIDER\_PATHS" android:resource="@xml/file\_paths"></meta-data>  </provider>  <meta-data android:name="com.google.android.maps.v2.API\_KEY" android:value="AIzaSyAtxvXVhbzV9OTwZh8UxVsW2A58WYf-Btc" />  <uses-library android:name="org.apache.http.legacy" android:required="false" />  </application>  </manifest> |  |

## Clase CustomPin

En el Proyecto Prism creamos la Clase **CustomPin**

|  |  |
| --- | --- |
| **CustomPin** | **Comentarios** |
| using Xamarin.Forms.Maps;  namespace GenericApp.Prism  {  public class CustomPin : Pin  {  public string Name { get; set; }  public string Url { get; set; }  }  } |  |

## Clase CustomMap

En el Proyecto Prism creamos la Clase **CustomMap**

|  |  |
| --- | --- |
| **CustomMap** | **Comentarios** |
| using System.Collections.Generic;  using Xamarin.Forms.Maps;  namespace GenericApp.Prism  {  public class CustomMap : Map  {  public List<CustomPin> CustomPins { get; set; }  }  } |  |

## Clase CustomMapRenderer

Dentro del **Proyecto Android** creamos la Clase **CustomMapRenderer**

|  |  |
| --- | --- |
| **CustomMapRenderer** | **Comentarios** |
| using System.Collections.Generic;  using Android.Content;  using Android.Gms.Maps;  using Android.Gms.Maps.Model;  using CustomRenderer.Droid;  using GenericApp.Prism;  using GenericApp.Prism.Droid;  using GenericApp.Prism.ViewModels;  using Xamarin.Forms;  using Xamarin.Forms.Maps;  using Xamarin.Forms.Maps.Android;  [assembly: ExportRenderer(typeof(CustomMap), typeof(CustomMapRenderer))]  namespace CustomRenderer.Droid  {  public class CustomMapRenderer : MapRenderer, GoogleMap.IInfoWindowAdapter  {  private int opc;  List<CustomPin> customPins;  public CustomMapRenderer(Context context) : base(context)  {  }  protected override void OnElementChanged(Xamarin.Forms.Platform.Android.ElementChangedEventArgs<Map> e)  {  base.OnElementChanged(e);  if (e.OldElement != null)  {  NativeMap.InfoWindowClick -= OnInfoWindowClick;  }  if (e.NewElement != null)  {  var formsMap = (CustomMap)e.NewElement;  customPins = formsMap.CustomPins;  }  }  protected override void OnMapReady(GoogleMap map)  {  base.OnMapReady(map);  NativeMap.InfoWindowClick += OnInfoWindowClick;  NativeMap.SetInfoWindowAdapter(this);  }  protected override MarkerOptions CreateMarker(Pin pin)  {  var marker = new MarkerOptions();  marker.SetPosition(new LatLng(pin.Position.Latitude, pin.Position.Longitude));  marker.SetTitle(pin.Address);  marker.SetSnippet(pin.Label);  if (pin.ClassId == "Item1")  {  opc = 1;  }  if (pin.ClassId == "Item2")  {  opc = 2;  }  if (pin.ClassId == "Item3")  {  opc = 3;  }  if (pin.ClassId == "Item4")  {  opc = 4;  }  if (pin.ClassId == "Item5")  {  opc = 5;  }  if (pin.ClassId == "Item6")  {  opc = 6;  }  if (pin.StyleId == "PinesAmarillos")  {  marker.SetIcon(BitmapDescriptorFactory.FromResource(Resource.Drawable.pinamarillo));  }  if (pin.StyleId == "PinesAzules")  {  marker.SetIcon(BitmapDescriptorFactory.FromResource(Resource.Drawable.pinazul));  }  if (pin.StyleId == "PinesCelestes")  {  marker.SetIcon(BitmapDescriptorFactory.FromResource(Resource.Drawable.pinceleste));  }  if (pin.StyleId == "PinesNaranjas")  {  marker.SetIcon(BitmapDescriptorFactory.FromResource(Resource.Drawable.pinnaranja));  }  if (pin.StyleId == "PinesRojos")  {  marker.SetIcon(BitmapDescriptorFactory.FromResource(Resource.Drawable.pinrojo));  }  if (pin.StyleId == "PinesRosas")  {  marker.SetIcon(BitmapDescriptorFactory.FromResource(Resource.Drawable.pinrosa));  }  if (pin.StyleId == "PinesVerdes")  {  marker.SetIcon(BitmapDescriptorFactory.FromResource(Resource.Drawable.pinverde));  }  if (pin.StyleId == "PinesVerdesClaro")  {  marker.SetIcon(BitmapDescriptorFactory.FromResource(Resource.Drawable.pinverdeclaro));  }  if (pin.StyleId == "PinesVioletas")  {  marker.SetIcon(BitmapDescriptorFactory.FromResource(Resource.Drawable.pinvioleta));  }  return marker;  }  void OnInfoWindowClick(object sender, GoogleMap.InfoWindowClickEventArgs e)  {  var txtBuscar = e.Marker.Title;  if (opc == 1)  {  ProductsPageViewModel.GetInstance().Search = txtBuscar;  ProductsPageViewModel.GetInstance().Search = txtBuscar;  ProductsPageViewModel.GetInstance().CerrarMapa();  }  if (opc == 2)  {  ProductsPageViewModel.GetInstance().Search = txtBuscar;  ProductsPageViewModel.GetInstance().CerrarMapa();  }  if (opc == 3)  {  ProductsPageViewModel.GetInstance().Search = txtBuscar;  ProductsPageViewModel.GetInstance().CerrarMapa();  }  if (opc == 4)  {  ProductsPageViewModel.GetInstance().Search = txtBuscar;  ProductsPageViewModel.GetInstance().CerrarMapa();  }  }  public Android.Views.View GetInfoContents(Marker marker)  {  var inflater = Android.App.Application.Context.GetSystemService(Context.LayoutInflaterService) as Android.Views.LayoutInflater;  if (inflater != null)  {  Android.Views.View view;  return null;// view;  }  return null;  }  public Android.Views.View GetInfoWindow(Marker marker)  {  return null;  }  CustomPin GetCustomPin(Marker annotation)  {  var position = new Position(annotation.Position.Latitude, annotation.Position.Longitude);  foreach (var pin in customPins)  {  if (pin.Position == position)  {  return pin;  }  }  return null;  }  }  } |  |

