**Universidad ORT Uruguay**

**Facultad de Ingeniería**

**Escuela de Tecnología**

**OBLIGATORIO PROGRAMACIÓN 3**

**DOCUMENTO**

Hombre con barba y bigote

Descripción generada automáticamente

**Nicolás Giménez – 291950**

Un hombre con una playera de color azul

Descripción generada automáticamente

**Cristian García – 317010**

**Grupo N3D**

**Docente: Liliana Pino**

**Analista en tecnologías de la información**

**21/10/2024**

​​**Índice**

[1. Diagrama de casos de uso. 4](#_Toc183659770)

[2. Diagrama de clases (UML). 5](#_Toc183659771)

[3.1 Lógica de negocio. 5](#_Toc183659772)

[3.2 Lógica de acceso a datos. 7](#_Toc183659773)

[3.3 Lógica de aplicación. 7](#_Toc183659774)

[3.4 Compartido. 11](#_Toc183659775)

[3.5 WebApi. 13](#_Toc183659776)

[3.6 MVC. 15](#_Toc183659777)

[3. Código fuente. 17](#_Toc183659778)

[4.1 Lógica de negocio. 17](#_Toc183659779)

[4.1.1 Entidades. 17](#_Toc183659780)

[4.1.2 Enums. 21](#_Toc183659781)

[4.1.3 Excepciones Entidades. 22](#_Toc183659782)

[4.1.4 Interfaces Entidades. 23](#_Toc183659783)

[4.1.5 Interfaces Repositorios. 23](#_Toc183659784)

[4.1.6 Value Objects 25](#_Toc183659785)

[4.2 Lógica de acceso a datos. 26](#_Toc183659786)

[4.2.1 Repositorios. 28](#_Toc183659787)

[4.3 Lógica de aplicación. 36](#_Toc183659788)

[4.3.1 Interfaces de casos de usos. 36](#_Toc183659789)

[4.3.2 Casos de usos. 40](#_Toc183659790)

[4.3.3 Validadores. 55](#_Toc183659791)

[4.4 Compartido. 57](#_Toc183659792)

[4.4.1 DTOs. 57](#_Toc183659793)

[4.4.2 Mappers. 62](#_Toc183659794)

[4.5 MVC. 70](#_Toc183659795)

[4.5.1 Controllers. 70](#_Toc183659796)

[4.5.2 Models. 106](#_Toc183659797)

[4.5.3 Utils. 112](#_Toc183659798)

[4.6 WebApi. 113](#_Toc183659799)

[4.6.1 Controllers. 113](#_Toc183659800)

[4. Prompts y datos. 114](#_Toc183659801)

# Diagrama de casos de uso.

Diagrama

Descripción generada automáticamente

# Diagrama de clases (UML).

## 2.1 Lógica de negocio.

Diagrama

Descripción generada automáticamente

Excepciones e interface de entity.

Diagrama

Descripción generada automáticamente

Interfaces de repositorios.

Diagrama

Descripción generada automáticamente

Entidades y value objects.

Diagrama

Descripción generada automáticamente

## 2.2 Lógica de acceso a datos.

Escala de tiempo

Descripción generada automáticamente

## 2.3 Lógica de aplicación.

Diagrama

Descripción generada automáticamenteUsuario.

Diagrama

Descripción generada automáticamenteAtleta.

Diagrama

Descripción generada automáticamenteEvento.

Diagrama

Descripción generada automáticamenteDisciplina.

Diagrama

Descripción generada automáticamentePaís.

Auditoría.

Diagrama

Descripción generada automáticamente

## 2.4 Compartido.

Usuario.

Diagrama

Descripción generada automáticamente

País.

Texto, Carta

Descripción generada automáticamente

Diagrama

Descripción generada automáticamenteAtleta, evento y disciplina.

Auditoría.

Diagrama

Descripción generada automáticamente

## 2.5 WebApi.

AtletaController.

Diagrama

Descripción generada automáticamente

DisciplinaController.

Diagrama

Descripción generada automáticamente

EventoController.

Diagrama

Descripción generada automáticamente

UsuarioController.

Diagrama

Descripción generada automáticamente

## 2.6 MVC.

UsuarioController, ErrorController y HomeController.

Diagrama

Descripción generada automáticamente

AtletaController.

Diagrama

Descripción generada automáticamente

EventoController.

Diagrama

Descripción generada automáticamente

DisciplinaController.

Diagrama

Descripción generada automáticamente

# Código fuente.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: JuegosOlimpicosDBContext.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaAccesoDatos\JuegosOlimpicosDBContext.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using LogicaNegocio.Entidades;

using Microsoft.EntityFrameworkCore;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaAccesoDatos

{

public class JuegosOlimpicosDBContext : DbContext

{

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

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

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

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

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

public DbSet<Auditoria> Auditorias { get; set; }

public JuegosOlimpicosDBContext(DbContextOptions opt) : base(opt)

{

//Pregunta si la base está creada

if (Database.EnsureCreated())

{

//Se ejecuta el método para cargar los datos desde el principio

InicializarDatos();

}

}

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

modelBuilder.Entity<Usuario>()

.OwnsOne(u => u.Email)

.HasIndex(e => e.Valor)

.IsUnique();

modelBuilder.Entity<Disciplina>()

.OwnsOne(d => d.Nombre)

.HasIndex(n => n.Valor)

.IsUnique();

modelBuilder.Entity<Auditoria>()

.Property(a => a.Accion)

.HasConversion<string>();

}

private void InicializarDatos()

{

//Consultar si las tablas ya tienen datos, en caso de estar vacías se cargan datos

if (!Usuarios.Any())

{

EjecutarScript("Usuarios\_Admin.sql");

}

if (!Paises.Any())

{

EjecutarScript("Paises.sql");

}

if (!Atletas.Any())

{

EjecutarScript("Atletas.sql");

EjecutarScript("Disciplinas.sql");

