GenericApp

**INDICE**

1 Matriz de funcionalidad 4

2 Arquitectura 4

3 Repositorio en GitHub 5

4 Crear Solución y Proyectos Common, Web y Prism 5

4.1 Solución en blanco 5

4.2 Proyecto Common 6

4.3 Proyecto Web (Net Core) 7

4.4 Proyectos Prism 9

5 Diagrama Entidad Relación 11

6 Creación de la Base de Datos 11

6.1 Entities 12

6.2 DataContext 13

6.3 Cadena de conexión 14

7 CRUD para Countries 15

7.1 Controlador 15

7.2 Mejorar CRUD Countries 17

8 Entities Department, City y Team 22

8.1 DepartmentEntity 22

8.2 CityEntity 22

8.3 CountryEntity 22

8.4 TeamEntity 23

8.5 Actualizamos DataContext 23

8.6 Actualizar Base de Datos 24

9 ViewModels para manejar imágenes y/o combos 24

9.1 CategoryViewModel 24

9.2 ProductViewModel 24

9.3 AddProductImageViewModel 25

9.4 CountryViewModel 25

9.5 TeamViewModel 25

10 ImageHelper 26

11 CombosHelper 26

12 ConverterHelper 29

13 Maestro detalle MVC para Countries 31

13.1 Vista Index 31

13.2 Métodos Index y Details del controlador CountriesController: 31

13.3 Vista Details del controlador Countries: 32

13.4 Método AddDepartment 34

13.5 Método AddTeam 35

13.6 Método EditDepartment 37

13.7 Método EditTeam 38

13.8 Método DeleteDepartment 39

13.9 Método DeleteTeam 39

13.10 Borrado en cascada de Países que tengan Provincias y/o Equipos 40

13.11 Método DetailsDepartment 40

13.12 Método AddCity 41

13.13 Método EditCity 43

13.14 Método DeleteCity 44

14 Seeder 44

14.1 Clase SeedDb 44

14.2 Inyección de la Clase SeedDb 46

15 CRUD para Categories 47

15.1 Controlador 47

15.2 Vistas 49

16 CRUD para Products 51

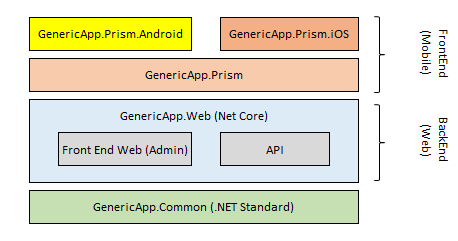
16.1 Controlador 51

16.2 Vistas 55

# 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

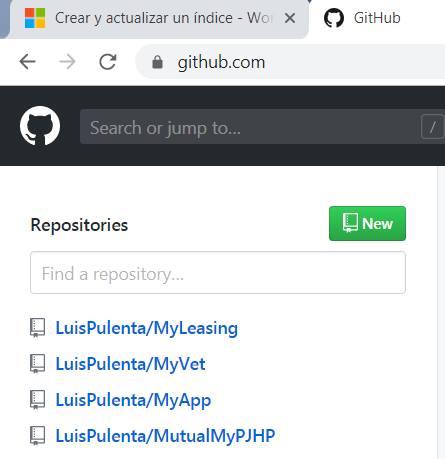
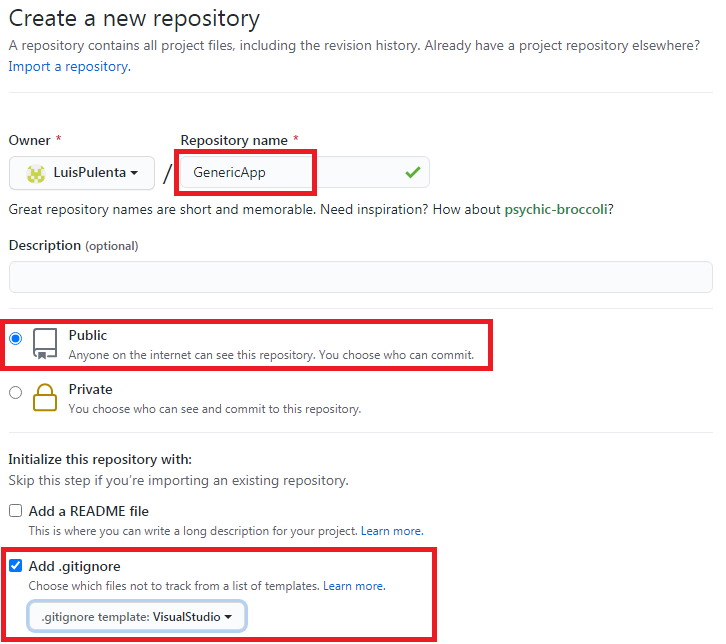
Empezamos haciendo el Repositorio en GitHub

