



El Proyecto estará basado en un nivel general para el proyecto propuesto en clase:

Directorios color **Rojo**

Proyecto y sus componentes a crear color **azul**

Comando a ejecutar por consola color **verde**

|  |  |
| --- | --- |
|  | **CRUD EN PROYECTOS RAZOR Y ENTITY-FRAME-CORE** |
|  | Una vez garantizado el diseño de la base de datos, se procederá a desarrollar el aplicativo en un lenguaje de programación, con conexión a un S.G.B.D. servidor de bases de datos para el almacenamiento permanente de la información. |
|  |  |
|  | Para cada una de las entidades (Tablas – Clases), se debe generar una GUI que permita la Administración y control de registros:   * **C** reate : Permite adicionar un nuevo registro * **R** ead : Permite consultar un registro de los almacenados y mostrarlo * **U** update : Permite actualizar la información del registro previamente Editado * **D** elete : Permite remover o eliminar un registro previamente Editado |
|  |  |

|  |  |  |
| --- | --- | --- |
|  | **CRUD EN PROYECTOS RAZOR Y ENTITY-FRAME-CORE** | |
|  | El siguiente proyecto está direccionado a las versiones recomendadas del curso (.NET Core 6.0.400), si trabaja con otras versiones, debe hacer las adecuaciones respectivas entre versiones que se requieren para la instalación de paquetes adicionales.Directorio raíz: proyectos-webNombre del proyecto: Notas.App | |
|  |  | |
|  | **Verificar que se encuentre en del directorio** **proyectos-web** |  |
|  | **Crear capa solución (sln):** **dotnet new sln -o Notas.App** | |
|  | **Verificar que se crea el directorio** para la **solución** **proyectos-web\Notas.App** | |
|  |  | |
|  | **Crear capa dominio (classlib):** **dotnet new classlib -o Notas.App.Dominio** | |
|  | **Crear capa Persistencia (classlib):** **dotnet new classlib -o Notas.App.Persistencia** | |
|  | **Crear capa Servicios (webapi):** **dotnet new webapi -o Notas.App.Servicios** | |
|  | **Crear capa Presentacion (webapp):** **dotnet new webapp -o Notas.App.Presentacion** | |
|  | **Crear capa Consola (console):** **dotnet new console -o Notas.App.Consola** | |
|  | No continuar si al verificar se obtiene el siguiente resultado (dir) y (tree): | |
|  |  | |
|  | **Verificar que se encuentre en del directorio** **solución** e ingresar a **Visual Studio Code** **proyectos-web\Notas.App\code .** | |
|  |  | |
|  | **Instalación de librerías globales, verifique que se encuentre en la capa de raíz del proyecto Notas.App, donde se encuentra el archivo.sln; si al instalar las librerías ya existen se indicara con un mensaje y se procede a actualizar con update.**  **Abrir una nueva terminal o ejecutar desde la ventana CMD,**  **Por seguridad y no incurrir en error, seleccione la línea del comando completa y al hacer clic con el mouse derecho en la consola automáticamente se pegará** | |
|  | **dotnet tool install --global dotnet-ef**  **o**  **dotnet tool update --global dotnet-ef** | |
|  |  | |
|  | **Instalar librerías particulares de la capa de Persistencia, verificar que NO salgan errores de compatibilidad de versiones. Son las requeridas para trabajar el Code First con el EntyFrameWorkCore.**  **Verificar que se encuentre en la capa de Notas.App.Persistencia** | |
|  |  | |
|  | **dotnet add package Microsoft.EntityFrameworkCore --version 5.0.9** | |
|  | **dotnet add package Microsoft.EntityFrameworkCore.Tools --version 5.0.9** | |
|  | **dotnet add package Microsoft.EntityFrameworkCore.Design --version 5.0.9** | |
|  | **dotnet add package Microsoft.EntityFrameworkCore.SqlServer --version 5.0.9** | |
|  | **Verificar su correcta inclusión de los paquetes en el archivo csproj de la capa respectiva** | |
|  |  | |
|  | **Repetir el mismo proceso de instalar las mismas librerías(4) particulares de la capa de Presentacion, ya que también requiere trabajar el Code First con el EntyFrameWorkCore.**  **Verificar que se encuentre en la capa de Notas.App.Presentacion** | |
|  | **dotnet add package Microsoft.EntityFrameworkCore --version 5.0.9** | |
|  | **dotnet add package Microsoft.EntityFrameworkCore.Tools --version 5.0.9** | |