EjecutarScript("AtletasDisciplina.sql");

}

if(!Eventos.Any())

{

EjecutarScript("Eventos.sql");

EjecutarScript("PuntajeEventoAtleta.sql");

}

}

private void EjecutarScript(string nombreScript)

{

//Ruta relativa para acceder a los scripts

string rutaCompleta = Path.Combine(Directory.GetCurrentDirectory(), "..", "ScriptsDatos", nombreScript);

//Leer el archivo

string sql = File.ReadAllText(rutaCompleta);

//Ejecutar el archivo en la base

Database.ExecuteSqlRaw(sql);

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: AtletaMapper.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\Compartido\Mappers\AtletaMapper.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using Compartido.DTOs.Atletas;

using LogicaNegocio.Entidades;

using LogicaNegocio.ExcepcionesEntidades;

using LogicaNegocio.Enums;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Compartido.Mappers

{

public static class AtletaMapper

{

public static IEnumerable<AtletaListaDTO> AtletasToListaDTO(List<Atleta> atletas)

{

return atletas.Select(a => new AtletaListaDTO

{

Id = a.Id,

Nombre = a.Nombre,

Apellido = a.Apellido,

NombreCompleto = a.Nombre + " " + a.Apellido,

NombrePais = a.Pais.Nombre,

Sexo = a.Sexo.ToString(),

});

}

public static AtletaDTO AtletaToDTO(Atleta atleta)

{

if (atleta == null)

{

throw new AtletaException("Atleta vacío en mapper");

}

AtletaDTO res = new AtletaDTO

{

Id = atleta.Id,

Nombre = atleta.Nombre,

Apellido = atleta.Apellido,

NombrePais = atleta.Pais.Nombre,

Sexo = atleta.Sexo.ToString(),

DisciplinasAtleta = DisciplinaMapper.DisciplinasToListaDTO(atleta.LiDisciplinas)

};

return res;

}

public static AtletaDTO AtletaToPuntajeEventoDTO(Atleta atleta)

{

if (atleta == null)

{

throw new AtletaException("Atleta vacío en mapper");

}

AtletaDTO res = new AtletaDTO

{

Id = atleta.Id,

Nombre = atleta.Nombre,

Apellido = atleta.Apellido,

NombrePais = atleta.Pais.Nombre,

Sexo = atleta.Sexo.ToString()

};

return res;

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: AuditoriaMapper.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\Compartido\Mappers\AuditoriaMapper.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using Compartido.DTOs.Auditorias;

using LogicaNegocio.Entidades;

using LogicaNegocio.Enums;

using LogicaNegocio.ExcepcionesEntidades;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Compartido.Mappers

{

public static class AuditoriaMapper

{

public static Auditoria DtoInsertToAuditoria(AuditoriaInsertDTO dto)

{

if (dto == null) throw new AuditoriaException("DTO vacío en Auditoria mapper");

return new Auditoria

{

Accion = dto.Accion,

EmailUsuario = dto.EmailUsuario,

Entidad = dto.Entidad,

EntidadId = dto.EntidadId,

Fecha = DateTime.Now

};

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: DisciplinaMapper.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\Compartido\Mappers\DisciplinaMapper.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using Compartido.DTOs.Disciplinas;

using LogicaNegocio.Entidades;

using LogicaNegocio.ExcepcionesEntidades;

using LogicaNegocio.ValueObjects.Disciplina;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Compartido.Mappers

{

public static class DisciplinaMapper

{

public static IEnumerable<DisciplinaDTO> DisciplinasToListaDTO(List<Disciplina> disciplinas)

{

IEnumerable<DisciplinaDTO> disciplinasListaDTO = disciplinas.Select(d => new DisciplinaDTO

{

Id = d.Id,

Nombre = d.Nombre.Valor,

AnioIntegracion = d.AnioIntegracion

});

return disciplinasListaDTO;

}

public static Disciplina InsertDTOToDisciplina(DisciplinaInsertDTO disciplinaInsertDTO)

{

if (disciplinaInsertDTO == null) throw new DisciplinaException("Discplina insert vacia en mapper");

Disciplina res = new Disciplina

{

Nombre = new RDisciplinaNombre(disciplinaInsertDTO.Nombre),

AnioIntegracion = disciplinaInsertDTO.AnioIntegracion,

};

return res;

}

public static Disciplina DTOToDisciplina(DisciplinaUpdateDTO dto)

{

if (dto == null) throw new DisciplinaException("Discplina update vacia en mapper");

return new Disciplina

{

Id = dto.Id,

Nombre = new RDisciplinaNombre(dto.Nombre),

AnioIntegracion = dto.AnioIntegracion

};

}

public static DisciplinaDTO DisciplinaToDTO(Disciplina disciplina)

{

if (disciplina == null) throw new DisciplinaException("Discplina vacia en mapper");

return new DisciplinaDTO

{

Id = disciplina.Id,

Nombre = disciplina.Nombre.Valor,

AnioIntegracion = disciplina.AnioIntegracion

};

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: EventoMapper.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\Compartido\Mappers\EventoMapper.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using Compartido.DTOs.Eventos;

using LogicaNegocio.Entidades;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using LogicaNegocio.ExcepcionesEntidades;

namespace Compartido.Mappers

{

public static class EventoMapper

{

public static Evento DtoToEvento(EventoInsertDTO dto)

{

if (dto == null) throw new EventoException("Evento insert DTO vacío en mapper");

Evento res = new Evento

{

DisciplinaId = dto.DisciplinaId,

FchInicio = dto.FchInicio,

FchFin = dto.FchFin,

NombrePrueba = dto.NombrePrueba,

LiPuntajes = dto.AtletasId.Select(id => new PuntajeEventoAtleta

{

AtletaId = id,

Puntaje = 0

}).ToList()

};

return res;

}

public static IEnumerable<EventoListaDTO> EventosToListaDTO(List<Evento> listaEventos)