## ProductsMapPage

Creamos la **ProductsMapPage**

|  |  |
| --- | --- |
| **ProductsMapPage** | **Comentarios** |
| <?xml version="1.0" encoding="utf-8" ?>  <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  prism:ViewModelLocator.AutowireViewModel="True"  x:Class="GenericApp.Prism.Views.ProductsMapPage"  xmlns:prism="clr-namespace:Prism.Mvvm;assembly=Prism.Forms"  xmlns:maps="clr-namespace:GenericApp.Prism;assembly=GenericApp.Prism"  BackgroundColor="Blue"  Title="{Binding Title}">  <ContentPage.ToolbarItems>  <ToolbarItem Icon="ic\_action\_streetview"  Clicked="MapStreetCommand" />  <ToolbarItem Icon="ic\_action\_satellite"  Clicked="MapSateliteCommand" />  <ToolbarItem Icon="ic\_action\_looks\_two"  Clicked="MapHybridCommand" />  </ContentPage.ToolbarItems>  <ContentPage.Content>  <StackLayout>  <maps:CustomMap  x:Name="MyMap"  MapType="Street"  HorizontalOptions="FillAndExpand"  VerticalOptions="FillAndExpand">  </maps:CustomMap>  </StackLayout>  </ContentPage.Content>  </ContentPage> |  |

