GenericApp

**INDICE**

1 Matriz de funcionalidad 3

2 Arquitectura 3

3 Repositorio en GitHub 3

4 Crear Solución y Proyectos Common, Web y Prism 4

4.1 Solución en blanco 4

4.2 Proyecto Common 5

4.3 Proyecto Web (Net Core) 6

4.4 Proyectos Prism 8

5 Diagrama Entidad Relación 10

6 Creación de la Base de Datos 10

6.1 Entities 11

6.2 DataContext 11

6.3 Cadena de conexión 11

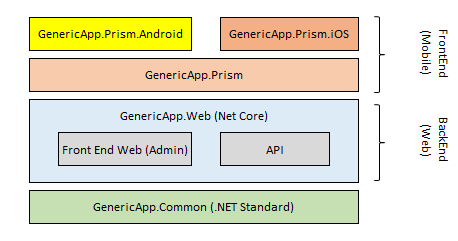
7 CRUD para Countries 12

7.1 Controlador 12

# 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

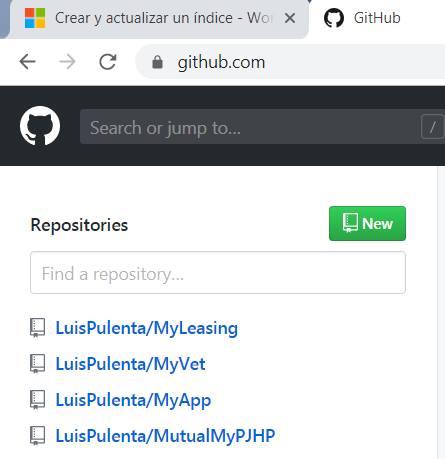
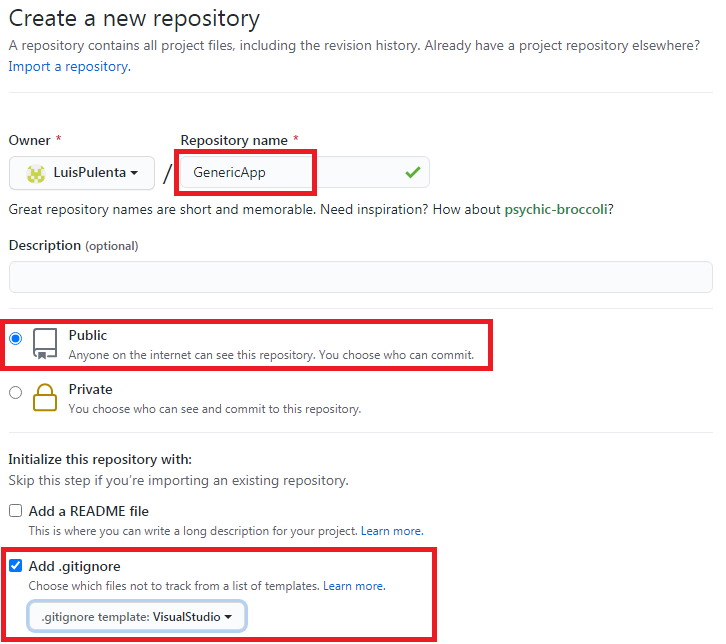
Empezamos haciendo el Repositorio en GitHub

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

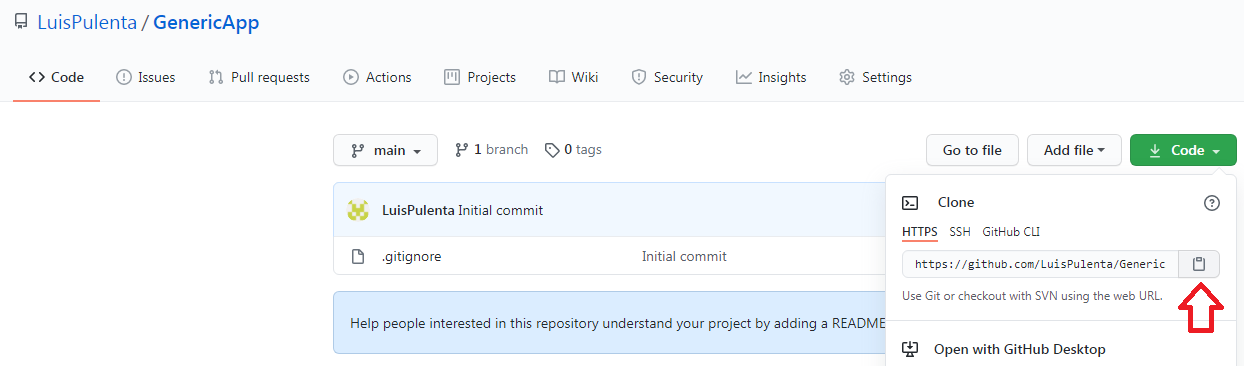
<https://github.com/> **Usuario:** LuisPulenta **Contraseña:** Talleres2306

