Universidad ORT

**Facultad de Ingeniería**

**Programación 3**

**Obligatorio**

|  |  |
| --- | --- |
|  | Mujer sonriendo con una playera de color naranja  Descripción generada automáticamente |
| Andrés Alvarez - N.º 295384 | Lucía Layes - N.º 291613 |

Grupo N3A

Prof. Plinio Gañi

21 de octubre 2024

Índice

Contenido

[**1.- Diagrama de casos de uso.** 3](#_Toc180092125)

[**2.- Descripción narrativa de los siguientes casos de uso** 4](#_Toc180092126)

[2.1.- Alta Atleta. 4](#_Toc180092127)

[2.1.- Ingreso de puntaje de atleta. 5](#_Toc180092128)

[**3.- Diagrama de clases (UML) para todas las capas implementadas.** 6](#_Toc180092129)

[3.1.- Capa DTO. 6](#_Toc180092130)

[3.2.- Capa ExcepcionesPropias 7](#_Toc180092131)

[3.3.- Capa LogicaAplicacion 8](#_Toc180092132)

[3.4.- Capa LogicaDatos 16](#_Toc180092133)

[3.5.- Capa LogicaNegocio 19](#_Toc180092134)

[3.6.- Capa Presentación 24](#_Toc180092135)

[3.7.- Capa Api 29](#_Toc180092136)

[**4.- Código fuente de toda la aplicación incluido en la documentación.** 30](#_Toc180092137)

[4.1.-DTO. 30](#_Toc180092138)

[4.1.1- Mappers. 36](#_Toc180092139)

[4.2.- Capa ExcepcionesPropias. 42](#_Toc180092140)

[4.3.- Capa LogicaAplicacion. 46](#_Toc180092141)

[4.4.- Capa LogicaDatos. 97](#_Toc180092142)

[4.5.- Capa LogicaNegocio. 105](#_Toc180092143)

[4.6.- Capa Presentación. 118](#_Toc180092144)

[4.7.- Capa Api. 153](#_Toc180092145)

# 

# **1.- Diagrama de casos de uso.**

Diagrama

Descripción generada automáticamente

# **2.- Descripción narrativa de los siguientes casos de uso**

## 2.1.- Alta Atleta.

Tabla

Descripción generada automáticamente

## 2.1.- Ingreso de puntaje de atleta.

Tabla

Descripción generada automáticamente

# **3.- Diagrama de clases (UML) para todas las capas implementadas.**

## 3.1.- Capa DTO.

Imagen que contiene Diagrama

Descripción generada automáticamente

Diagrama

Descripción generada automáticamente

Diagrama

Descripción generada automáticamente

## 3.2.- Capa ExcepcionesPropias

Diagrama

Descripción generada automáticamente

## 3.3.- Capa LogicaAplicacion

Diagrama, Dibujo de ingeniería

Descripción generada automáticamente

Diagrama

Descripción generada automáticamente

Diagrama

Descripción generada automáticamente

Diagrama

Descripción generada automáticamente

Diagrama

Descripción generada automáticamente

Diagrama

Descripción generada automáticamente con confianza media

Diagrama

Descripción generada automáticamente

Diagrama

Descripción generada automáticamente

Diagrama

Descripción generada automáticamente

Diagrama

Descripción generada automáticamente

Diagrama

Descripción generada automáticamente

Diagrama

Descripción generada automáticamente

Diagrama

Descripción generada automáticamente

Diagrama

Descripción generada automáticamente

## 3.4.- Capa LogicaDatos

Diagrama

Descripción generada automáticamente

Diagrama

Descripción generada automáticamente  
  
Diagrama

Descripción generada automáticamente  
  
Diagrama

Descripción generada automáticamente

Diagrama

Descripción generada automáticamente

Diagrama

Descripción generada automáticamente

Diagrama

Descripción generada automáticamente  
  
  
Diagrama

Descripción generada automáticamente

## 3.5.- Capa LogicaNegocio

Diagrama

Descripción generada automáticamente  
  
Diagrama, Dibujo de ingeniería

Descripción generada automáticamente

Diagrama

Descripción generada automáticamente

Diagrama

Descripción generada automáticamente

Diagrama

Descripción generada automáticamente  
  
Diagrama

Descripción generada automáticamente

Diagrama

Descripción generada automáticamente

## 3.6.- Capa Presentación

Diagrama, Dibujo de ingeniería

Descripción generada automáticamente

Diagrama

Descripción generada automáticamente  
  
Diagrama

Descripción generada automáticamente

Diagrama

Descripción generada automáticamente  
  
Diagrama

Descripción generada automáticamente

Diagrama

Descripción generada automáticamente  
  
  
Una captura de pantalla de un celular con letras

Descripción generada automáticamente con confianza media  
  
Imagen de la pantalla de un celular con letras

Descripción generada automáticamente con confianza media  
  
Diagrama

Descripción generada automáticamente  
  
Diagrama

Descripción generada automáticamente

## 3.7.- Capa Api

Diagrama

Descripción generada automáticamente  
  
Diagrama

Descripción generada automáticamente

# **4.- Código fuente de toda la aplicación incluido en la documentación.**

## 4.1.-DTO.

using System.ComponentModel.DataAnnotations;

namespace DTO

{

public class AltaDisciplinaDTO

{

[Required]

[Display(Name = "Id")]

public int IdDisciplina { get; set; }

[Required]

[Display(Name = "Nombre")]

public string NombreDisciplina { get; set; }

[Required]

[Display(Name = "Año")]

public int AnioDisciplina { get; set; }

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DTO

{

public class AltaEventoAtletaPuntajeDTO

{

public int Id { get; set; }

[Required]

public int IdAtleta { get; set; }

[Required]

public decimal? Puntaje { get; set; }

[Required]

public int IdEvento { get; set; }

}

}

using System.ComponentModel.DataAnnotations;

namespace DTO

{

public class AltaEventoDTO

{

[Display(Name = "Id")]

public int IdEvento { get; set; }

[Required]

[Display(Name = "Nombre del Evento")]

public string NombreEvento { get; set; }

[Required]

[Display(Name = "Disciplina")]

public int IdDisciplina { get; set; }

[Required]

[Display(Name = "Fecha de Inicio")]

public DateTime FechaInicioEvento { get; set; }

[Required]

[Display(Name = "Fecha de Finalización")]

public DateTime FechaFinalEvento { get; set; }

[Required]

[Display(Name = "Atletas Participantes")]

public List<int> IdsAtletas { get; set; } = new List<int>();

}

}

using System.ComponentModel.DataAnnotations;

namespace DTO

{

public class AltaUsuarioDTO

{

[Required]

[Display(Name = "Email")]

public string EmailUsuario { get; set; }

[Required]

[Display(Name = "Contraseña")]

public string ContraseniaUsuario { get; set; }

[Required]

[Display(Name = "Rol")]

public int IdRol { get; set; }

public int IdUser { get; set; }

public string? NombreRol { get; set; }

public string? Date { get; set; }

public string? Hora { get; set; }

public string? EmailAdmin { get; set; }

}

}

using System.ComponentModel.DataAnnotations;

namespace DTO

{

public class ListadoAtletasDTO

{

[Display(Name = "Id")]

public int IdAtleta { get; set; }

[Display(Name = "Nombre")]

public string NombreAtleta { get; set; }

[Display(Name = "Apellido")]

public string ApellidoAtleta { get; set; }

[Display(Name = "Sexo")]

public bool SexoAtleta { get; set; }

[Display(Name = "Nombre-Pais")]

public string NombrePais { get; set; }

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

public int IdDisciplina { get; set; }

}

}

using System.ComponentModel.DataAnnotations;

namespace DTO

{

public class ListadoDisciplinaDTO

{

[Required]

[Display(Name = "Id")]

public int IdDisciplina { get; set; }

[Required]

[Display(Name = "Nombre")]

public string NombreDisciplina { get; set; }

[Required]

[Display(Name = "Año")]

public int AnioDisciplina { get; set; }

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DTO.Mappers

{

public class ListadoEventoAtletaDTO

{

[Display(Name = "Id Evento Atleta")]

public int Id { get; set; }

[Display(Name = "Id Atleta")]

public int IdAtleta { get; set; }

[Display(Name = "Nombre Atleta")]

public string NombreAtleta { get; set; }

[Display(Name = "Id Evento")]

public int IdEvento { get; set; }

[Display(Name = "Nombre Evento")]

public string NombreEvento { get; set; }

[Display(Name = "Puntaje Atleta")]

public decimal? PuntajeAtleta { get; set; }

[Display(Name = "Fecha del Evento")]

public DateTime FechaEvento { get; set; }

}

}

using System.ComponentModel.DataAnnotations;

namespace DTO

{

public class ListadoEventosDTO

{

[Display(Name = "Id Evento")]

public int IdEvento { get; set; }

[Display(Name = "Nombre Evento")]

public string NombreEvento { get; set; }

[Display(Name = "Nombre Disciplina")]

public string NombreDisciplina { get; set; }

[Display(Name = "Fecha Inicio")]

public DateTime FechaInicio { get; set; }

[Display(Name = "Fecha Final")]

public DateTime FechaFinal { get; set; }

}

}

using System.ComponentModel.DataAnnotations;

namespace DTO

{

public class ListadoUsuariosDTO

{

[Display(Name = "Email")]

public string EmailUsuario { get; set; }

[Display(Name = "Rol")]

public string NombreRol { get; set; }

[Display(Name = "ID-Usuario")]

public int IdUser { get; set; }

[Display(Name = "Fecha")]

public string Date { get; set; }

[Display(Name = "Hora")]

public string Hora { get; set; }

[Display(Name = "Admin-Email")]

public string EmailAdmin { get; set; }

}

}

using System.ComponentModel.DataAnnotations;

namespace DTO

{

public class LoginUsuarioDTO

{

[Required]

[Display(Name = "Email")]

public string EmailUsuario { get; set; }

[Required]

[Display(Name = "Contraseña")]

public string ContraseniaUsuario { get; set; }

[Required]

[Display(Name = "Rol")]

public int IdRol { get; set; }

}

}

using System.ComponentModel.DataAnnotations;

namespace DTO

{

public class RolDTO

{

[Display(Name = "Nombre Rol")]

public string NombreRol { get; set; }

[Display(Name = "IdRol")]

public int IdRol { get; set; }

}

}

## 4.1.1- Mappers.

using LogicaNegocio.EntidadesDominio;

namespace DTO.Mappers

{

public class MappersAtleta

{

public static Atleta FromDTO(ListadoAtletasDTO dto)

{

if (dto != null)

{

Atleta atleta = new Atleta

{

IdDisciplinas = dto.IdDisciplinas,

Disciplinas = new List<Disciplina>()

};

atleta.Id = dto.IdAtleta;

return atleta;

}

return null;

}

public static ListadoAtletasDTO FromAtleta(Atleta atleta)

{

ListadoAtletasDTO dto = new ListadoAtletasDTO()

{

IdAtleta = atleta.Id,

NombreAtleta = atleta.Nombre,

ApellidoAtleta = atleta.Apellido,

SexoAtleta = atleta.EsMasculino,

NombrePais = atleta.Pais.Nombre,

IdDisciplina = atleta.IdDisciplina

};

return dto;

}

public static IEnumerable<ListadoAtletasDTO> FromAtletas(IEnumerable<Atleta> atletas)

{

List<ListadoAtletasDTO> dtos = new List<ListadoAtletasDTO>();

foreach (Atleta item in atletas)

{

dtos.Add(FromAtleta(item));

}

return dtos;

}

}

}

using LogicaNegocio.EntidadesDominio;

using LogicaNegocio.ValueObjects;

namespace DTO.Mappers

{

public class MappersDisciplina

{

public static Disciplina FromDTO(AltaDisciplinaDTO dto)

{

if (dto != null)

{

NombreDisciplina nombreDisciplina = new NombreDisciplina(dto.NombreDisciplina);

AnioDisciplina anioDisciplina = new AnioDisciplina(dto.AnioDisciplina);

Disciplina disciplina = new Disciplina

{

Nombre = nombreDisciplina,

AnioDisciplina = anioDisciplina

};

return disciplina;

}

return null;

}

public static ListadoDisciplinaDTO FromDisciplina(Disciplina disciplina)

{

ListadoDisciplinaDTO dto = new ListadoDisciplinaDTO()

{

NombreDisciplina = disciplina.Nombre.Valor,

AnioDisciplina = disciplina.AnioDisciplina.Valor,

IdDisciplina = disciplina.Id

};

return dto;

}

public static IEnumerable<ListadoDisciplinaDTO> FromDisciplinas(IEnumerable<Disciplina> disciplinas)

{

List<ListadoDisciplinaDTO> disc = new List<ListadoDisciplinaDTO>();

foreach (Disciplina item in disciplinas)

{

disc.Add(FromDisciplina(item));

}

return disc;

}

}

}

using LogicaNegocio.EntidadesDominio;

using LogicaNegocio.ValueObjects;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DTO.Mappers

{

public class MappersEventoAtleta

{

public static EventoAtleta FromDTO(AltaEventoAtletaPuntajeDTO dto)

{

if (dto != null)

{

EventoAtleta eventoAtleta = new EventoAtleta

{

Id = dto.Id,

Atleta = null,

PuntajeAtleta = dto.Puntaje,

Evento = null

};

return eventoAtleta;

}

return null;

}

public static ListadoEventoAtletaDTO ToDTO(EventoAtleta eventoAtleta)

{

if (eventoAtleta != null)

{

ListadoEventoAtletaDTO dto = new ListadoEventoAtletaDTO

{

Id = eventoAtleta.Id,

IdAtleta = eventoAtleta.Atleta.Id,

NombreAtleta = eventoAtleta.Atleta.Nombre,

IdEvento = eventoAtleta.Evento.Id,

NombreEvento = eventoAtleta.Evento.NombreEvento.Valor,

PuntajeAtleta = eventoAtleta.PuntajeAtleta,

FechaEvento = eventoAtleta.Evento.FechaFinal

};

return dto;

}

return null;

}

public static IEnumerable<ListadoEventoAtletaDTO> ToDTOs(IEnumerable<EventoAtleta> eventosAtletas)

{

List<ListadoEventoAtletaDTO> dto = new List<ListadoEventoAtletaDTO>();

foreach (EventoAtleta item in eventosAtletas)

{

dto.Add(ToDTO(item));

}

return dto;

}

}

}

using LogicaNegocio.EntidadesDominio;

using LogicaNegocio.ValueObjects;

namespace DTO.Mappers

{

public class MappersEventos

{

public static Evento FromDTO(AltaEventoDTO dto)

{

if (dto != null)

{

NombreEvento nombreEvento = new NombreEvento(dto.NombreEvento);

Evento evento = new Evento(nombreEvento, dto.IdDisciplina, dto.FechaInicioEvento, dto.FechaFinalEvento, dto.IdsAtletas)

{

Id = dto.IdEvento

};

evento.IdDisciplina = dto.IdDisciplina;

return evento;

}

return null;

}

public static ListadoEventosDTO ToDTO(Evento evento)

{

if (evento == null) return null;

ListadoEventosDTO dto = new ListadoEventosDTO

{

IdEvento = evento.Id,

NombreEvento = evento.NombreEvento.Valor,

NombreDisciplina = evento.Disciplina.Nombre.Valor,

FechaInicio = evento.FechaInicio,

FechaFinal = evento.FechaFinal

};

return dto;

}

public static IEnumerable<ListadoEventosDTO> ToDTOs(List<Evento> eventos)