{

IEnumerable<EventoListaDTO> eventos = listaEventos.Select(e => new EventoListaDTO

{

EventoId = e.Id,

NombrePrueba = e.NombrePrueba,

FchInicio = e.FchInicio,

FchFin = e.FchFin

});

return eventos;

}

public static EventoDTO EventoToDTO(Evento evento)

{

if (evento == null) throw new EventoException("Evento vacío en mapper");

EventoDTO eventoDto = new EventoDTO

{

Id = evento.Id,

FchInicio = evento.FchInicio,

FchFin = evento.FchFin,

NombrePrueba = evento.NombrePrueba,

LiAtletas = evento.LiPuntajes.Select(p => new PuntajeEventoAtletaDTO

{

Atleta = AtletaMapper.AtletaToPuntajeEventoDTO(p.Atleta),

Puntaje = p.Puntaje

})

};

return eventoDto;

}

public static List<PuntajeEventoAtleta> DtoListaModificadaToListaPuntaje(IEnumerable<PEAUpdateDTO> listaModificada)

{

if (listaModificada == null) throw new EventoException("Lista de puntajes DTO vacía en mapper");

return listaModificada.Select(p => new PuntajeEventoAtleta

{

AtletaId = p.AtletaId,

Puntaje = p.Puntaje,

}).ToList();

}

public static IEnumerable<AtletaEventoListaDTO> EventoToListaAtletaEventoDTO(List<Evento> lista)

{

if (lista == null) throw new EventoException("Lista de eventos vacía en mapper");

return lista.Select(e => new AtletaEventoListaDTO

{

Id = e.Id,

NombrePrueba = e.NombrePrueba,

NombreDisciplina = e.Disciplina.Nombre.Valor,

FchFin = e.FchFin,

FchInicio = e.FchInicio

});

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: PaisMapper.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\Compartido\Mappers\PaisMapper.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Compartido.Mappers

{

public static class PaisMapper

{

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: UsuarioMapper.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\Compartido\Mappers\UsuarioMapper.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using Compartido.DTOs.Usuarios;

using LogicaNegocio.Entidades;

using LogicaNegocio.Enums;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using LogicaNegocio.ExcepcionesEntidades;

using LogicaNegocio.ValueObjects.Usuario;

namespace Compartido.Mappers

{

public static class UsuarioMapper

{

public static UsuarioDTO UsuarioToDTO(Usuario usuario)

{

if (usuario == null)

{

throw new UsuarioException("Error en mapper: el Usuario ingresado está vacío");

}

return new UsuarioDTO

{

Id = usuario.Id,

Email = usuario.Email.Valor,

RolUsuario = usuario.RolUsuario.ToString(),

Nombre = usuario.Nombre,

FechaRegistro = usuario.FechaRegistro.ToString("dd/MM/yyyy")

};

}

public static Usuario InsertDTOToUsuario(UsuarioInsertDTO usuarioInsertDto)

{

if (usuarioInsertDto == null)

{

throw new UsuarioException("Error en mapper: el Usuario insert está vacío");

}

return new Usuario

{

Email = new RUsuarioEmail(usuarioInsertDto.Email),

Nombre = usuarioInsertDto.Nombre,

Contrasena = new RUsuarioContrasena(usuarioInsertDto.Contrasena),

RolUsuario = usuarioInsertDto.RolUsuario == 0 ? Rol.Administrador : Rol.Digitador,

IdAdminRegistro = usuarioInsertDto.IdAdminRegistro

};

}

public static IEnumerable<UsuarioDTO> ListaUsuariosToDTOListaUsuarios(List<Usuario> usuarios)

{

IEnumerable<UsuarioDTO> usuariosDTOs = usuarios.Select(u => new UsuarioDTO()

{

Email = u.Email.Valor,

Nombre = u.Nombre,

Id = u.Id,

RolUsuario = u.RolUsuario.ToString(),

FechaRegistro = u.FechaRegistro.ToString("dd/MM/yyyy")

});

return usuariosDTOs;

}

public static UsuarioUpdateDTO UsuarioToUpdateDTO(Usuario usuario)

{

if (usuario == null)

{

throw new UsuarioException("Error en mapper: el Usuario está vacío");

}

return new UsuarioUpdateDTO

{

Email = usuario.Email.Valor,

Nombre = usuario.Nombre

};

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: 20241127023058\_init.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaAccesoDatos\Migrations\20241127023058\_init.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using Microsoft.EntityFrameworkCore.Migrations;

#nullable disable

namespace LogicaAccesoDatos.Migrations

{

/// <inheritdoc />

public partial class init : Migration

{

/// <inheritdoc />

protected override void Up(MigrationBuilder migrationBuilder)

{

migrationBuilder.CreateTable(

name: "Auditorias",

columns: table => new

{

Id = table.Column<int>(type: "int", nullable: false)

.Annotation("SqlServer:Identity", "1, 1"),

Fecha = table.Column<DateTime>(type: "datetime2", nullable: false),

Accion = table.Column<string>(type: "nvarchar(max)", nullable: false),

Entidad = table.Column<string>(type: "nvarchar(max)", nullable: false),

EntidadId = table.Column<int>(type: "int", nullable: false),

EmailUsuario = table.Column<string>(type: "nvarchar(max)", nullable: false)

},

constraints: table =>

{

table.PrimaryKey("PK\_Auditorias", x => x.Id);

});

migrationBuilder.CreateTable(

name: "Disciplinas",

columns: table => new

{

Id = table.Column<int>(type: "int", nullable: false)

.Annotation("SqlServer:Identity", "1, 1"),

Nombre\_Valor = table.Column<string>(type: "nvarchar(50)", maxLength: 50, nullable: false),

AnioIntegracion = table.Column<int>(type: "int", nullable: false)

},

constraints: table =>

{

table.PrimaryKey("PK\_Disciplinas", x => x.Id);

});

migrationBuilder.CreateTable(

name: "Paises",

columns: table => new

{

Id = table.Column<int>(type: "int", nullable: false)