|  | **dotnet add package Microsoft.EntityFrameworkCore.Design --version 5.0.9** | |
|  | **dotnet add package Microsoft.EntityFrameworkCore.SqlServer --version 5.0.9** | |
|  |  | |
|  | **Repetir el proceso de instalar solamente la librería(1) Microsoft.EntityFrameworkCore.Design a la capa de Consola, ya que también requiere trabajar el Code First con el EntyFrameWorkCore.**  **Verificar que se encuentre en la capa de Notas.App.Consola** | |
|  | **dotnet add package Microsoft.EntityFrameworkCore.Design --version 5.0.9** | |
|  |  | |
|  | **Crear REFERENCES las cuales son requeridas entre capas, para comunicasen y hacer uso de sus clases y métodos, una clase ofrece el servicio con ADD REFERENCE y la otra lo utiliza indicando en la cabecera con USING la capa asociada.** | |
|  | **Verificar que se encuentre en la capa de Notas.App.Persistencia se recomienda usar ./tab para evitar errores y llegar directamente al destino correcto, en este caso persistencia requiere del dominio, para hacer uso de las clases creadas para cada entidad.** | |
|  | **dotnet add reference ..\Notas.App.Dominio\** | |
|  |  | |
|  | **Verificar que se encuentre en la capa de Notas.App.Presentacion se recomienda usar ./tab para evitar errores y llegar directamente al destino correcto, en este caso presentacion requiere del dominio para las clases y de persistencia para el DBContext encargado de la comunicación con la Base de Datos.** | |
|  | **dotnet add reference ..\Notas.App.Dominio\** | |
|  | **dotnet add reference ..\Notas.App.Persistencia\** | |
|  |  | |
|  | **Verificar que se encuentre en la capa de Notas.App.Consola se recomienda usar ./tab para evitar errores y llegar directamente al destino correcto, en este caso Consola requiere del dominio para las clases y de persistencia para el DBContext encargado de la comunicación con la Base de Datos.** | |
|  | **dotnet add reference ..\Notas.App.Dominio\** | |
|  | **dotnet add reference ..\Notas.App.Persistencia\** | |
|  |  | |
|  | **COMPILACIÓN**: Verificación de errores de sintaxis y referencias y paquees incluidos, sobre las capas trabajadas:  Como la capa Presentación contiene la referencia a Dominio y Persistencia, al compilarla se compilan las tres.  **Verificar que se encuentre en la capa de Notas.App.Presentacion** | |
|  | **dotnet build** | |
|  |  | |
|  | **CORRER – RUN - EJECUTAR**: Se obtienen los resultados de la primera página para adecuarla a un proyecto en particular, aplicando el CRUD completo sobre cada una de las entidades creadas. | |
|  | **dotnet watch run** | |
|  |  | |
|  |  | |
|  | **Si no permite abrir la página, permitir con localhost no seguro** | |
|  |  | |
|  | **CREACION MENUS**: Para que cada ítem del menú horizontal se visualizasen en la barra superior de la página creada, deben existir los siguientes dos archivos:   * Se ha creado el archivo **\_Layout\_cshml** \Notas.App\Notas.App.Presentacion\Pages\**Shared**\\_Layout.cshtml * La ruta indicada en asp-page, debe existir, por defecto se re direccionan a la sección de **Pages**, y se debe incluir la ruta especificada si van dentro de otros directorios | |
|  |  | |
|  | **CREACION DIRECTORIOS EN PAGE**: Por organización se creará una carpeta para cada entidad del proyecto y dentro se crearán los archivos asociados al **CRUD** respectivo; para el ejemplo se está tomando la entidad Asignaturas. | |
|  |  | |
|  | **CREACION ARCHIVOS RAZOR PARA EL CRUD**: Inicialmente cuando se selecciona la entidad Asignaturas, debe mostrar todos sus registros en una tabla, a la cual se le asociarán los eventos respectivos para el usuario interactuar.  Verificar que se encuentre en la capa de Presentación, incluir para la creación del namespace la ruta completa hasta la Entidad a crear, para diferenciar de las otras entidades a crear. | |
|  | **dotnet new page -n Listar -na Notas.App.Presentacion.Pages.Asignaturas -o .\Pages\Asignaturas\** | |
|  |  | |
|  | El archivo **Listar.cshml.cs** se encarga del código C#  El archivo **Listar.cshtml** se encarga del código Razor, que combina etiquetas **HTML** <etiqueta> con el lenguaje C# para llamada a variables y funciones con el símbolo **@** | |