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

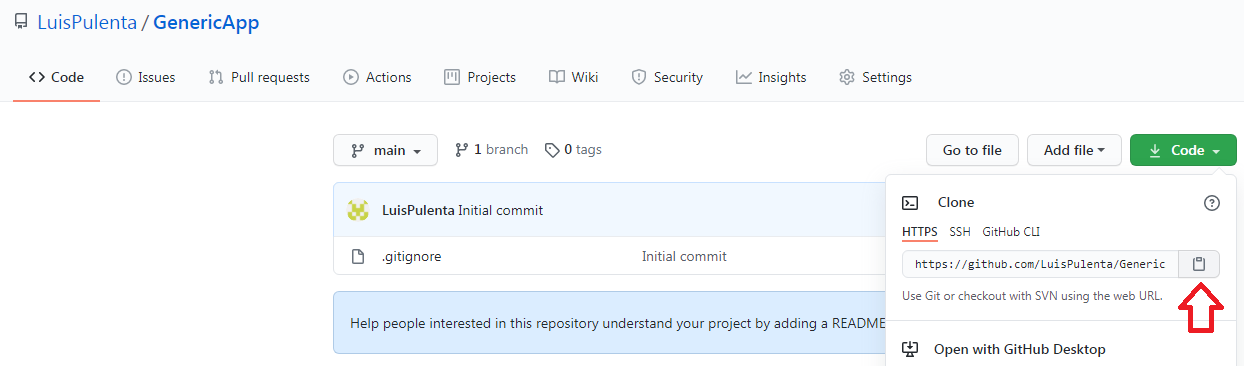
<https://github.com/> **Usuario:** LuisPulenta **Contraseña:** Talleres2306

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

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

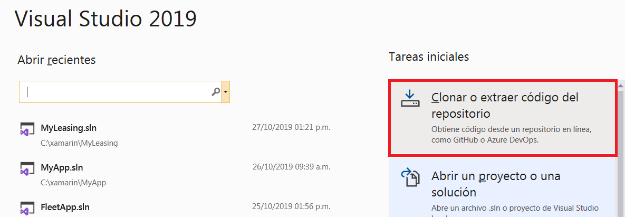
Luego copiamos la dirección para luego clonar en Visual Studio:



# Crear Solución y Proyectos Common, Web y Prism

## Solución en blanco

Abrimos Visual Studio y vamos a la opción Clonar

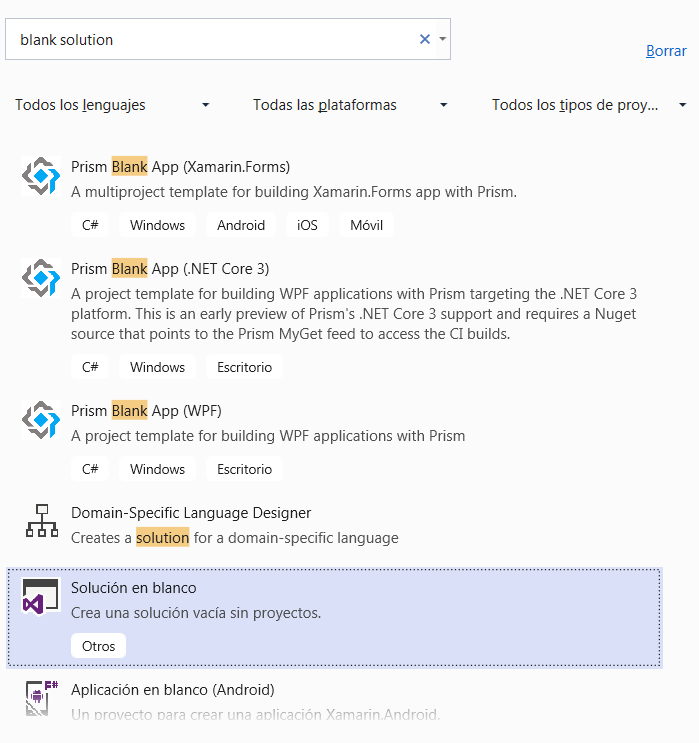
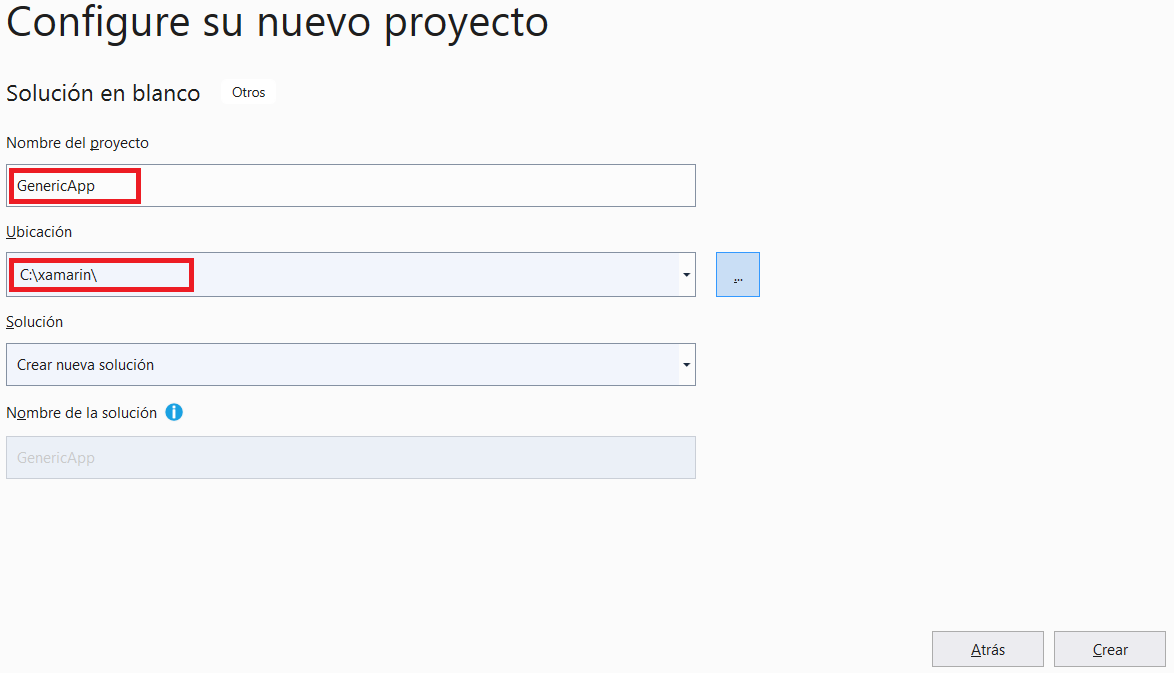