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

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

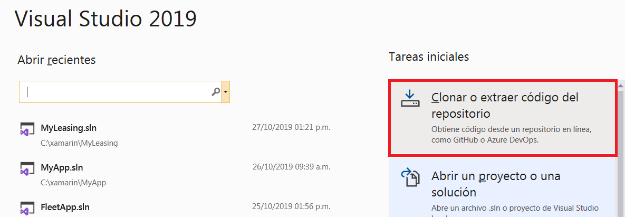
Luego copiamos la dirección para luego clonar en Visual Studio:



# Crear Solución y Proyectos Common, Web y Prism

## Solución en blanco

Abrimos Visual Studio y vamos a la opción Clonar

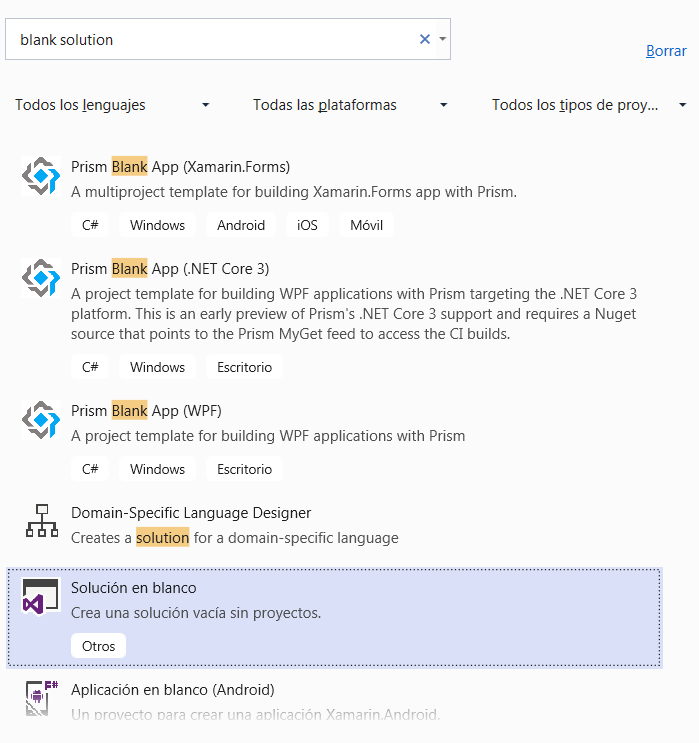
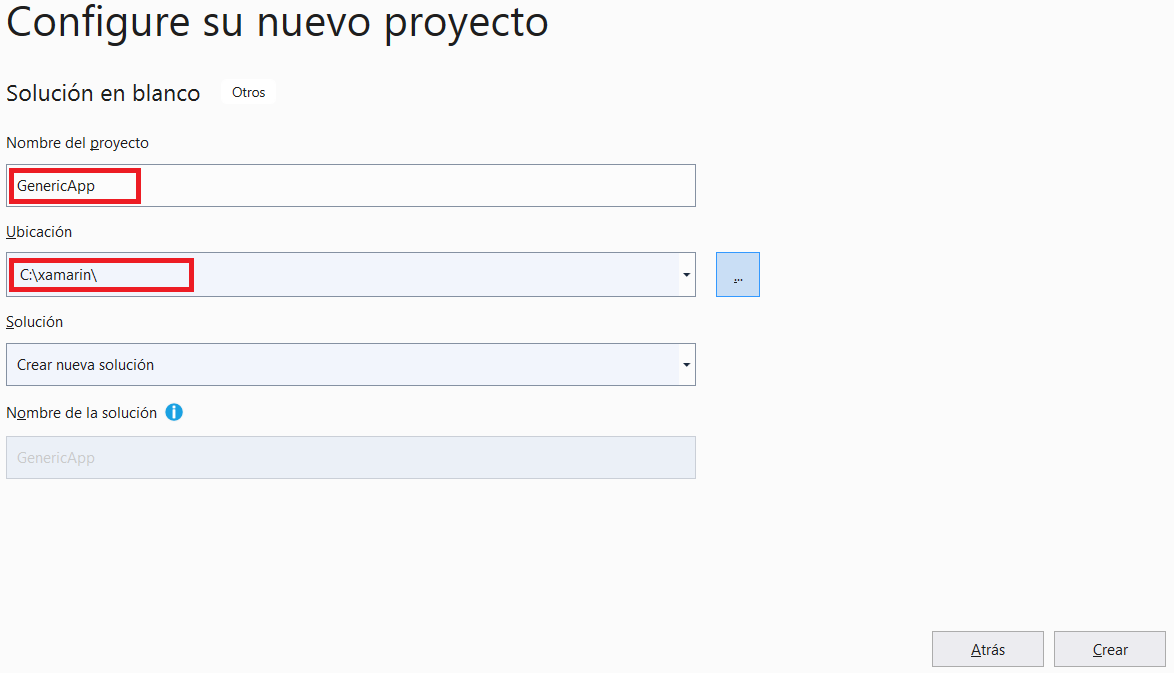




