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

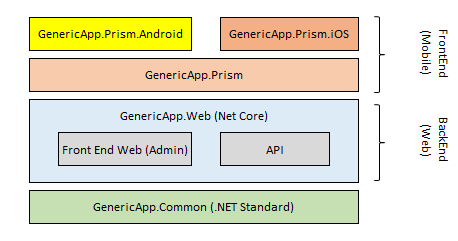
7 CRUD para Countries 12

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

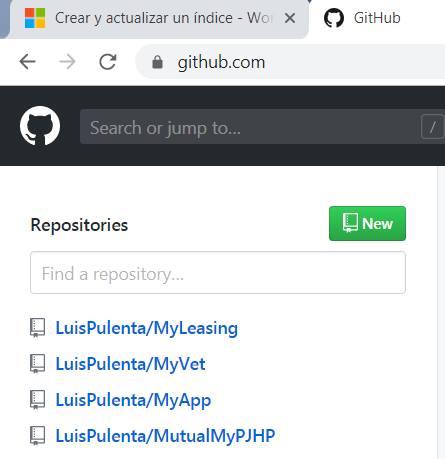
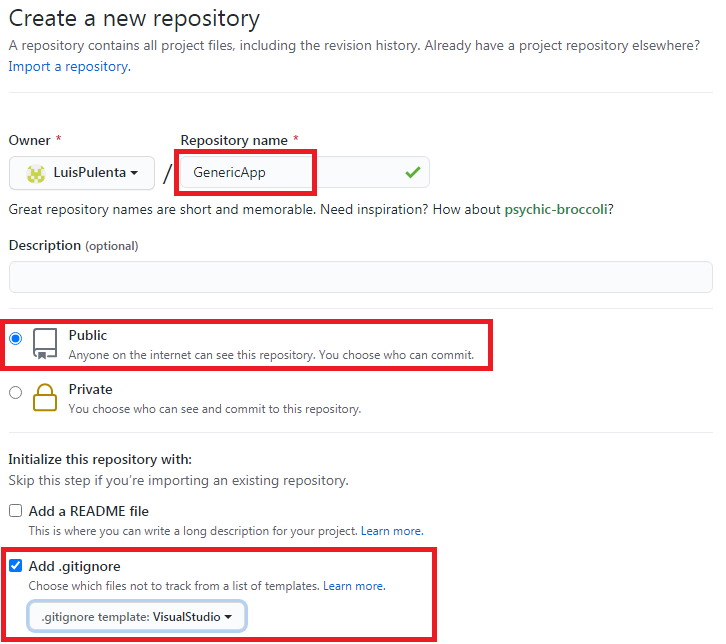
Empezamos haciendo el Repositorio en GitHub

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

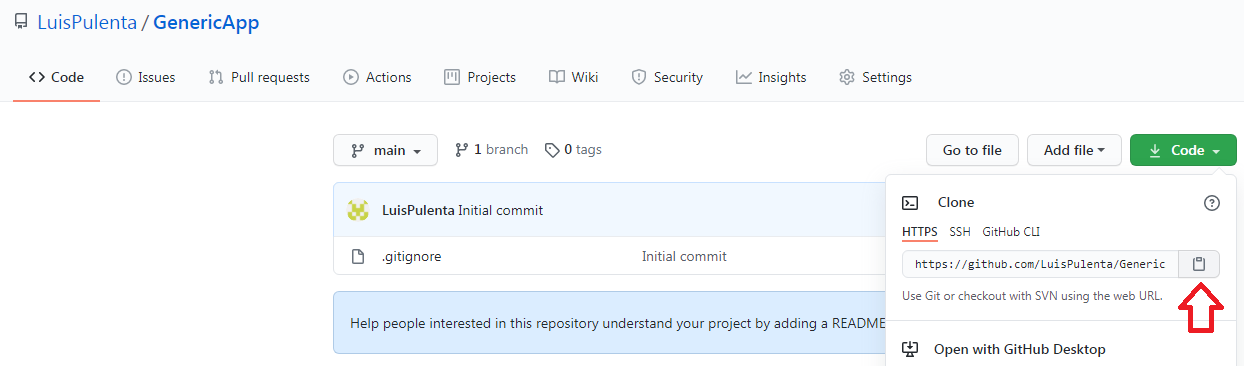
<https://github.com/> **Usuario:** LuisPulenta **Contraseña:** Talleres2306