## ProductsMapPageViewModel

|  |  |
| --- | --- |
| **ProductsMapPageViewModel** | **Comentarios** |
| using GenericApp.Common.Services;  using Prism.Navigation;  namespace GenericApp.Prism.ViewModels  {  public class ProductsMapPageViewModel : ViewModelBase  {  private readonly INavigationService \_navigationService;  private readonly IApiService \_apiService;  private static ProductsMapPageViewModel \_instance;  public static ProductsMapPageViewModel GetInstance()  {  return \_instance;  }  public ProductsMapPageViewModel(INavigationService navigationService, IApiService apiService) : base(navigationService)  {  \_navigationService = navigationService;  \_apiService = apiService;  \_instance = this;  Title = "Mapa";  }  public async void CerrarMapa()  {  await \_navigationService.GoBackAsync();  }  }  } |  |

## ProductsMapsPage.xaml.cs

|  |  |
| --- | --- |
| **ProductsMapsPage.xaml.cs** | **Comentarios** |
| using GenericApp.Common.Services;  using GenericApp.Prism.ItemViewModels;  using GenericApp.Prism.ViewModels;  using Plugin.Permissions;  using Plugin.Permissions.Abstractions;  using System;  using System.Collections.Generic;  using System.Linq;  using System.Threading.Tasks;  using Xamarin.Forms;  using Xamarin.Forms.Maps;  namespace GenericApp.Prism.Views  {  public partial class ProductsMapPage : ContentPage  {  private readonly IGeolocatorService \_geolocatorService;  public ProductsMapPage(IGeolocatorService geolocatorService)  {  InitializeComponent();  \_geolocatorService = geolocatorService;  MyMap.MapType = MapType.Street;  MyMap.IsVisible = false;  MoveMapToCurrentPositionAsync();  MyMap.IsVisible = true;  ShowPinsAsync();  }  protected override void OnAppearing()  {  base.OnAppearing();  MyMap.IsVisible = false;  MoveMapToCurrentPositionAsync();  MyMap.IsVisible = true;  }  private async Task<List<CustomPin>> ShowPinsAsync()  {  CustomPin pin = new CustomPin  {  Type = PinType.Place,  Position = new Position(-0, 0),  Label = " ",  Address = " ",  Name = " ",  StyleId = "",  Url = ""  };  var pins = new List<CustomPin> { pin };  ProductsPageViewModel productsViewModel = ProductsPageViewModel.GetInstance();  foreach (ProductItemViewModel product in productsViewModel.Products.ToList())  {  if (!string.IsNullOrEmpty(Convert.ToString(product.Latitude)) && !string.IsNullOrEmpty(Convert.ToString(product.Longitude)))  {  if (Convert.ToString(product.Latitude).Length > 5 && Convert.ToString(product.Longitude).Length > 5)  {  Position position = new Position(Convert.ToDouble(product.Latitude), Convert.ToDouble(product.Longitude));  if (product.State.Name == "Sin Iniciar")  {  MyMap.Pins.Add(new CustomPin  {  Label = Convert.ToString(product.Id) + '-' + product.State.Name,  Address = product.Name,  Position = position,  Type = PinType.Place,  StyleId = "PinesCelestes",  ClassId = "Item1",  });  }  else if (product.State.Name == "Iniciado")  {  MyMap.Pins.Add(new CustomPin  {  Label = Convert.ToString(product.Id) + '-' + product.State.Name,  Address = product.Name,  Position = position,  Type = PinType.Place,  StyleId = "PinesAmarillos",  ClassId = "Item1",  });  }  else if (product.State.Name == "Pendiente")  {  MyMap.Pins.Add(new CustomPin  {  Label = Convert.ToString(product.Id) + '-' + product.State.Name,  Address = product.Name,  Position = position,  Type = PinType.Place,  StyleId = "PinesRojos",  ClassId = "Item1",  });  }  else if (product.State.Name == "Terminado")  {  MyMap.Pins.Add(new CustomPin  {  Label = Convert.ToString(product.Id) + '-' + product.State.Name,  Address = product.Name,  Position = position,  Type = PinType.Place,  StyleId = "PinesVerdes",  ClassId = "Item1",  });  }  }  }  }  return pins;  }  private async void MoveMapToCurrentPositionAsync()  {  bool isLocationPermision = await CheckLocationPermisionsAsync();  if (isLocationPermision)  {  MyMap.IsShowingUser = true;  await \_geolocatorService.GetLocationAsync();  if (\_geolocatorService.Latitude != 0 && \_geolocatorService.Longitude != 0)  {  Position position = new Position(  \_geolocatorService.Latitude,  \_geolocatorService.Longitude);  MyMap.IsVisible = false;  MyMap.MoveToRegion(MapSpan.FromCenterAndRadius(  position,  Distance.FromKilometers(.5)));  MyMap.IsVisible = true;  }  }  }  private async Task<bool> CheckLocationPermisionsAsync()  {  PermissionStatus permissionLocation = await CrossPermissions.Current.CheckPermissionStatusAsync(Permission.Location);  PermissionStatus permissionLocationAlways = await CrossPermissions.Current.CheckPermissionStatusAsync(Permission.LocationAlways);  PermissionStatus permissionLocationWhenInUse = await CrossPermissions.Current.CheckPermissionStatusAsync(Permission.LocationWhenInUse);  bool isLocationEnabled = permissionLocation == PermissionStatus.Granted ||  permissionLocationAlways == PermissionStatus.Granted ||  permissionLocationWhenInUse == PermissionStatus.Granted;  if (isLocationEnabled)  {  return true;  }  await CrossPermissions.Current.RequestPermissionsAsync(Permission.Location);  permissionLocation = await CrossPermissions.Current.CheckPermissionStatusAsync(Permission.Location);  permissionLocationAlways = await CrossPermissions.Current.CheckPermissionStatusAsync(Permission.LocationAlways);  permissionLocationWhenInUse = await CrossPermissions.Current.CheckPermissionStatusAsync(Permission.LocationWhenInUse);  return permissionLocation == PermissionStatus.Granted ||  permissionLocationAlways == PermissionStatus.Granted ||  permissionLocationWhenInUse == PermissionStatus.Granted;  }  private void MapStreetCommand(object sender, EventArgs eventArgs)  {  MyMap.MapType = MapType.Street;  }  private void MapSateliteCommand(object sender, EventArgs eventArgs)  {  MyMap.MapType = MapType.Satellite;  }  private void MapHybridCommand(object sender, EventArgs eventArgs)  {  MyMap.MapType = MapType.Hybrid;  }  }  } |  |