Que sea el mismo nombre del Repositorio. Debe ser una carpeta que no existe

Pegar

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

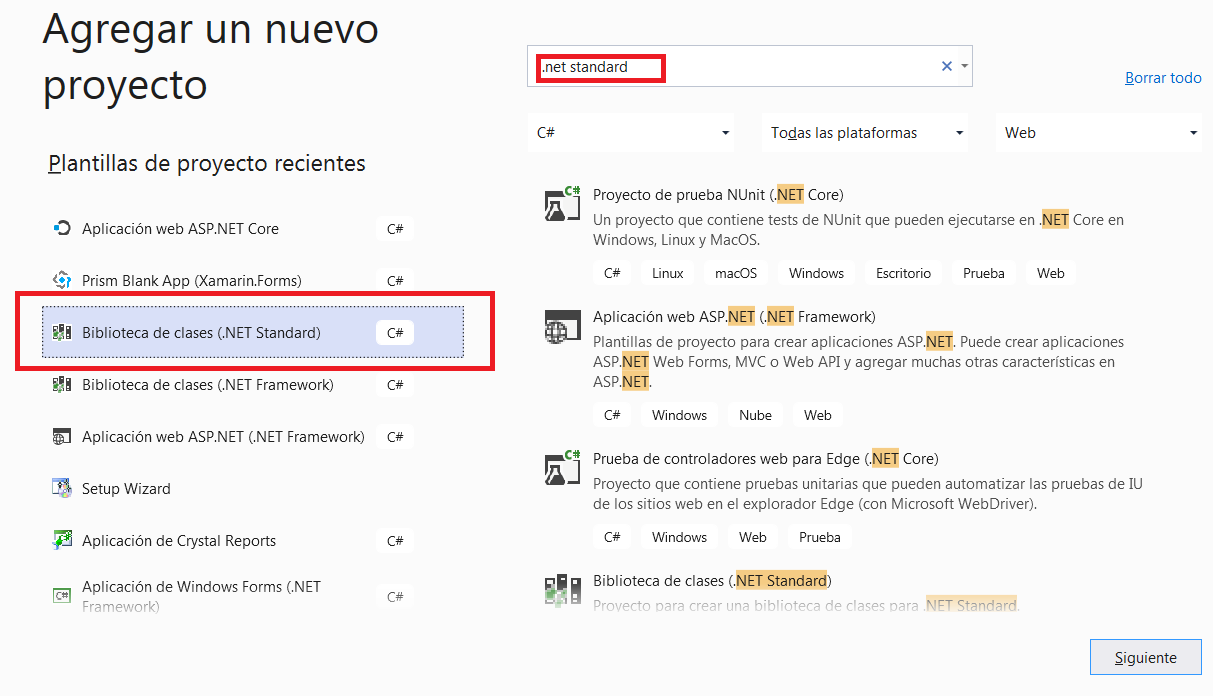
OJO!!! Debe ser la Carpeta CONTENEDORA

Que sea el mismo nombre del Repositorio