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

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

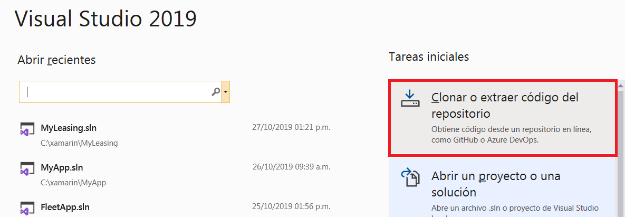
Luego copiamos la dirección para luego clonar en Visual Studio:



# Crear Solución y Proyectos Common, Web y Prism

## Solución en blanco

Abrimos Visual Studio y vamos a la opción Clonar

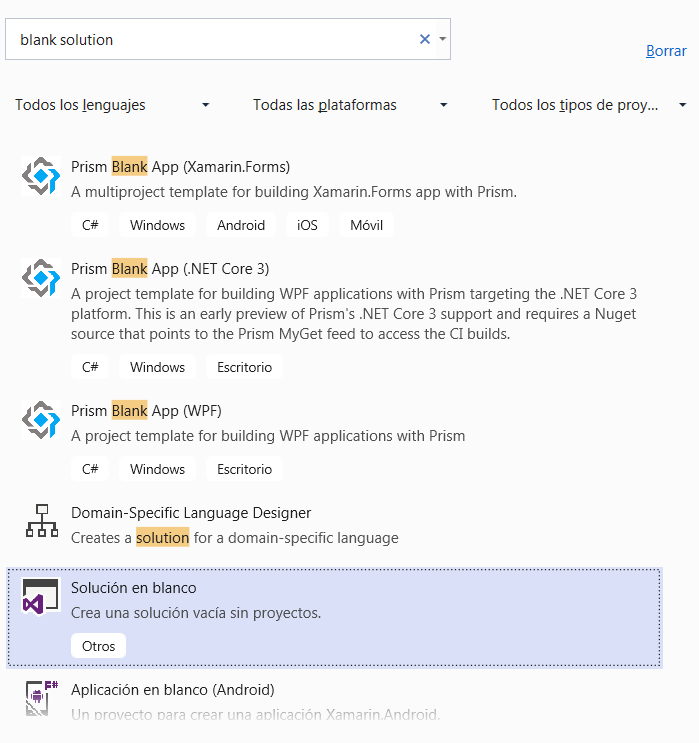
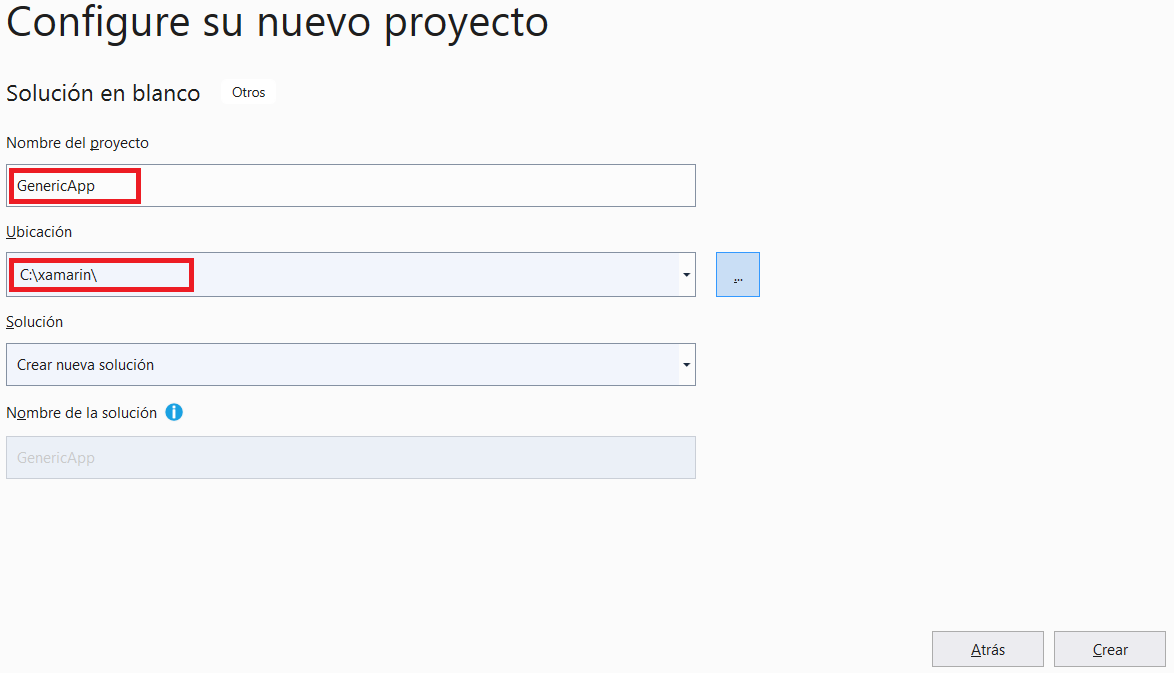