# Agregar Nuevo Producto

## AddProductPage

|  |  |
| --- | --- |
| **AddProductPage** | **Comentarios** |
| <?xml version="1.0" encoding="utf-8" ?>  <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  xmlns:prism="http://prismlibrary.com"  prism:ViewModelLocator.AutowireViewModel="True"  xmlns:ios="clr-namespace:Xamarin.Forms.PlatformConfiguration.iOSSpecific;assembly=Xamarin.Forms.Core"  ios:Page.UseSafeArea="true"  xmlns:busyindicator="clr-namespace:Syncfusion.SfBusyIndicator.XForms;assembly=Syncfusion.SfBusyIndicator.XForms"  x:Class="GenericApp.Prism.Views.AddProductPage"  BackgroundColor="White"  Title="{Binding Title}">  <AbsoluteLayout>  <StackLayout AbsoluteLayout.LayoutBounds="0,0,1,1"  AbsoluteLayout.LayoutFlags="All"  Padding="5">  <ScrollView>  <StackLayout  Padding="5">  <Grid Padding="5,5">  <Grid.ColumnDefinitions>  <ColumnDefinition Width="1.2\*" />  <ColumnDefinition Width="\*" />  <ColumnDefinition Width="\*" />  <ColumnDefinition Width="\*" />  <ColumnDefinition Width="48" />  </Grid.ColumnDefinitions>  <Grid.RowDefinitions>  <RowDefinition Height="Auto" />  <RowDefinition Height="Auto" />  <RowDefinition Height="Auto" />  <RowDefinition Height="Auto" />  <RowDefinition Height="Auto" />  </Grid.RowDefinitions>  <Label  Grid.Row="0"  Grid.Column="0"    Text="Categoría:"  FontSize="Small"  VerticalOptions="Center"  TextColor="Black"  FontAttributes="Bold"></Label>  <Picker x:Name="Categoría"  Grid.Row="0"  Grid.Column="1"  Grid.ColumnSpan="4"  FontSize="Small"  IsEnabled="True"  ItemDisplayBinding="{Binding Name}"  ItemsSource="{Binding Categories}"  SelectedItem="{Binding Category, Mode=TwoWay}"  HeightRequest="40"  Title="Elija Categoría"></Picker>  <Label  Grid.Row="1"  Grid.Column="0"  Grid.ColumnSpan="2"  Text="Nombre:"  FontSize="Small"  VerticalOptions="Center"  TextColor="Black"  FontAttributes="Bold"></Label>  <Entry HorizontalOptions="FillAndExpand"  Grid.Row="1"  Grid.Column="1"  Grid.ColumnSpan="3"  FontSize="Small"  Placeholder="Ingrese Nombre..."  HeightRequest="40"  Text="{Binding Name}"/>    <Label  Grid.Row="2"  Grid.Column="0"  Grid.ColumnSpan="2"  Text="Descripción:"  FontSize="Small"  VerticalOptions="Center"  TextColor="Black"  FontAttributes="Bold"></Label>    <Entry HorizontalOptions="FillAndExpand"  Grid.Row="3"  Grid.Column="0"  Grid.ColumnSpan="4"  FontSize="Small"  Placeholder="Ingrese Descripción..."  HeightRequest="40"  Text="{Binding Description}"/>  <ImageButton  Grid.Row="3"  Grid.Column="4"  Command="{Binding GetAddressCommand}"  Source="ic\_map\_black"/>  <Label  Grid.Row="4"  Grid.Column="0"  Text="Precio:"  FontSize="Small"  VerticalOptions="Center"  TextColor="Black"  FontAttributes="Bold"></Label>  <Entry  Grid.Row="4"  Grid.Column="1"  Grid.ColumnSpan="2"  Text="{Binding Price}"  FontSize="Small"  VerticalOptions="Center"  TextColor="Black"></Entry>    <Label  Grid.Row="5"  Grid.Column="0"  Text="Latitud:"  FontSize="Small"  VerticalOptions="Center"  TextColor="Black"  