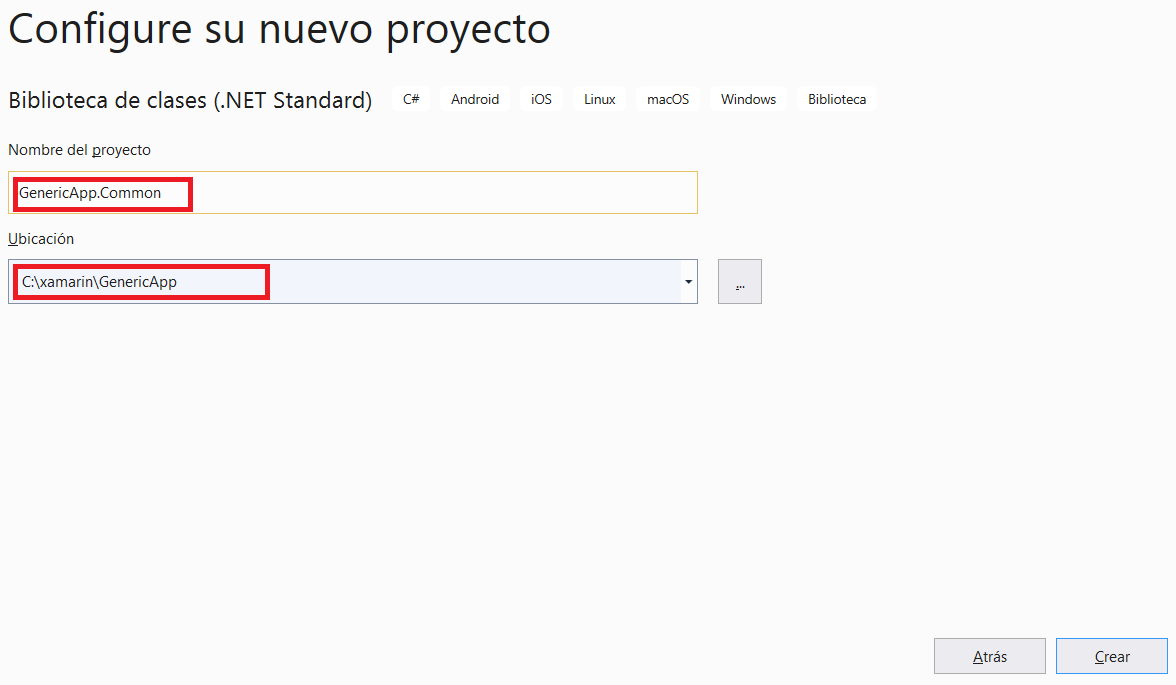
## Proyecto Common

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

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



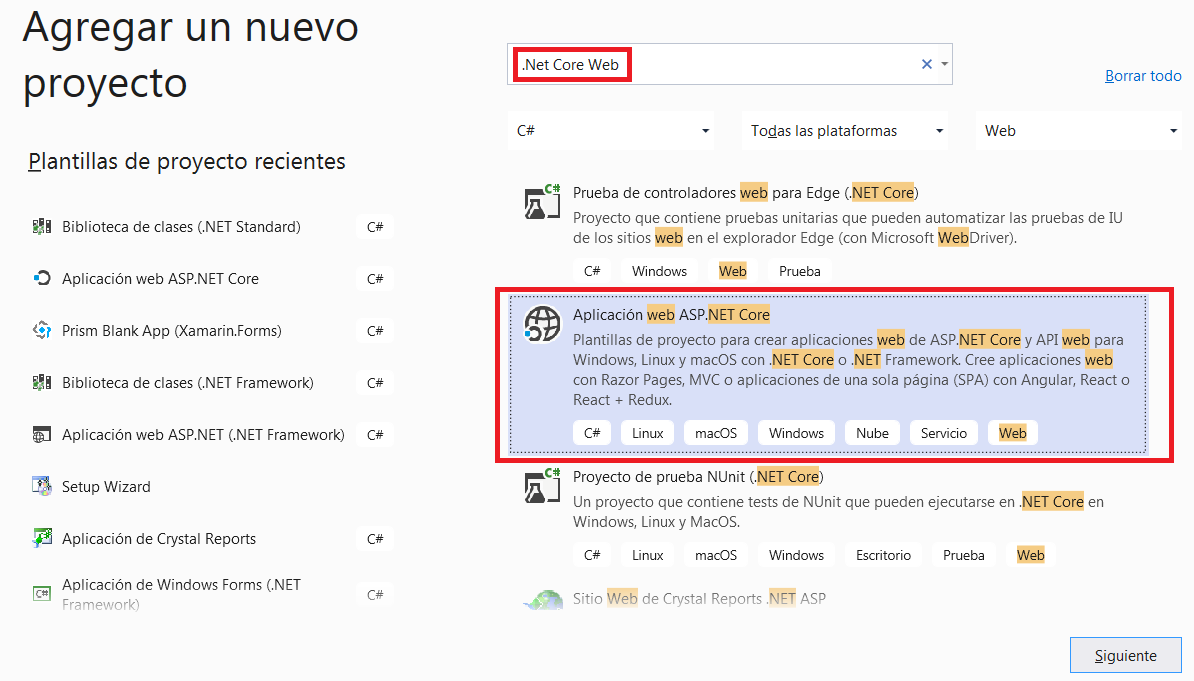
Y la llamamos **GenericApp.Common**

****

(La Clase Class1 que se crea la borramos)