Que sea el mismo nombre del Repositorio. Debe ser una carpeta que no existe

Pegar

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

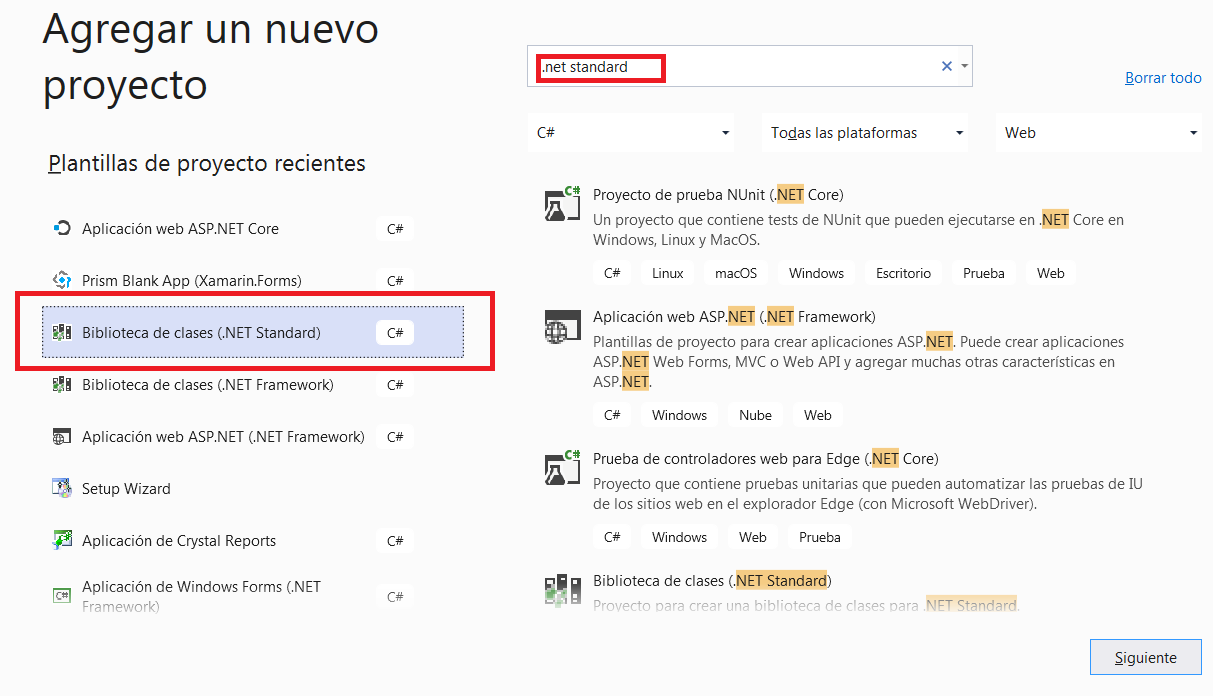
OJO!!! Debe ser la Carpeta CONTENEDORA

Que sea el mismo nombre del Repositorio

