En la carpeta /Entidades tengo la entidad **Nota**

using System.ComponentModel.DataAnnotations;

namespace Prueba.Entidades

{

public class Nota

{

public Nota() // Este es el constructor de la clase Nota

{

this.CategoriasNota = new List<CategoriaNota>(); // Inicializa CategoriasNota como una nueva lista de CategoriaNota

}

public int Id { get; set; }

[StringLength(250)]

[Required]

public string Titulo { get; set; } //Titulo de la Nota

public List<CategoriaNota> CategoriasNota { get; set; } //\*\* Propiedad de Navegación que nos lleva a la Data Relacionada CategoriasNota

}

}

y la entidad **CategoriaNota**

using System.ComponentModel.DataAnnotations;

namespace Prueba.Entidades

{

public class CategoriaNota

{

//\*\* Creamos entidad CategoriaNota con realación MUCHOS a MUCHOS

//\*\* Una CategoriaNota puede tener muchas Notas

public int Id { get; set; }

[StringLength(100)]

[Required]

public string Nombre { get; set; }

public List<Nota> Notas { get; set; } //\*\* Propiedad de Navegación que nos lleva a la Data Relacionada Notas

}

}

En la capeta /Models tengo el modelo **NotaViewModel**

using Prueba.Entidades;

namespace Prueba.Models

{

public class NotaViewModel

{

public int Id { get; set; }

public string Titulo { get; set; }

public List<CategoriaNota> CategoriasNota { get; set; } //\*\* Propiedad de Navegación que nos lleva a la Data Relacionada CategoriasNota

public List<CategoriaNota> CategoriaSeleccionadaId { get; set; }

}

}

el modelo **CreaNotaViewModel**

namespace Prueba.Models

{

public class CrearNotaViewModel

{

public string Titulo { get; set; }

public List<int> CategoriaSeleccionadaId {get; set;}

}

}

El modelo **CategoriaNotaViewModel**

namespace Prueba.Models

{

public class CategoriaNotaViewModel

{

public int Id { get; set; }

public string Nombre { get; set; }

public List<NotaViewModel> Notas { get; set; } //\*\* Propiedad de Navegación que nos lleva a la Data Relacionada Notas

}

}

En la carpeta /Controllers tengo un controlador **NotaController**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Rendering;

using Microsoft.EntityFrameworkCore;

using Prueba;

using Prueba.Models;

using Prueba.Entidades;

namespace Prueba.Controllers

{

public class NotasController : Controller

{

private readonly ApplicationDbContext context;

public NotasController(ApplicationDbContext context)

{

this.context = context;

}

// GET: Listado Notas

[HttpGet]

public async Task<IActionResult> Index()

{

return View(await context.Notas.AsNoTracking().ToListAsync()); ;

}

// GET: Notas/VerNota

public async Task<IActionResult> VerNota(int? id)

{

if (id is null || context.Notas is null)

{

return NotFound();

}

var nota = await context.Notas

.Include(c => c.CategoriasNota)

.FirstOrDefaultAsync(m => m.Id == id);

if (nota is null)

{

return NotFound();

}

return View(nota);

}

// GET: Notas/CrearNota

public IActionResult CrearNota()

{

var viewModel = new NotaViewModel ();

viewModel.CategoriasNota = context.CategoriasNota.ToList();

return View(viewModel);

}

// POST: Notas/CrearNota

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> CrearNota(CrearNotaViewModel crearNotaViewModel)

{

if (!ModelState.IsValid)

{

return View(crearNotaViewModel);

}

var nota = new Nota();

nota.Titulo = crearNotaViewModel.Titulo;

foreach (var id in crearNotaViewModel.CategoriaSeleccionadaId)

{

var categoria = await context.CategoriasNota.FindAsync(id);

if (categoria != null)

{

nota.CategoriasNota.Add(categoria);

}

}

context.Add(nota);

await context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

// GET: Notas/EditarNota

public async Task<IActionResult> EditarNota(int? id)

{

if (id == null || context.Notas == null)

{

return NotFound();

}

var viewModel = new NotaViewModel();

viewModel.CategoriasNota = context.CategoriasNota.ToList();

var nota = await context.Notas

.Include(c => c.CategoriasNota)