.Annotation("SqlServer:Identity", "1, 1"),

Nombre = table.Column<string>(type: "nvarchar(max)", nullable: false),

Habitantes = table.Column<int>(type: "int", nullable: false),

NombreDelegado = table.Column<string>(type: "nvarchar(max)", nullable: false),

TelDelegado = table.Column<string>(type: "nvarchar(max)", nullable: false)

},

constraints: table =>

{

table.PrimaryKey("PK\_Paises", x => x.Id);

});

migrationBuilder.CreateTable(

name: "Usuarios",

columns: table => new

{

Id = table.Column<int>(type: "int", nullable: false)

.Annotation("SqlServer:Identity", "1, 1"),

Email\_Valor = table.Column<string>(type: "nvarchar(255)", maxLength: 255, nullable: false),

RolUsuario = table.Column<int>(type: "int", nullable: false),

FechaRegistro = table.Column<DateTime>(type: "datetime2", nullable: false),

IdAdminRegistro = table.Column<int>(type: "int", nullable: false),

Nombre = table.Column<string>(type: "nvarchar(50)", maxLength: 50, nullable: true),

Contrasena\_Valor = table.Column<string>(type: "nvarchar(max)", nullable: false)

},

constraints: table =>

{

table.PrimaryKey("PK\_Usuarios", x => x.Id);

});

migrationBuilder.CreateTable(

name: "Eventos",

columns: table => new

{

Id = table.Column<int>(type: "int", nullable: false)

.Annotation("SqlServer:Identity", "1, 1"),

NombrePrueba = table.Column<string>(type: "nvarchar(max)", nullable: false),

DisciplinaId = table.Column<int>(type: "int", nullable: false),

FchInicio = table.Column<DateTime>(type: "datetime2", nullable: false),

FchFin = table.Column<DateTime>(type: "datetime2", nullable: false)

},

constraints: table =>

{

table.PrimaryKey("PK\_Eventos", x => x.Id);

table.ForeignKey(

name: "FK\_Eventos\_Disciplinas\_DisciplinaId",

column: x => x.DisciplinaId,

principalTable: "Disciplinas",

principalColumn: "Id",

onDelete: ReferentialAction.Cascade);

});

migrationBuilder.CreateTable(

name: "Atletas",

columns: table => new

{

Id = table.Column<int>(type: "int", nullable: false)

.Annotation("SqlServer:Identity", "1, 1"),

Nombre = table.Column<string>(type: "nvarchar(max)", nullable: false),

Apellido = table.Column<string>(type: "nvarchar(max)", nullable: false),

Sexo = table.Column<int>(type: "int", nullable: false),

PaisId = table.Column<int>(type: "int", nullable: false)

},

constraints: table =>

{

table.PrimaryKey("PK\_Atletas", x => x.Id);

table.ForeignKey(

name: "FK\_Atletas\_Paises\_PaisId",

column: x => x.PaisId,

principalTable: "Paises",

principalColumn: "Id",

onDelete: ReferentialAction.Cascade);

});

migrationBuilder.CreateTable(

name: "AtletaDisciplina",

columns: table => new

{

LiAtletasId = table.Column<int>(type: "int", nullable: false),

LiDisciplinasId = table.Column<int>(type: "int", nullable: false)

},

constraints: table =>

{

table.PrimaryKey("PK\_AtletaDisciplina", x => new { x.LiAtletasId, x.LiDisciplinasId });

table.ForeignKey(

name: "FK\_AtletaDisciplina\_Atletas\_LiAtletasId",

column: x => x.LiAtletasId,

principalTable: "Atletas",

principalColumn: "Id",

onDelete: ReferentialAction.Cascade);

table.ForeignKey(

name: "FK\_AtletaDisciplina\_Disciplinas\_LiDisciplinasId",

column: x => x.LiDisciplinasId,

principalTable: "Disciplinas",

principalColumn: "Id",

onDelete: ReferentialAction.Cascade);

});

migrationBuilder.CreateTable(

name: "PuntajeEventoAtleta",

columns: table => new

{

AtletaId = table.Column<int>(type: "int", nullable: false),

EventoId = table.Column<int>(type: "int", nullable: false),

Puntaje = table.Column<decimal>(type: "decimal(18,2)", nullable: false)

},

constraints: table =>

{

table.PrimaryKey("PK\_PuntajeEventoAtleta", x => new { x.AtletaId, x.EventoId });

table.ForeignKey(

name: "FK\_PuntajeEventoAtleta\_Atletas\_AtletaId",

column: x => x.AtletaId,

principalTable: "Atletas",

principalColumn: "Id",

onDelete: ReferentialAction.Cascade);

table.ForeignKey(

name: "FK\_PuntajeEventoAtleta\_Eventos\_EventoId",

column: x => x.EventoId,

principalTable: "Eventos",

principalColumn: "Id",

onDelete: ReferentialAction.Cascade);

});

migrationBuilder.CreateIndex(

name: "IX\_AtletaDisciplina\_LiDisciplinasId",

table: "AtletaDisciplina",

column: "LiDisciplinasId");

migrationBuilder.CreateIndex(

name: "IX\_Atletas\_PaisId",

table: "Atletas",

column: "PaisId");

migrationBuilder.CreateIndex(

name: "IX\_Disciplinas\_Nombre\_Valor",

table: "Disciplinas",

column: "Nombre\_Valor",

unique: true);

migrationBuilder.CreateIndex(

name: "IX\_Eventos\_DisciplinaId",

table: "Eventos",

column: "DisciplinaId");

migrationBuilder.CreateIndex(

name: "IX\_PuntajeEventoAtleta\_EventoId",

table: "PuntajeEventoAtleta",

column: "EventoId");

migrationBuilder.CreateIndex(

name: "IX\_Usuarios\_Email\_Valor",

table: "Usuarios",

column: "Email\_Valor",

unique: true);