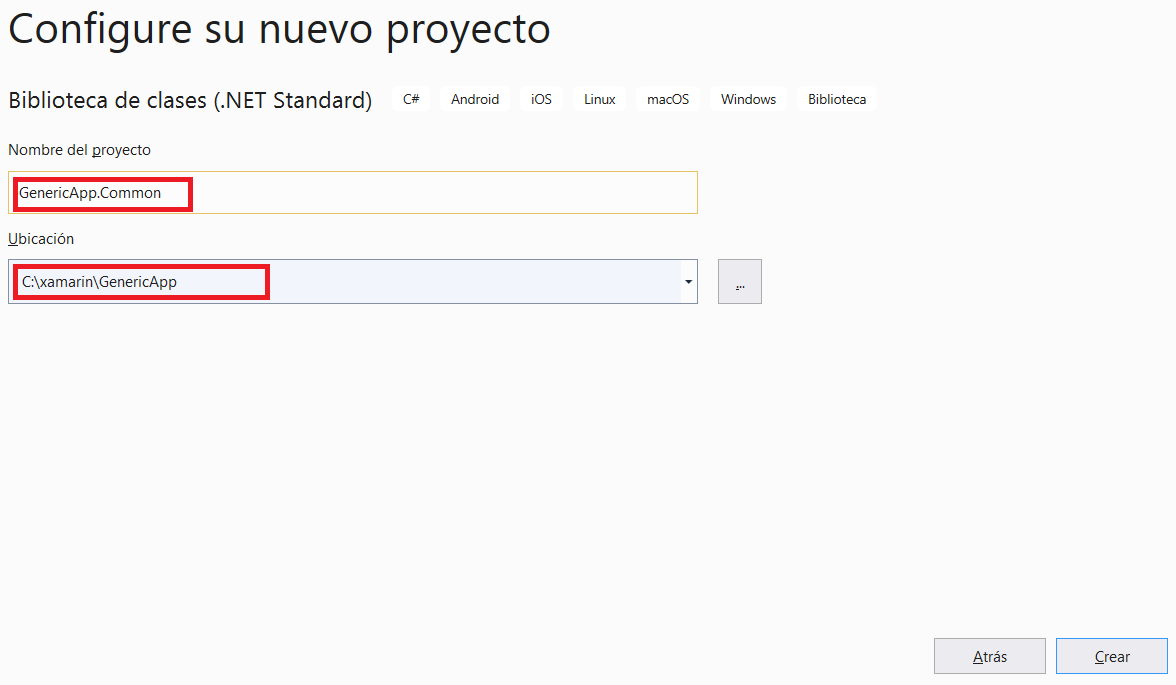
## Proyecto Common

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

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



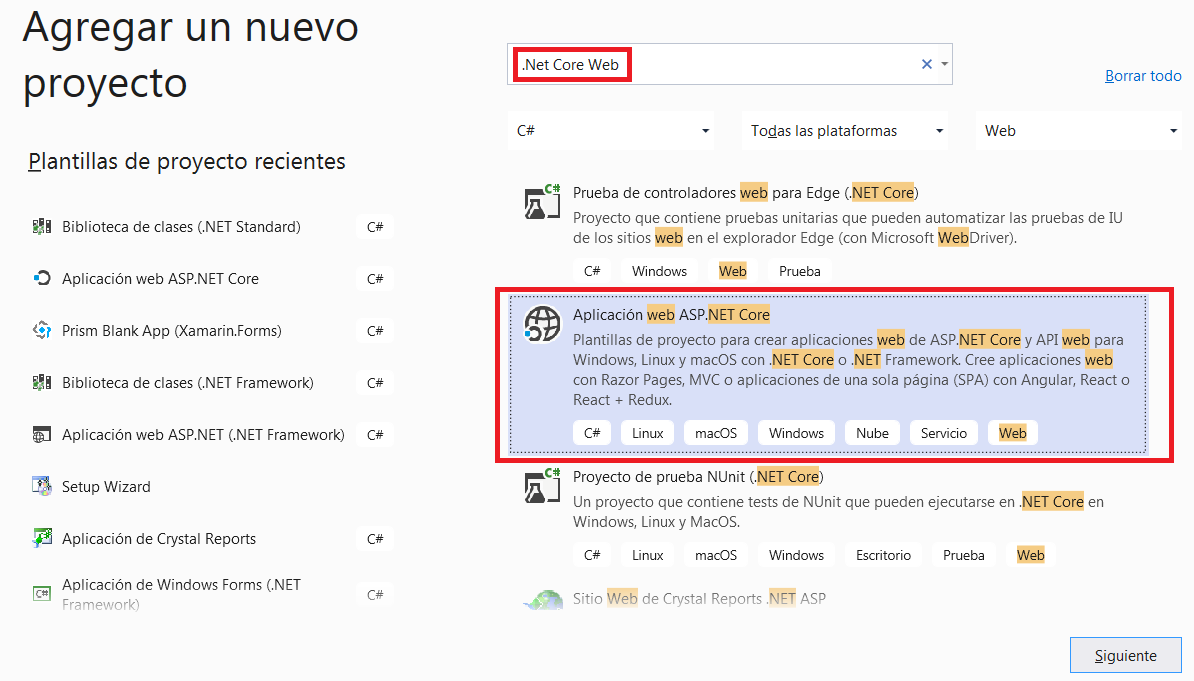
Y la llamamos **GenericApp.Common**

****

(La Clase Class1 que se crea la borramos)