FontAttributes="Bold"></Label>  <Label  Grid.Row="5"  Grid.Column="1"  Text="{Binding Latitude}"  FontSize="Small"  VerticalOptions="Center"  LineBreakMode="TailTruncation"  TextColor="Black"></Label>  <Label  Grid.Row="5"  Grid.Column="2"  Text="Longitud:"  FontSize="Small"  VerticalOptions="Center"  TextColor="Black"  FontAttributes="Bold"></Label>  <Label  Grid.Row="5"  Grid.Column="3"  Text="{Binding Longitude}"  LineBreakMode="TailTruncation"  FontSize="Small"  VerticalOptions="Center"  TextColor="Black"></Label>  <Label  Grid.Row="6"  Grid.Column="0"  Text="Estado:"  FontSize="Small"  VerticalOptions="Center"  TextColor="Black"  FontAttributes="Bold"></Label>  <Picker x:Name="Estado"  Grid.Row="6"  Grid.Column="1"  Grid.ColumnSpan="4"  FontSize="Small"  IsEnabled="True"  ItemDisplayBinding="{Binding Name}"  ItemsSource="{Binding States}"  SelectedItem="{Binding State}"  HeightRequest="40"  Title="Elija el Estado"></Picker>  </Grid>  <!--\*\*\*\*\* TOMAR FOTOS \*\*\*\*\*-->  <Grid Padding="5,5"  VerticalOptions="EndAndExpand">  <Grid.ColumnDefinitions>  <ColumnDefinition Width="\*" />  </Grid.ColumnDefinitions>  <Grid.RowDefinitions>  <RowDefinition Height="Auto" />  <RowDefinition Height="Auto" />  <RowDefinition Height="Auto" />  </Grid.RowDefinitions>  <Label  Grid.Row="0"  Grid.Column="0"  Text="Toque la imagen para cargar/cambiar una foto"  FontSize="Small"  TextColor="Blue"></Label>  <Image  Grid.Row="1"  Grid.Column="0"  HeightRequest="250"  HorizontalOptions="CenterAndExpand"  Source="{Binding ImageSource, Mode=TwoWay}">  <Image.GestureRecognizers>  <TapGestureRecognizer Command="{Binding TakePhotoCommand}" />  </Image.GestureRecognizers>  </Image>  </Grid>  </StackLayout>  </ScrollView>  <Grid Padding="5,5"  VerticalOptions="EndAndExpand">  <Grid.ColumnDefinitions>  <ColumnDefinition Width="\*" />  <ColumnDefinition Width="\*" />  </Grid.ColumnDefinitions>  <Grid.RowDefinitions>  <RowDefinition Height="Auto" />  <RowDefinition Height="Auto" />  <RowDefinition Height="Auto" />  </Grid.RowDefinitions>  <Button  Grid.Row="0"  Grid.Column="0"  Text="cancelar"  BackgroundColor="Red"  IsEnabled="{Binding IsEnabled}"  CornerRadius="30"  HeightRequest="60"  FontSize="Small"  TextColor="White"  FontAttributes="Bold"  VerticalOptions="EndAndExpand"  Command="{Binding CancelCommand}">  </Button>  <Button  Grid.Row="0"  Grid.Column="1"  Text="grabar"  BackgroundColor="Blue"  IsEnabled="{Binding IsEnabled}"  CornerRadius="30"  HeightRequest="60"  FontSize="Small"  TextColor="White"  FontAttributes="Bold"  VerticalOptions="End"  Command="{Binding SaveCommand}"></Button>  </Grid>  </StackLayout>  <busyindicator:SfBusyIndicator AnimationType="Gear"  AbsoluteLayout.LayoutBounds=".5,.5,.5,.5"  AbsoluteLayout.LayoutFlags="All"  BackgroundColor="Transparent"  HorizontalOptions="Center"  TextColor="Blue"  IsVisible="{Binding IsRunning}"  Title="Guardando..."  VerticalOptions="Center"  ViewBoxWidth="80"  ViewBoxHeight="80" />  </AbsoluteLayout>  </ContentPage> |  |