}

/// <inheritdoc />

protected override void Down(MigrationBuilder migrationBuilder)

{

migrationBuilder.DropTable(

name: "AtletaDisciplina");

migrationBuilder.DropTable(

name: "Auditorias");

migrationBuilder.DropTable(

name: "PuntajeEventoAtleta");

migrationBuilder.DropTable(

name: "Usuarios");

migrationBuilder.DropTable(

name: "Atletas");

migrationBuilder.DropTable(

name: "Eventos");

migrationBuilder.DropTable(

name: "Paises");

migrationBuilder.DropTable(

name: "Disciplinas");

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: 20241127023058\_init.Designer.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaAccesoDatos\Migrations\20241127023058\_init.Designer.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// <auto-generated />

using System;

using System.Collections.Generic;

using LogicaAccesoDatos;

using Microsoft.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore.Infrastructure;

using Microsoft.EntityFrameworkCore.Metadata;

using Microsoft.EntityFrameworkCore.Migrations;

using Microsoft.EntityFrameworkCore.Storage.ValueConversion;

#nullable disable

namespace LogicaAccesoDatos.Migrations

{

[DbContext(typeof(JuegosOlimpicosDBContext))]

[Migration("20241127023058\_init")]

partial class init

{

/// <inheritdoc />

protected override void BuildTargetModel(ModelBuilder modelBuilder)

{

#pragma warning disable 612, 618

modelBuilder

.HasAnnotation("ProductVersion", "8.0.11")

.HasAnnotation("Relational:MaxIdentifierLength", 128);

SqlServerModelBuilderExtensions.UseIdentityColumns(modelBuilder);

modelBuilder.Entity("AtletaDisciplina", b =>

{

b.Property<int>("LiAtletasId")

.HasColumnType("int");

b.Property<int>("LiDisciplinasId")

.HasColumnType("int");

b.HasKey("LiAtletasId", "LiDisciplinasId");

b.HasIndex("LiDisciplinasId");

b.ToTable("AtletaDisciplina");

});

modelBuilder.Entity("LogicaNegocio.Entidades.Atleta", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int");

SqlServerPropertyBuilderExtensions.UseIdentityColumn(b.Property<int>("Id"));

b.Property<string>("Apellido")

.IsRequired()

.HasColumnType("nvarchar(max)");

b.Property<string>("Nombre")

.IsRequired()

.HasColumnType("nvarchar(max)");

b.Property<int>("PaisId")

.HasColumnType("int");

b.Property<int>("Sexo")

.HasColumnType("int");

b.HasKey("Id");

b.HasIndex("PaisId");

b.ToTable("Atletas");

});

modelBuilder.Entity("LogicaNegocio.Entidades.Auditoria", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int");

SqlServerPropertyBuilderExtensions.UseIdentityColumn(b.Property<int>("Id"));

b.Property<string>("Accion")

.IsRequired()

.HasColumnType("nvarchar(max)");

b.Property<string>("EmailUsuario")

.IsRequired()

.HasColumnType("nvarchar(max)");

b.Property<string>("Entidad")

.IsRequired()

.HasColumnType("nvarchar(max)");

b.Property<int>("EntidadId")

.HasColumnType("int");

b.Property<DateTime>("Fecha")

.HasColumnType("datetime2");

b.HasKey("Id");

b.ToTable("Auditorias");

});

modelBuilder.Entity("LogicaNegocio.Entidades.Disciplina", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int");

SqlServerPropertyBuilderExtensions.UseIdentityColumn(b.Property<int>("Id"));

b.Property<int>("AnioIntegracion")

.HasColumnType("int");

b.HasKey("Id");

b.ToTable("Disciplinas");

});

modelBuilder.Entity("LogicaNegocio.Entidades.Evento", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int");

SqlServerPropertyBuilderExtensions.UseIdentityColumn(b.Property<int>("Id"));

b.Property<int>("DisciplinaId")

.HasColumnType("int");

b.Property<DateTime>("FchFin")

.HasColumnType("datetime2");

b.Property<DateTime>("FchInicio")

.HasColumnType("datetime2");

b.Property<string>("NombrePrueba")

.IsRequired()

.HasColumnType("nvarchar(max)");

b.HasKey("Id");

b.HasIndex("DisciplinaId");

b.ToTable("Eventos");

});

modelBuilder.Entity("LogicaNegocio.Entidades.Pais", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int");

SqlServerPropertyBuilderExtensions.UseIdentityColumn(b.Property<int>("Id"));

b.Property<int>("Habitantes")

.HasColumnType("int");

b.Property<string>("Nombre")

.IsRequired()

.HasColumnType("nvarchar(max)");

b.Property<string>("NombreDelegado")

.IsRequired()

.HasColumnType("nvarchar(max)");

b.Property<string>("TelDelegado")

.IsRequired()

.HasColumnType("nvarchar(max)");

b.HasKey("Id");

b.ToTable("Paises");

});

modelBuilder.Entity("LogicaNegocio.Entidades.PuntajeEventoAtleta", b =>

{

b.Property<int>("AtletaId")

.HasColumnType("int");

b.Property<int>("EventoId")

.HasColumnType("int");

b.Property<decimal>("Puntaje")

.HasColumnType("decimal(18,2)");

b.HasKey("AtletaId", "EventoId");

b.HasIndex("EventoId");

b.ToTable("PuntajeEventoAtleta");

});

modelBuilder.Entity("LogicaNegocio.Entidades.Usuario", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int");

SqlServerPropertyBuilderExtensions.UseIdentityColumn(b.Property<int>("Id"));

b.Property<DateTime>("FechaRegistro")

.HasColumnType("datetime2");

b.Property<int>("IdAdminRegistro")

.HasColumnType("int");

b.Property<string>("Nombre")

.HasMaxLength(50)

.HasColumnType("nvarchar(50)");

b.Property<int>("RolUsuario")