## Proyecto Web (Net Core)

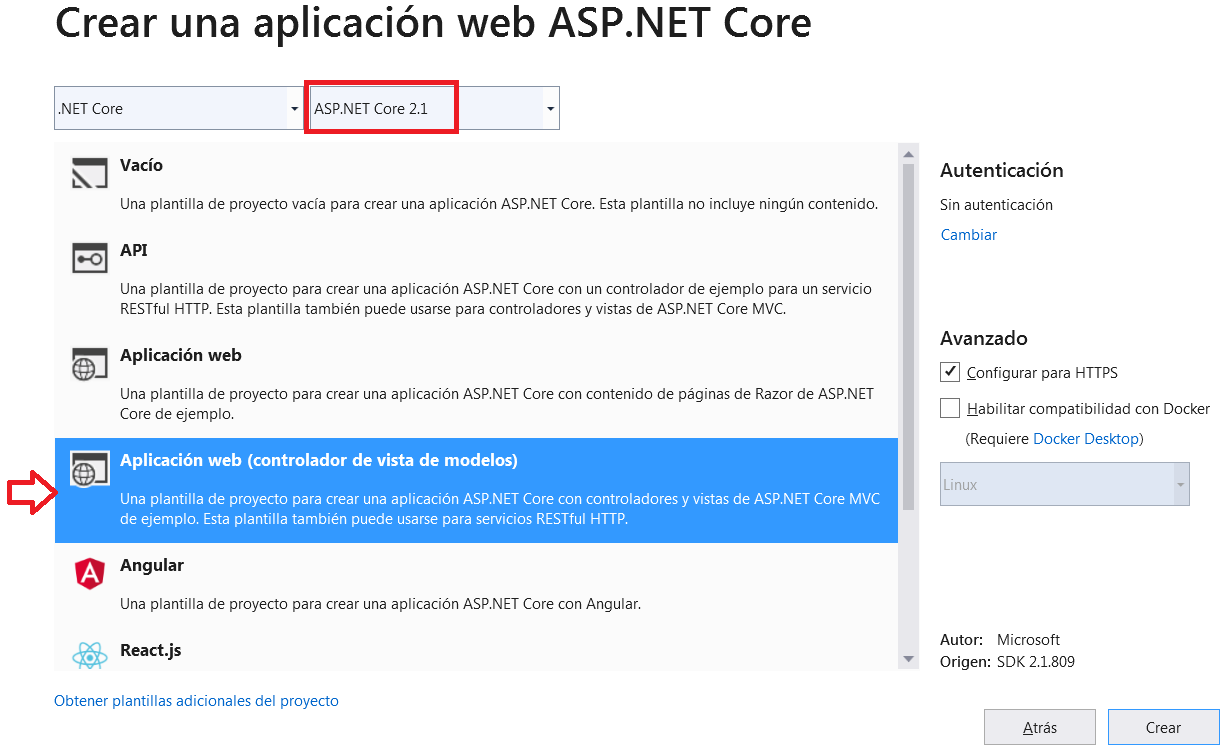
Volvemos a hacer clic derecho en la Solución y ahora ponemos **.Net Core Web** en la plantilla y elegimos **Aplicación Web.Net Core**