Que sea el mismo nombre del Repositorio. Debe ser una carpeta que no existe

Pegar

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

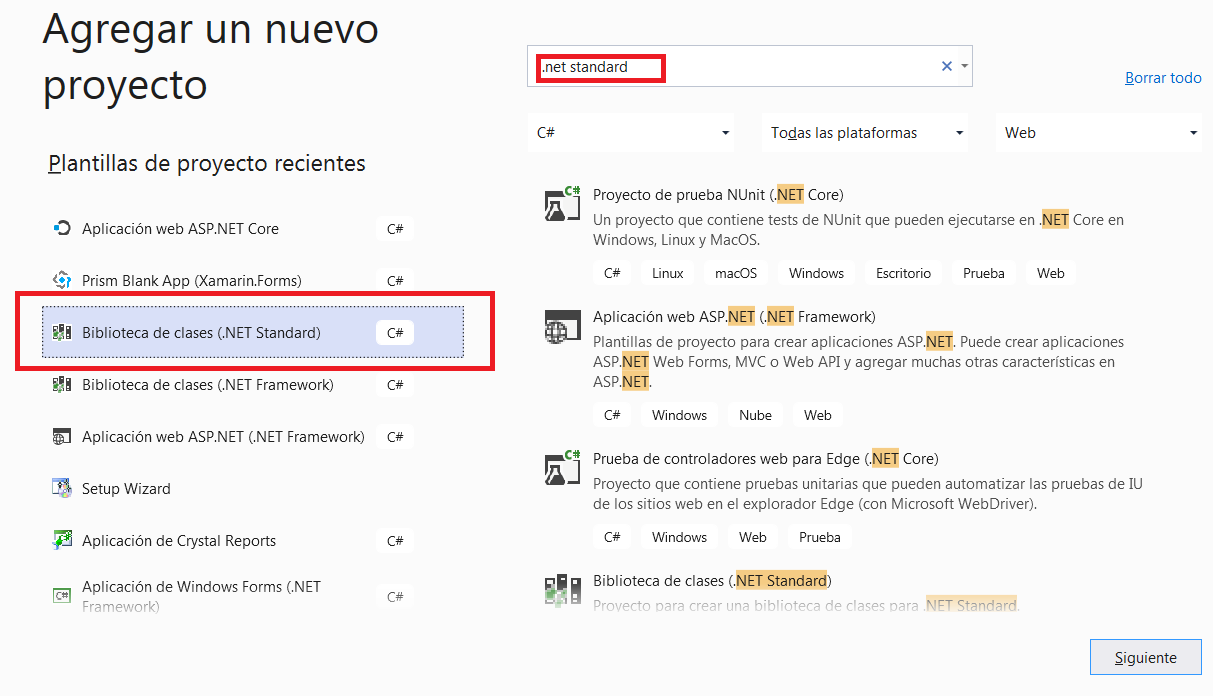
OJO!!! Debe ser la Carpeta CONTENEDORA

Que sea el mismo nombre del Repositorio