## AddProductPageViewModel

|  |  |
| --- | --- |
| **AddProductPageViewModel** | **Comentarios** |
| using GenericApp.Common.Helpers;  using GenericApp.Common.Requests;  using GenericApp.Common.Responses;  using GenericApp.Common.Services;  using Newtonsoft.Json;  using Plugin.Media;  using Plugin.Media.Abstractions;  using Prism.Commands;  using Prism.Navigation;  using System;  using System.Collections.Generic;  using System.Collections.ObjectModel;  using System.Linq;  using Xamarin.Essentials;  using Xamarin.Forms;  using Xamarin.Forms.Maps;  namespace GenericApp.Prism.ViewModels  {  public class AddProductPageViewModel : ViewModelBase  {  private readonly INavigationService \_navigationService;  private readonly IGeolocatorService \_geolocatorService;  private readonly IApiService \_apiService;  private readonly IFilesHelper \_filesHelper;  private bool \_isRunning;  public bool IsRunning  {  get => \_isRunning;  set => SetProperty(ref \_isRunning, value);  }  private bool \_isEnabled;  public bool IsEnabled  {  get => \_isEnabled;  set => SetProperty(ref \_isEnabled, value);  }    private bool \_isRefreshing;  public bool IsRefreshing  {  get => \_isRefreshing;  set => SetProperty(ref \_isRefreshing, value);  }    private MediaFile \_file;  public MediaFile File  {  get => \_file;  set => SetProperty(ref \_file, value);  }  private Position \_position;  private ImageSource \_imageSource;  public ImageSource ImageSource  {  get => \_imageSource;  set => SetProperty(ref \_imageSource, value);  }  private StateResponse \_state;  public StateResponse State  {  get => \_state;  set => SetProperty(ref \_state, value);  }  private CategoryResponse \_category;  public CategoryResponse Category  {  get => \_category;  set => SetProperty(ref \_category, value);  }  private ObservableCollection<CategoryResponse> \_categories;  public ObservableCollection<CategoryResponse> Categories  {  get => \_categories;  set => SetProperty(ref \_categories, value);  }  private ObservableCollection<StateResponse> \_states;  public ObservableCollection<StateResponse> States  {  get => \_states;  set => SetProperty(ref \_states, value);  }  private string \_name;  public string Name  {  get => \_name;  set => SetProperty(ref \_name, value);  }  private string \_description;  public string Description  {  get => \_description;  set => SetProperty(ref \_description, value);  }  private decimal \_price;  public decimal Price  {  get => \_price;  set => SetProperty(ref \_price, value);  }  private double \_latitude;  public double Latitude  {  get => \_latitude;  set => SetProperty(ref \_latitude, value);  }  private double \_longitude;  public double Longitude  {  get => \_longitude;  set => SetProperty(ref \_longitude, value);  }  private string \_filter;  public string Filter  {  get => \_filter;  set => SetProperty(ref \_filter, value);  }  private DelegateCommand \_takePhotoCommand;  public DelegateCommand TakePhotoCommand => \_takePhotoCommand ?? (\_takePhotoCommand = new DelegateCommand(TakePhoto));  private DelegateCommand \_cancelCommand;  public DelegateCommand CancelCommand => \_cancelCommand ?? (\_cancelCommand = new DelegateCommand(Cancel));  private DelegateCommand \_saveCommand;  public DelegateCommand SaveCommand => \_saveCommand ?? (\_saveCommand = new DelegateCommand(Save));  private DelegateCommand \_getAddressCommand;  public DelegateCommand GetAddressCommand => \_getAddressCommand ?? (\_getAddressCommand = new DelegateCommand(LoadSourceAsync));  public AddProductPageViewModel(INavigationService navigationService, IGeolocatorService geolocatorService, IApiService apiService, IFilesHelper filesHelper) : base(navigationService)  {  \_navigationService = navigationService;  \_geolocatorService = geolocatorService;  \_apiService = apiService;  \_filesHelper = filesHelper;  IsEnabled = true;  Title = "Agregar Nuevo Producto";  instance = this;  ImageSource = "noimage.png";  LoadCategories();  LoadStates();  }  #region Singleton  private static AddProductPageViewModel instance;  public static AddProductPageViewModel GetInstance()  {  return instance;  }  #endregion  private async void LoadSourceAsync()  {  IsEnabled = false;  await \_geolocatorService.GetLocationAsync();  if (\_geolocatorService.Latitude == 0 && \_geolocatorService.Longitude == 0)  {  IsEnabled = true;  await App.Current.MainPage.DisplayAlert(  "Error",  "Error de Geolocalización",  "Aceptar");  //await \_navigationService.GoBackAsync();  return;  }  \_position = new Position(\_geolocatorService.Latitude, \_geolocatorService.Longitude);  Latitude = \_geolocatorService.Latitude;  Longitude = \_geolocatorService.Longitude;  Geocoder