## Proyecto Web (Net Core)

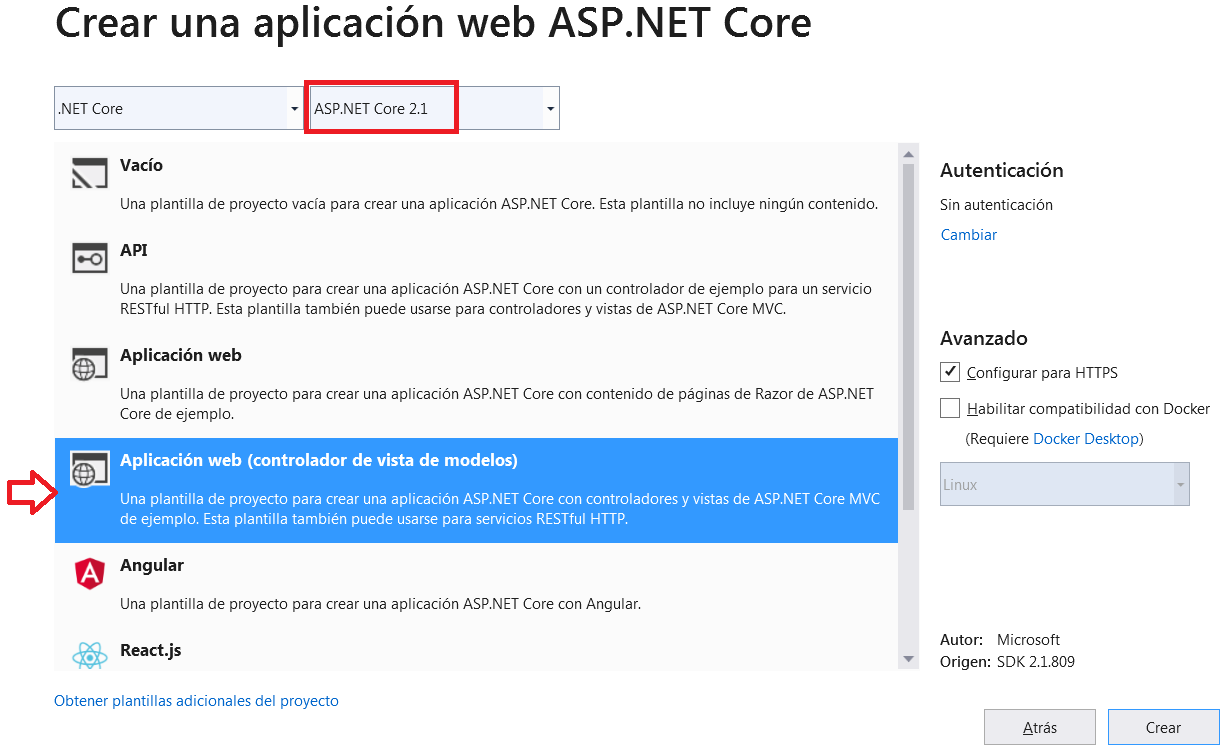
Volvemos a hacer clic derecho en la Solución y ahora ponemos **.Net Core Web** en la plantilla y elegimos **Aplicación Web.Net Core**



Como nombre le ponemos **GenericApp.Web**

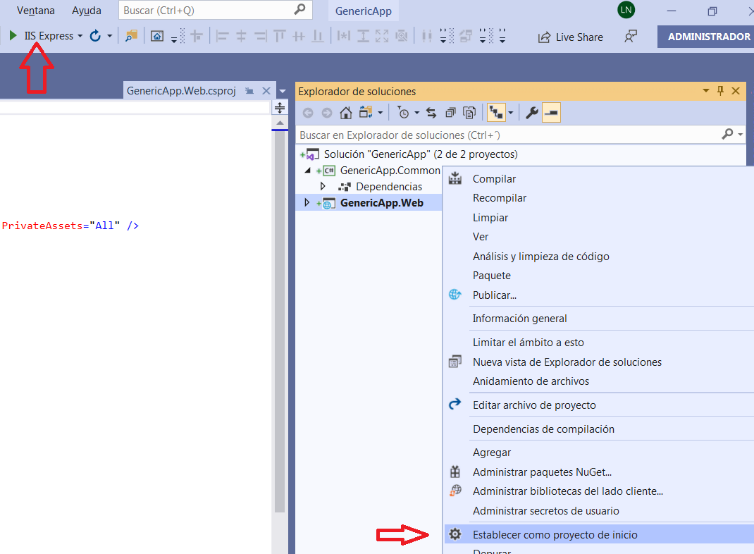
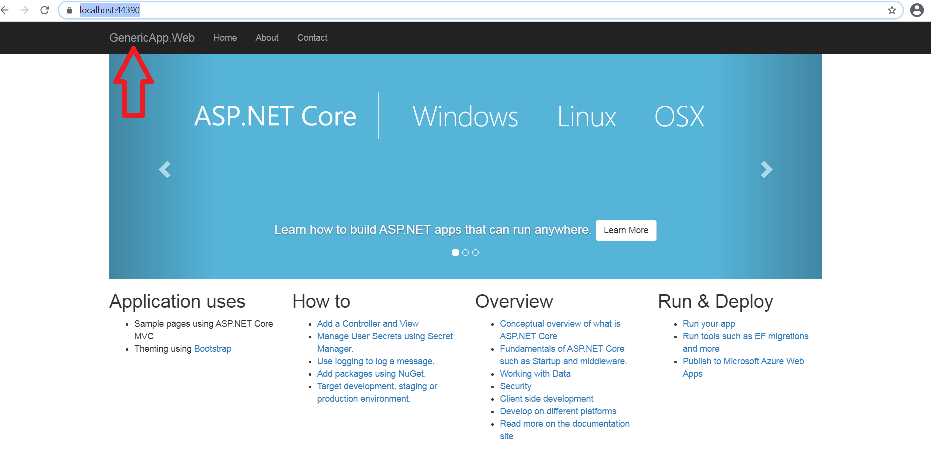


Debe ser Core 2.1 y elegimos la opción Aplicación web (controlador de vista de modelos)