Como nombre le ponemos **GenericApp.Web**

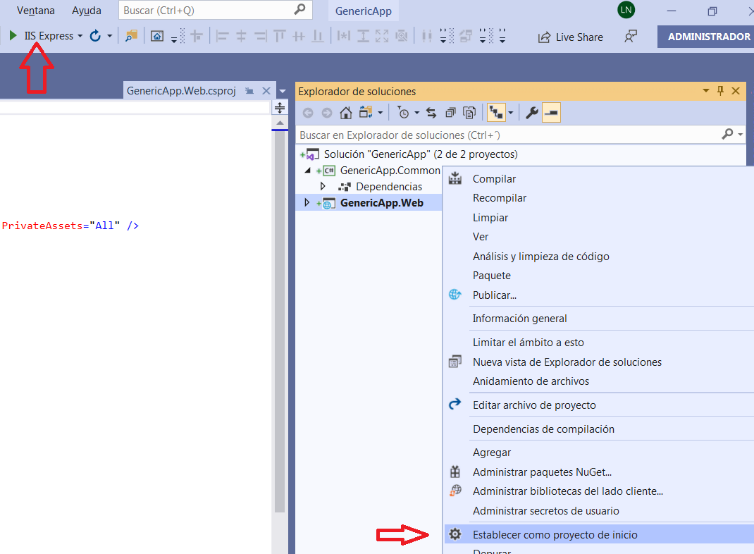
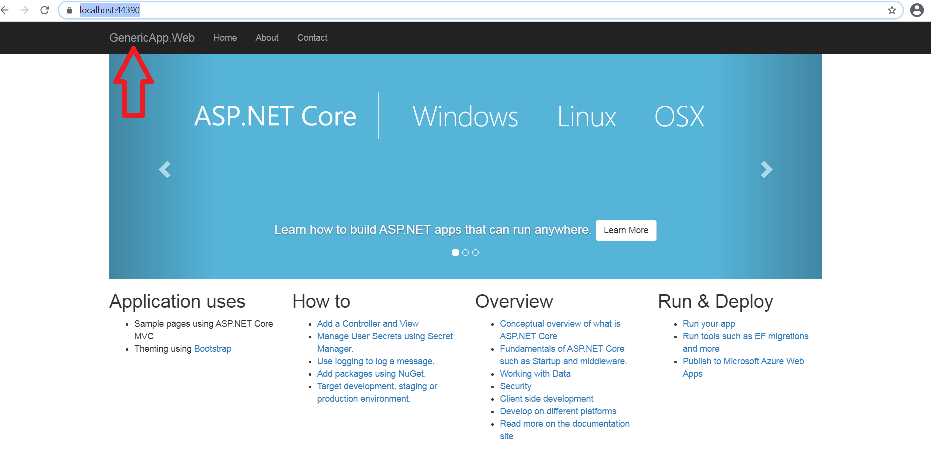


Debe ser Core 2.1 y elegimos la opción Aplicación web (controlador de vista de modelos)



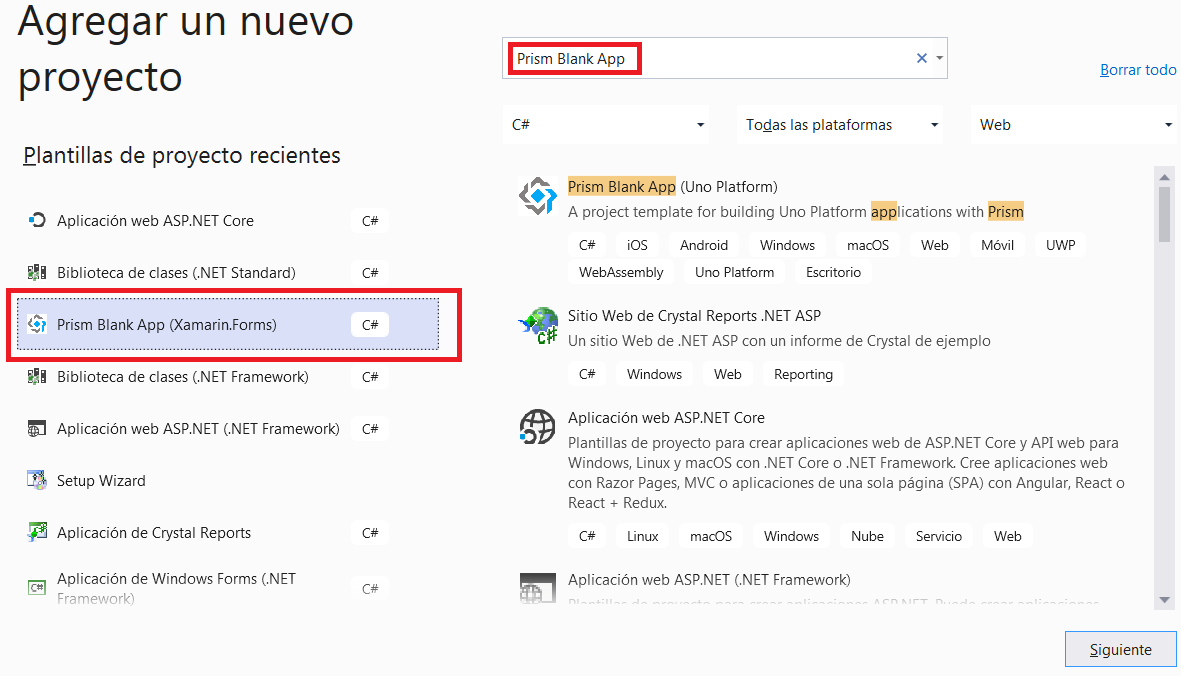
Este Proyecto ya debe arrrancar.