.FirstOrDefaultAsync(m => m.Id == id);

if (nota == null)

{

return NotFound();

}

viewModel.Titulo = nota.Titulo.ToString();

viewModel.CategoriasNota = context.CategoriasNota.ToList();

viewModel.CategoriaSeleccionadaId = nota.CategoriasNota;

return View(viewModel);

}

// POST: Notas/EditarNota

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> EditarNota(int id, [Bind("Id,Titulo")] Nota nota)

{

if (id != nota.Id)

{

return NotFound();

}

if (ModelState.IsValid)

{

try

{

context.Update(nota);

await context.SaveChangesAsync();

}

catch (DbUpdateConcurrencyException)

{

if (!NotaExists(nota.Id))

{

return NotFound();

}

else

{

throw;

}

}

return RedirectToAction(nameof(Index));

}

return View(nota);

}

// GET: Notas/BorrarNota

public async Task<IActionResult> BorrarNota(int? id)

{

if (id == null || context.Notas == null)

{

return NotFound();

}

var nota = await context.Notas

.FirstOrDefaultAsync(m => m.Id == id);

if (nota == null)

{

return NotFound();

}

return View(nota);

}

// POST: Notas/BorrarNota

[HttpPost, ActionName("BorrarNota")]

[ValidateAntiForgeryToken]

public async Task<IActionResult> DeleteConfirmed(int id)

{

if (context.Notas == null)

{

return Problem("La Entidad 'ApplicationDbContext.Categoria' es null.");

}

var nota = await context.Notas.FindAsync(id);

if (nota != null)

{

context.Notas.Remove(nota);

}

await context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

private bool NotaExists(int id)

{

return context.Notas.Any(e => e.Id == id);

}

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

C A T E G O R I A S

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

// GET: Listado CategoriasNota

public async Task<IActionResult> Categorias()

{

return View(await context.CategoriasNota.ToListAsync());

}

// GET: Notas/CrearCategoria

public IActionResult CrearCategoria()

{

return View();

}

// POST: Notas/CrearCategoria

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> CrearCategoria([Bind("Id,Nombre")] CategoriaNota categoriaNota)

{

if (ModelState.IsValid)

{

context.Add(categoriaNota);

await context.SaveChangesAsync();

return RedirectToAction(nameof(Categorias));

}

return View(categoriaNota);

}

// GET: Notas/VerCategoria

public async Task<IActionResult> VerCategoria(int? id)

{

if (id == null || context.CategoriasNota == null)

{

return NotFound();

}

var categoriaNota = await context.CategoriasNota

.FirstOrDefaultAsync(m => m.Id == id);

if (categoriaNota == null)

{

return NotFound();

}

return View(categoriaNota);

}

// GET: Notas/EditarCategoria

public async Task<IActionResult> EditarCategoria(int? id)

{

if (id == null || context.CategoriasNota == null)

{

return NotFound();

}

var categoriaNota = await context.CategoriasNota.FindAsync(id);

if (categoriaNota == null)

{

return NotFound();

}

return View(categoriaNota);

}

// POST: Notas/EditarCategoriaNota

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> EditarCategoria(int id, [Bind("Id,Nombre")] CategoriaNota categoriaNota)

{

if (id != categoriaNota.Id)

{

return NotFound();

}

if (ModelState.IsValid)

{

try

{

context.Update(categoriaNota);

await context.SaveChangesAsync();

}

catch (DbUpdateConcurrencyException)

{

if (!CategoriaNotaExiste(categoriaNota.Id))

{

return NotFound();

}

else

{

throw;

}

}

return RedirectToAction(nameof(Categorias));

}

return View(categoriaNota);

}

// GET: Notas/BorrarCategoria

public async Task<IActionResult> BorrarCategoria(int? id)

{

if (id == null || context.CategoriasNota == null)

{

return NotFound();

}

var categoriaNota = await context.CategoriasNota

.FirstOrDefaultAsync(m => m.Id == id);

if (categoriaNota == null)

{

return NotFound();

}

return View(categoriaNota);

}

// POST: Notas/BorrarNota

[HttpPost, ActionName("BorrarCategoria")]

[ValidateAntiForgeryToken]

public async Task<IActionResult> ConfirmarBorrado(int id)