{

List<ListadoEventosDTO> dto = new List<ListadoEventosDTO>();

foreach (Evento evento in eventos)

{

dto.Add(ToDTO(evento));

}

return dto;

}

}

}

using LogicaNegocio.EntidadesDominio;

namespace DTO.Mappers

{

public class MappersRol

{

public static RolDTO ToDTO(Rol rol)

{

RolDTO dto = new RolDTO

{

IdRol = rol.Id,

NombreRol = rol.Nombre

};

return dto;

}

public static IEnumerable<RolDTO> FromRoles(IEnumerable<Rol> roles)

{

List<RolDTO> dtos = new List<RolDTO>();

foreach (Rol role in roles)

{

dtos.Add(ToDTO(role));

}

return dtos;

}

}

}

using LogicaNegocio.EntidadesDominio;

using LogicaNegocio.ValueObjects;

namespace DTO.Mappers

{

public class MappersUsuario

{

public static Usuario FromDTO(AltaUsuarioDTO dto)

{

if (dto != null)

{

EmailUsuario email = new EmailUsuario(dto.EmailUsuario);

ContraseniaUsuario contrasenia = new ContraseniaUsuario(dto.ContraseniaUsuario);

Rol rol = new Rol

{

Id = dto.IdRol,

Nombre = dto.NombreRol

};

Usuario usu = new Usuario(email, contrasenia, null, dto.Date, dto.Hora, dto.EmailAdmin);

usu.RolId = dto.IdRol;

usu.Id = dto.IdUser;

return usu;

}

return null;

}

public static AltaUsuarioDTO ToDTO(Usuario usuario)

{

AltaUsuarioDTO dto = new AltaUsuarioDTO

{

EmailUsuario = usuario.Email.Valor,

ContraseniaUsuario = usuario.Contrasenia.Valor,

IdRol = usuario.Rol.Id,

IdUser = usuario.Id,

NombreRol = usuario.Rol.Nombre,

Date = usuario.Date,

Hora = usuario.Hour,

EmailAdmin = usuario.EmailAdmin

};

return dto;

}

public static ListadoUsuariosDTO FromUsuario(Usuario usuario)

{

ListadoUsuariosDTO dto = new ListadoUsuariosDTO()

{

EmailUsuario = usuario.Email.Valor,

NombreRol = usuario.Rol.Nombre,

IdUser = usuario.Id,

Date = usuario.Date,

Hora = usuario.Hour,

EmailAdmin = usuario.EmailAdmin

};

return dto;

}

public static IEnumerable<ListadoUsuariosDTO> FromUsuarios(IEnumerable<Usuario> usuarios)

{

List<ListadoUsuariosDTO> dtos = new List<ListadoUsuariosDTO>();

foreach (Usuario usu in usuarios)

{

dtos.Add(FromUsuario(usu));

}

return dtos;

}

}

}

## 4.2.- Capa ExcepcionesPropias.

namespace ExcepcionesPropias

{

public class ExcepcionesAtleta : Exception

{

public ExcepcionesAtleta() { }

public ExcepcionesAtleta(string Mensaje) : base(Mensaje) { }

public ExcepcionesAtleta(string Mensaje, Exception innerException) : base(Mensaje, innerException) { }

}

}

namespace ExcepcionesPropias

{

public class ExcepcionesDisciplina : Exception

{

public ExcepcionesDisciplina()

{

}

public ExcepcionesDisciplina(string mensaje) : base(mensaje)

{

}

public ExcepcionesDisciplina(string mensaje, Exception innerException) : base(mensaje, innerException)

{

}

}

}

namespace ExcepcionesPropias

{

public class ExcepcionesEvento : Exception

{

public ExcepcionesEvento()

{

}

public ExcepcionesEvento(string mensaje) : base(mensaje)

{

}

public ExcepcionesEvento(string mensaje, Exception innerException) : base(mensaje, innerException)

{

}

}

}

namespace ExcepcionesPropias

{

public class ExcepcionesPais : Exception

{

public ExcepcionesPais() { }

public ExcepcionesPais(string Mensaje) : base(Mensaje) { }

public ExcepcionesPais(string Mensaje, Exception innerException) : base(Mensaje, innerException) { }

}

}

namespace ExcepcionesPropias

{

public class ExcepcionesRol : Exception

{

public ExcepcionesRol() { }

public ExcepcionesRol(string Mensaje) : base(Mensaje) { }

public ExcepcionesRol(string Mensaje, Exception innerException) : base(Mensaje, innerException) { }

}

}

namespace ExcepcionesPropias

{

public class ExcepcionesUsuario : Exception

{

public ExcepcionesUsuario() { }

public ExcepcionesUsuario(string Mensaje) : base(Mensaje) { }

public ExcepcionesUsuario(string Mensaje, Exception innerException) : base(Mensaje, innerException) { }

}

}

using DTO;

using DTO.Mappers;

using ExcepcionesPropias;

using LogicaAplicacion.InterfacesCU;

using LogicaNegocio.EntidadesDominio;

using LogicaNegocio.InterfacesRepositorios;

namespace LogicaAplicacion.CU

{

public class AltaAtleta : IAltaAtleta

{

public IRepositorioAtleta RepositorioAtleta { get; set; }

public AltaAtleta(IRepositorioAtleta repositorioAtleta)

{

RepositorioAtleta = repositorioAtleta;

}

public void UpdateAtleta(ListadoAtletasDTO dto)

{

if (dto != null)

{

Atleta atleta = MappersAtleta.FromDTO(dto);

RepositorioAtleta.Update(atleta.Id, dto.IdDisciplina);

}

else

{

throw new ExcepcionesAtleta("Atleta no puede ser vacío");