geoCoder = new Geocoder();  IEnumerable<string> sources = await geoCoder.GetAddressesForPositionAsync(\_position);  List<string> addresses = new List<string>(sources);  if (addresses.Count > 1)  {  Description = addresses[0];  }  IsEnabled = true;  }  private async void LoadCategories()  {  string url = App.Current.Resources["UrlAPI"].ToString();  if (Connectivity.NetworkAccess != NetworkAccess.Internet)  {  IsRunning = false;  IsEnabled = true;  await App.Current.MainPage.DisplayAlert(  "Error",  "Revise su conexión a Internet",  "Aceptar");  return;  }  Response response = await \_apiService.GetListAsync<CategoryResponse>(url, "api", "/Categories");  if (!response.IsSuccess)  {  await App.Current.MainPage.DisplayAlert(  "Error",  response.Message,  "Aceptar");  return;  }  List<CategoryResponse> list = (List<CategoryResponse>)response.Result;  Categories = new ObservableCollection<CategoryResponse>(list.OrderBy(t => t.Name));  }  private async void LoadStates()  {  string url = App.Current.Resources["UrlAPI"].ToString();  if (Connectivity.NetworkAccess != NetworkAccess.Internet)  {  IsRunning = false;  IsEnabled = true;  await App.Current.MainPage.DisplayAlert(  "Error",  "Revise su conexión a Internet",  "Aceptar");  return;  }  Response response = await \_apiService.GetListAsync<StateResponse>(url, "api", "/States");  if (!response.IsSuccess)  {  await App.Current.MainPage.DisplayAlert(  "Error",  response.Message,  "Aceptar");  return;  }  List<StateResponse> list = (List<StateResponse>)response.Result;  States = new ObservableCollection<StateResponse>(list.OrderBy(t => t.Id));  }  private async void Save()  {  if (Category == null)  {  await App.Current.MainPage.DisplayAlert("Error", "Debe seleccionar una Categoría.", "Aceptar");  return;  }  if (string.IsNullOrEmpty(Name))  {  await App.Current.MainPage.DisplayAlert("Error", "Debe ingresar el Nombre.", "Aceptar");  return;  }  if (string.IsNullOrEmpty(Description))  {  await App.Current.MainPage.DisplayAlert("Error", "Debe ingresar la Descripción.", "Aceptar");  return;  }  if (Latitude == 0 || Longitude == 0)  {  await App.Current.MainPage.DisplayAlert("Error", "Se necesitan las Coordenadas de Geolocalización.", "Aceptar");  return;  }  if (State == null)  {  await App.Current.MainPage.DisplayAlert("Error", "Debe seleccionar el Estado.", "Aceptar");  return;  }  if (\_file == null)  {  await App.Current.MainPage.DisplayAlert("Error", "Debe cargar la foto del Producto.", "Aceptar");  return;  }  IsRunning = true;  IsEnabled = false;  string url = App.Current.Resources["UrlAPI"].ToString();  if (Connectivity.NetworkAccess != NetworkAccess.Internet)  {  IsRunning = true;  IsEnabled = false;  await App.Current.MainPage.DisplayAlert(  "Error",  "Revise su conexión a Internet",  "Aceptar");  return;  }  byte[] ImageArray = null;  if (File != null)  {  ImageArray = \_filesHelper.ReadFully(File.GetStream());  File.Dispose();  }  TokenResponse token = JsonConvert.DeserializeObject<TokenResponse>(Settings.Token);  UserResponse user = token.User;  ProductRequest myproduct = new ProductRequest  {  Name = Name,  Description =Description,  Price=Price,  Category=Category,  Latitude = Latitude,  Longitude = Longitude,  PhotoArray = ImageArray,  State = State,  };  ResponseT<object> response = await \_apiService.PostAsync(  url,  "api",  "/Products",  myproduct,  "bearer",  token.Token);  if (!response.IsSuccess)  {  IsRunning = false;  IsEnabled = true;  await App.Current.MainPage.DisplayAlert(  "Error",  response.Message,  "Aceptar");  return;  }  IsRunning = false;  IsEnabled = true;  await App.Current.MainPage.DisplayAlert(  "Ok",  "Guardado con éxito!!",  "Aceptar");  ProductsPageViewModel productsPageViewModel = ProductsPageViewModel.GetInstance();  productsPageViewModel.LoadProductsAsync();  productsPageViewModel.RefreshList();  await \_navigationService.GoBackAsync();  }  private async void Cancel()  {  await \_navigationService.GoBackAsync();  }  private async void TakePhoto()  {  await CrossMedia.Current.Initialize();  \_file = await CrossMedia.Current.TakePhotoAsync(  new StoreCameraMediaOptions  {  Directory = "Sample",  Name = "test.jpg",  PhotoSize = PhotoSize.Small,  }  );  if (\_file != null)  {  ImageSource = ImageSource.FromStream(() =>  {  var stream = \_file.GetStream();  return stream;  });  }  IsRunning = false;  }  }  } |  |

# Editar Producto

## EditProductPage

|  |  |
| --- | --- |
| **EditProductPage** | **Comentarios** |
|  |  |

## EditProductPageViewModel

|  |  |
| --- | --- |
| **EditAddProductPageViewModel** | **Comentarios** |
|  |  |

|  |  |
| --- | --- |
| **CountryEntity** | **Comentarios** |
|  |  |