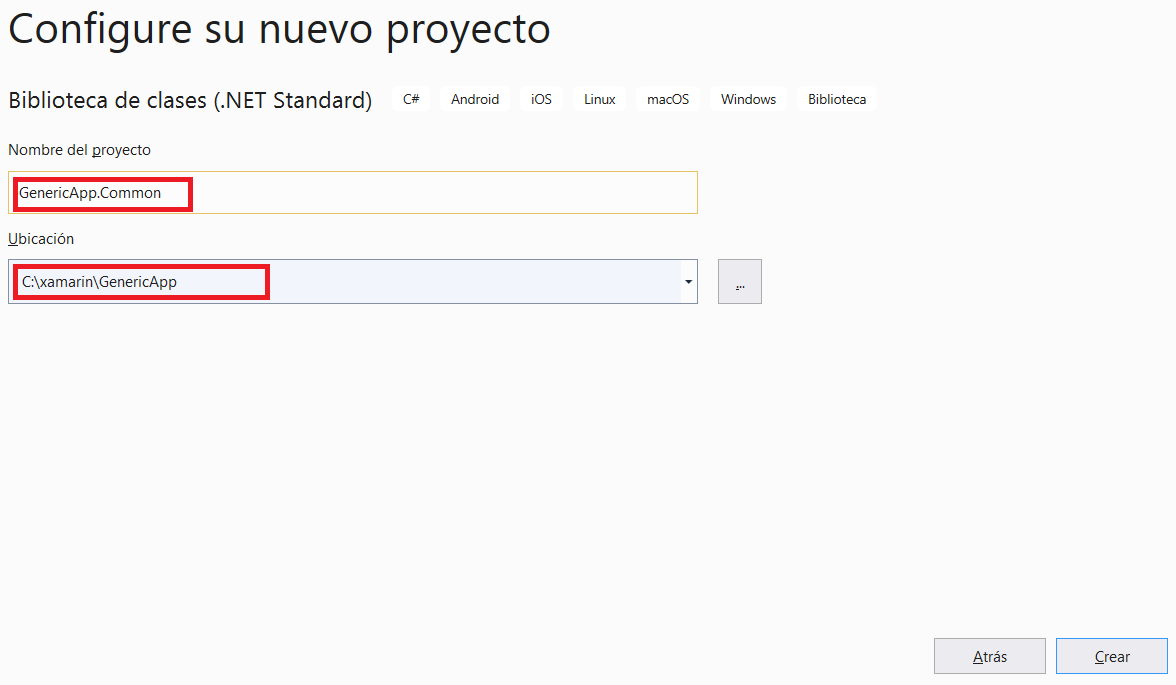
## Proyecto Common

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

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



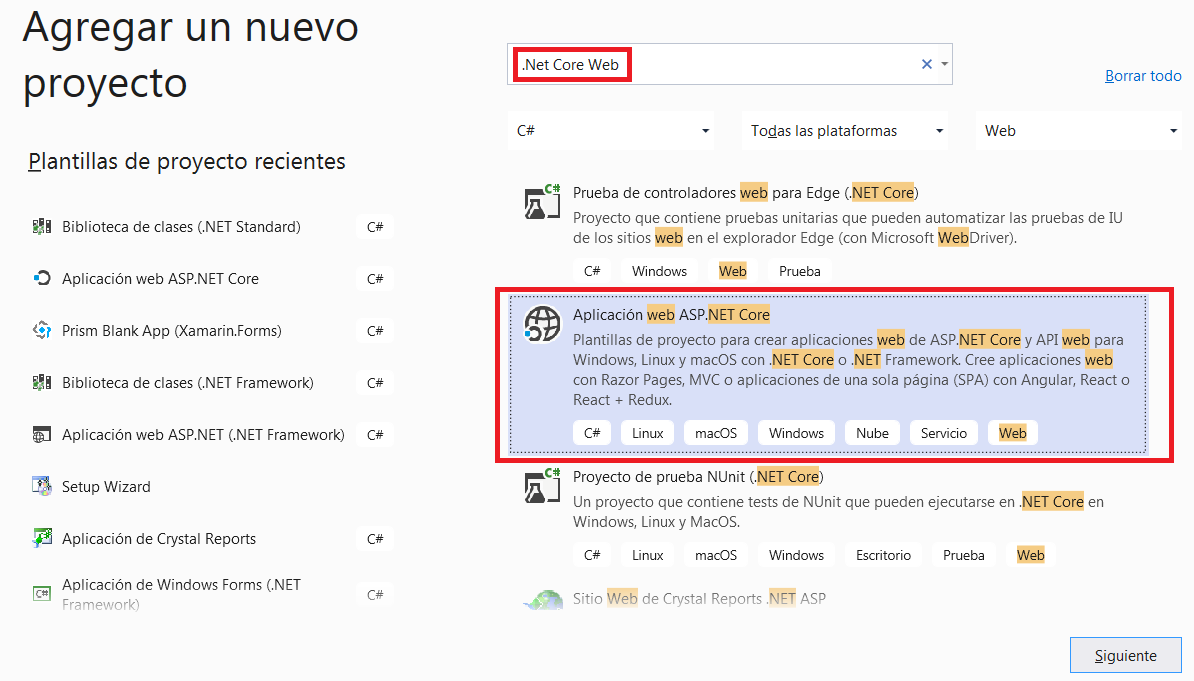
Y la llamamos **GenericApp.Common**

****

(La Clase Class1 que se crea la borramos)