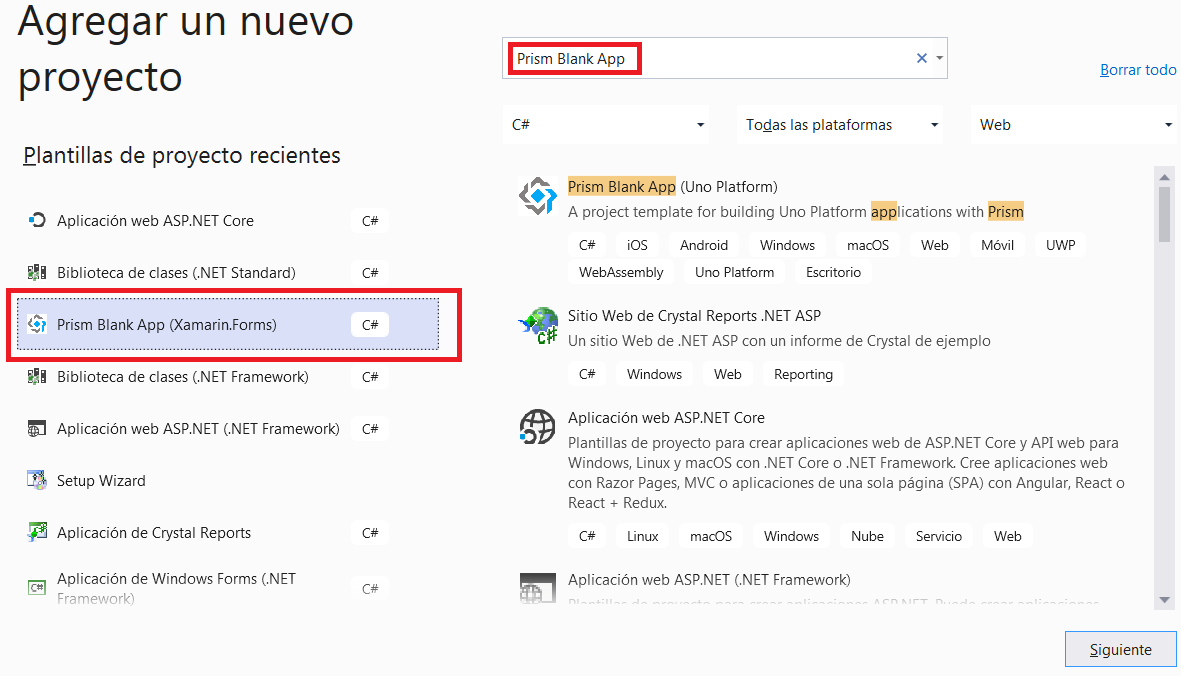
Este Proyecto ya debe arrrancar.