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

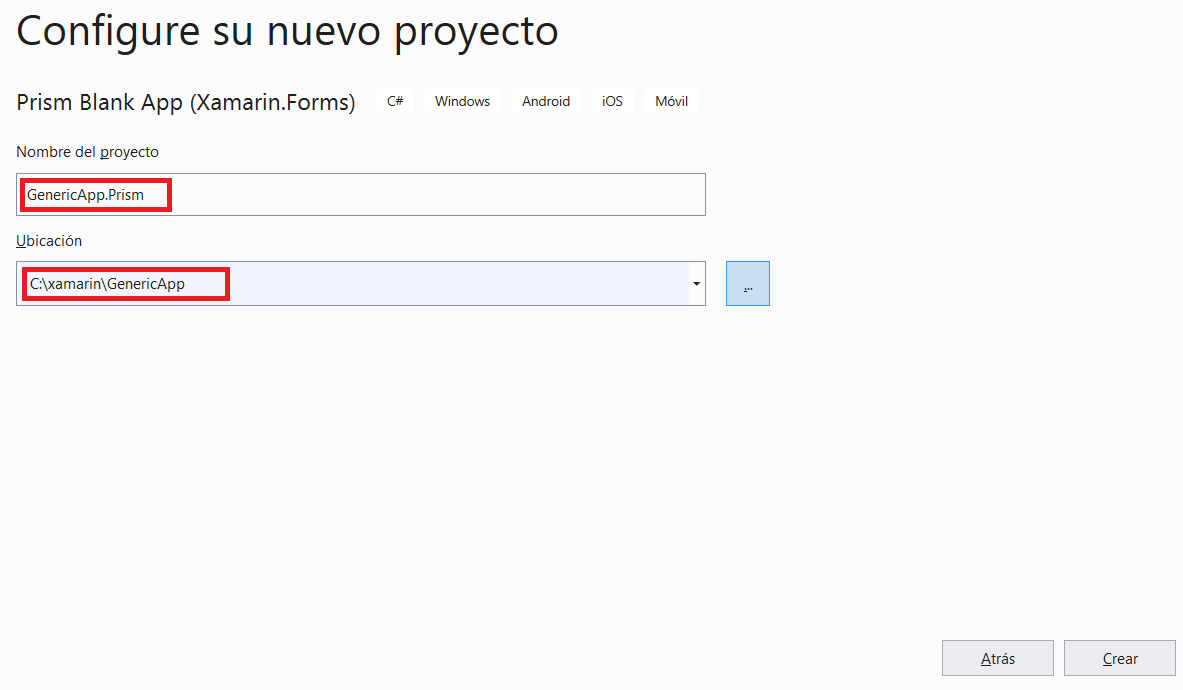
 

## Proyectos Prism

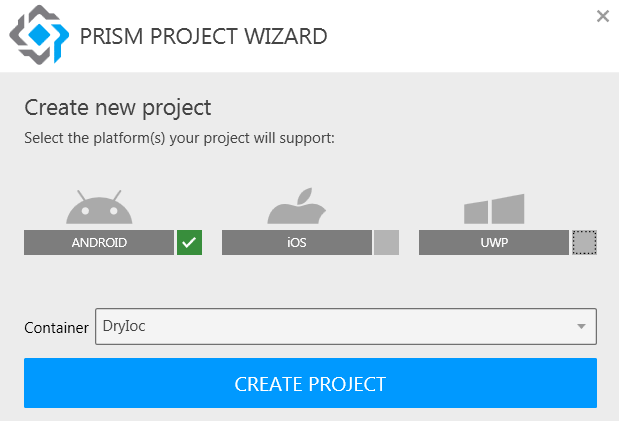
Volvemos a hacer clic derecho en la Solución y ahora ponemos **Prism Blank App** en la plantilla y elegimos **Prism Blank App (Xamarin.Forms)**