## Proyecto Web (Net Core)

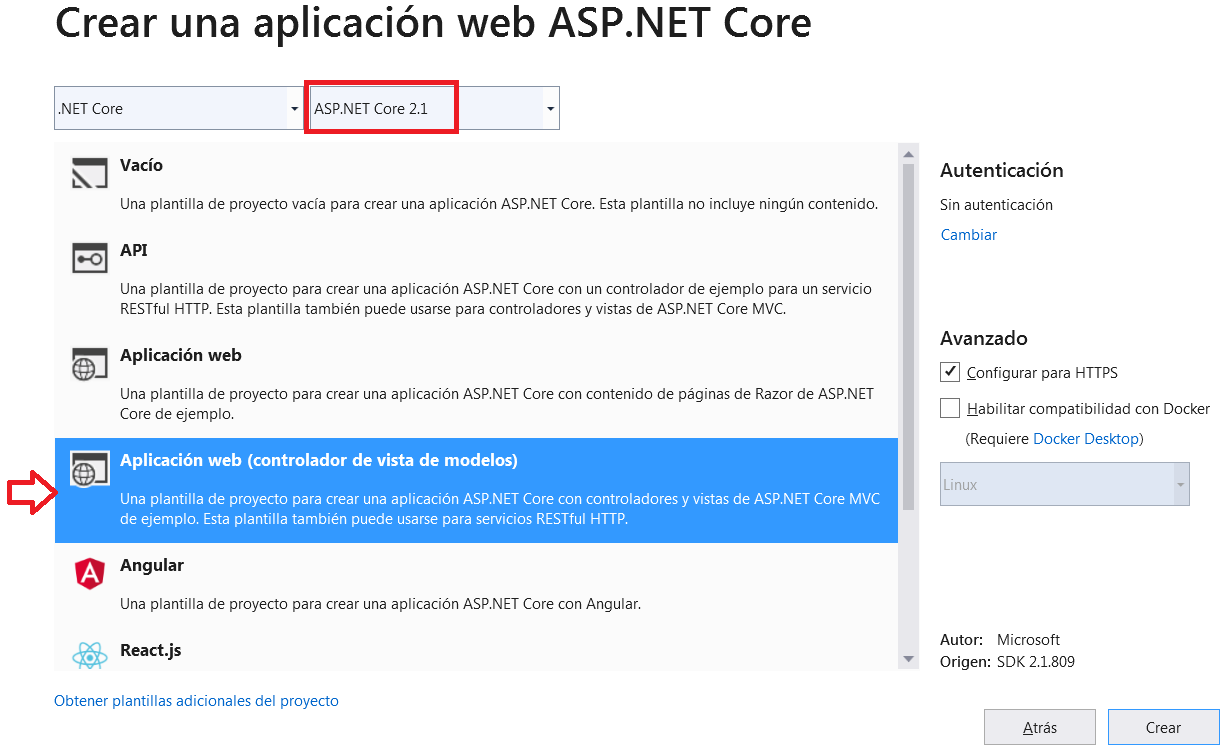
Volvemos a hacer clic derecho en la Solución y ahora ponemos **.Net Core Web** en la plantilla y elegimos **Aplicación Web.Net Core**



Como nombre le ponemos **GenericApp.Web**

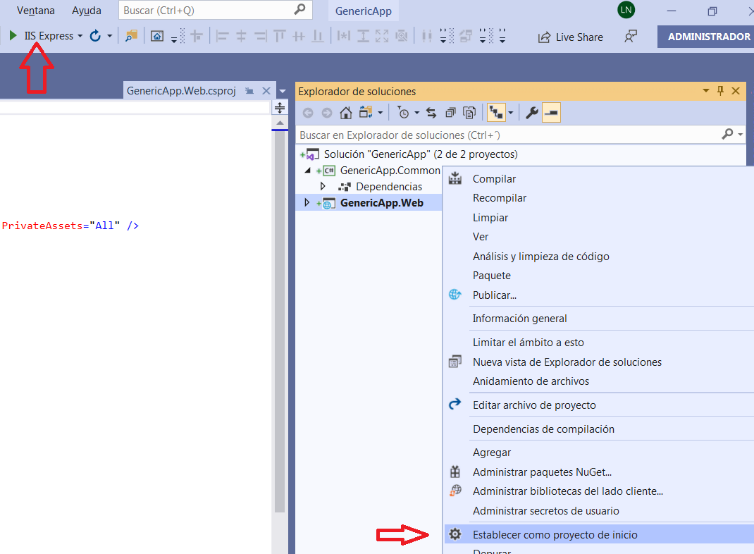
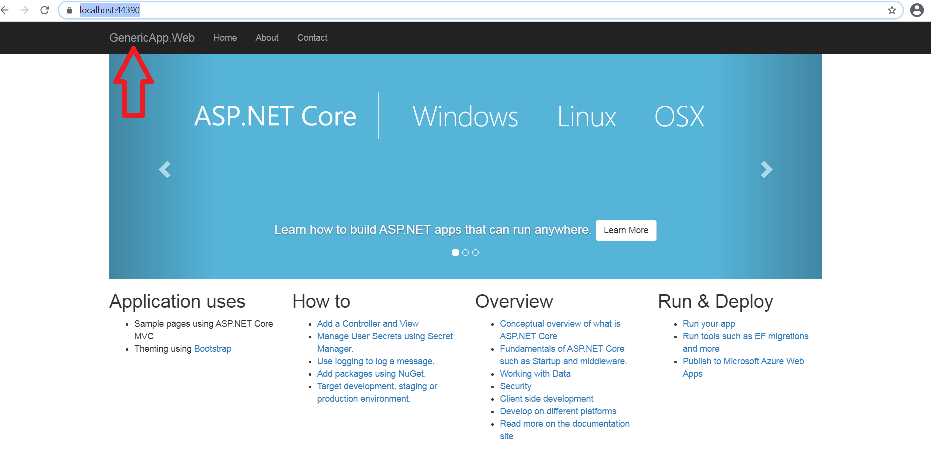


Debe ser Core 2.1 y elegimos la opción Aplicación web (controlador de vista de modelos)