.HasColumnType("int");

b.ComplexProperty<Dictionary<string, object>>("Contrasena", "LogicaNegocio.Entidades.Usuario.Contrasena#RUsuarioContrasena", b1 =>

{

b1.IsRequired();

b1.Property<string>("Valor")

.IsRequired()

.HasColumnType("nvarchar(max)");

});

b.HasKey("Id");

b.ToTable("Usuarios");

});

modelBuilder.Entity("AtletaDisciplina", b =>

{

b.HasOne("LogicaNegocio.Entidades.Atleta", null)

.WithMany()

.HasForeignKey("LiAtletasId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.HasOne("LogicaNegocio.Entidades.Disciplina", null)

.WithMany()

.HasForeignKey("LiDisciplinasId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("LogicaNegocio.Entidades.Atleta", b =>

{

b.HasOne("LogicaNegocio.Entidades.Pais", "Pais")

.WithMany()

.HasForeignKey("PaisId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.Navigation("Pais");

});

modelBuilder.Entity("LogicaNegocio.Entidades.Disciplina", b =>

{

b.OwnsOne("LogicaNegocio.ValueObjects.Disciplina.RDisciplinaNombre", "Nombre", b1 =>

{

b1.Property<int>("DisciplinaId")

.HasColumnType("int");

b1.Property<string>("Valor")

.IsRequired()

.HasMaxLength(50)

.HasColumnType("nvarchar(50)");

b1.HasKey("DisciplinaId");

b1.HasIndex("Valor")

.IsUnique();

b1.ToTable("Disciplinas");

b1.WithOwner()

.HasForeignKey("DisciplinaId");

});

b.Navigation("Nombre")

.IsRequired();

});

modelBuilder.Entity("LogicaNegocio.Entidades.Evento", b =>

{

b.HasOne("LogicaNegocio.Entidades.Disciplina", "Disciplina")

.WithMany()

.HasForeignKey("DisciplinaId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.Navigation("Disciplina");

});

modelBuilder.Entity("LogicaNegocio.Entidades.PuntajeEventoAtleta", b =>

{

b.HasOne("LogicaNegocio.Entidades.Atleta", "Atleta")

.WithMany()

.HasForeignKey("AtletaId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.HasOne("LogicaNegocio.Entidades.Evento", null)

.WithMany("LiPuntajes")

.HasForeignKey("EventoId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.Navigation("Atleta");

});

modelBuilder.Entity("LogicaNegocio.Entidades.Usuario", b =>

{

b.OwnsOne("LogicaNegocio.ValueObjects.Usuario.RUsuarioEmail", "Email", b1 =>

{

b1.Property<int>("UsuarioId")

.HasColumnType("int");

b1.Property<string>("Valor")

.IsRequired()

.HasMaxLength(255)

.HasColumnType("nvarchar(255)");

b1.HasKey("UsuarioId");

b1.HasIndex("Valor")

.IsUnique();

b1.ToTable("Usuarios");

b1.WithOwner()

.HasForeignKey("UsuarioId");

});

b.Navigation("Email")

.IsRequired();

});

modelBuilder.Entity("LogicaNegocio.Entidades.Evento", b =>

{

b.Navigation("LiPuntajes");

});

#pragma warning restore 612, 618

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: JuegosOlimpicosDBContextModelSnapshot.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaAccesoDatos\Migrations\JuegosOlimpicosDBContextModelSnapshot.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// <auto-generated />

using System;

using System.Collections.Generic;

using LogicaAccesoDatos;

using Microsoft.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore.Infrastructure;

using Microsoft.EntityFrameworkCore.Metadata;

using Microsoft.EntityFrameworkCore.Storage.ValueConversion;

#nullable disable

namespace LogicaAccesoDatos.Migrations

{

[DbContext(typeof(JuegosOlimpicosDBContext))]

partial class JuegosOlimpicosDBContextModelSnapshot : ModelSnapshot

{

protected override void BuildModel(ModelBuilder modelBuilder)

{

#pragma warning disable 612, 618

modelBuilder

.HasAnnotation("ProductVersion", "8.0.11")

.HasAnnotation("Relational:MaxIdentifierLength", 128);

SqlServerModelBuilderExtensions.UseIdentityColumns(modelBuilder);

modelBuilder.Entity("AtletaDisciplina", b =>

{

b.Property<int>("LiAtletasId")

.HasColumnType("int");

b.Property<int>("LiDisciplinasId")

.HasColumnType("int");

b.HasKey("LiAtletasId", "LiDisciplinasId");

b.HasIndex("LiDisciplinasId");

b.ToTable("AtletaDisciplina");

});

modelBuilder.Entity("LogicaNegocio.Entidades.Atleta", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int");

SqlServerPropertyBuilderExtensions.UseIdentityColumn(b.Property<int>("Id"));

b.Property<string>("Apellido")

.IsRequired()

.HasColumnType("nvarchar(max)");

b.Property<string>("Nombre")

.IsRequired()

.HasColumnType("nvarchar(max)");

b.Property<int>("PaisId")

.HasColumnType("int");

b.Property<int>("Sexo")

.HasColumnType("int");

b.HasKey("Id");

b.HasIndex("PaisId");

b.ToTable("Atletas");

});

modelBuilder.Entity("LogicaNegocio.Entidades.Auditoria", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int");

SqlServerPropertyBuilderExtensions.UseIdentityColumn(b.Property<int>("Id"));

b.Property<string>("Accion")

.IsRequired()

.HasColumnType("nvarchar(max)");

b.Property<string>("EmailUsuario")

.IsRequired()

.HasColumnType("nvarchar(max)");

b.Property<string>("Entidad")

.IsRequired()

.HasColumnType("nvarchar(max)");

b.Property<int>("EntidadId")

.HasColumnType("int");

b.Property<DateTime>("Fecha")

.HasColumnType("datetime2");

b.HasKey("Id");

b.ToTable("Auditorias");

});

modelBuilder.Entity("LogicaNegocio.Entidades.Disciplina", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int");