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

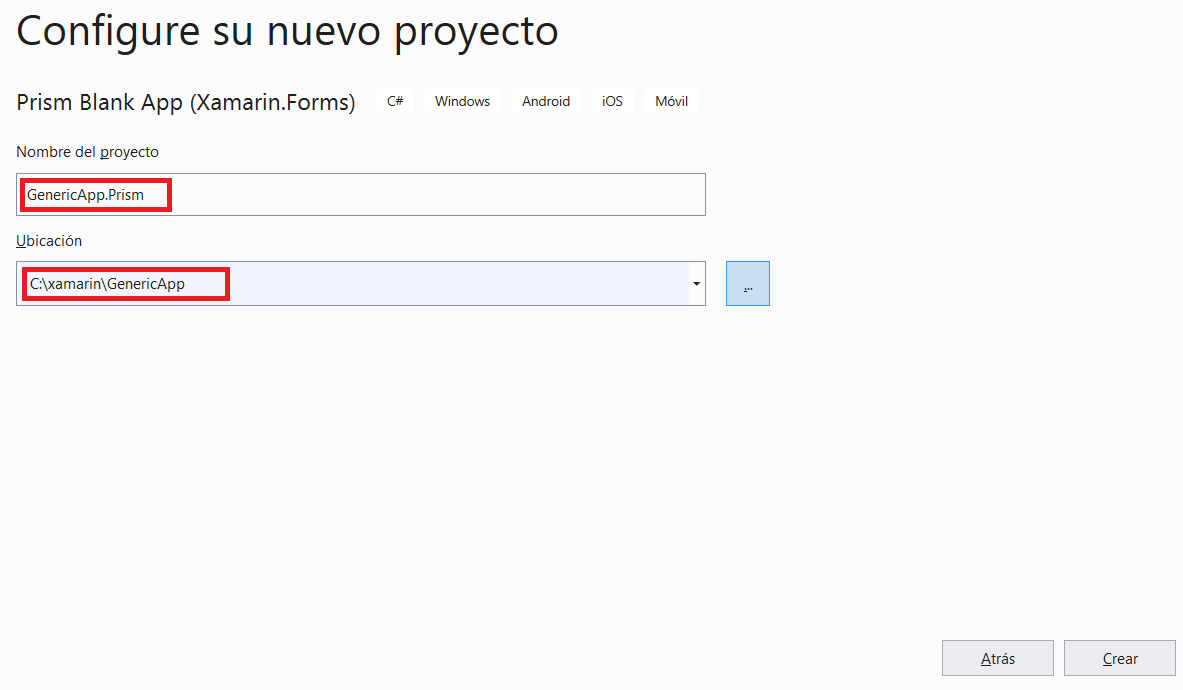
 

## Proyectos Prism

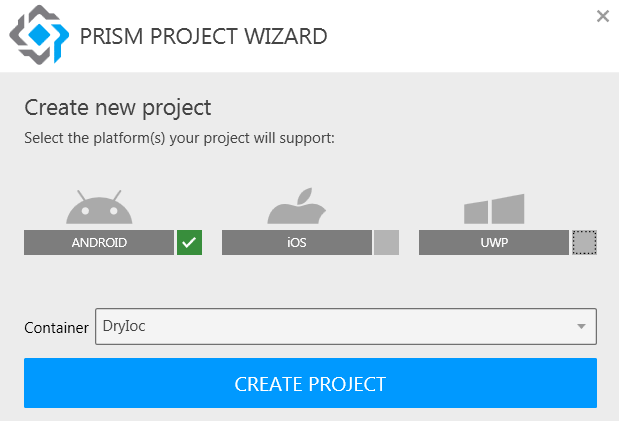
Volvemos a hacer clic derecho en la Solución y ahora ponemos **Prism Blank App** en la plantilla y elegimos **Prism Blank App (Xamarin.Forms)**

****

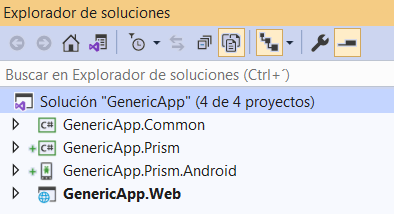
Como nombre le ponemos **GenericApp.Prism**



Elegimos **Container 🡪 DryIoc**



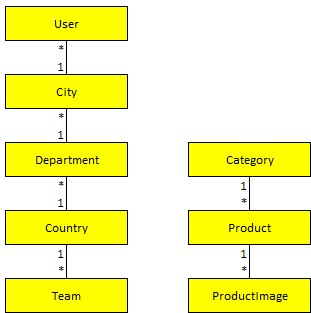
Una vez finalizado, así nos debe quedar la solución con todos sus Proyectos:



# Agregar Nuggets

|  |  |  |
| --- | --- | --- |
| **NUGGET** | **VERSION** | **AGREGAR A** |
| MailKit | 2.5.1 | Proyecto Web |
|  |  |  |
|  |  |  |

# 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 **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 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; }  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<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 |

## 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

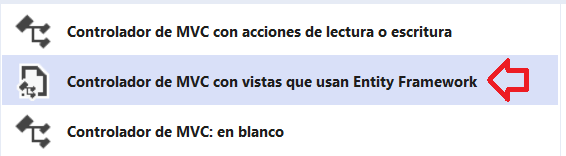
Verificamos en el SQL Server que la Base de Datos se haya creado.