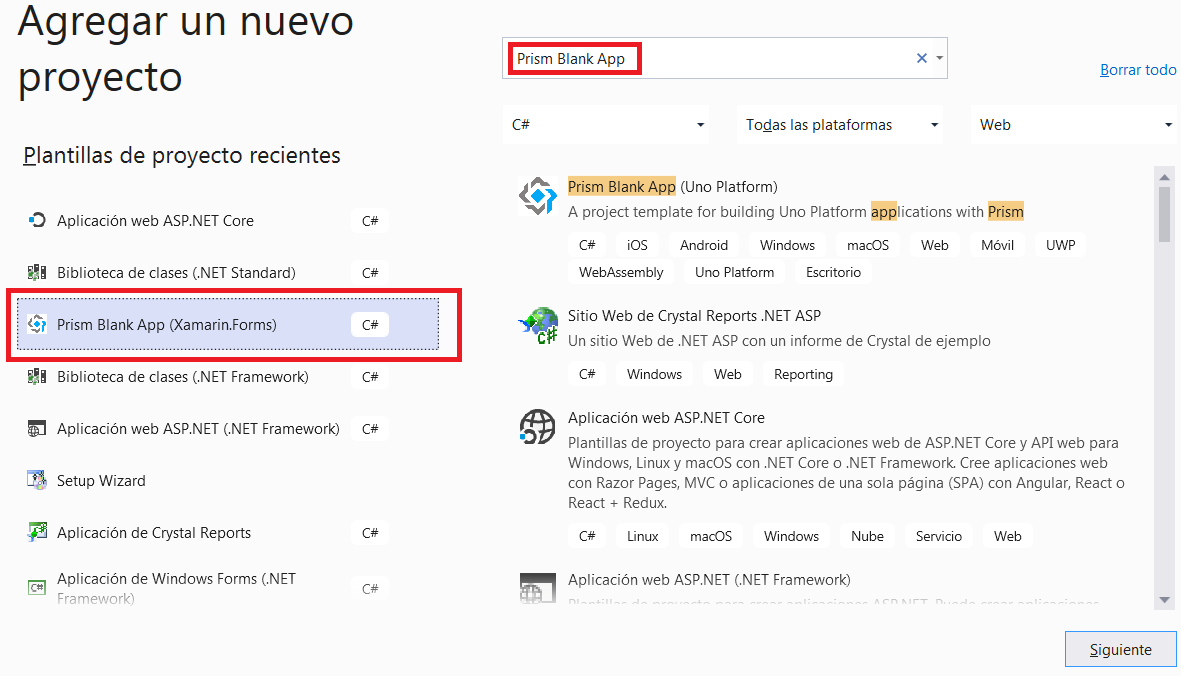
Este Proyecto ya debe arrrancar.