{

if (context.CategoriasNota == null)

{

return Problem("La Entidad 'ApplicationDbContext.CategoriaNota' es null.");

}

var categoriaNota = await context.CategoriasNota.FindAsync(id);

if (categoriaNota != null)

{

context.CategoriasNota.Remove(categoriaNota);

}

await context.SaveChangesAsync();

return RedirectToAction(nameof(Categorias));

}

private bool CategoriaNotaExiste(int id)

{

return context.CategoriasNota.Any(e => e.Id == id);

}

}

}

En la carpeta /Views/Nota tengo las vistas Index que muestra un listado de las Notas

@using Prueba.Entidades

@model IEnumerable<Nota>

@{

ViewData["Title"] = "Index";

}

<h1>@ViewBag.Title</h1>

<p>

<**a** class="btn btn-primary mb-3" **asp-action**="CrearNota">Crear Nota</**a**>

</p>

<table class="table">

<thead>

<tr>

<th>

@Html.DisplayNameFor(model => model.Titulo)

</th>

<th></th>

</tr>

</thead>

<tbody>

@foreach (var item in Model)

{

<tr>

<td>

@Html.DisplayFor(modelItem => item.Titulo)

</td>

<td>

<**a** class="btn btn-primary mb-3" **asp-action**="EditarNota" **asp-route-id**="@item.Id">Editar</**a**>

<**a** class="btn btn-success mb-3" **asp-action**="VerNota" **asp-route-id**="@item.Id">Ver Nota</**a**>

<**a** class="btn btn-warning mb-3" **asp-action**="BorrarNota" **asp-route-id**="@item.Id">Borrar</**a**>

</td>

</tr>

}

</tbody>

</table>

La vista CrearNota que muestra un formulario para Crear Notas con sus CategoriasNota

@using Prueba.Models

@model NotaViewModel

@{

ViewData["Title"] = "Crear Nota";

}

<h4>@ViewBag.Title</h4>

<hr />

<div class="row">

<div class="col-md-4">

<**form** **asp-action**="CrearNota">

<**div** **asp-validation-summary**="ModelOnly" class="text-danger"></**div**>

<div class="form-group">

<**label** **asp-for**="Titulo" class="control-label"></**label**>

<**input** **asp-for**="Titulo" class="form-control" /><br />

<**span** **asp-validation-for**="Titulo" class="text-danger"></**span**>

</div>

<div>

@foreach (var categoria in Model.CategoriasNota)

{

<input type="checkbox" name="CategoriaSeleccionadaId" value="@categoria.Id" /> @categoria.Nombre <br>

}

</div>

<br />

<div class="form-group">

<button type="submit" value="CrearNota" class="btn btn-primary mb-3">Crear</button>

<**a** class="btn btn-secondary mb-3" **asp-action**="Index">Cancelar</**a**>

</div>

</**form**>

</div>

</div>

Las vistas VerNota, VerCategoria, CrearCategoria, Categorías que muestra un listado de las Categorías, la vista BorrarNota, la vista BorrarCategoria y también la vista EditarCategoria. Todas ellas funcionan correctamente.

Ahora estoy haciendo la vista EditarNota. Parece que funciona, pero tengo problemas con el método HttpGET EditarNota y el HttpPost EditarNota, además de la vista EditarNota.   
La vista EditarNota es un formulario que debe mostrar los datos de una nota que se ha seleccionado, incluidos todos los checkbox de todas las categorías que existen, pero debe aparecer el Titulo de la categoría y deben estar chequeados solo los checkbox de su data relacionada. De esta forma el usuario podrá modificar el Titulo y seleccionar o deseleccionar los checkbox.

Puedes verificar todos los documentos y recomendar los cambios necesarios para poder recuperar los datos del formulario EditarNota con sus datos relacionados, poder modificarlos y que se graben correctamente.

Debes dar por hecho que las tablas ya están creadas correctamente en SQL y el ApplicationDbContext también funciona correctamente. Solamente céntrate en solucionar lo que antes te he pedido.  
  
Recuerda que es para la creación de la Web en la que estoy trabajando con VisualStudio con Net Core 7 MVC y EntityFramework Core 7.0 y C#