|  | **COMUNICACIÓN DEL FRONT-END CON EL BACK-END:** Se crean los Modelos de las clases en la capa de Dominio:   * Una sola carpeta **Entidades** * Una clase por cada entidad, nombre inicial mayúsculas, ejm **Asignatura.cs** * Se escribe el código de la clase con sus atributos y tipos de datos respectivos * Tener presente que para el Entity Core, se exige tener un atributo obligado llamado id. | |
| 34.1. |  | |
|  |  | |
|  | //permisos para validaciones y display  using System.ComponentModel.DataAnnotations;  namespace Notas.App.Dominio.Entidades  {      public class Asignatura      {          //atributos de la clase asignatura - ver diseño          [Key]          [Display(Name = "ID")]          public int id { get; set; }          [Required]          [Display(Name = "Nombre")]          public string nombre { get; set; }          [Required]          [Display(Name = "Intensidad")]          public int intensidad { get; set; }      }  } | |
|  | **Compilar y verificar que NO se tengan errores en la capa de dominio** | |
|  |  | |
|  | **CAPA DE PERSISTENCIA – INTERFACES – REPOSITORIOS - APPCONTEXT:** Se crean los archivos de las clases e interfaces en la capa de Persistencia:   * Una sola carpeta **AppRepositorios** * El alcance de los namespaces hasta AppRepositorios * Una Interface **IRepositorios** para las firmas o contratos de los métodos CRUD * Una Clase **Repositorios** para la implementación de los métodos del CRUD * Una clase para el **AppContext** | |
|  |  | |
|  | Código para la Clase AppContext | |
|  | using Microsoft.EntityFrameworkCore;  using Notas.App.Dominio.Entidades;  namespace Notas.App.Persistencia.AppRepositorios  {      public class AppContext  : DbContext      {          public DbSet<Formador> Formador { get; set; }            //crear el deContext          protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)          {              if (!optionsBuilder.IsConfigured)              {              optionsBuilder             .UseSqlServer("Server=localhost; user id=sa; password=12345; Initial Catalog=BDNotas;");              }          }      }  } | |
|  | Código para la Interface IRepositorios - Firma de los métodos a implementar en los Repositorios | |
|  | //Directivas  using System;  using System.Collections;  using System.Collections.Generic;  using System.Linq;  //permisos acceder al modelo de la clase  using Notas.App.Dominio.Entidades;  namespace Notas.App.Persistencia.AppRepositorios  {      public interface IRepositorios      {          //contratos o firmas para los metodos Asignatura          Asignatura InsertarAsignatura(Asignatura Asignatura);          IEnumerable<Asignatura> ListarAsignaturas();          Asignatura ConsultarAsignatura(int? idAsignatura);          Asignatura ActualizarAsignatura(Asignatura Asignatura);          void EliminarAsignatura(int idAsignatura);      }  } | |
|  | Código para la clase Repositorios | |
|  | using System;  using System.Collections.Generic;  using System.Linq;  using Microsoft.EntityFrameworkCore;  using Notas.App.Dominio.Entidades; //permisos acceder al modelo de la clase  namespace Notas.App.Persistencia.AppRepositorios  {      public class Repositorios : IRepositorios      {         private readonly AppContext \_appContext;         public Repositorios(AppContext appContext)          {              \_appContext = appContext;          }          //AQUÍ CADA UNO DE LOS MÉTODOS DEL CRUD, REFERENCIADOS EN LA INTERFACE          Asignatura IRepositorios.InsertarAsignatura(Asignatura asignatura)          {          try           {              var adicionado = \_appContext.asignatura.Add( asignatura );  //INSERT en la BD              \_appContext.SaveChanges();              return adicionado.Entity;            }catch              {                  throw;              }          }          IEnumerable<Asignatura> IRepositorios.ListarAsignaturas(string? cadena)          {              if (cadena == null)                  