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

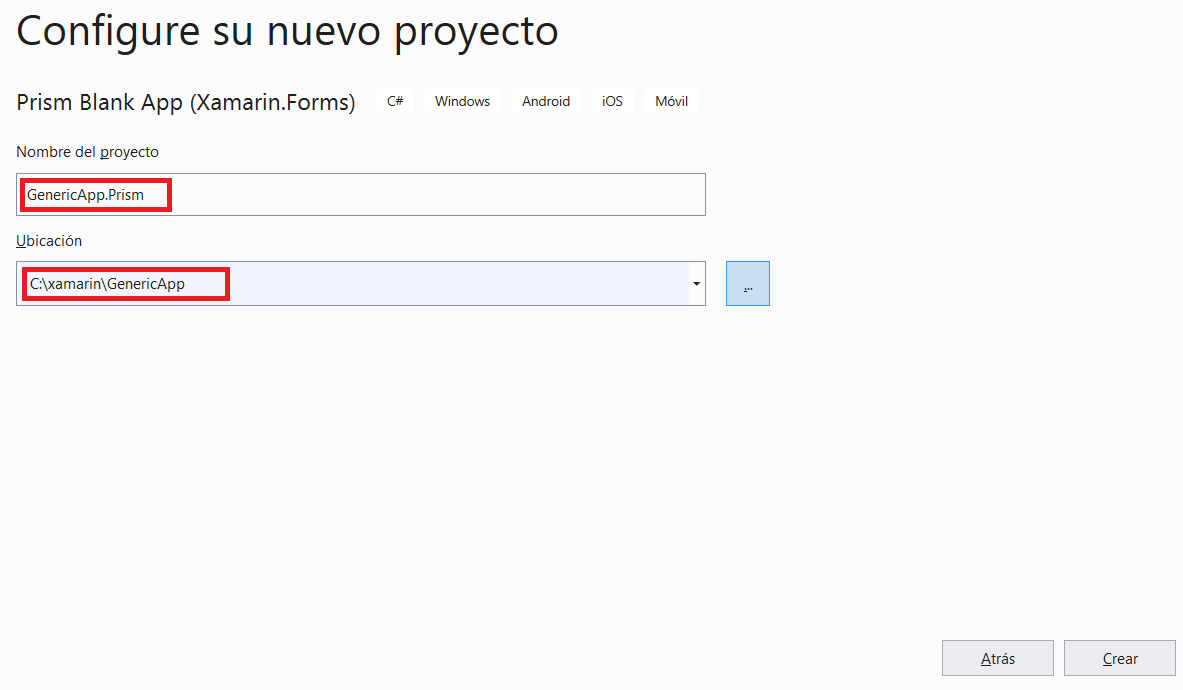
 

## Proyectos Prism

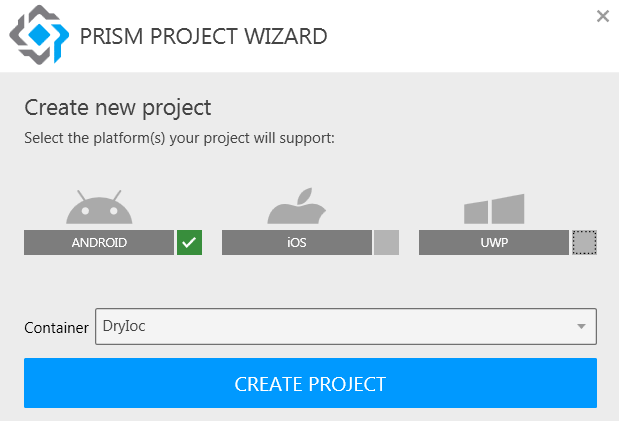
Volvemos a hacer clic derecho en la Solución y ahora ponemos **Prism Blank App** en la plantilla y elegimos **Prism Blank App (Xamarin.Forms)**

****

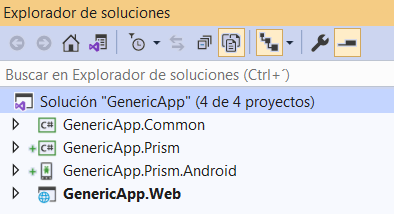
Como nombre le ponemos **GenericApp.Prism**



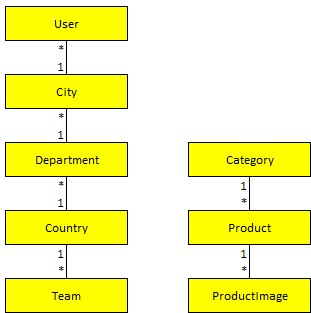
Elegimos **Container 🡪 DryIoc**



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



# 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

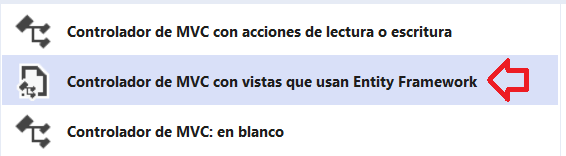
Verificamos en el SQL Server que la Base de Datos se haya creado.

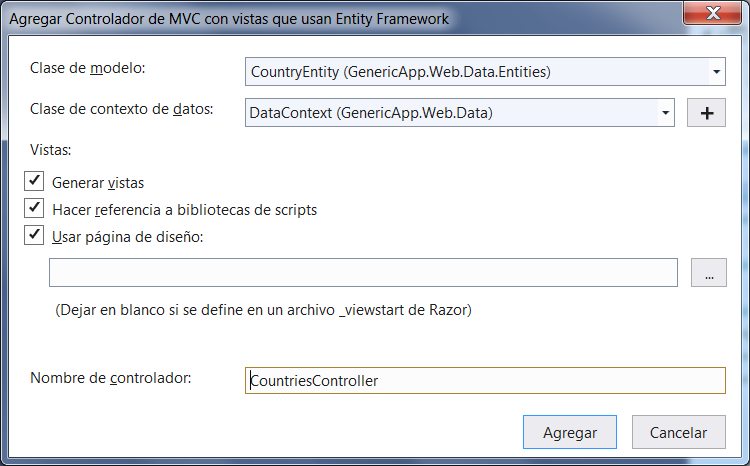
# CRUD para Countries

## 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) |

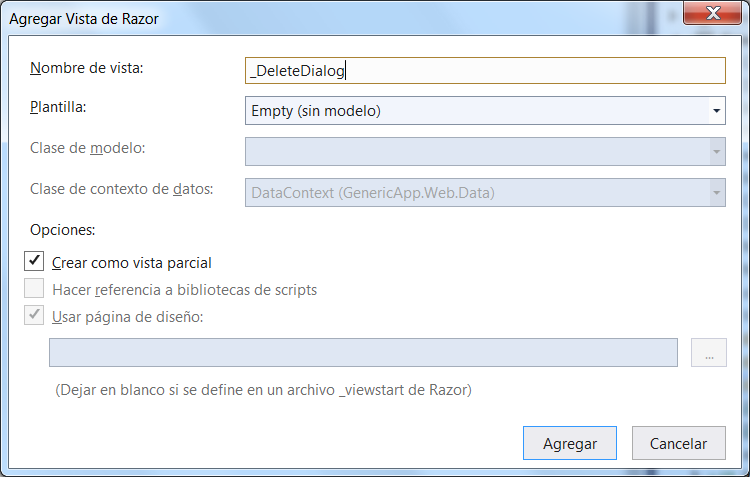
Modificamos el menú para poder probar lo que llevamos.