SqlServerPropertyBuilderExtensions.UseIdentityColumn(b.Property<int>("Id"));

b.Property<int>("AnioIntegracion")

.HasColumnType("int");

b.HasKey("Id");

b.ToTable("Disciplinas");

});

modelBuilder.Entity("LogicaNegocio.Entidades.Evento", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int");

SqlServerPropertyBuilderExtensions.UseIdentityColumn(b.Property<int>("Id"));

b.Property<int>("DisciplinaId")

.HasColumnType("int");

b.Property<DateTime>("FchFin")

.HasColumnType("datetime2");

b.Property<DateTime>("FchInicio")

.HasColumnType("datetime2");

b.Property<string>("NombrePrueba")

.IsRequired()

.HasColumnType("nvarchar(max)");

b.HasKey("Id");

b.HasIndex("DisciplinaId");

b.ToTable("Eventos");

});

modelBuilder.Entity("LogicaNegocio.Entidades.Pais", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int");

SqlServerPropertyBuilderExtensions.UseIdentityColumn(b.Property<int>("Id"));

b.Property<int>("Habitantes")

.HasColumnType("int");

b.Property<string>("Nombre")

.IsRequired()

.HasColumnType("nvarchar(max)");

b.Property<string>("NombreDelegado")

.IsRequired()

.HasColumnType("nvarchar(max)");

b.Property<string>("TelDelegado")

.IsRequired()

.HasColumnType("nvarchar(max)");

b.HasKey("Id");

b.ToTable("Paises");

});

modelBuilder.Entity("LogicaNegocio.Entidades.PuntajeEventoAtleta", b =>

{

b.Property<int>("AtletaId")

.HasColumnType("int");

b.Property<int>("EventoId")

.HasColumnType("int");

b.Property<decimal>("Puntaje")

.HasColumnType("decimal(18,2)");

b.HasKey("AtletaId", "EventoId");

b.HasIndex("EventoId");

b.ToTable("PuntajeEventoAtleta");

});

modelBuilder.Entity("LogicaNegocio.Entidades.Usuario", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int");

SqlServerPropertyBuilderExtensions.UseIdentityColumn(b.Property<int>("Id"));

b.Property<DateTime>("FechaRegistro")

.HasColumnType("datetime2");

b.Property<int>("IdAdminRegistro")

.HasColumnType("int");

b.Property<string>("Nombre")

.HasMaxLength(50)

.HasColumnType("nvarchar(50)");

b.Property<int>("RolUsuario")

.HasColumnType("int");

b.ComplexProperty<Dictionary<string, object>>("Contrasena", "LogicaNegocio.Entidades.Usuario.Contrasena#RUsuarioContrasena", b1 =>

{

b1.IsRequired();

b1.Property<string>("Valor")

.IsRequired()

.HasColumnType("nvarchar(max)");

});

b.HasKey("Id");

b.ToTable("Usuarios");

});

modelBuilder.Entity("AtletaDisciplina", b =>

{

b.HasOne("LogicaNegocio.Entidades.Atleta", null)

.WithMany()

.HasForeignKey("LiAtletasId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.HasOne("LogicaNegocio.Entidades.Disciplina", null)

.WithMany()

.HasForeignKey("LiDisciplinasId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("LogicaNegocio.Entidades.Atleta", b =>

{

b.HasOne("LogicaNegocio.Entidades.Pais", "Pais")

.WithMany()

.HasForeignKey("PaisId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.Navigation("Pais");

});

modelBuilder.Entity("LogicaNegocio.Entidades.Disciplina", b =>

{

b.OwnsOne("LogicaNegocio.ValueObjects.Disciplina.RDisciplinaNombre", "Nombre", b1 =>

{

b1.Property<int>("DisciplinaId")

.HasColumnType("int");

b1.Property<string>("Valor")

.IsRequired()

.HasMaxLength(50)

.HasColumnType("nvarchar(50)");

b1.HasKey("DisciplinaId");

b1.HasIndex("Valor")

.IsUnique();

b1.ToTable("Disciplinas");

b1.WithOwner()

.HasForeignKey("DisciplinaId");

});

b.Navigation("Nombre")

.IsRequired();

});

modelBuilder.Entity("LogicaNegocio.Entidades.Evento", b =>

{

b.HasOne("LogicaNegocio.Entidades.Disciplina", "Disciplina")

.WithMany()

.HasForeignKey("DisciplinaId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.Navigation("Disciplina");

});

modelBuilder.Entity("LogicaNegocio.Entidades.PuntajeEventoAtleta", b =>

{

b.HasOne("LogicaNegocio.Entidades.Atleta", "Atleta")

.WithMany()

.HasForeignKey("AtletaId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.HasOne("LogicaNegocio.Entidades.Evento", null)

.WithMany("LiPuntajes")

.HasForeignKey("EventoId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.Navigation("Atleta");

});

modelBuilder.Entity("LogicaNegocio.Entidades.Usuario", b =>

{

b.OwnsOne("LogicaNegocio.ValueObjects.Usuario.RUsuarioEmail", "Email", b1 =>

{

b1.Property<int>("UsuarioId")

.HasColumnType("int");

b1.Property<string>("Valor")

.IsRequired()

.HasMaxLength(255)

.HasColumnType("nvarchar(255)");

b1.HasKey("UsuarioId");

b1.HasIndex("Valor")

.IsUnique();

b1.ToTable("Usuarios");

b1.WithOwner()

.HasForeignKey("UsuarioId");

});

b.Navigation("Email")

.IsRequired();

});

modelBuilder.Entity("LogicaNegocio.Entidades.Evento", b =>

{

b.Navigation("LiPuntajes");

});