}

}

}

}

## 4.3.- Capa LogicaAplicacion.

## LOGICAAPLICACION.CU

## using DTO;

## using DTO.Mappers;

## using ExcepcionesPropias;

## using LogicaAplicacion.InterfacesCU;

## using LogicaNegocio.EntidadesDominio;

## using LogicaNegocio.InterfacesRepositorios;

## namespace LogicaAplicacion.CU

## {

## public class AltaAtleta : IAltaAtleta

## {

## public IRepositorioAtleta RepositorioAtleta { get; set; }

## public AltaAtleta(IRepositorioAtleta repositorioAtleta)

## {

## RepositorioAtleta = repositorioAtleta;

## }

## public void UpdateAtleta(ListadoAtletasDTO dto)

## {

## if (dto != null)

## {

## Atleta atleta = MappersAtleta.FromDTO(dto);

## RepositorioAtleta.Update(atleta.Id, dto.IdDisciplina);

## }

## else

## {

## throw new ExcepcionesAtleta("Atleta no puede ser vacío");

## }

## }

## }

## }

## using DTO;

## using DTO.Mappers;

## using ExcepcionesPropias;

## using LogicaAplicacion.InterfacesCU;

## using LogicaNegocio.EntidadesDominio;

## using LogicaNegocio.InterfacesRepositorios;

## namespace LogicaAplicacion.CU

## {

## public class AltaDisciplina : IAltaDisciplina

## {

## public IRepositorioDisciplina Repositorio { get; set; }

## public AltaDisciplina(IRepositorioDisciplina repositorio)

## {

## Repositorio = repositorio;

## }

## public void AltaDisci(AltaDisciplinaDTO dto)

## {

## if (dto != null)

## {

## Disciplina disci = MappersDisciplina.FromDTO(dto);

## Repositorio.Add(disci);

## }

## else

## {

## throw new ExcepcionesDisciplina("No existe disciplina");

## }

## }

## }

## }

## using DTO;

## using DTO.Mappers;

## using ExcepcionesPropias;

## using LogicaAplicacion.InterfacesCU;

## using LogicaNegocio.EntidadesDominio;

## using LogicaNegocio.InterfacesRepositorios;

## namespace LogicaAplicacion.CU

## {

## public class AltaEvento : IAltaEvento

## {

## public IRepositorioEvento Repositorio { get; set; }

## public IRepositorioDisciplina RepositorioDisciplina { get; set; }

## public IRepositorioAtleta RepositorioAtleta { get; set; }

## public AltaEvento(IRepositorioEvento repositorio, IRepositorioDisciplina repositorioDisciplina,

## IRepositorioAtleta repositorioAtleta)

## {

## Repositorio = repositorio;

## RepositorioDisciplina = repositorioDisciplina;

## RepositorioAtleta = repositorioAtleta;

## }

## public void AltaEve(AltaEventoDTO dto)

## {

## if (Repositorio.FindByName(dto.NombreEvento) != null)

## {

## throw new ExcepcionesEvento("Ya existe un evento con el mismo nombre.");

## }

## Evento evento = MappersEventos.FromDTO(dto);

## Disciplina disciplina = RepositorioDisciplina.FindById(evento.IdDisciplina);

## if (disciplina == null)

## {

## throw new ExcepcionesEvento("La disciplina seleccionada no existe.");

## }

## evento.Disciplina = disciplina;

## if (dto.IdsAtletas.Count < 3)

## {

## throw new ExcepcionesEvento("Se requiere un mínimo de tres atletas para registrar un evento.");

## }

## List<EventoAtleta> EventoAtletas = new List<EventoAtleta>();

## foreach (int atletaId in dto.IdsAtletas)

## {

## Atleta atleta = RepositorioAtleta.FindById(atletaId);

## if (atleta == null)

## {

## throw new ExcepcionesEvento($"El atleta con ID {atletaId} no está registrado en el sistema.");

## }

## if (!atleta.Disciplinas.Any(d => d.Id == evento.Disciplina.Id))

## {

## throw new ExcepcionesEvento($"El atleta con ID {atletaId} no está registrado para la disciplina del evento.");

## }

## EventoAtleta eventoAtleta = new EventoAtleta

## {

## Atleta = atleta,

## Evento = null

## };

## EventoAtletas.Add(eventoAtleta);

## }

## evento.EventosAtletas = EventoAtletas;

## Repositorio.Add(evento);

## }

## }

## }

## using DTO;

## using DTO.Mappers;

## using ExcepcionesPropias;

## using LogicaAplicacion.InterfacesCU;

## using LogicaNegocio.EntidadesDominio;

## using LogicaNegocio.InterfacesRepositorios;

## using System;

## using System.Collections.Generic;

## using System.Linq;

## using System.Text;

## using System.Threading.Tasks;

## namespace LogicaAplicacion.CU

## {

## public class AltaEventoPuntaje : IAltaEventoAtletaPuntaje

## {

## public IRepositorioEventoAtleta Repositorio { get; set; }

## public AltaEventoPuntaje(IRepositorioEventoAtleta repositorio)

## {

## Repositorio = repositorio;

## }

## public void AltaPuntaje(AltaEventoAtletaPuntajeDTO dto)

## {

## if (dto != null)

## {

## if (dto.Puntaje < 0)

## {

## throw new ExcepcionesEvento("Puntaje no puede ser menor a cero");

## }

## if (dto.Puntaje > 100)

## {

## throw new ExcepcionesEvento("Puntaje no puede ser mayor a cien");

## }

## Repositorio.Update(MappersEventoAtleta.FromDTO(dto));

## }

## }

## }

## }

## using DTO;

## using DTO.Mappers;

## using ExcepcionesPropias;

## using LogicaAplicacion.InterfacesCU;

## using LogicaNegocio.EntidadesDominio;

## using LogicaNegocio.InterfacesRepositorios;

## namespace LogicaAplicacion.CU

## {

## public class AltaUsuario : IAltaUsuario

## {

## public IRepositorioUsuario Repositorio { get; set; }

## public IRepositorioRoles RepositorioRoles { get; set; }

## public AltaUsuario(IRepositorioUsuario repositorio, IRepositorioRoles repositorioRoles)

## {

## Repositorio = repositorio;

## RepositorioRoles = repositorioRoles;

## }

## public void Alta(AltaUsuarioDTO dto)

## {

## Usuario usuario = MappersUsuario.FromDTO(dto);

## Rol rol = RepositorioRoles.FindById(usuario.RolId);

## if (rol == null)

## {

## throw new ExcepcionesUsuario("El rol que usted selecciono no existe");

## }

## usuario.Rol = rol;

## Repositorio.Add(usuario);

## }

## }

## }

## using ExcepcionesPropias;

## using LogicaAplicacion.InterfacesCU;

## using LogicaNegocio.EntidadesDominio;

## using LogicaNegocio.InterfacesRepositorios;

## namespace LogicaAplicacion.CU

## {

## public class BajaUsuario : IBajaUsuario

## {

## public IRepositorioUsuario RepositorioUsuario { get; set; }

## public BajaUsuario(IRepositorioUsuario repositorioUsuario)

## {

## RepositorioUsuario = repositorioUsuario;

## }

## public void BajaUser(string email)

## {

## if (email != null)

## {

## Usuario user = RepositorioUsuario.FindByMail(email);

## if (user != null)

## {

## RepositorioUsuario.Remove(user.Id);

## }

## else

## {

## throw new ExcepcionesUsuario("Usuario no encontrado");

## }

## }

## else

## {

## throw new ExcepcionesUsuario("Email no puede ser nulo");

## }

## }

## }

## }

## using DTO;

## using DTO.Mappers;

## using LogicaAplicacion.InterfacesCU;

## using LogicaNegocio.InterfacesRepositorios;

## namespace LogicaAplicacion.CU

## {

## public class BuscarUsuarioPorId : IBuscarUsuarioPorID

## {

## public IRepositorioUsuario RepositorioUsuario { get; set; }

## public BuscarUsuarioPorId(IRepositorioUsuario repositorioUsuario)

## {

## RepositorioUsuario = repositorioUsuario;

## }

## public AltaUsuarioDTO BuscarUsuarioId(int id)

## {

## return MappersUsuario.ToDTO(RepositorioUsuario.FindById(id));

## }

## }

## }

## using DTO;

## using DTO.Mappers;

## using ExcepcionesPropias;

## using LogicaAplicacion.InterfacesCU;

## using LogicaNegocio.InterfacesRepositorios;

## namespace LogicaAplicacion.CU

## {

## public class ListadoAtletas : IListadoAtletas

## {

## public IRepositorioAtleta Repositorio { get; set; }

## public ListadoAtletas(IRepositorioAtleta repositorio)

## {

## Repositorio = repositorio;

## }

## public ListadoAtletasDTO GetAtletaPorId(int id)

## {

## if (id != 0 || id > 0)

## {

## return MappersAtleta.FromAtleta(Repositorio.FindById(id));

## }

## throw new ExcepcionesAtleta("Atleta no encontrado");

## }

## public IEnumerable<ListadoAtletasDTO> GetAtletas()

## {

## return MappersAtleta.FromAtletas(Repositorio.FindAll());

## }

## 

## }

## }

## using DTO;

## using DTO.Mappers;

## using ExcepcionesPropias;

## using LogicaAplicacion.InterfacesCU;

## using LogicaNegocio.EntidadesDominio;

## using LogicaNegocio.InterfacesRepositorios;

## namespace LogicaAplicacion.CU

## {

## public class ListadoDisciplina : IListadoDisciplina

## {

## public IRepositorioDisciplina Repositorio { get; set; }

## public IRepositorioAtleta RepositorioAtleta { get; set; }

## public ListadoDisciplina(IRepositorioDisciplina repositorio, IRepositorioAtleta repositorioAtleta)

## {

## Repositorio = repositorio;

## RepositorioAtleta = repositorioAtleta;

## }

## public IEnumerable<ListadoDisciplinaDTO> GetDisciplinas()

## {

## return MappersDisciplina.FromDisciplinas(Repositorio.FindAll());

## }

## public IEnumerable<ListadoDisciplinaDTO> GetDisciplinasDisponibles(int idAtleta)

## {

## return MappersDisciplina.FromDisciplinas(RepositorioAtleta.FindAllDisponible(idAtleta));

## }

## public IEnumerable<ListadoDisciplinaDTO> GetDisciplinasRegistradas(int idAtleta)

## {

## return MappersDisciplina.FromDisciplinas(RepositorioAtleta.FindAllRegistrada(idAtleta));

## }

## public Disciplina FindById(int idDisciplina)

## {

## var disciplina = Repositorio.FindById(idDisciplina);

## if (disciplina == null)

## {

## throw new ExcepcionesDisciplina("Disciplina no encontrada.");

## }

## return disciplina;

## }

## }

## }

## using DTO.Mappers;

## using LogicaAplicacion.InterfacesCU;

## using LogicaNegocio.EntidadesDominio;

## using LogicaNegocio.InterfacesRepositorios;

## using System;

## using System.Collections.Generic;

## using System.Linq;

## using System.Text;

## using System.Threading.Tasks;

## namespace LogicaAplicacion.CU

## {

## public class ListadoEventoAtletas : IListadoEventosAtletas

## {

## public IRepositorioEventoAtleta RepositorioEventoAtleta { get; set; }

## public ListadoEventoAtletas(IRepositorioEventoAtleta repositorioEventoAtleta)

## {

## RepositorioEventoAtleta = repositorioEventoAtleta;

## }

## public IEnumerable<ListadoEventoAtletaDTO> GetAtletasPorEvento(int idEvento)

## {

## IEnumerable<EventoAtleta> eventoAtletas = new List<EventoAtleta>();

## eventoAtletas = RepositorioEventoAtleta.ObtenerAtletasPorEvento(idEvento);

## return MappersEventoAtleta.ToDTOs(eventoAtletas);

## }

## public IEnumerable<ListadoEventoAtletaDTO> GetEventosPorAtleta(int idAtleta)

## {

## IEnumerable<EventoAtleta> EventosPorAtleta = new List<EventoAtleta>();

## EventosPorAtleta = RepositorioEventoAtleta.ObtenerEventosPorAtleta(idAtleta);

## return MappersEventoAtleta.ToDTOs(EventosPorAtleta);

## }

## }

## }

## using DTO;

## using DTO.Mappers;

## using ExcepcionesPropias;

## using LogicaAplicacion.InterfacesCU;

## using LogicaNegocio.EntidadesDominio;

## using LogicaNegocio.InterfacesRepositorios;

## namespace LogicaAplicacion.CU

## {

## public class ListadoEventos : IListadoEventos

## {

## public IRepositorioEvento RepositorioEvento { get; set; }

## public IRepositorioAtleta RepositorioAtleta { get; set; }

## public IRepositorioDisciplina RepositorioDisciplina { get; set; }

## public ListadoEventos(IRepositorioEvento repositorioEvento,

## IRepositorioAtleta repositorioAtleta, IRepositorioDisciplina repositorioDisciplina)

## {

## RepositorioEvento = repositorioEvento;

## RepositorioAtleta = repositorioAtleta;

## RepositorioDisciplina = repositorioDisciplina;

## }

## public IEnumerable<ListadoEventosDTO> GetEventosPorFecha(DateTime fecha)

## {

## IEnumerable<Evento> eventos = RepositorioEvento.GetEventosPorFecha(fecha);

## var eventosDTO = MappersEventos.ToDTOs(eventos.ToList());

## return eventosDTO;

## }

## }

## }

## using DTO;

## using DTO.Mappers;

## using LogicaAplicacion.InterfacesCU;

## using LogicaNegocio.InterfacesRepositorios;

## namespace LogicaAplicacion.CU

## {

## public class ListadoRoles : IListadoRoles

## {

## public IRepositorioRoles Repositorio { get; set; }

## public ListadoRoles(IRepositorioRoles repositorio)

## {

## Repositorio = repositorio;

## }

## public IEnumerable<RolDTO> ObtenerListado()

## {

## return MappersRol.FromRoles(Repositorio.FindAll());

## }

## }

## }

## using DTO;

## using DTO.Mappers;

## using LogicaAplicacion.InterfacesCU;

## using LogicaNegocio.EntidadesDominio;

## using LogicaNegocio.InterfacesRepositorios;

## namespace LogicaAplicacion.CU

## {

## public class ListadoUsuarios : IListadoUsuarios

## {

## public IRepositorioUsuario Repositorio { get; set; }

## public ListadoUsuarios(IRepositorioUsuario repositorio)

## {

## Repositorio = repositorio;

## }

## public IEnumerable<ListadoUsuariosDTO> ObtenerListado()

## {

## IEnumerable<Usuario> usuarios = Repositorio.FindAll();

## return MappersUsuario.FromUsuarios(usuarios);

## }

## }

## }

## using ExcepcionesPropias;

## using LogicaAplicacion.InterfacesCU;

## using LogicaNegocio.EntidadesDominio;

## using LogicaNegocio.InterfacesRepositorios;

## namespace LogicaAplicacion.CU

## {

## public class LoginUsuario : ILoginUsuario

## {

## public IRepositorioUsuario RepositorioUsuario { get; set; }

## public LoginUsuario(IRepositorioUsuario repositorioUsuario)

## {

## RepositorioUsuario = repositorioUsuario;

## }

## public Usuario FindByMail(string email)

## {

## if (email != null)

## {

## Usuario user = RepositorioUsuario.FindByMail(email);

## return user;

## }

## else

## {

## throw new ExcepcionesUsuario("Email y/o Contraseña no pueden ser vacíos");

## }

## }

## }

## }

## using DTO;

## using DTO.Mappers;

## using ExcepcionesPropias;

## using LogicaAplicacion.InterfacesCU;

## using LogicaNegocio.InterfacesRepositorios;

## namespace LogicaAplicacion.CU

## {

## public class UpdateUsuario : IUpdateUsuario

## {

## public IRepositorioUsuario RepositorioUsuario { get; set; }

## public UpdateUsuario(IRepositorioUsuario repositorioUsuario)

## {

## RepositorioUsuario = repositorioUsuario;

## }

## public void UpdateUser(AltaUsuarioDTO dto)

## {

## if (dto != null)

## {

## RepositorioUsuario.Update(MappersUsuario.FromDTO(dto));

## }

## else

## {

## throw new ExcepcionesUsuario("Usuario no encontrado");

## }

## }

## }

## }

## LOGICAAPLIACION.INTERFACESCU

## using DTO;

## namespace LogicaAplicacion.InterfacesCU

## {

## public interface IAltaAtleta

## {

## void UpdateAtleta(ListadoAtletasDTO dto);

## }

## }

## using DTO;

## namespace LogicaAplicacion.InterfacesCU

## {

## public interface IAltaDisciplina

## {

## void AltaDisci(AltaDisciplinaDTO dto);

## }

## }

## using DTO;

## namespace LogicaAplicacion.InterfacesCU

## {

## public interface IAltaEvento

## {

## void AltaEve(AltaEventoDTO dto);

## }

## }

## using DTO;

## using System;

## using System.Collections.Generic;

## using System.Linq;

## using System.Text;

## using System.Threading.Tasks;

## namespace LogicaAplicacion.InterfacesCU

## {

## public interface IAltaEventoAtletaPuntaje

## {

## void AltaPuntaje(AltaEventoAtletaPuntajeDTO dto);

## }

## }

## using DTO;

## namespace LogicaAplicacion.InterfacesCU

## {

## public interface IAltaUsuario

## {

## void Alta(AltaUsuarioDTO nuevo);

## }

## }

## namespace LogicaAplicacion.InterfacesCU

## {

## public interface IBajaUsuario

## {

## void BajaUser(string email);

## }

## }

## using DTO;

## namespace LogicaAplicacion.InterfacesCU

## {

## public interface IBuscarUsuarioPorID

## {

## AltaUsuarioDTO BuscarUsuarioId(int id);

## }

## }

## using DTO;

## namespace LogicaAplicacion.InterfacesCU

## {

## public interface IListadoAtletas

## {

## ListadoAtletasDTO GetAtletaPorId(int id);

## IEnumerable<ListadoAtletasDTO> GetAtletas();

## }

## }

## using DTO;

## using LogicaNegocio.EntidadesDominio;

## namespace LogicaAplicacion.InterfacesCU

## {

## public interface IListadoDisciplina

## {

## Disciplina FindById(int idDisciplina);

## IEnumerable<ListadoDisciplinaDTO> GetDisciplinas();

## IEnumerable<ListadoDisciplinaDTO> GetDisciplinasDisponibles(int idAtleta);

## IEnumerable<ListadoDisciplinaDTO> GetDisciplinasRegistradas(int idAtleta);

## }

## }

## using DTO;

## using LogicaNegocio.EntidadesDominio;

## namespace LogicaAplicacion.InterfacesCU

## {

## public interface IListadoEventos

## {

## IEnumerable<ListadoEventosDTO> GetEventosPorFecha(DateTime fecha);

## }

## }

## using DTO.Mappers;

## using System;

## using System.Collections.Generic;

## using System.Linq;

## using System.Text;

## using System.Threading.Tasks;

## namespace LogicaAplicacion.InterfacesCU

## {

## public interface IListadoEventosAtletas

## {

## IEnumerable<ListadoEventoAtletaDTO> GetAtletasPorEvento(int idEvento);

## IEnumerable<ListadoEventoAtletaDTO> GetEventosPorAtleta(int idAtleta);

## }

## }

## using DTO;

## namespace LogicaAplicacion.InterfacesCU

## {

## public interface IListadoRoles

## {

## IEnumerable<RolDTO> ObtenerListado();

## }

## }

## using DTO;

## namespace LogicaAplicacion.InterfacesCU

## {

## public interface IListadoUsuarios

## {

## IEnumerable<ListadoUsuariosDTO> ObtenerListado();

## }

## }

## using LogicaNegocio.EntidadesDominio;

## namespace LogicaAplicacion.InterfacesCU

## {

## public interface ILoginUsuario

## {

## Usuario FindByMail(string email);

## }

## }

## using DTO;

## namespace LogicaAplicacion.InterfacesCU

## {

## public interface IUpdateUsuario

## {

## void UpdateUser(AltaUsuarioDTO dto);

## }

## }

## LOGICADATOS.REPOSITORIOS

## using LogicaNegocio.EntidadesDominio;

## using Microsoft.EntityFrameworkCore;

## namespace LogicaDatos.Repositorios

## {

## public class OlimpiadasContext : DbContext

## {

## public DbSet<Usuario> Usuarios { get; set; }

## public DbSet<Rol> Roles { get; set; }

## public DbSet<Pais> Paises { get; set; }

## public DbSet<Evento> Eventos { get; set; }

## public DbSet<Disciplina> Disciplinas { get; set; }

## public DbSet<Delegado> Delegados { get; set; }

## public DbSet<Atleta> Atletas { get; set; }

## public DbSet<EventoAtleta> EventoAtleta { get; set; }

## public OlimpiadasContext(DbContextOptions options) : base(options)

## {

## }

## }

## }

## using ExcepcionesPropias;

## using LogicaNegocio.EntidadesDominio;

## using LogicaNegocio.InterfacesRepositorios;

## using Microsoft.EntityFrameworkCore;

## namespace LogicaDatos.Repositorios

## {

## public class RepositorioAtletaBD : IRepositorioAtleta

## {

## public OlimpiadasContext Context { get; set; }

## public RepositorioAtletaBD(OlimpiadasContext context)

## {

## Context = context;

## }

## public Atleta FindById(int id)

## {

## if (id != 0 || id > 0)

## {

## return Context.Atletas

## .Include(atleta => atleta.Pais)

## .Include(atleta => atleta.Disciplinas)

## .Include(atleta => atleta.Pais.Delegado)

## .Where(atleta => atleta.Id == id)

## .SingleOrDefault();

## }

## else

## {

## throw new ExcepcionesAtleta("No existe Atleta");

## }

## }

## public IEnumerable<Atleta> FindAll()

## {

## return Context.Atletas

## .Include(atleta => atleta.Pais)

## .OrderBy(atleta => atleta.Pais.Nombre)

## .ThenBy(atleta => atleta.Apellido)

## .ThenBy(atleta => atleta.Nombre)

## .ToList();

## }

## public void Update(int atletaId, int idDisciplina)

## {

## var atleta = Context.Atletas.Include(atleta => atleta.Disciplinas)

## .FirstOrDefault(atleta => atleta.Id == atletaId);

## if (atleta != null)

## {

## var nuevaDisciplina = Context.Disciplinas.FirstOrDefault(disiciplina => disiciplina.Id == idDisciplina);

## if (nuevaDisciplina != null)

## {

## atleta.Disciplinas.Add(nuevaDisciplina);

## Context.SaveChanges();

## }

## else

## {

## throw new ExcepcionesDisciplina($"Disciplina con ID {idDisciplina} no encontrada.");

## }

## }

## else

## {

## throw new ExcepcionesAtleta("Atleta no encontrado");

## }

## }

## public IEnumerable<Disciplina> FindAllDisponible(int idAtleta)

## {

## if (idAtleta != 0 || idAtleta > 0)

## {

## var disciplinasRegistradas = Context.Atletas

## .Where(at => at.Id == idAtleta)

## .SelectMany(atleta => atleta.Disciplinas)

## .Select(disci => disci.Id)

## .ToList();

## return Context.Disciplinas

## .Where(disci => !disciplinasRegistradas.Contains(disci.Id))

## .ToList();

## }

## else

## {

## throw new ExcepcionesAtleta("Atleta no encontrado");

## }

## }

## public IEnumerable<Disciplina> FindAllRegistrada(int idAtleta)

## {

## if (idAtleta != 0 || idAtleta > 0)

## {

## var disciplinasRegistradas = Context.Atletas

## .Where(atleta => atleta.Id == idAtleta)

## .SelectMany(atleta => atleta.Disciplinas)

## .Select(disciplina => disciplina.Id)

## .ToList();

## return Context.Disciplinas

## .Where(disciplina => disciplinasRegistradas.Contains(disciplina.Id))

## .ToList();

## }

## else

## {

## throw new ExcepcionesAtleta("Atleta no encontrado");

## }

## }

## }

## }

## using LogicaNegocio.EntidadesDominio;

## using LogicaNegocio.InterfacesRepositorios;

## namespace LogicaDatos.Repositorios

## {

## public class RepositorioDelegadoBD : IRepositorioDelegado

## {

## public OlimpiadasContext Context { get; set; }

## public RepositorioDelegadoBD(OlimpiadasContext context)

## {

## Context = context;

## }

## public IEnumerable<Delegado> FindAll()

## {

## return Context.Delegados.ToList();

## }

## public Delegado FindById(int id)

## {

## return Context.Delegados.

## Where(dele => dele.Id == id).

## SingleOrDefault();

## }

## public Delegado FindByName(string nombre)

## {

## return Context.Delegados.

## Where(dele => dele.Nombre == nombre).

## SingleOrDefault();

## }

## }

## }

## using ExcepcionesPropias;

## using LogicaNegocio.EntidadesDominio;

## using LogicaNegocio.InterfacesRepositorios;

## namespace LogicaDatos.Repositorios

## {

## public class RepositorioDisciplinaBD : IRepositorioDisciplina

## {

## public OlimpiadasContext Context { get; set; }

## public RepositorioDisciplinaBD(OlimpiadasContext context)

## {

## Context = context;

## }

## public Disciplina Add(Disciplina obj)

## {

## if (obj != null)

## {

## if (FindByName(obj.Nombre.Valor) == null)

## {

## Context.Disciplinas.Add(obj);

## Context.SaveChanges();

## return obj;

## }

## else

## {

## throw new ExcepcionesDisciplina("Disciplina ya existente");

## }

## }

## else

## {

## throw new ExcepcionesDisciplina("No se encuentra la disciplina");

## }

## }

## public IEnumerable<Disciplina> FindAll()

## {

## return Context.Disciplinas.

## OrderBy(d => d.Nombre.Valor).

## ToList();

## }

## public List<Disciplina> FindAllDisciplinasById(List<int> ids)

## {

## return Context.Disciplinas.Where(disci => ids.Contains(disci.Id)).ToList();

## }

## public Disciplina FindById(int id)

## {

## return Context.Disciplinas.

## Where(disci => disci.Id == id).

## SingleOrDefault();

## }

## public Disciplina FindByName(string name)

## {

## if (name != null)

## {

## return Context.Disciplinas.Where(disc => disc.Nombre.Valor == name).SingleOrDefault();

## }

## else

## {

## throw new ExcepcionesDisciplina("Nombre de disciplina no encontrado");

## }

## }

## public List<Disciplina> FindAllByAtletaId(int atletaId)

## {

## return Context.Disciplinas

## .Where(disci => disci.Atletas.Any(atle => atle.Id == atletaId))

## .ToList();

## }

## }

## }

## 4.4.- Capa LogicaDatos.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaDatos.Repositorios

{

using ExcepcionesPropias;

using LogicaNegocio.EntidadesDominio;

using LogicaNegocio.InterfacesRepositorios;

using Microsoft.EntityFrameworkCore;

namespace LogicaDatos.Repositorios

{

public class RepositorioEventoAtletaBD : IRepositorioEventoAtleta

{

public OlimpiadasContext Context { get; set; }

public RepositorioEventoAtletaBD(OlimpiadasContext context)