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";  }  <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</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></th>  </tr>  </thead>  <tbody>  @foreach (var item in Model)  {  <tr>  <td>  @Html.DisplayFor(modelItem => item.Name)  </td>  <td>  <a **asp-action**="Edit" **asp-route-id**="@item.Id" class="btn btn-warning"><i class="glyphicon glyphicon-pencil"></i></a>  <a **asp-action**="Details" **asp-route-id**="@item.Id" class="btn btn-info"><i class="glyphicon glyphicon-align-justify"></i></a>  <button data-id="@item.Id" class="btn btn-danger deleteItem" data-toggle="modal" data-target="#deleteDialog"><i class="glyphicon glyphicon-trash"></i></button>  </td>  </tr>  }  </tbody>  </table>  </div>  </div>  </div>  </div>  <**partial** **name**="\_DeleteDialog" />  @section Scripts {  @{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}  <script src="//cdn.datatables.net/1.10.19/js/jquery.dataTables.min.js"></script>  <script src="/js/deleteDialog.js"></script>  <script type="text/javascript">  $(document).ready(function () {  $('#MyTable').DataTable();  // Delete item  sc\_deleteDialog.openModal('deleteItem', true, 'btnYesDelete', '/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.Data.Entities.CountryEntity  @{  ViewData["Title"] = "Create";  }  <h2>Agregar nuevo País</h2>  <h4></h4>  <hr />  <div class="row">  <div class="col-md-4">  <form **asp-action**="Create">  <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; }  }  } |  |

# 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);  }  } |  |

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;  }  }  } |  |

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);  }  } |  |

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  };  }  }  } |  |

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);  } |  |

# 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>Detalle de un País</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.DepartmentsNumber)  </dt>  <dd>  @Html.DisplayFor(model => model.DepartmentsNumber)  </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></th>  </tr>  </thead>  <tbody>  @foreach (var item in Model.Teams)  {  <tr>  <td>  @Html.DisplayFor(modelItem => item.Name)  </td>  <td>  <a **asp-action**="EditTeam" **asp-route-id**="@item.Id" class="btn btn-warning"><i class="glyphicon glyphicon-pencil"></i></a>  <a **asp-action**="DetailsTeam" **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 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();  }  TeamEntity model = new TeamEntity { IdCountry = country.Id };  return View(model);  }  [HttpPost]  [ValidateAntiForgeryToken]  public async Task<IActionResult> AddTeam(TeamEntity team)  {  if (ModelState.IsValid)  {  CountryEntity country = await \_context.Countries  .Include(c => c.Teams)  .FirstOrDefaultAsync(c => c.Id == team.IdCountry);  if (country == null)  {  return NotFound();  }  try  {  team.Id = 0;  country.Teams.Add(team);  \_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(team);  } |  |

Adicionamos la vista parcial **\_Team**:

|  |  |
| --- | --- |
| **\_Team** | **Comentarios** |
| @model GenericApp.Web.Data.Entities.TeamEntity  <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 **AddTeam**

|  |  |
| --- | --- |
| **AddTeam** | **Comentarios** |
| @model GenericApp.Web.Data.Entities.TeamEntity  @{  ViewData["Title"] = "Add Team";  }  <h2>Agregar nuevo Equipo</h2>  <h4></h4>  <hr />  <div class="row">  <div class="col-md-4">  <form **asp-action**="AddTeam">  <div **asp-validation-summary**="ModelOnly" class="text-danger"></div>  <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.IdCountry" 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;  return View(team);  }  [HttpPost]  [ValidateAntiForgeryToken]  public async Task<IActionResult> EditTeam(TeamEntity team)  {  if (ModelState.IsValid)  {  try  {  \_context.Update(team);  await \_context.SaveChangesAsync();  return RedirectToAction($"{nameof(Details)}/{team.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(team);  } |  |

Adicionamos la vista **EditTeam**:

|  |  |
| --- | --- |
| **EditTeam** | **Comentarios** |
| @model GenericApp.Web.Data.Entities.TeamEntity  @{  ViewData["Title"] = "Edit";  }  <h2>Editar Equipo</h2>  <h4></h4>  <hr />  <div class="row">  <div class="col-md-4">  <form **asp-action**="EditTeam">  <div **asp-validation-summary**="ModelOnly" class="text-danger"></div>  <input **type**="hidden" **asp-for**="Id" />  <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.IdCountry" 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 Microsoft.AspNetCore.Mvc;  using Microsoft.EntityFrameworkCore;  using System.Threading.Tasks;  namespace GenericApp.Web.Controllers  {  public class CategoriesController : Controller  {  private readonly DataContext \_context;  public CategoriesController(DataContext context)  {  \_context = context;  }  public async Task<IActionResult> Index()  {  return View(await \_context.Categories.ToListAsync());  }  }  } |  |

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>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">Editar</a>  <button data-id="@item.Id" class="btn btn-danger deleteItem" data-toggle="modal" data-target="#deleteDialog">Borrar</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>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";  }  <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> Add New</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:100px;height:100px;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>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:200px;height:200px;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 **AddProduct**

|  |  |
| --- | --- |
| **AddProduct** | **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");}  } |  |

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