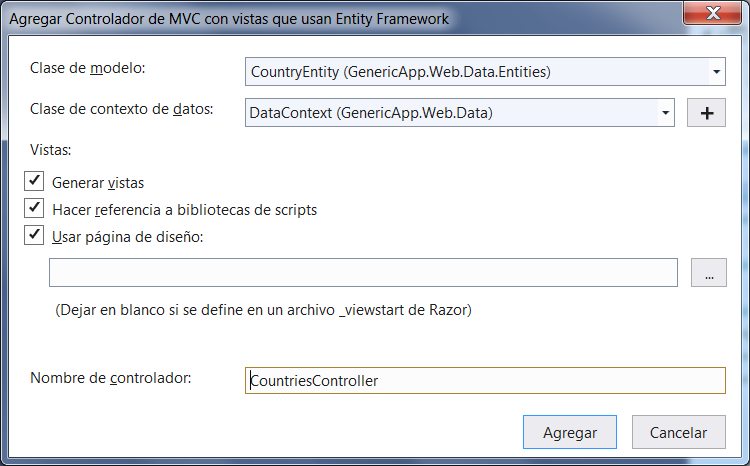
# CRUD para Countries

## 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 GenericApp.Web.Data;  using GenericApp.Web.Data.Entities;  using Microsoft.AspNetCore.Mvc;  using Microsoft.EntityFrameworkCore;  using System.Linq;  using System.Threading.Tasks;  namespace GenericApp.Web.Controllers  {  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.ToListAsync());  }  // GET: Countries/Details/5  public async Task<IActionResult> Details(int? id)  {  if (id == null)  {  return NotFound();  }  var countryEntity = await \_context.Countries  .FirstOrDefaultAsync(m => m.Id == id);  if (countryEntity == null)  {  return NotFound();  }  return View(countryEntity);  }  // GET: Countries/Create  public IActionResult Create()  {  return View();  }  // POST: Countries/Create  [HttpPost]  [ValidateAntiForgeryToken]  public async Task<IActionResult> Create([Bind("Id,Name,FlagImagePath")] CountryEntity countryEntity)  {  if (ModelState.IsValid)  {  \_context.Add(countryEntity);  await \_context.SaveChangesAsync();  return RedirectToAction(nameof(Index));  }  return View(countryEntity);  }  // GET: Countries/Edit/5  public async Task<IActionResult> Edit(int? id)  {  if (id == null)  {  return NotFound();  }  var countryEntity = await \_context.Countries.FindAsync(id);  if (countryEntity == null)  {  return NotFound();  }  return View(countryEntity);  }  // POST: Countries/Edit/5  [HttpPost]  [ValidateAntiForgeryToken]  public async Task<IActionResult> Edit(int id, [Bind("Id,Name,FlagImagePath")] CountryEntity countryEntity)  {  if (id != countryEntity.Id)  {  return NotFound();  }  if (ModelState.IsValid)  {  try  {  \_context.Update(countryEntity);  await \_context.SaveChangesAsync();  }  catch (DbUpdateConcurrencyException)  {  if (!CountryEntityExists(countryEntity.Id))  {  return NotFound();  }  else  {  throw;  }  }  return RedirectToAction(nameof(Index));  }  return View(countryEntity);  }  // GET: Countries/Delete/5  public async Task<IActionResult> Delete(int? id)  {  if (id == null)  {  return NotFound();  }  var countryEntity = await \_context.Countries  .FirstOrDefaultAsync(m => m.Id == id);  if (countryEntity == null)  {  return NotFound();  }  return View(countryEntity);  }  // POST: Countries/Delete/5  [HttpPost, ActionName("Delete")]  [ValidateAntiForgeryToken]  public async Task<IActionResult> DeleteConfirmed(int id)  {  var countryEntity = await \_context.Countries.FindAsync(id);  \_context.Countries.Remove(countryEntity);  await \_context.SaveChangesAsync();  return RedirectToAction(nameof(Index));  }  private bool CountryEntityExists(int id)  {  return \_context.Countries.Any(e => e.Id == 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) |

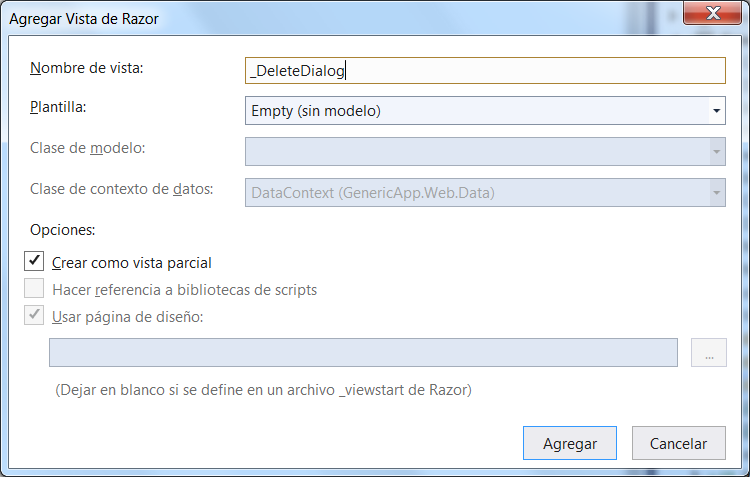
Modificamos el menú para poder probar lo que llevamos.

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; }  public IFormFile ImageFile { get; set; }  public IEnumerable<SelectListItem> Categories { 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; }  }  } |  |

# ImageHelper

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

Dentro creamos la Interfaz **IimageHelper**

|  |  |
| --- | --- |
| **IimageHelper** | **Comentarios** |
| using System.Threading.Tasks;  using Microsoft.AspNetCore.Http;  namespace GenericApp.Web.Helpers  {  public interface IImageHelper  {  Task<string> UploadImageAsync(IFormFile 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}";  }  }  } |  |

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();  }  } |  |

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;  }  }  } |  |

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.Web.Data.Entities;  using GenericApp.Web.Models;  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);  }  } |  |

Creamos la implementación **ConverterHelper**

|  |  |
| --- | --- |
| **ConverterHelper** | **Comentarios** |
| using GenericApp.Web.Data;  using GenericApp.Web.Data.Entities;  using GenericApp.Web.Models;  using System.Globalization;  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,  Name = model.Name,  Price = ToPrice(model.PriceString),  ProductImages = model.ProductImages  };  }  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,  Name = product.Name,  PriceString = $"{product.Price}",  ProductImages = product.ProductImages,  };  }  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,  };  }  }  } |  |

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.Mvc;  using Microsoft.EntityFrameworkCore;  using System;  using System.Threading.Tasks;  //using Vereyon.Web;  namespace GenericApp.Web.Controllers  {  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;  namespace GenericApp.Web.Controllers  {  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(p => p.Category)  .Include(p => p.ProductImages)  .ToListAsync());  }  public IActionResult Create()  {  ProductViewModel model = new ProductViewModel  {  Categories = \_combosHelper.GetComboCategories(),  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();  return View(model);  }  public async Task<IActionResult> Edit(int? id)  {  if (id == null)  {  return NotFound();  }  ProductEntity product = await \_context.Products  .Include(p => p.Category)  .Include(p => p.ProductImages)  .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(c => c.ProductImages)  .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 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>  <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">  <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>  </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 UserEntity

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

|  |  |
| --- | --- |
| **UserEntity** | **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, UserType.User.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(),  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">New 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">Confirm</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");}  } |  |

## 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> |  |

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