{

Context = context;

}

public void Add(EventoAtleta obj)

{

Context.EventoAtleta.Add(obj);

Context.SaveChanges();

}

public void Update(EventoAtleta obj)

{

var Existe = Context.EventoAtleta

.FirstOrDefault(eveatl => eveatl.Id == obj.Id);

if (Existe != null)

{

Context.Entry(Existe).State = EntityState.Detached;

}

Context.EventoAtleta.Update(obj);

Context.SaveChanges();

}

public EventoAtleta FindById(int id)

{

return Context.EventoAtleta.Find(id);

}

public IEnumerable<EventoAtleta> FindAll()

{

return Context.EventoAtleta.ToList();

}

public void Remove(int id)

{

var eventoAtleta = FindById(id);

if (eventoAtleta != null)

{

Context.EventoAtleta.Remove(eventoAtleta);

Context.SaveChanges();

}

}

public IEnumerable<EventoAtleta> ObtenerAtletasPorEvento(int idEvento)

{

return Context.EventoAtleta

.Where(eve => eve.Evento.Id == idEvento)

.Include(eve => eve.Atleta)

.Include(eve => eve.Evento)

.ToList();

}

public int FindIdByAtletaEvento(int idEvento, int idAtleta)

{

var eventoAtleta = Context.EventoAtleta

.FirstOrDefault(eveatl => eveatl.Evento.Id == idEvento && eveatl.Atleta.Id == idAtleta);

if (eventoAtleta != null)

{

return eventoAtleta.Id;

}

throw new ExcepcionesEvento("No se encontró la relación entre el atleta y el evento.");

}

public IEnumerable<EventoAtleta> ObtenerEventosPorAtleta(int idAtleta)

{

return Context.EventoAtleta

.Where(eveatl => eveatl.Atleta.Id == idAtleta)

.Include(eveatl => eveatl.Evento)

.Include(eveatl => eveatl.Evento.Disciplina)

.Include(eveatl => eveatl.Atleta)

.OrderBy(eveatl => eveatl.Evento.Disciplina.Nombre.Valor)

.ToList();

}

}

}

}

using ExcepcionesPropias;

using LogicaNegocio.EntidadesDominio;

using LogicaNegocio.InterfacesRepositorios;

using Microsoft.EntityFrameworkCore;

namespace LogicaDatos.Repositorios

{

public class RepositorioEventoBD : IRepositorioEvento

{

public OlimpiadasContext Context { get; set; }

public RepositorioEventoBD(OlimpiadasContext context)

{

Context = context;

}

public void Add(Evento obj)

{

obj.Validate();

if (obj != null)

{

Context.Eventos.Add(obj);

Context.SaveChanges();

}

else

{

throw new ExcepcionesEvento("No se encuentra el evento");

}

}

public void Update(Evento obj)

{

if (obj != null)

{

Context.Eventos.Update(obj);

Context.SaveChanges();

}

else

{

throw new ExcepcionesEvento("No se encuentra el evento");

}

}

public Evento FindById(int id)

{

var evento = Context.Eventos

.Include(eve => eve.Disciplina)

.SingleOrDefault(eve => eve.Id == id);

if (evento == null)

{

throw new ExcepcionesEvento("Evento no encontrado.");

}

return evento;

}

public IEnumerable<Evento> FindAll()

{

return Context.Eventos

.Include(eve => eve.Disciplina)

.Include(eve => eve.EventosAtletas)

.ToList();

}

public void Remove(int id)

{

var eve = Context.Eventos.Find(id);

if (eve != null)

{

Context.Eventos.Remove(eve);

Context.SaveChanges();

}

else

{

throw new ExcepcionesEvento("No se encuentra el evento");

}

}

public IEnumerable<Evento> GetEventosPorFecha(DateTime fecha)

{

return Context.Eventos

.Where(eve => eve.FechaFinal == fecha)

.Include(eve => eve.Disciplina)

.Include(eve => eve.EventosAtletas)

.ToList();

}

public Evento FindByName(string nombreEvento)

{

if (nombreEvento != null)

{

return Context.Eventos

.Include(eve => eve.Disciplina)

.Where(eve => eve.NombreEvento.Valor == nombreEvento)

.SingleOrDefault();

}

else

{

throw new ExcepcionesEvento("Nombre no encontrado");

}

}

}

}

using LogicaNegocio.EntidadesDominio;

using LogicaNegocio.InterfacesRepositorios;

using Microsoft.EntityFrameworkCore;

using System.Data;

namespace LogicaDatos.Repositorios

{

public class RepositorioPaisBD : IRepositorioPais

{

public OlimpiadasContext Context { get; set; }

public RepositorioPaisBD(OlimpiadasContext context)

{

Context = context;

}

public IEnumerable<Pais> FindAll()

{

return Context.Paises.ToList();

}

public Pais FindById(int id)

{

return Context.Paises.Include(pais => pais.Delegado).

Where(pais => pais.Id == id).

SingleOrDefault();

}

public Pais FindByName(string nombre)

{

return Context.Paises.Include(pais => pais.Delegado).

Where(pais => pais.Nombre == nombre).

SingleOrDefault();

}

}

}

using ExcepcionesPropias;

using LogicaNegocio.EntidadesDominio;

using LogicaNegocio.InterfacesRepositorios;

namespace LogicaDatos.Repositorios

{

public class RepositorioRolesBD : IRepositorioRoles

{

public OlimpiadasContext Context { get; set; }

public RepositorioRolesBD(OlimpiadasContext context)

{

Context = context;

}

public void Add(Rol obj)

{

if (obj == null)

{

throw new ExcepcionesRol("No se encuentra el rol");

}

if (FindByName(obj.Nombre) != null)

{

throw new ExcepcionesRol("Este nombre ya está en uso");

}

Context.Roles.Add(obj);

Context.SaveChanges();

}

public IEnumerable<Rol> FindAll()

{

return Context.Roles.ToList();

}

public Rol FindById(int id)

{

if (id != null)

{

return Context.Roles.Find(id);

}

throw new ExcepcionesRol("Rol nulo");

}

public void Remove(int id)

{

Rol rol = Context.Roles.Find(id);

if (rol != null)

{

Context.Roles.Remove(rol);

Context.SaveChanges();

}

else

{

throw new ExcepcionesRol("No se encuentra el Rol");

}

}

public void Update(Rol obj)

{

Rol rol = FindByName(obj.Nombre);

if (rol == null)

{

Context.Roles.Update(obj);

Context.SaveChanges();

}

else

{

throw new Exception("Rol no encontrado.");

}

}

public Rol FindByName(string name)

{

if (name != null)

{

return Context.Roles.Where(rol => rol.Nombre == name).SingleOrDefault();

}

else

{

throw new ExcepcionesRol("Nombre de Rol no encontrado");

}

}

}

}

using ExcepcionesPropias;

using LogicaNegocio.EntidadesDominio;

using LogicaNegocio.InterfacesRepositorios;

using Microsoft.EntityFrameworkCore;

namespace LogicaDatos.Repositorios

{

public class RepositorioUsuarioBD : IRepositorioUsuario

{

public OlimpiadasContext Context { get; set; }

public RepositorioUsuarioBD(OlimpiadasContext context)

{

Context = context;

}

public void Add(Usuario obj)

{

obj.Validate();

if (obj == null)

{

throw new ExcepcionesUsuario("No se encuentra el usuario");

}

if (FindByMail(obj.Email.Valor) != null)

{

throw new ExcepcionesUsuario("Este email ya está en uso");

}

Context.Usuarios.Add(obj);

Context.SaveChanges();

}

public void Update(Usuario obj)

{

if (obj != null)

{

obj.Validate();

Context.Usuarios.Update(obj);

Context.SaveChanges();

}

else

{

throw new ExcepcionesUsuario("Usuario no Encontrado");

}

}

public Usuario FindById(int id)

{

return Context.Usuarios.Include(usu => usu.Rol).Where(usu => usu.Id == id).SingleOrDefault();

}

public IEnumerable<Usuario> FindAll()

{

return Context.Usuarios.Include(usu => usu.Rol).ToList();

}

public void Remove(int id)

{

Usuario user = Context.Usuarios.Find(id);

if (user != null)

{

Context.Usuarios.Remove(user);

Context.SaveChanges();

}

else

{

throw new ExcepcionesUsuario("No se encuentra el usuario");

}

}

public Usuario FindByMail(string mail)

{

if (mail != null)

{

return Context.Usuarios.Include(usu => usu.Rol).Where(usu => usu.Email.Valor == mail).SingleOrDefault();

}

else

{

throw new ExcepcionesUsuario("Email no encontrado");

}

}

}

}

## 4.5.- Capa LogicaNegocio.

LOGICANEGOCIO.ENTIDADESDOMINIO

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace LogicaNegocio.EntidadesDominio

{

[Table("Atletas")]

public class Atleta

{

public int Id { get; set; }

[Required]

public string Nombre { get; set; }

[Required]

public string Apellido { get; set; }

[Required]

public bool EsMasculino { get; set; }

[Required]

public Pais Pais { get; set; }

[Required]

public List<Disciplina> Disciplinas { get; set; }

[NotMapped]

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

[NotMapped]

public int IdDisciplina { get; set; }

public List<EventoAtleta>? EventosAtletas { get; set; }

[NotMapped]

public List<int>? IdsEventosAtletas { get; set; }

}

}

using Microsoft.EntityFrameworkCore;

namespace LogicaNegocio.EntidadesDominio

{

[Owned]

public class Delegado

{

public int Id { get; set; }

public string Nombre { get; set; }

public int Telefono { get; set; }

}

}

using LogicaNegocio.ValueObjects;

namespace LogicaNegocio.EntidadesDominio

{

public class Disciplina

{

public int Id { get; set; }

public NombreDisciplina Nombre { get; set; }

public AnioDisciplina AnioDisciplina { get; set; }

public List<Atleta> Atletas { get; set; }

}

}

using ExcepcionesPropias;

using LogicaNegocio.InterfacesDominio;

using LogicaNegocio.ValueObjects;

using System.ComponentModel.DataAnnotations.Schema;

namespace LogicaNegocio.EntidadesDominio

{

public class Evento : IValidable

{

public int Id { get; set; }

public NombreEvento NombreEvento { get; set; }

public Disciplina Disciplina { get; set; }

[NotMapped]

public int IdDisciplina { get; set; }

public DateTime FechaInicio { get; set; }

public DateTime FechaFinal { get; set; }

public List<EventoAtleta> EventosAtletas { get; set; }

[NotMapped]

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

protected Evento() { }

public Evento(NombreEvento nombreEvento, int idDisciplina, DateTime fechaInicio, DateTime fechaFinal, List<int> idsEventosAtletas)

{

NombreEvento = nombreEvento;

IdDisciplina = idDisciplina;

FechaInicio = fechaInicio;

FechaFinal = fechaFinal;

IdsEventosAtletas = idsEventosAtletas;

Validate();

}

public void Validate()

{

if (FechaInicio > FechaFinal)

{

throw new ExcepcionesEvento("La fecha de inicio no puede ser mayor a la fecha final.");

}

if (FechaInicio.Year != 2024 || FechaFinal.Year != 2024)

{

throw new ExcepcionesEvento("Los años de las fechas deben ser 2024.");

}

}

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations.Schema;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaNegocio.EntidadesDominio

{

public class EventoAtleta

{

public int Id { get; set; }

public Atleta Atleta { get; set; }

public Evento? Evento { get; set; }

public decimal? PuntajeAtleta { get; set; }

}

}

using Microsoft.EntityFrameworkCore;

namespace LogicaNegocio.EntidadesDominio

{

[Owned]

public class Pais

{

public int Id { get; set; }

public string Nombre { get; set; }

public int CantidadHabitantes { get; set; }

public Delegado Delegado { get; set; }

}

}

using Microsoft.EntityFrameworkCore;

namespace LogicaNegocio.EntidadesDominio

{

[Owned]

public class Rol

{

public int Id { get; set; }

public string Nombre { get; set; }

}

}

using ExcepcionesPropias;

using LogicaNegocio.InterfacesDominio;

using LogicaNegocio.ValueObjects;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace LogicaNegocio.EntidadesDominio

{

[Table("Usuarios")]

public class Usuario : IValidable

{

public int Id { get; set; }

[Required]

public EmailUsuario Email { get; set; } // VO

[Required]

[MinLength(6, ErrorMessage = "La Contraseña debe tener al menos 6 caracteres.")]