asignaturas = \_appContext.asignatura;              else{                  //busca coincidencias entre los registros y la cadena enviada                  asignaturas = \_appContext.asignatura.Where(s => s.nombre.Contains(cadena));                  //busca solamente los que son exactamente igual a la cadena enviada                  //asignaturas = \_appContext.asignatura.Where(s => s.nombre.Equals(cadena));              }              return asignaturas;          }         Asignatura IRepositorios.ConsultarAsignatura(int? id)         {              return \_appContext.asignatura.FirstOrDefault(p => p.id == id);         }         Asignatura IRepositorios.ActualizarAsignatura(Asignatura asignatura)          {              var encontrado = \_appContext.asignatura.FirstOrDefault(p => p.id == asignatura.id);              if (encontrado != null)              {                  encontrado.nombre     = asignatura.nombre;                  encontrado.intensidad = asignatura.intensidad;                  \_appContext.SaveChanges();              }              return encontrado;          }          void IRepositorios.EliminarAsignatura(int id)          {              var encontrado = \_appContext.asignatura.FirstOrDefault(p => p.id == id);              if (encontrado == null)                  return;              \_appContext.asignatura.Remove(encontrado);              \_appContext.SaveChanges();          }      }  } | |
|  | **Compilar y verificar que NO se tengan errores en la capa de Persistencia** | |
|  | Asociar los Repositorios a la capa de presentación, para el uso del servicio del DBContext | |
|  |  | |
|  | using Notas.App.Persistencia.AppRepositorios; | |
|  | services.AddScoped<IRepositorios, Repositorios>(); | |
|  | services.AddSingleton<Notas.App.Persistencia.AppRepositorios.AppContext>(); | |
|  | **CONEXIÓN Y MIGRACIÓN DE LA BASE DE DATOS:** Se crean los archivos de migración para la base de datos con los comandos respectivos:   * Verificar en la capa de consola, las librerías y referencias requeridas | |
|  |  | |
|  | Para la creación y migración de la Base de datos, tener en cuenta:   * **Servidor de Bases de Datos** * **Usuario** * **Clave** * **Nombre de la Base de Datos** | |
|  |  | |
|  | **Verificar que se encuentre en la capa de Notas.App.Persistencia se recomienda usar .\tab para evitar errores y llegar directamente al destino correcto.** | |
|  | Crear la migración de los modelos de clases, (NO crea la Base de Datos en el Motor)  **dotnet ef migrations add Inicial --startup-project ..\Notas.App.Consola\** | |
|  | Compilar la capa de persistencia | |
|  | Crear la Base de datos  **dotnet ef database update --startup-project ..\Notas.App.Consola\** | |
|  |  | |
|  | Verificar la creación de la Base de Datos “**Proyectos**” en el servidor de bases de datos SQLServer o Azure, se crean la tabla Asignaturas y una adicional de migración controlada por el EntityFrameworkcore. Cuando se trabaja con Code First NO se debe alterar la estructura de la Base de Datos por el Servidor, sino recurrir a los scripts sql que genera el EF, al correr una actualización. | |
|  |  | |
|  | Probar en la capa de presentación nuevamente la página web | |
|  |  | |
|  | Complementar el código de la capa de presentación para las operaciones del CRUD | |
|  | **Listar.cshtml.cs** | |
|  | using System;  using System.Collections.Generic;  using System.Linq;  using System.Threading.Tasks;  using Microsoft.AspNetCore.Mvc;  using Microsoft.AspNetCore.Mvc.RazorPages;  using Microsoft.AspNetCore.Authorization;  using Notas.App.Dominio;  using Notas.App.Persistencia.AppRepositorios;  using Notas.App.Persistencia;  namespace Notas.App.Presentacion.Pages  {      //[Authorize]      public class ListarModel : PageModel      {          private readonly IRepositorios \_appContext;          public IEnumerable<Formador> Asignaturas {get; set;}          public ListarModel()          {              this.\_appContext = new Repositorios(new Notas.App.Persistencia.AppRepositorios.AppContext());          }            public void OnGet(string filtroBusqueda)          {              Asignaturas = \_appContext.GetAllAsignaturas();          }      }  } | |
