CTC Analista Programador. Programación 2

Obligatorio 3

Alumna: Paula Camacho

Instituto Tecnológico Colonia

ÍNDICE:

Introducción

Este proyecto es una aplicación web desarrollada con **ASP.NET Core** que busca gestionar la información de agricultores, maquinarias y registros de uso dentro de una cooperativa. A pesar de que gran parte del sistema está funcional, hay ciertos detalles que no logran ejecutarse correctamente, y he documentado esto con sinceridad para dejar en claro qué funciona y qué necesita revisión.

Estructura del Proyecto

Capas del Sistema

- Controladores: Gestionan las operaciones CRUD de cada entidad (Agricultores, Maquinarias y Registros de Uso).
- Modelos: Representan las entidades de la base de datos y sus propiedades.
- Contexto (DbContext): Configura la conexión con la base de datos y define los DbSets para las tablas.
- Configuración del Proyecto: Configurado en Program.cs, donde se registra el contexto de la base de datos y se definen los servicios necesarios.

Entidades Principales

Agricultor

```
public class Agricultor
{
    public int Id { get; set; }
    public string Nombre { get; set; }
    public int TamanoCampo { get; set; }
}
```

Maquinaria

```
public class Maquinaria
{
    public int Id { get; set; }
    public string Nombre { get; set; }
    public string Tipo { get; set; }
```

```
    public decimal CostoPorHora { get; set; }
    public int AgricultorPropietarioId { get; set; }
}
```

Registro de Uso

```
public class RegistroUso
{
    public int Id { get; set; }
    public int MaquinariaId { get; set; }
    public DateTime FechaUso { get; set; }
    public string Observaciones { get; set;
```

Controladores:

AgricultorController

Gestiona las operaciones CRUD para los agricultores:

- **GetAll:** Obtiene todos los agricultores.
- **GetById:** Busca un agricultor por su ID.
- **Create:** Crea un nuevo agricultor.
- **Update:** Actualiza un agricultor existente.
- **Delete:** Elimina un agricultor.

Ejemplo de un método:

```
[HttpGet]
public IActionResult GetAll()
{
    var agricultores = _context.Agricultores.ToList();
    return Ok(agricultores);
}
```

Problemas Encontrados:

- Error en el Contexto: A veces _context.Agricultores no se inicializa correctamente. Esto puede deberse a problemas en la configuración del contexto en Program.cs o en la base de datos.

Base de Datos

Se utiliza SQL Server y la conexión se configura en appsettings.json:

```
{
  "ConnectionStrings": {
                                                                "DefaultConnection":
"Server=(localdb)\\MSSQLLocalDB;Database=ACooperativa;Trusted_Connection=True;"
  }
}
```

Se generaron migraciones utilizando Entity Framework Core, aunque algunas tablas no se sincronizan correctamente con el modelo actual.

Atributos de Data Notation y Relaciones

Agricultor

```
public class Agricultor{

    public int Id { get; set; }

    [Required]

        [StringLength(50, ErrorMessage = "El nombre no puede superar los 50 caracteres.")]

    public string Nombre { get; set; }

     [Range(1, int.MaxValue, ErrorMessage = "El tamaño del campo debe ser mayor a 0.")]
```

```
    public int TamanoCampo { get; set; }

    public List<Maquinaria> Maquinarias { get; set; }

}
```

Maquinaria

```
public class Maquinaria

{

 public int Id { get; set; }



    [Required]

[StringLength(50, ErrorMessage = "El nombre no puede superar los 50 caracteres.")]

public string Nombre { get; set; }

 [Required]

    [StringLength(30, ErrorMessage = "El tipo no puede superar los 30 caracteres.")]

     public string Tipo {     [Range(0.01, double.MaxValue, ErrorMessage = "El costo por hora debe ser mayor a 0.")]

    public decimal CostoPorHora { get; set; }

    [Required]

    public int AgricultorPropietarioId { get; set; }

    [ForeignKey("AgricultorPropietarioId")]

    public Agricultor AgricultorPropietario { get; set; }

    public List<RegistroUso> RegistrosUso { get; set; }
```

```
}
```

RegistroUso

```
public class RegistroUso
{
    public int Id { get; set; }
    [Required]
    public int MaquinariaId { get; set; }

    [Required]
    [DataType(DataType.DateTime)]
    public DateTime FechaUso { get; set; }

     [StringLength(200, ErrorMessage = "Las observaciones no pueden superar los 200 caracteres.")]
    public string Observaciones { get; set; }

    [ForeignKey("MaquinariaId")]
    public Maquinaria Maquinaria { get; set; }
        }
```

Problemas No Resueltos

1. Mensaje de Bienvenida: No logre eliminar el mensaje "Welcome" de la página principal, a pesar de editar las vistas relacionadas. Podría estar cacheado o relacionado con el diseño por defecto de Razor Pages.
2. Controladores con Error: En el AgricultorController, las operaciones CRUD lanzan errores relacionados con _context. Esto sugiere que la inyección de dependencias no está funcionando adecuadamente.
3. No se Abre Swagger: A pesar de estar configurado en Program.cs, Swagger no se inicializa correctamente para probar los endpoints.
4. Errores de Datatype: Se solicitó agregar atributos de tipo de dato (DataType) a los modelos, pero estos no parecen reflejarse en la base de datos ni en las validaci

Conclusión

Este proyecto tiene una base funcional para la gestión de agricultores, maquinarias y registros de uso, pero necesita revisión en:

- **Contexto de Base de Datos:** Verificar la configuración y asegurar que las tablas reflejen los modelos.
- **Endpoints:** Resolver errores en los controladores para garantizar que las operaciones CRUD sean accesibles.
- **Interfaz:** Ajustar las vistas para eliminar el contenido por defecto y personalizarlas según los requerimientos.

Espero que esta documentación explique los pasos seguidos y los problemas encontrados. Estoy abierto a feedback para mejorar el proyecto y lograr que todo funcione correctamente.