[Display(Name = "Contraseña")]

public ContraseniaUsuario Contrasenia { get; set; } // VO

[Required] //CAMBIO 27-9-10:37

public Rol Rol { get; set; } // ED

public int RolId { get; set; }

public string? Date { get; set; }

public string? Hour { get; set; }

public string? EmailAdmin { get; set; }

public Usuario(EmailUsuario email, ContraseniaUsuario contrasenia, Rol rol, string? date, string? hour, string? emailAdmin)

{

Email = email;

Contrasenia = contrasenia;

Rol = rol;

Date = date;

Hour = hour;

EmailAdmin = emailAdmin;

Validate();

}

protected Usuario() { }

public void Validate()

{

if (Email == null) throw new ExcepcionesUsuario("El email es Obligatorio");

if (Contrasenia == null) throw new ExcepcionesUsuario("La contraseña es obligatoria");

}

}

}

LOGICANEGOCIO.INTERFACESDOMINO

namespace LogicaNegocio.InterfacesDominio

{

public interface IValidable

{

void Validate();

}

}

LOGICANEGOCIO. InterfacesRepositorios

namespace LogicaNegocio.InterfacesRepositorios

{

public interface IRepositorio<T>

{

void Add(T obj);

void Update(T obj);

T FindById(int id);

IEnumerable<T> FindAll();

void Remove(int id);

}

}

using LogicaNegocio.EntidadesDominio;

namespace LogicaNegocio.InterfacesRepositorios

{

public interface IRepositorioAtleta

{

Atleta FindById(int id);

IEnumerable<Atleta> FindAll();

void Update(int atletaId, int idDisciplina);

IEnumerable<Disciplina> FindAllDisponible(int idAtleta);

IEnumerable<Disciplina> FindAllRegistrada(int idAtleta);

}

}

using LogicaNegocio.EntidadesDominio;

namespace LogicaNegocio.InterfacesRepositorios

{

public interface IRepositorioDelegado

{

IEnumerable<Delegado> FindAll();

Delegado FindById(int id);

}

}

using LogicaNegocio.EntidadesDominio;

namespace LogicaNegocio.InterfacesRepositorios

{

public interface IRepositorioDisciplina

{

List<Disciplina> FindAllDisciplinasById(List<int> ids);

Disciplina Add(Disciplina obj);

IEnumerable<Disciplina> FindAll();

List<Disciplina> FindAllByAtletaId(int idAtleta);

Disciplina FindById(int idDisciplina);

}

}

using LogicaNegocio.EntidadesDominio;

namespace LogicaNegocio.InterfacesRepositorios

{

public interface IRepositorioEvento : IRepositorio<Evento>

{

Evento FindByName(string nombreEvento);

IEnumerable<Evento> GetEventosPorFecha(DateTime fecha);

}

}

using LogicaNegocio.EntidadesDominio;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaNegocio.InterfacesRepositorios

{

public interface IRepositorioEventoAtleta:IRepositorio<EventoAtleta>

{

IEnumerable<EventoAtleta> ObtenerAtletasPorEvento(int idEvento);

IEnumerable<EventoAtleta> ObtenerEventosPorAtleta(int idAtleta);

int FindIdByAtletaEvento(int idEvento, int idAtleta);

}

}

using LogicaNegocio.EntidadesDominio;

namespace LogicaNegocio.InterfacesRepositorios

{

public interface IRepositorioPais

{

Pais FindByName(string name);

IEnumerable<Pais> FindAll();

Pais FindById(int id);

}

}

using LogicaNegocio.EntidadesDominio;

namespace LogicaNegocio.InterfacesRepositorios

{

public interface IRepositorioRoles : IRepositorio<Rol>

{

Rol FindByName(string name);

}

}

using LogicaNegocio.EntidadesDominio;

namespace LogicaNegocio.InterfacesRepositorios

{

public interface IRepositorioUsuario : IRepositorio<Usuario>

{

Usuario FindByMail(string mail);

}

}

LOGICANEGOCIO.INTERFACESREPOSITORIO

using ExcepcionesPropias;

using Microsoft.EntityFrameworkCore;

namespace LogicaNegocio.ValueObjects

{

[Owned]

public class AnioDisciplina : IEquatable<AnioDisciplina>

{

public int Valor { get; private set; }

protected AnioDisciplina() { }

public AnioDisciplina(int valor)

{

Valor = valor;

Validate();

}

public void Validate()

{

if (Valor == null)

{

throw new ExcepcionesDisciplina("El año no puede ser Vacío.");

}

if (Valor == 0)

{

throw new ExcepcionesDisciplina("El año no puede ser igual a Cero.");

}

if (Valor < 0)

{

throw new ExcepcionesDisciplina("El año no puede ser menor a Cero.");

}

if (Valor < 1896)

{

throw new ExcepcionesDisciplina("El año no puede ser menor al de la creación de los juegos olímpicos.");

}

if (Valor > DateTime.Now.Year)

{

throw new ExcepcionesDisciplina("El año no puede ser mayor al que se está cursando.");

}

}

public bool Equals(AnioDisciplina? other)

{

if (other != null)

{

return Valor == other.Valor;

}

return false;

}

}

}

using ExcepcionesPropias;

using Microsoft.EntityFrameworkCore;

namespace LogicaNegocio.ValueObjects

{

[Owned]

public class ContraseniaUsuario : IEquatable<ContraseniaUsuario>

{

public string Valor { get; private set; }

public ContraseniaUsuario(string contrasenia)

{

Valor = contrasenia;

Validate();

}

protected ContraseniaUsuario() { }

private void Validate()

{

Valor = Valor.Trim(); //CAMBIO REALIZADO 27-9-10:30

if (string.IsNullOrEmpty(Valor) || Valor.Length < 6)

{

throw new ExcepcionesUsuario("La contraseña debe tener al menos 6 caracteres.");

}

if (!Valor.Contains(".") && !Valor.Contains(";") && !Valor.Contains(",") && !Valor.Contains("!"))

{

throw new ExcepcionesUsuario("La contraseña debe tener al menos un caracter especificos");

}

if (!ExisteCaracteres())

{

throw new ExcepcionesUsuario("La contraseña debe tener al menos una mayúscula,una minúscula y un digito.");

}

}

public bool Equals(ContraseniaUsuario? other)

{

if (other != null)

{

return Valor.Equals(other.Valor);

}

return false;

}

public bool ExisteCaracteres()

{

bool existeMayus = false;

bool existeMinus = false;

bool existeNumero = false;

foreach (char item in Valor)

{

if (char.IsUpper(item))

{

existeMayus = true;

}

if (char.IsLower(item))

{

existeMinus = true;

}

if (char.IsDigit(item))

{

existeNumero = true;

}

if (existeMayus && existeMinus && existeNumero)

{

break;

}

}

return existeMayus && existeMinus && existeNumero;

}

}

}

using ExcepcionesPropias;

using Microsoft.EntityFrameworkCore;

namespace LogicaNegocio.ValueObjects

{

[Owned]

public class EmailUsuario : IEquatable<EmailUsuario>

{

public string Valor { get; private set; }

public EmailUsuario(string correo)

{

Valor = correo;

Validate();

}

protected EmailUsuario() { }

private void Validate()//CAMBIO REALIZADO 27-9-10:30

{

Valor = Valor.Trim();

if (string.IsNullOrEmpty(Valor))

{

throw new ExcepcionesUsuario("El email no puede estar vacío.");

}

if (!Valor.Contains("@"))

{

throw new ExcepcionesUsuario("El email debe contener un arroba (@).");

}

int indexAt = Valor.IndexOf("@");

if (indexAt == -1 || !Valor.Substring(indexAt).Contains("."))

{

throw new ExcepcionesUsuario("El email debe contener un punto (.) después del arroba.");

}

if (Valor.Contains(" "))

{

throw new ExcepcionesUsuario("El email no puede contener espacios en blanco.");

}

}

public bool Equals(EmailUsuario? other)

{

if (other != null)

{

return this.Valor == other.Valor;

}

return false;

}

}

}

using ExcepcionesPropias;

using Microsoft.EntityFrameworkCore;

namespace LogicaNegocio.ValueObjects

{

[Owned]

public class NombreDisciplina : IEquatable<NombreDisciplina>

{

public string Valor { get; private set; }

protected NombreDisciplina() { }

public NombreDisciplina(string valor)

{

Valor = valor;

Validate();

}

public void Validate()

{

if (Valor == "" || Valor == null || string.IsNullOrEmpty(Valor))

{

throw new ExcepcionesDisciplina("El nombre no puede ser vacío");

}

if (Valor.Length < 10 || Valor.Length > 50)

{

throw new ExcepcionesDisciplina("El largo del nombre no puede ser menor a 10 ni mayor a 50");

}

}

public bool Equals(NombreDisciplina? other)

{

if (other != null)

{

return Valor == other.Valor;

}

return false;

}

}

}

using ExcepcionesPropias;

using Microsoft.EntityFrameworkCore;

namespace LogicaNegocio.ValueObjects

{

[Owned]

public class NombreEvento : IEquatable<NombreEvento>

{

public string Valor { get; private set; }

protected NombreEvento() { }

public NombreEvento(string valor)

{

Valor = valor;

Validate();

}

public void Validate()

{

if (string.IsNullOrWhiteSpace(Valor))

{

throw new ExcepcionesEvento("El nombre no puede ser vacío o estar compuesto solo por espacios en blanco");

}

if (Valor.Length < 3)

{

throw new ExcepcionesEvento("El nombre debe tener al menos 3 caracteres");

}

if (Valor.Length > 100)

{

throw new ExcepcionesEvento("El nombre no puede exceder los 100 caracteres");

}

}

public bool Equals(NombreEvento? other)

{

if (other != null)

{

return Valor == other.Valor;

}

return false;

}

}

}

## 4.6.- Capa Presentación.

PRESENTACION.CONTROLLERS

using DTO;

using ExcepcionesPropias;

using LogicaAplicacion.InterfacesCU;

using LogicaNegocio.EntidadesDominio;

using Microsoft.AspNetCore.Mvc;

using Presentacion.Models;

namespace Presentacion.Controllers

{

public class AtletaController : Controller

{

public IAltaAtleta CUAltaAtleta { get; set; }

public IListadoAtletas CUListadoAtleta { get; set; }

public ILoginUsuario CULoginUsuario { get; set; }

public IListadoDisciplina CUlistadoDisciplina { get; set; }

public AtletaController(IAltaAtleta cUAltaAtleta, IListadoAtletas cUListadoAtleta,

ILoginUsuario cULoginUsuario, IListadoDisciplina cuListadoDisciplina)

{

CUAltaAtleta = cUAltaAtleta;

CUListadoAtleta = cUListadoAtleta;

CULoginUsuario = cULoginUsuario;

CUlistadoDisciplina = cuListadoDisciplina;

}

public bool EstaLogueado()

{

string email = HttpContext.Session.GetString("emailUsuarioLogueado");

try

{

Usuario userLogueado = CULoginUsuario.FindByMail(email);

if (userLogueado == null)

{

return false;

}

else

{

return true;

}

}

catch (ExcepcionesUsuario ex)

{

ViewBag.Error = ex.Message;

return false;

}

}

public bool EsDigitador()

{

string admin = HttpContext.Session.GetString("rolUsuarioLogueado");

if (admin == "Digitador")

{

return true;

}

else

{

return false;

}

}

// GET: AtletaController

public ActionResult Index()

{

if (EstaLogueado() && EsDigitador())

{

IEnumerable<ListadoAtletasDTO> atletasOrdenados = CUListadoAtleta.GetAtletas();

return View(atletasOrdenados);

}

else

{

return RedirectToAction("Index", "Atleta");

}

}

// GET: AtletaController/Edit/5

public ActionResult Edit(int id)

{

if (EstaLogueado() && EsDigitador())

{

try

{

ListadoAtletasDTO dto = CUListadoAtleta.GetAtletaPorId(id);

AtletaDisciplinasViewModel vm = new AtletaDisciplinasViewModel();

vm.DTOAtleta = CUListadoAtleta.GetAtletaPorId(dto.IdAtleta);

vm.DTODisciplinasDisponibles = CUlistadoDisciplina.GetDisciplinasDisponibles(dto.IdAtleta);

vm.DTODisciplinasRegistradas = CUlistadoDisciplina.GetDisciplinasRegistradas(dto.IdAtleta);

return View(vm);

}

catch (ExcepcionesAtleta ex)

{

ViewBag.Mensaje = ex.Message;

return View();

}

}

else

{

return RedirectToAction("Index", "Atleta");

}

}

// POST: AtletaController/Edit/5

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Edit(AtletaDisciplinasViewModel vm)

{

try

{

CUAltaAtleta.UpdateAtleta(vm.DTOAtleta);

return RedirectToAction("Edit", "Atleta");

}

catch (ExcepcionesAtleta ex)

{

ViewBag.Mensaje = ex.Message;

}

vm.DTODisciplinasDisponibles = CUlistadoDisciplina.GetDisciplinasDisponibles(vm.DTOAtleta.IdAtleta);

vm.DTODisciplinasRegistradas = CUlistadoDisciplina.GetDisciplinasRegistradas(vm.DTOAtleta.IdAtleta);

return View(vm);

}

}

}

using DTO;

using ExcepcionesPropias;

using LogicaAplicacion.InterfacesCU;

using LogicaNegocio.EntidadesDominio;

using Microsoft.AspNetCore.Mvc;

namespace Presentacion.Controllers

{

public class DisciplinaController : Controller

{

public IAltaDisciplina CUAltaDisciplina { get; set; }

public IListadoDisciplina CUListadoDisciplina { get; set; }

public ILoginUsuario CULoginUsuario { get; set; }

public DisciplinaController(IAltaDisciplina cUAltaDisciplina, IListadoDisciplina cUListadoDisciplina,

ILoginUsuario cULoginUsuario)

{

CUAltaDisciplina = cUAltaDisciplina;

CUListadoDisciplina = cUListadoDisciplina;

CULoginUsuario = cULoginUsuario;

}

public bool EstaLogueado()

{

string email = HttpContext.Session.GetString("emailUsuarioLogueado");

try

{

Usuario userLogueado = CULoginUsuario.FindByMail(email);

if (userLogueado == null)

{

return false;

}

else

{

return true;

}

}

catch (ExcepcionesUsuario ex)

{

ViewBag.Error = ex.Message;

return false;

}

}

public bool EsDigitador()

{

string admin = HttpContext.Session.GetString("rolUsuarioLogueado");

if (admin == "Digitador")

{

return true;

}

else

{

return false;

}

}

// GET: DisciplinaController

public ActionResult Index()

{

if (EstaLogueado() && EsDigitador())

{

return View(CUListadoDisciplina.GetDisciplinas());

}

else

{

return RedirectToAction("Index", "Home");

}

}

// GET: DisciplinaController/Create

public ActionResult Create()

{

if (EstaLogueado() && EsDigitador())

{

AltaDisciplinaDTO dto = new AltaDisciplinaDTO();

return View(dto);

}

else

{

return RedirectToAction("Index", "Home");

}

}

// POST: DisciplinaController/Create

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Create(AltaDisciplinaDTO dto)

{

try

{

CUAltaDisciplina.AltaDisci(dto);

return RedirectToAction(nameof(Index));

}

catch (ExcepcionesDisciplina ex)

{

ViewBag.Error = ex.Message;

return View();

}

}

}

}

using DTO;

using DTO.Mappers;

using ExcepcionesPropias;

using LogicaAplicacion.CU;

using LogicaAplicacion.InterfacesCU;

using LogicaNegocio.EntidadesDominio;

using LogicaNegocio.InterfacesRepositorios;

using Microsoft.AspNetCore.Mvc;

using Presentacion.Models;

using static System.Runtime.InteropServices.JavaScript.JSType;

namespace Presentacion.Controllers

{

public class EventoController : Controller

{

public IAltaEvento CUAltaEvento { get; set; }

public IListadoEventos CUListadoEventos { get; set; }

public IListadoAtletas CUListadoAtleta { get; set; }

public IListadoDisciplina CUListadoDisciplina { get; set; }

public IListadoEventosAtletas CUListadoEventoAtleta { get; set; }

public IAltaEventoAtletaPuntaje CUAltaEventoPuntaje { get; set; }

public IRepositorioEventoAtleta CURepositorioEventoAtleta { get; set; }

public ILoginUsuario CULoginUsuario { get; set; }

public EventoController(IAltaEvento cualtaEvento, IListadoEventos cuListadoEventos,

IListadoAtletas cUListadoAtleta, IListadoDisciplina cuListadoDisciplina,

IListadoEventosAtletas cuListadoEventoAtleta, IAltaEventoAtletaPuntaje cUAltaEventoPuntaje,