|  | **Listar.cshtml** | |
|  | @page  @model Notas.App.Presentacion.Pages.ListarModel  @{    }  <h1>Listarado de Asignaturas</h1>  <table class="table">      @foreach (var formador in Model.Asignaturas)      {          <tr>              <td>@formador.id</td>              <td>@formador.identificacion</td>              <td>@formador.nombre</td>              <td>                  <a class="btn btn-primary table-btn" asp-page="./Details" asp-route-formadorId="@formador.id">                      Detalle </a>              </td>              <td>                  <a class="btn btn-primary table-btn" asp-page="./Edit" asp-route-formadorId="@formador.id">                      Editar                  </a>              </td </tr>      }  </table>  <a asp-page=".\Edit" class="btn btn-primary">Adicionar un formador</a> | |
|  | Crear los archivos para Edit, que se encargará del INSERT y el UPDATE. | |
|  | **dotnet new page -n Edit -na Notas.App.Presentacion -o .\Pages\Asignaturas\** | |
|  | **Código Edit.cshtml.cs** | |
|  | using System;  using System.Collections.Generic;  using System.Linq;  using System.Threading.Tasks;  using Microsoft.AspNetCore.Mvc;  using Microsoft.AspNetCore.Mvc.RazorPages;  using Notas.App.Dominio;  using Notas.App.Persistencia.AppRepositorios;  namespace Notas.App.Presentacion.Pages  {      public class EditModel : PageModel      {         private readonly IRepositorios \_appContext;          [BindProperty]          public Formador formador  { get; set; }          public EditModel()         {              this.\_appContext  =new Repositorios(new Notas.App.Persistencia.AppRepositorios.AppContext());         }            public IActionResult OnGet(int? formadorId)          {              if (formadorId.HasValue)              {                  formador = \_appContext.GetFormador(formadorId.Value);              }              else              {                  formador = new Formador();              }              if (formador == null)              {                  return RedirectToPage("./NotFound");              }              else                  return Page();          }          public IActionResult OnPost()          {              if(!ModelState.IsValid)              {                  return Page();              }              if(formador.id > 0)              {                 formador = \_appContext.UpdateFormador(formador);              }              else              {                 \_appContext.AddFormador(formador);              }              return Page();          }      }  } | |
|  | **Código Edit.cshtml** | |
|  | @page  @model Notas.App.Presentacion.Pages.EditModel  @{      int formadorId = Model.formador.id;  }  <h2>Edición de los datos del formador @Model.formador.nombre @Model.formador.identificacion</h2>  <form method="post">      <input type="hidden" asp-for="formador.id" />      <div class="form-group">          <label asp-for="formador.identificacion"></label>          <input asp-for="formador.identificacion" class="form-control">          <span class="text-danger" asp-validation-for="formador.identificacion"></span>      </div>      <div class="form-group">          <label asp-for="formador.nombre"></label>          <input asp-for="formador.nombre" class="form-control">          <span class="text-danger" asp-validation-for="formador.nombre"></span>      </div>      <button type="submit" class="btn btn-primary">Grabar</button>  </form>  <a asp-page="./Listar" class="btn-default">Regresar a la Listara de Asignaturas </a> | |
|  | **Verificar Insertando varios registros por la Interfaz** | |
|  |  | |
|  | **Al Editar el registro, puede actualizar los campos del formulario y al guardar se actualiza la Base de Datos.** | |
|  |  | |
|  | **Al Detalle del registro, puede Visualizar los campos del formulario cargados desde la base de datos, es la oportunidad de utilizar un CARD para su presentación.** | |
|  |  | |
|  |  | |
|  |  | |
|  |  | |