****

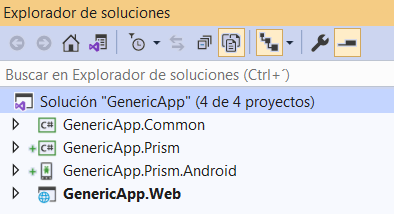
Como nombre le ponemos **GenericApp.Prism**



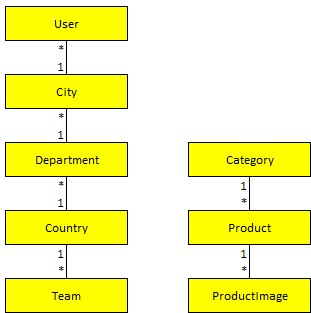
Elegimos **Container 🡪 DryIoc**



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 FlagImageUrl { get; set; }  public string FlagImageFullPath => string.IsNullOrEmpty(FlagImageUrl)  ? $"http://keypress.serveftp.net:88/GenericAppApi/images/Flags/noimage.png"  : $"http://keypress.serveftp.net:88/GenericAppApi{FlagImageUrl.Substring(1)}";  }  } |  |

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

Dentro colocamos el archivo **noimage.png**

## 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<CountryEntity> Countries { get; set; }  protected override void OnModelCreating(ModelBuilder modelBuilder)  {  base.OnModelCreating(modelBuilder);  modelBuilder.Entity<CountryEntity>()  .HasIndex(t => t.Name)  .IsUnique();  }  }  } | 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

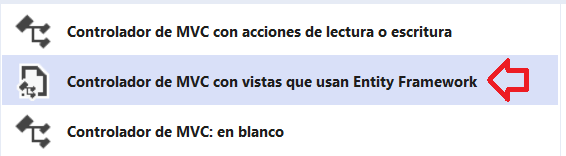
Verificamos en el SQL Server que la Base de Datos se haya creado.

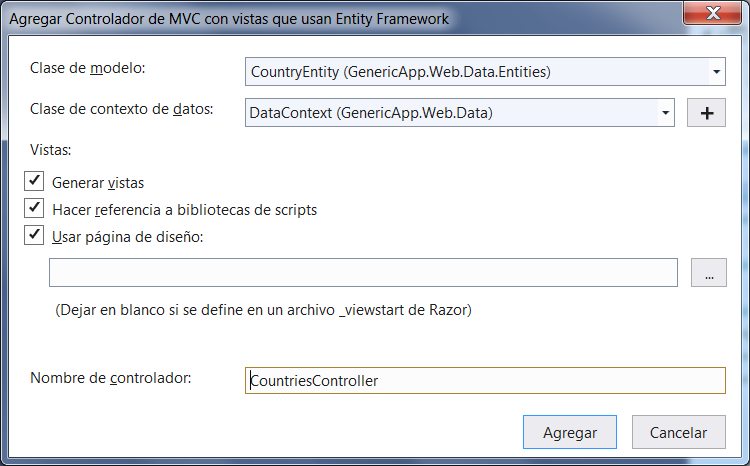
# CRUD para Countries

## 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,FlagImageUrl")] 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,FlagImageUrl")] 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) |

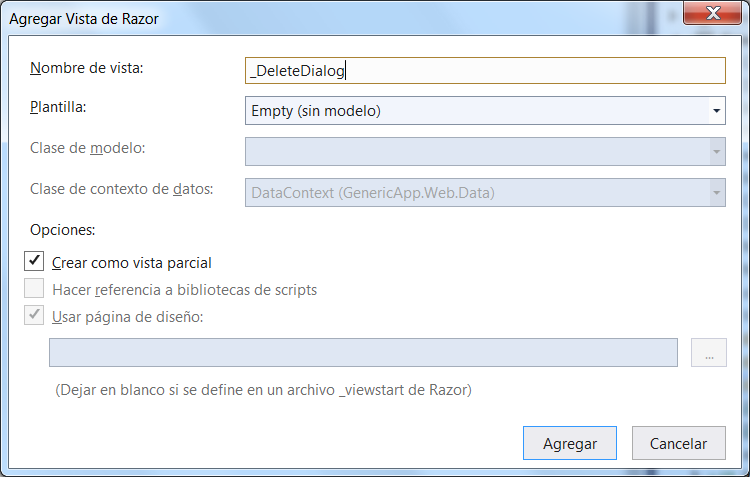
Modificamos el menú para poder probar lo que llevamos.

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

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

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