IRepositorioEventoAtleta curepositorioEventoAtleta, ILoginUsuario cULoginUsuario)

{

CUAltaEvento = cualtaEvento;

CUListadoEventos = cuListadoEventos;

CUListadoAtleta = cUListadoAtleta;

CUListadoDisciplina = cuListadoDisciplina;

CUListadoEventoAtleta = cuListadoEventoAtleta;

CUAltaEventoPuntaje = cUAltaEventoPuntaje;

CURepositorioEventoAtleta = curepositorioEventoAtleta;

CULoginUsuario = cULoginUsuario;

}

public bool EstaLogueado()

{

string email = HttpContext.Session.GetString("emailUsuarioLogueado");

try

{

Usuario userLogueado = CULoginUsuario.FindByMail(email);

if (userLogueado == null)

{

return false;

}

else

{

return true;

}

}

catch (ExcepcionesUsuario ex)

{

ViewBag.Error = ex.Message;

return false;

}

}

// GET: Evento/Index

public ActionResult Index()

{

if (EstaLogueado())

{

IEnumerable<ListadoEventosDTO> eventos = new List<ListadoEventosDTO>();

return View(eventos);

}

return RedirectToAction("Index", "Home");

}

[HttpPost]

public ActionResult Index(DateTime? fecha)

{

if (EstaLogueado())

{

DateTime fechaConsulta = fecha ?? DateTime.Now;

IEnumerable<ListadoEventosDTO> eventos = CUListadoEventos.GetEventosPorFecha(fechaConsulta);

return View(eventos);

}

return RedirectToAction("Index", "Home");

}

// GET: EventoController/Create

public ActionResult Create()

{

if (EstaLogueado())

{

try

{

var vm = new AltaEventoViewModel

{

DTOAltaEvento = new AltaEventoDTO(),

DTOAtleta = CUListadoAtleta.GetAtletas(),

DTODisciplinas = CUListadoDisciplina.GetDisciplinas()

};

return View(vm);

}

catch (ExcepcionesEvento ex)

{

ViewBag.Error = ex.Message;

return RedirectToAction("Index", "Home");

}

}

return RedirectToAction("Index", "Home");

}

// POST: EventoController/Create

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Create(AltaEventoViewModel vm)

{

try

{

AltaEventoDTO dto = new AltaEventoDTO

{

NombreEvento = vm.DTOAltaEvento.NombreEvento,

IdDisciplina = vm.DTOAltaEvento.IdDisciplina,

FechaInicioEvento = vm.DTOAltaEvento.FechaInicioEvento,

FechaFinalEvento = vm.DTOAltaEvento.FechaFinalEvento,

IdsAtletas = vm.DTOAltaEvento.IdsAtletas

};

CUAltaEvento.AltaEve(dto);

return RedirectToAction("Index");

}

catch (ExcepcionesEvento ex)

{

ViewBag.Error = ex.Message;

}

catch (Exception)

{

ViewBag.Error = "No es posible registrar el evento.";

}

vm.DTOAtleta = CUListadoAtleta.GetAtletas();

vm.DTODisciplinas = CUListadoDisciplina.GetDisciplinas();

return View(vm);

}

public ActionResult Edit(int id)

{

if (EstaLogueado())

{

try

{

IEnumerable<ListadoEventoAtletaDTO> dto = CUListadoEventoAtleta.GetAtletasPorEvento(id);

if (dto == null || !dto.Any())

{

ViewBag.Error = "No hay atletas registrados para este evento.";

return RedirectToAction("Index");

}

ViewBag.IdEvento = id;

return View(dto);

}

catch (ExcepcionesAtleta ex)

{

ViewBag.Error = ex.Message;

return RedirectToAction("Index");

}

catch (Exception ex)

{

ViewBag.Error = "Ocurrió un error al obtener la lista de atletas: " + ex.Message;

return RedirectToAction("Index");

}

}

return RedirectToAction("Index", "Home");

}

[HttpPost]

public ActionResult RegistrarPuntaje(int IdAtleta, decimal? puntajeAtleta, int idEvento)

{

try

{

AltaEventoAtletaPuntajeDTO dto = new AltaEventoAtletaPuntajeDTO

{

IdAtleta = IdAtleta,

Puntaje = puntajeAtleta,

IdEvento = idEvento

};

dto.Id = CURepositorioEventoAtleta.FindIdByAtletaEvento(idEvento, IdAtleta);

CUAltaEventoPuntaje.AltaPuntaje(dto);

}

catch (ExcepcionesEvento ex)

{

ViewBag.Mensaje = ex.Message;

}

return RedirectToAction("Edit", "Evento", new { id = idEvento });

}

}

}

using DTO;

using LogicaAplicacion.InterfacesCU;

using LogicaNegocio.EntidadesDominio;

using Microsoft.AspNetCore.Mvc;

namespace Presentacion.Controllers

{

public class HomeController : Controller

{

public ILoginUsuario CULoginUsuario { get; }

public HomeController(ILoginUsuario cuLoginUsuario)

{

CULoginUsuario = cuLoginUsuario;

}

public IActionResult Index()

{

return View();

}

public IActionResult Login()

{

return View();

}

[HttpPost]

public IActionResult Login(LoginUsuarioDTO loginUserDTO)

{

try

{

if (string.IsNullOrWhiteSpace(loginUserDTO.EmailUsuario) || string.IsNullOrWhiteSpace(loginUserDTO.ContraseniaUsuario))

{

ViewBag.Error = "Email y contraseña son requeridos.";

return View();

}

Usuario usuarioLogueado = CULoginUsuario.FindByMail(loginUserDTO.EmailUsuario);

if (usuarioLogueado == null || usuarioLogueado.Contrasenia.Valor != loginUserDTO.ContraseniaUsuario)

{

ViewBag.Error = "Email o contraseña incorrectos.";

return View();

}

HttpContext.Session.SetString("rolUsuarioLogueado", usuarioLogueado.Rol.Nombre);

HttpContext.Session.SetString("emailUsuarioLogueado", usuarioLogueado.Email.Valor);

HttpContext.Session.SetInt32("idUsuarioLogueado", usuarioLogueado.Id);

return RedirectToAction("Index", "Home");

}

catch (Exception ex)

{

ViewBag.Error = "Ocurrió un error inesperado. Intente de nuevo.";

return View();

}

}

public IActionResult Logout()

{

HttpContext.Session.Clear();

return RedirectToAction("Index", "Home");

}

}

}

using DTO;

using ExcepcionesPropias;

using LogicaAplicacion.InterfacesCU;

using LogicaNegocio.EntidadesDominio;

using Microsoft.AspNetCore.Mvc;

using Presentacion.Models;

namespace Presentacion.Controllers

{

public class UsuariosController : Controller

{

public IAltaUsuario CUAltaUsuario { get; set; }

public IListadoUsuarios CUListadoUsuarios { get; set; }

public IListadoRoles CUListadoRoles { get; set; }

public IBajaUsuario CUBajaUsuario { get; set; }

public IUpdateUsuario CUUpdateUsuario { get; set; }

public ILoginUsuario CULoginUsuario { get; set; }

public IBuscarUsuarioPorID CUBuscarUsuarioPorID { get; set; }

public UsuariosController(IAltaUsuario cuAltaUsuarios, IListadoUsuarios cuListadoUsuarios, IListadoRoles cuListadoRoles,

IBajaUsuario cuBajaUsuario, IUpdateUsuario cUUpdateUsuario, ILoginUsuario cULoginUsuario, IBuscarUsuarioPorID cuBuscarUsuarioPorID)

{

CUAltaUsuario = cuAltaUsuarios;

CUListadoUsuarios = cuListadoUsuarios;

CUListadoRoles = cuListadoRoles;

CUBajaUsuario = cuBajaUsuario;

CUUpdateUsuario = cUUpdateUsuario;

CULoginUsuario = cULoginUsuario;

CUBuscarUsuarioPorID = cuBuscarUsuarioPorID;

}

public bool EstaLogueado()

{

string email = HttpContext.Session.GetString("emailUsuarioLogueado");

try

{

Usuario userLogueado = CULoginUsuario.FindByMail(email);

if (userLogueado == null)

{

return false;

}

else

{

return true;

}

}

catch (ExcepcionesUsuario ex)

{

ViewBag.Error = ex.Message;

return false;

}

}

public bool EsAdmin()

{

string admin = HttpContext.Session.GetString("rolUsuarioLogueado");

if (admin == "Administrador")

{

return true;

}

else

{

return false;

}

}

// GET: UsuariosController

public ActionResult ListaUsuarios()

{

if (EstaLogueado() && EsAdmin())

{

string emailUsuarioLogueado = HttpContext.Session.GetString("emailUsuarioLogueado");

Usuario userLogueado = CULoginUsuario.FindByMail(emailUsuarioLogueado);

ViewBag.Usuario = userLogueado;

IEnumerable<ListadoUsuariosDTO> dtos = CUListadoUsuarios.ObtenerListado();

return View(dtos);

}

else

{

return RedirectToAction("Index", "Home");

}

}

// GET: UsuariosController/Details/5

public ActionResult Details(int id)

{

if (EstaLogueado() && EsAdmin())

{

AltaUsuarioDTO user = CUBuscarUsuarioPorID.BuscarUsuarioId(id);

return View(user);

}

else

{

return RedirectToAction("Index", "Home");

}

}

// GET: UsuariosController/Create

public ActionResult Create()

{

if (EstaLogueado() && EsAdmin())

{

try

{

AltaUsuarioViewModel vm = new AltaUsuarioViewModel();

vm.DTOAltaUsuario = new AltaUsuarioDTO();

vm.RolesDTO = CUListadoRoles.ObtenerListado();

return View(vm);

}

catch (ExcepcionesUsuario ex)

{

ViewBag.Error = ex.Message;

return RedirectToAction("Index", "Home");

}

}

else

{

return RedirectToAction("Index", "Home");

}

}

// POST: UsuariosController/Create

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Create(AltaUsuarioViewModel vm)

{

try

{

string? emailUsuarioLogueado = HttpContext.Session.GetString("emailUsuarioLogueado");

DateTime fecha = DateTime.Now.Date;

string fechaFormateada = fecha.ToString("dd/MM/yyyy");

string horaFormateada = DateTime.Now.ToString("HH:mm");

vm.DTOAltaUsuario.EmailAdmin = emailUsuarioLogueado;

vm.DTOAltaUsuario.Date = fechaFormateada;

vm.DTOAltaUsuario.Hora = horaFormateada;

CUAltaUsuario.Alta(vm.DTOAltaUsuario);

return RedirectToAction(nameof(ListaUsuarios));

}

catch (ExcepcionesUsuario ex)

{

ViewBag.Error = ex.Message;

}

catch (Exception ex)

{

ViewBag.Error = "No es posible realizar el alta al Usuario";

}

vm.RolesDTO = CUListadoRoles.ObtenerListado();

return View(vm);

}

// GET: UsuariosController/Edit/5

public ActionResult Edit(int id)

{

if (EstaLogueado() && EsAdmin())

{

AltaUsuarioDTO user = CUBuscarUsuarioPorID.BuscarUsuarioId(id);

AltaUsuarioViewModel vm = new AltaUsuarioViewModel

{

DTOAltaUsuario = user,

RolesDTO = CUListadoRoles.ObtenerListado()

};

return View(vm);

}

else

{

return RedirectToAction("Index", "Home");

}

}

// POST: UsuariosController/Edit/5

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Edit(AltaUsuarioViewModel vm)

{

//AltaUsuarioViewModel vmPost = new AltaUsuarioViewModel

//{

// DTOAltaUsuario = vm.DTOAltaUsuario,

// RolesDTO = CUListadoRoles.ObtenerListado()

//};

try

{

string? emailUsuarioLogueado = HttpContext.Session.GetString("emailUsuarioLogueado");

DateTime fecha = DateTime.Now.Date;

string fechaFormateada = fecha.ToString("dd/MM/yyyy");

string horaFormateada = DateTime.Now.ToString("HH:mm");

vm.DTOAltaUsuario.EmailAdmin = "Admin-Actualizador " + emailUsuarioLogueado;

vm.DTOAltaUsuario.Date = "Fecha-Actualización " + fechaFormateada;

vm.DTOAltaUsuario.Hora = "Hora-Actualización " + horaFormateada;

CUUpdateUsuario.UpdateUser(vm.DTOAltaUsuario);

return RedirectToAction(nameof(ListaUsuarios));

}

catch (ExcepcionesUsuario ex)

{

ViewBag.Error = ex.Message;

}

return View(vm);

}

// GET: UsuariosController/Delete/5

public ActionResult Delete(string email)

{

return View((object)email);

}

// POST: UsuariosController/DeleteConfirmed

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult DeleteConfirmed(string email)

{

try

{

CUBajaUsuario.BajaUser(email);

return RedirectToAction("ListaUsuarios", "Usuarios");

}

catch (ExcepcionesUsuario ex)

{

ViewBag.Error = ex.Message;

return View("Delete", "Usuarios");

}

catch (Exception ex)

{

ViewBag.Error = "Ocurrió un error al intentar eliminar al usuario: " + ex.Message;

return View("Delete", "Usuarios");

}

}

}

}

PRESENTACION.MODELS

using DTO;

namespace Presentacion.Models

{

public class AltaEventoViewModel

{

public AltaEventoDTO DTOAltaEvento { get; set; }

public IEnumerable<ListadoAtletasDTO> DTOAtleta { get; set; }

public IEnumerable<ListadoDisciplinaDTO> DTODisciplinas { get; set; }

}

}

using DTO;

namespace Presentacion.Models

{

public class AltaUsuarioViewModel

{

public AltaUsuarioDTO DTOAltaUsuario { get; set; }

public IEnumerable<RolDTO> RolesDTO { get; set; }

}

}

using DTO;

namespace Presentacion.Models

{

public class AtletaDisciplinasViewModel

{

public ListadoAtletasDTO DTOAtleta { get; set; }

public IEnumerable<ListadoDisciplinaDTO> DTODisciplinasDisponibles { get; set; }

public IEnumerable<ListadoDisciplinaDTO> DTODisciplinasRegistradas { get; set; }

}

}

namespace Presentacion.Models

{

public class ErrorViewModel

{

public string? RequestId { get; set; }

public bool ShowRequestId => !string.IsNullOrEmpty(RequestId);

}

}

PRESENTACION.VIEWS.ATLETA

@model Presentacion.Models.AtletaDisciplinasViewModel

<h1>DISCIPLINAS DE ATLETA</h1>

@ViewBag.Mensaje

<hr />

<div class="row">

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

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

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

<div class="form-group">

<**input** **asp-for**="DTOAtleta.IdAtleta" class="form-control" readonly **type**="hidden" />

</div>

<div class="form-group">

<**label** **asp-for**="DTOAtleta.NombreAtleta" class="control-label"></**label**>

<**input** **asp-for**="DTOAtleta.NombreAtleta" class="form-control" readonly />

</div>

<div class="form-group">

<**label** **asp-for**="DTOAtleta.ApellidoAtleta" class="control-label"></**label**>

<**input** **asp-for**="DTOAtleta.ApellidoAtleta" class="form-control" readonly />

</div>

<div class="form-group">

<**label** **asp-for**="DTOAtleta.SexoAtleta" class="control-label"></**label**>

<span class="form-control">

@(Model.DTOAtleta.SexoAtleta == true ? "Hombre" : "Mujer")

</span>

<**input** **type**="hidden" **asp-for**="DTOAtleta.SexoAtleta" />

</div>

<div class="form-group">

<**label** **asp-for**="DTOAtleta.NombrePais" class="control-label"></**label**>

<**input** **asp-for**="DTOAtleta.NombrePais" class="form-control" readonly />

</div>

<hr />

<br />

<!-- Disciplinas Registradas -->

<h5>Disciplinas Registradas</h5>

<ul class="list-group">

@foreach (var disciplina in Model.DTODisciplinasRegistradas)

{

<li class="list-group-item">@disciplina.NombreDisciplina (@disciplina.AnioDisciplina)</li>

}

</ul>

<hr />

<br />

<!-- Disciplinas Disponibles -->

<h5>Disciplinas Disponibles para Registro</h5>

<ul class="list-group">

@foreach (var disciplina in Model.DTODisciplinasDisponibles)

{

<li class="list-group-item">

<**input** **type**="radio" **asp-for**="DTOAtleta.IdDisciplina" **value**="@disciplina.IdDisciplina" />

@disciplina.NombreDisciplina (@disciplina.AnioDisciplina)

</li>

}

</ul>

<br />

<div class="form-group">

<input type="submit" value="Guardar" class="btn btn-primary" />

</div>

</**form**>

</div>

</div>

@section Scripts {

@{

await Html.RenderPartialAsync("\_ValidationScriptsPartial");

}

}

@model IEnumerable<DTO.ListadoAtletasDTO>

<h1>ATLETAS</h1>

<table class="table">

<thead>

<tr>

<th>

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

</th>

<th>

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

</th>

<th>

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

</th>

<th>

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

</th>

<th>

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

</th>

<th></th>

</tr>

</thead>

<tbody>

@foreach (var item in Model)

{

<tr>

<td>

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

</td>

<td>

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

</td>

<td>

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

</td>

<td>

@(item.SexoAtleta ? "Masculino" : "Femenino")

</td>

<td>

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

</td>

<td>

@Html.ActionLink("Ver Disciplinas", "Edit", new { id = item.IdAtleta } , new { @class = "btn btn-primary" })

</td>

</tr>

}

</tbody>

</table>

PRESENTACION.VIEWS.DISCIPLINA

@model DTO.AltaDisciplinaDTO

<h1>REGISTRAR DISCIPLINA</h1>

@ViewBag.Error

<hr />

<div class="row">

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

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

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

<div class="form-group">

<**input** **asp-for**="IdDisciplina" class="form-control" **type**="hidden" readonly />

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

</div>

<div class="form-group">

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

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

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

</div>

<div class="form-group">

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

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

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

</div>

<br />

<div class="form-group">

<input type="submit" value="Create" class="btn btn-primary" />

</div>

</**form**>

</div>

</div>

@section Scripts {

@{

await Html.RenderPartialAsync("\_ValidationScriptsPartial");

}

}

@model IEnumerable<DTO.ListadoDisciplinaDTO>

<h1>DISCIPLINAS</h1>

<table class="table">

<thead>

<tr>

<th>

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

</th>

<th>

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

</th>

<th>

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

</th>

<th></th>

</tr>

</thead>

<tbody>

@foreach (var item in Model) {

<tr>

<td>

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

</td>

<td>

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

</td>

<td>

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

</td>

@\* <td>

@Html.ActionLink("Edit", "Edit", new { /\* id=item.PrimaryKey \*/ }) |

@Html.ActionLink("Details", "Details", new { /\* id=item.PrimaryKey \*/ }) |

@Html.ActionLink("Delete", "Delete", new { /\* id=item.PrimaryKey \*/ })

</td> \*@

</tr>

}

</tbody>

</table>

PRESENTACION.VIEWS.EVENTO

@model Presentacion.Models.AltaEventoViewModel

@{

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

}

<h2>Crear Evento</h2>

@ViewBag.Error

<br />

<**form** **asp-action**="Create" method="post">

<div class="form-group">

<**label** **asp-for**="DTOAltaEvento.NombreEvento" class="control-label"></**label**>

<**input** **asp-for**="DTOAltaEvento.NombreEvento" class="form-control" />

<**span** **asp-validation-for**="DTOAltaEvento.NombreEvento" class="text-danger"></**span**>

</div>

<br />

<div class="form-group">

<label for="IdDisciplina" class="control-label">Disciplina</label>

<select id="IdDisciplina" name="DTOAltaEvento.IdDisciplina" class="form-control">

<**option** **value**="">Seleccione una Disciplina</**option**>

@foreach (var disciplina in Model.DTODisciplinas)

{

<**option** **value**="@disciplina.IdDisciplina">

@disciplina.NombreDisciplina

</**option**>

}

</select>

<**span** **asp-validation-for**="DTOAltaEvento.IdDisciplina" class="text-danger"></**span**>

</div>

<br />

<div class="form-group">

<**label** **asp-for**="DTOAltaEvento.FechaInicioEvento" class="control-label"></**label**>

<**input** **asp-for**="DTOAltaEvento.FechaInicioEvento" **type**="date" class="form-control"

min="2024-01-01" **value**="@DateTime.Today.ToString("yyyy-MM-dd")" />

<**span** **asp-validation-for**="DTOAltaEvento.FechaInicioEvento" class="text-danger"></**span**>

</div>

<br />

<div class="form-group">

<**label** **asp-for**="DTOAltaEvento.FechaFinalEvento" class="control-label"></**label**>

<**input** **asp-for**="DTOAltaEvento.FechaFinalEvento" **type**="date" class="form-control"

min="2024-01-01" **value**="@DateTime.Today.ToString("yyyy-MM-dd")" />

<**span** **asp-validation-for**="DTOAltaEvento.FechaFinalEvento" class="text-danger"></**span**>

</div>

<br />

<div class="form-group">

<label class="control-label" for="IdsAtletas">Atletas Participantes</label>

<select id="IdsAtletas" name="DTOAltaEvento.IdsAtletas" class="form-control" multiple size="5">

@foreach (var atleta in Model.DTOAtleta)

{

<**option** **value**="@atleta.IdAtleta">

@atleta.NombreAtleta @atleta.ApellidoAtleta

</**option**>

}

</select>

<**span** **asp-validation-for**="DTOAltaEvento.IdsAtletas" class="text-danger"></**span**>

</div>

<br />

<button type="submit" class="btn btn-primary">Guardar</button>

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

</**form**>

@section Scripts {

@await Html.PartialAsync("\_ValidationScriptsPartial")

}

@model IEnumerable<DTO.Mappers.ListadoEventoAtletaDTO>

@{

ViewData["Title"] = "Atletas en Evento";

}

<h1>Atletas en Evento</h1>

<table class="table">

<thead>

<tr>

<th>@Html.DisplayNameFor(model => model.IdAtleta)</th>

<th>@Html.DisplayNameFor(model => model.NombreAtleta)</th>

<th>@Html.DisplayNameFor(model => model.IdEvento)</th>

<th>@Html.DisplayNameFor(model => model.NombreEvento)</th>

<th>@Html.DisplayNameFor(model => model.FechaEvento)</th>

<th>@Html.DisplayNameFor(model => model.PuntajeAtleta)</th>

<th>Nuevo Puntaje</th>

<th></th>

</tr>

</thead>

<tbody>

@foreach (var item in Model)

{

var puntajeActual = item.PuntajeAtleta ?? 0;

<tr>

<td>@Html.DisplayFor(modelItem => item.IdAtleta)</td>

<td>@Html.DisplayFor(modelItem => item.NombreAtleta)</td>

<td>@Html.DisplayFor(modelItem => item.IdEvento)</td>

<td>@Html.DisplayFor(modelItem => item.NombreEvento)</td>

<td>@item.FechaEvento.ToString("dd/MM/yyyy")</td>

<td>@puntajeActual</td>

<td>

<**form** **asp-action**="RegistrarPuntaje" method="post" onsubmit="return validateScore(this)">

<input type="hidden" name="IdAtleta" value="@item.IdAtleta" />

<input type="hidden" name="idEvento" value="@item.IdEvento" />

<input type="text" name="puntajeAtleta" class="form-control" required />

<button type="submit" class="btn btn-primary mt-2">Guardar</button>

</**form**>

</td>

</tr>

}

@ViewBag.Mensaje

</tbody>

</table>

<script>

function validateScore(form) {

let puntaje = parseFloat(form.puntajeAtleta.value);

if (isNaN(puntaje) || puntaje < 0 || puntaje > 100) {

alert("El puntaje debe estar entre 0 y 100.");

return false;

}

return true;

}

</script>

@model IEnumerable<DTO.ListadoEventosDTO>

@{

ViewBag.Title = "Buscar Eventos";

var fechaSeleccionada = ViewBag.Fecha ?? DateTime.Now;

}

<h2>Buscar Eventos por Fecha</h2>

@if (ViewBag.Error != null)

{

<div class="alert alert-danger">

@ViewBag.Error

</div>

}

<**form** method="post" action="@Url.Action("Index", "Evento")">

<div class="form-group">

<label for="fecha">Seleccione una fecha:</label>

<input type="date" id="fecha" name="fecha" class="form-control"

value="@fechaSeleccionada.ToString("yyyy-MM-dd")" required />

</div>

<button type="submit" class="btn btn-primary mt-2">Buscar</button>

</**form**>

@if (Model != null && Model.Any())

{

<h3 class="mt-4">Eventos Encontrados:</h3>

<table class="table table-bordered">

<thead>

<tr>

<th>Nombre</th>

<th>Fecha</th>

<th>Acciones</th>

</tr>

</thead>

<tbody>

@foreach (var evento in Model)

{

<tr>

<td>@evento.NombreEvento</td>

<td>@evento.FechaFinal.ToString("dd/MM/yyyy")</td>

<td>

@Html.ActionLink("Ir a Evento", "Edit", new { id = evento.IdEvento }, new { @class = "btn btn-primary" })

</td>

</tr>

}

</tbody>

</table>

}

else if (!Model.Any())

{

<div class="alert alert-info mt-4">

Seleccione una fecha con Eventos.

</div>

}

PRESENTACION.VIEWS.HOME

@{

ViewData["Title"] = "Inicio";

}

<h1>Bienvenidos</h1>

<div class="text-center">

<img src="https://sociologiac.net/sociology/wp-content/uploads/2008/08/bandera-juegos-olimpicos-olimpiadas-768x512.png" alt="Bienvenida" class="img-fluid" />

</div>

@model DTO.LoginUsuarioDTO

@{

ViewData["Title"] = "Login";

}

<div class="container">

<div class="row justify-content-center">

<div class="col-md-6">

<h2 class="text-center">Iniciar Sesión</h2>

@if (ViewBag.Error != null)

{

<div class="alert alert-danger">@ViewBag.Error</div>

}

<**form** **asp-controller**="Home" **asp-action**="Login" method="post">

<div class="form-group">

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

<**input** **asp-for**="EmailUsuario" class="form-control" **value**="carlos.martinez@comiteolimpico.org" />

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

</div>

<div class="form-group">

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

<**input** **asp-for**="ContraseniaUsuario" class="form-control" **type**="password" **value**="Inicio1234.!"/>

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

</div>

<br />

<button type="submit" class="btn btn-primary">Ingresar</button>

</**form**>

</div>

</div>

</div>

@{

ViewData["Title"] = "Privacy Policy";

}

<h1>@ViewData["Title"]</h1>

<p>Use this page to detail your site's privacy policy.</p>

PRESENTACION.VIEWS.SHARED

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>Olimpiadas 2024</title>

<**link** rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />

<**link** rel="stylesheet" **href**="~/css/site.css" **asp-append-version**="true" />

<**link** rel="stylesheet" **href**="~/WebApp.styles.css" **asp-append-version**="true" />

</head>

<body>

<header>

<nav class="navbar navbar-expand-sm navbar-light bg-white border-bottom box-shadow mb-3">

<div class="container-fluid">

<**a** class="navbar-brand" **asp-area**="" **asp-controller**="" **asp-action**="">Olimpiadas 2024</**a**>

<button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target=".navbar-collapse" aria-controls="navbarSupportedContent"

aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="navbar-collapse collapse d-sm-inline-flex justify-content-between">

<ul class="navbar-nav flex-grow-1">

@if (Context.Session.GetString("rolUsuarioLogueado") == null)

{

<li class="nav-item">

<**a** class="nav-link text-dark" **asp-controller**="Home" **asp-action**="Login">Inicio de Sesión</**a**>

</li>

}

else

{

if (Context.Session.GetString("rolUsuarioLogueado") == "Digitador")

{

<li class="nav-item">

<**a** class="nav-link text-dark" **asp-controller**="Disciplina" **asp-action**="Create">Registrar Disciplina</**a**>

</li>

<li class="nav-item">

<**a** class="nav-link text-dark" **asp-controller**="Disciplina" **asp-action**="Index">Disciplinas</**a**>

</li>

<li class="nav-item">

<**a** class="nav-link text-dark" **asp-controller**="Atleta" **asp-action**="Index">Atletas</**a**>

</li>

<li class="nav-item">

<**a** class="nav-link text-dark" **asp-controller**="Evento" **asp-action**="Create">Crear Evento</**a**>

</li>

<li class="nav-item">

<**a** class="nav-link text-dark" **asp-controller**="Evento" **asp-action**="Index">Ver Eventos</**a**>

</li>

<li class="nav-item ms-auto">

<**a** class="nav-link text-dark" **asp-controller**="Home" **asp-action**="Logout">Salir</**a**>

</li>

}

else if (Context.Session.GetString("rolUsuarioLogueado") == "Administrador")

{

<li class="nav-item">

<**a** class="nav-link text-dark" **asp-controller**="Usuarios" **asp-action**="Create">Registro</**a**>

</li>

<li class="nav-item">

<**a** class="nav-link text-dark" **asp-controller**="Usuarios" **asp-action**="ListaUsuarios">Listado Usuarios</**a**>

</li>

<li class="nav-item">

<**a** class="nav-link text-dark" **asp-controller**="Evento" **asp-action**="Create">Crear Evento</**a**>

</li>

<li class="nav-item">

<**a** class="nav-link text-dark" **asp-controller**="Evento" **asp-action**="Index">Ver Eventos</**a**>

</li>

<li class="nav-item ms-auto">

<**a** class="nav-link text-dark" **asp-controller**="Home" **asp-action**="Logout">Salir</**a**>

</li>

}

}

</ul>

</div>

</div>

</nav>

</header>

<div class="container">

<main role="main" class="pb-3">

@RenderBody()

</main>

</div>

<footer class="border-top footer text-muted">

<div class="container">

<p class="text-center">© Obligatorio Olimpiadas. Todos los derechos reservados.</p>

</div>

</footer>

<**script** src="~/lib/jquery/dist/jquery.min.js"></**script**>

<**script** src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></**script**>

<**script** **src**="~/js/site.js" **asp-append-version**="true"></**script**>

@await RenderSectionAsync("Scripts", required: false)

</body>

</html>

<**script** src="~/lib/jquery-validation/dist/jquery.validate.min.js"></**script**>

<**script** src="~/lib/jquery-validation-unobtrusive/jquery.validate.unobtrusive.min.js"></**script**>

@model ErrorViewModel

@{

ViewData["Title"] = "Error";

}

<h1 class="text-danger">Error.</h1>

<h2 class="text-danger">An error occurred while processing your request.</h2>

@if (Model.ShowRequestId)

{

<p>

<strong>Request ID:</strong> <code>@Model.RequestId</code>

</p>

}

<h3>Development Mode</h3>

<p>

Swapping to <strong>Development</strong> environment will display more detailed information about the error that occurred.

</p>

<p>

<strong>The Development environment shouldn't be enabled for deployed applications.</strong>

It can result in displaying sensitive information from exceptions to end users.

For local debugging, enable the <strong>Development</strong> environment by setting the <strong>ASPNETCORE\_ENVIRONMENT</strong> environment variable to <strong>Development</strong>

and restarting the app.

</p>

PRESENTACION.VIEWS.USUARIOS

@model Presentacion.Models.AltaEventoViewModel

@{

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

}

<h2>Crear Evento</h2>

<br />

<**form** **asp-action**="Create" method="post">

<div class="form-group">

<**label** **asp-for**="DTOAltaEvento.NombreEvento" class="control-label"></**label**>

<**input** **asp-for**="DTOAltaEvento.NombreEvento" class="form-control" />

<**span** **asp-validation-for**="DTOAltaEvento.NombreEvento" class="text-danger"></**span**>

</div>

<br />

<div class="form-group">

<**label** **asp-for**="DTOAltaEvento.IdDisciplina" class="control-label">Disciplina</**label**>

<**select** **asp-for**="DTOAltaEvento.IdDisciplina" class="form-control" **asp-items**="@(new SelectList(Model.DTODisciplinas, "IdDisciplina", "NombreDisciplina"))">

<**option** **value**="">Seleccione una Disciplina</**option**>

</**select**>

<**span** **asp-validation-for**="DTOAltaEvento.IdDisciplina" class="text-danger"></**span**>

</div>

<br />

<div class="form-group">

<**label** **asp-for**="DTOAltaEvento.FechaInicioEvento" class="control-label"></**label**>

<**input** **asp-for**="DTOAltaEvento.FechaInicioEvento" **type**="date" class="form-control"

min="2024-01-01" **value**="@DateTime.Today.ToString("yyyy-MM-dd")" />

<**span** **asp-validation-for**="DTOAltaEvento.FechaInicioEvento" class="text-danger"></**span**>

</div>

<br />

<div class="form-group">

<**label** **asp-for**="DTOAltaEvento.FechaFinalEvento" class="control-label"></**label**>

<**input** **asp-for**="DTOAltaEvento.FechaFinalEvento" **type**="date" class="form-control"

min="@Model.DTOAltaEvento.FechaInicioEvento.ToString("yyyy-MM-dd")" />

<**span** **asp-validation-for**="DTOAltaEvento.FechaFinalEvento" class="text-danger"></**span**>

</div>

<br />

<div class="form-group">

<label class="control-label">Atletas Participantes</label>

@foreach (var atleta in Model.DTOAtleta)

{

<div class="form-check">

<input type="checkbox" class="form-check-input" id="atleta\_@atleta.IdAtleta"

name="DTOAltaEvento.IdsAtletasPuntajes[]" value="@atleta.IdAtleta" />

<label class="form-check-label" for="atleta\_@atleta.IdAtleta">

@atleta.NombreAtleta @atleta.ApellidoAtleta

</label>

</div>

}

<**span** **asp-validation-for**="DTOAltaEvento.IdsAtletas" class="text-danger"></**span**>

</div>

<br />

<button type="submit" class="btn btn-primary">Guardar</button>

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

</**form**>

@section Scripts {

@await Html.PartialAsync("\_ValidationScriptsPartial")

}

@model string

@{

ViewData["Title"] = "Eliminar Usuario";

}

<h1>Eliminar Usuario</h1>

<p>¿Estás seguro de que deseas eliminar al usuario con el email: @Model?</p>

<**form** **asp-action**="DeleteConfirmed" method="post">

@Html.Hidden("email", Model)

<input type="submit" value="Eliminar" class="btn btn-danger" />

<**a** **asp-action**="ListaUsuarios" class="btn btn-secondary">Cancelar</**a**>

</**form**>

@if (ViewBag.Error != null)

{

<div class="alert alert-danger">@ViewBag.Error</div>

}

@model DTO.AltaUsuarioDTO

@{

ViewData["Title"] = "Details";

}

<h1>Details</h1>

<div>

<h4>AltaUsuarioDTO</h4>

<hr />

<dl class="row">

<dt class="col-sm-2">

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

</dt>

<dd class="col-sm-10">

@Html.DisplayFor(model => model.EmailUsuario)

</dd>

<dt class="col-sm-2">

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

</dt>

<dd class="col-sm-10">

@Html.DisplayFor(model => model.ContraseniaUsuario)

</dd>

<dt class="col-sm-2">

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

</dt>

<dd class="col-sm-10">

@Html.DisplayFor(model => model.IdRol)

</dd>

<dt class="col-sm-2">

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

</dt>

<dd class="col-sm-10">

@Html.DisplayFor(model => model.IdUser)

</dd>

<dt class="col-sm-2">

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

</dt>

<dd class="col-sm-10">

@Html.DisplayFor(model => model.NombreRol)

</dd>

<dt class="col-sm-2">

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

</dt>

<dd class="col-sm-10">

@Html.DisplayFor(model => model.Date)

</dd>

<dt class="col-sm-2">

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

</dt>

<dd class="col-sm-10">

@Html.DisplayFor(model => model.Hora)

</dd>

<dt class="col-sm-2">

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

</dt>

<dd class="col-sm-10">

@Html.DisplayFor(model => model.EmailAdmin)

</dd>

</dl>

</div>

@model Presentacion.Models.AltaUsuarioViewModel

@{

ViewData["Title"] = "Edit";

}

<h1>Edit</h1>

<span>@ViewBag.Error</span>

<h4>AltaUsuarioDTO</h4>

<hr />

<div class="row">

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

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

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

<div class="form-group">

<**label** **asp-for**="DTOAltaUsuario.IdUser" class="control-label"></**label**>

<**input** **asp-for**="DTOAltaUsuario.IdUser" class="form-control" readonly />

<**span** **asp-validation-for**="DTOAltaUsuario.IdUser" class="text-danger"></**span**>

</div>

<div class="form-group">

<**label** **asp-for**="DTOAltaUsuario.EmailUsuario" class="control-label"></**label**>

<**input** **asp-for**="DTOAltaUsuario.EmailUsuario" class="form-control" />

<**span** **asp-validation-for**="DTOAltaUsuario.EmailUsuario" class="text-danger"></**span**>

</div>

<div class="form-group">

<**label** **asp-for**="DTOAltaUsuario.ContraseniaUsuario" class="control-label"></**label**>

<**input** **asp-for**="DTOAltaUsuario.ContraseniaUsuario" class="form-control" />

<**span** **asp-validation-for**="DTOAltaUsuario.ContraseniaUsuario" class="text-danger"></**span**>

</div>

<div class="form-group">

<**label** **asp-for**="DTOAltaUsuario.IdRol" class="control-label"></**label**>

<**select** **asp-for**="DTOAltaUsuario.IdRol" class="form-control">

<**option** **value**="">Seleccione un rol</**option**>

@foreach (var item in Model.RolesDTO)

{

<**option** **value**="@item.IdRol">@item.NombreRol</**option**>

}

</**select**>

<**span** **asp-validation-for**="DTOAltaUsuario.IdRol" class="text-danger"></**span**>

</div>

<div class="form-group">

<input type="submit" value="Save" class="btn btn-primary" />

</div>

</**form**>

</div>

</div>

@section Scripts {

@{

await Html.RenderPartialAsync("\_ValidationScriptsPartial");

}

}

@model IEnumerable<DTO.ListadoUsuariosDTO>

@{

ViewData["Title"] = "Listado de Usuarios";

}

<h1>Listado de Usuarios</h1>

<table class="table">

<thead>

<tr>

<th>@Html.DisplayNameFor(model => model.IdUser)</th>

<th>

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

</th>

<th>

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

</th>

<th>@Html.DisplayNameFor(model => model.EmailAdmin) </th>

<th>@Html.DisplayNameFor(model => model.Date) </th>

<th>@Html.DisplayNameFor(model => model.Hora) </th>

</tr>

</thead>

<tbody>

@foreach (var item in Model)

{

@if (ViewBag.Usuario.Email.Valor != item.EmailUsuario)

{

<tr>

<td>@Html.DisplayFor(modelItem => item.IdUser)</td>

<td>

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

</td>

<td>

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

</td>

<td>

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

</td>

<td>

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

</td>

<td>

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

</td>

<td>

@Html.ActionLink("Edit", "Edit", new { id = item.IdUser }) |

@Html.ActionLink("Details", "Details", new { id = item.IdUser }) |

@Html.ActionLink("Delete", "Delete", new { email = item.EmailUsuario }, new { @class = "text-danger" })

</td>

</tr>

}

else

{

<tr>

<td>@Html.DisplayFor(modelItem => item.IdUser)</td>

<td>

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

</td>

<td>

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

</td>

<td>-</td>

<td>-</td>

<td>-</td>

<td>

@Html.ActionLink("Edit", "Edit", new { id = item.IdUser }) |

@Html.ActionLink("Details", "Details", new { id = item.IdUser })

</td>

</tr>

}

}

@if (!string.IsNullOrEmpty(ViewBag.Error))

{

<tr>

<td colspan="3" class="text-danger">

@ViewBag.Error

</td>

</tr>

}

</tbody>

</table>

PRESENTACION.VIEWS

@using Presentacion

@using Presentacion.Models

@addTagHelper \*, Microsoft.AspNetCore.Mvc.TagHelpers

@{

Layout = "\_Layout";

}

PRESENTACION

{

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft.AspNetCore": "Warning"

}

},

"AllowedHosts": "\*",

"ConnectionStrings": {

"MiConexion": "Server=(localdb)\\MSSQLLOCALDB; Database=Obligatorio\_Olimpiadas; Integrated Security=SSPI"

}

}

using LogicaAplicacion.CU;

using LogicaAplicacion.InterfacesCU;

using LogicaDatos.Repositorios;

using LogicaDatos.Repositorios.LogicaDatos.Repositorios;

using LogicaNegocio.EntidadesDominio;

using LogicaNegocio.InterfacesRepositorios;

using Microsoft.EntityFrameworkCore;

namespace Presentacion

{

public class Program

{

public static void Main(string[] args)

{

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddControllersWithViews();

builder.Services.AddSession();

//USUARIO

builder.Services.AddScoped<IAltaUsuario, AltaUsuario>();

builder.Services.AddScoped<IListadoUsuarios, ListadoUsuarios>();

builder.Services.AddScoped<IListadoRoles, ListadoRoles>();

builder.Services.AddScoped<IRepositorioUsuario, RepositorioUsuarioBD>();

builder.Services.AddScoped<IRepositorioRoles, RepositorioRolesBD>();

builder.Services.AddScoped<IBajaUsuario, BajaUsuario>();

builder.Services.AddScoped<IUpdateUsuario, UpdateUsuario>();

builder.Services.AddScoped<ILoginUsuario, LoginUsuario>();

builder.Services.AddScoped<IBuscarUsuarioPorID, BuscarUsuarioPorId>();

//ATLETA

builder.Services.AddScoped<IAltaAtleta, AltaAtleta>();

builder.Services.AddScoped<IListadoAtletas, ListadoAtletas>();

builder.Services.AddScoped<IRepositorioAtleta, RepositorioAtletaBD>();

//EVENTO

builder.Services.AddScoped<IRepositorioEvento, RepositorioEventoBD>();

builder.Services.AddScoped<IAltaEvento, AltaEvento>();

builder.Services.AddScoped<IListadoEventos, ListadoEventos>();

//EVENTO ATLETA

builder.Services.AddScoped<IRepositorioEventoAtleta, RepositorioEventoAtletaBD>();

builder.Services.AddScoped<IListadoEventosAtletas, ListadoEventoAtletas>();

builder.Services.AddScoped<IAltaEventoAtletaPuntaje, AltaEventoPuntaje>();

//DISCIPLINA

builder.Services.AddScoped<IRepositorioDisciplina, RepositorioDisciplinaBD>();

builder.Services.AddScoped<IAltaDisciplina, AltaDisciplina>();

builder.Services.AddScoped<IListadoDisciplina, ListadoDisciplina>();

// CONF DE LA BASE DE DATOS

string strCon = builder.Configuration.GetConnectionString("MiConexion");

builder.Services.AddDbContext<OlimpiadasContext>(options => options.UseSqlServer(strCon));

var app = builder.Build();

// Configure the HTTP request pipeline.

if (!app.Environment.IsDevelopment())

{

app.UseExceptionHandler("/Home/Error");

}

app.UseStaticFiles();

app.UseRouting();

app.UseSession();

app.UseAuthorization();

app.MapControllerRoute(

name: "default",

pattern: "{controller=Home}/{action=Index}/{id?}");

app.Run();

}

}

}

## 4.7.- Capa Api.

WEBAPI

using DTO;

using DTO.Mappers;

using ExcepcionesPropias;

using LogicaAplicacion.CU;

using LogicaAplicacion.InterfacesCU;

using LogicaNegocio.InterfacesRepositorios;

using Microsoft.AspNetCore.Mvc;

using System.Linq.Expressions;

// For more information on enabling Web API for empty projects, visit https://go.microsoft.com/fwlink/?LinkID=397860

namespace Api.Controllers

{

[Route("api/[controller]")]

[ApiController]

public class EventoAtletaController : ControllerBase

{

public IListadoEventosAtletas CUListadoEventoAtleta { get; set; }

public EventoAtletaController(IListadoEventosAtletas cUListadoEventoAtleta)

{

CUListadoEventoAtleta = cUListadoEventoAtleta;

}

// GET api/<EventoAtletaController>/5

[HttpGet("{id}")]

[ProducesResponseType(StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status400BadRequest)]

[ProducesResponseType(StatusCodes.Status404NotFound)]

[ProducesResponseType(StatusCodes.Status500InternalServerError)]

public IActionResult Get(int id)

{

if (id <= 0)

{

return BadRequest("El id debe ser un número mayor o igual a 1.");

}

try

{

IEnumerable<ListadoEventoAtletaDTO> eventos = CUListadoEventoAtleta.GetEventosPorAtleta(id);

if (eventos is not null && eventos.Any())

{

return Ok(eventos);

}

else

{

return NotFound("No se encontraron eventos para el atleta especificado.");

}

}

catch (Exception)

{

return StatusCode(500, "Ocurrió un error al procesar la solicitud.");

}

}

}

}

{

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft.AspNetCore": "Warning"

}

},

"AllowedHosts": "\*",

"ConnectionStrings": {

"MiConexion": "Server=(localdb)\\MSSQLLOCALDB; Database=Obligatorio\_Olimpiadas; Integrated Security=SSPI"

}

}

using LogicaAplicacion.CU;

using LogicaAplicacion.InterfacesCU;

using LogicaDatos.Repositorios;

using LogicaDatos.Repositorios.LogicaDatos.Repositorios;

using LogicaNegocio.EntidadesDominio;

using LogicaNegocio.InterfacesRepositorios;

using Microsoft.EntityFrameworkCore;

namespace Api

{

public class Program

{

public static void Main(string[] args)

{

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

//EVENTO ATLETA

builder.Services.AddScoped<IRepositorioEventoAtleta, RepositorioEventoAtletaBD>();

builder.Services.AddScoped<IListadoEventosAtletas, ListadoEventoAtletas>();

// CONF DE LA BASE DE DATOS

string strCon = builder.Configuration.GetConnectionString("MiConexion");

builder.Services.AddDbContext<OlimpiadasContext>(options => options.UseSqlServer(strCon));

builder.Services.AddControllers();

// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen();

var app = builder.Build();

// Configure the HTTP request pipeline.

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

}

app.UseAuthorization();

app.MapControllers();

app.Run();

}

}

}