#pragma warning restore 612, 618

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: RepositorioAtleta.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaAccesoDatos\Repositorios\RepositorioAtleta.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using LogicaNegocio.Entidades;

using LogicaNegocio.IRepositorios;

using Microsoft.EntityFrameworkCore;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using LogicaNegocio.ExcepcionesEntidades;

namespace LogicaAccesoDatos.Repositorios

{

public class RepositorioAtleta : IRepositorioAtleta

{

private readonly JuegosOlimpicosDBContext \_context;

public RepositorioAtleta(JuegosOlimpicosDBContext context)

{

\_context = context;

}

public void Add(Atleta item)

{

throw new NotImplementedException();

}

public void Delete(Atleta item)

{

throw new NotImplementedException();

}

public List<Atleta> GetAll() =>

\_context.Atletas

.Include(a => a.Pais)

.OrderBy(a => a.Pais.Nombre)

.ThenBy(a => a.Apellido)

.ThenBy(a => a.Nombre)

.ToList();

public Atleta GetById(int id)

{

Atleta atleta = \_context.Atletas

.Include(a => a.Pais)

.Include(a => a.LiDisciplinas)

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

if (atleta == null)

{

throw new AtletaException("Atleta no encontrado por id");

}

return atleta;

}

public void Update(Atleta item)

{

throw new NotImplementedException();

}

public void GuardarCambios()

{

\_context.SaveChanges();

}

public List<Atleta>? SelectByDisciplinaId(int disciplinaId)

{

Disciplina? encontrado = \_context.Disciplinas

.Include(d => d.LiAtletas)

.ThenInclude(a => a.Pais)

.FirstOrDefault(d => d.Id == disciplinaId);

return encontrado == null

? null

: encontrado.LiAtletas;

}

//RF2 – Listado de Atletas por Disciplina(API Web + HttpClient) – Sin autenticación

//- Crear un listado de atletas filtrado por disciplina.Este endpoint permitirá consultar todos los atletas

//que están registrados en una disciplina dado su Id.El listado se retornará ordenado alfabéticamente

//por nombre completo de atleta.Se incluirán como mínimo su Id (o número, según haya utilizado), su

//nombre completo y el nombre de su país.

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: RepositorioAuditoria.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaAccesoDatos\Repositorios\RepositorioAuditoria.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using LogicaNegocio.Entidades;

using LogicaNegocio.IRepositorios;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaAccesoDatos.Repositorios

{

public class RepositorioAuditoria:IRepositorioAuditoria

{

private readonly JuegosOlimpicosDBContext \_context;

public RepositorioAuditoria(JuegosOlimpicosDBContext context)

{

\_context = context;

}

public void Add(Auditoria item)

{

\_context.Add(item);

\_context.SaveChanges();

}

public void Delete(Auditoria item)

{

throw new NotImplementedException();

}

public List<Auditoria> GetAll()

{

throw new NotImplementedException();

}

public Auditoria? GetById(int id)

{

throw new NotImplementedException();

}

public void Update(Auditoria item)

{

throw new NotImplementedException();

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: RepositorioDisciplina.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaAccesoDatos\Repositorios\RepositorioDisciplina.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using LogicaNegocio.Entidades;

using LogicaNegocio.ExcepcionesEntidades;

using LogicaNegocio.IRepositorios;

using Microsoft.EntityFrameworkCore;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaAccesoDatos.Repositorios

{

public class RepositorioDisciplina : IRepositorioDisciplina

{

private readonly JuegosOlimpicosDBContext \_context;

public RepositorioDisciplina(JuegosOlimpicosDBContext context)

{

\_context = context;

}

public Disciplina Insert(Disciplina item)

{

\_context.Disciplinas.Add(item);

\_context.SaveChanges();

return item;

}

public void Delete(Disciplina item)

{

\_context.Disciplinas.Remove(item);

\_context.SaveChanges();

}

public List<Disciplina> GetAll() =>

\_context.Disciplinas

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

.ToList();

public List<Atleta> GetAtletasDisciplina(int idDisciplina)

{

Disciplina buscar = \_context.Disciplinas

.Include(d => d.LiAtletas)

.ThenInclude(a => a.Pais)

.SingleOrDefault(d => d.Id == idDisciplina);

if (buscar == null) throw new DisciplinaException("Disciplina no encontrada con ese id");

return buscar.LiAtletas;

}

public Disciplina? GetById(int id) =>

\_context.Disciplinas

.Include(d => d.LiAtletas)

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

public void Update(Disciplina item)

{

\_context.Disciplinas.Update(item);

\_context.SaveChanges();

}

public Disciplina? GetByNombre(string nombre) =>

\_context.Disciplinas.SingleOrDefault(disiplina => disiplina.Nombre.Valor == nombre);

public void Add(Disciplina item)

{

\_context.Disciplinas.Add(item);

\_context.SaveChanges();

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: RepositorioEvento.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaAccesoDatos\Repositorios\RepositorioEvento.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using LogicaNegocio.Entidades;

using LogicaNegocio.IRepositorios;

using Microsoft.EntityFrameworkCore;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaAccesoDatos.Repositorios

{

public class RepositorioEvento : IRepositorioEvento

{

private readonly JuegosOlimpicosDBContext \_dbContext;

public RepositorioEvento(JuegosOlimpicosDBContext dbContext)

{

\_dbContext = dbContext;

}

public void Add(Evento item)

{

\_dbContext.Eventos.Add(item);

\_dbContext.SaveChanges();

}

public void Delete(Evento item)

{

throw new NotImplementedException();

}

public List<Evento> GetAll()

{

throw new NotImplementedException();

}

public Evento GetById(int id) =>

\_dbContext.Eventos

.Include(e => e.LiPuntajes)

.ThenInclude(p => p.Atleta)

.ThenInclude(a => a.Pais)

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

public Evento GetByNombre(string nombre) => \_dbContext.Eventos.SingleOrDefault(e => e.NombrePrueba == nombre);

public List<Evento> GetByFecha(DateTime fecha)

{

return \_dbContext.Eventos.Where(e => e.FchInicio <= fecha && e.FchFin >= fecha).ToList();

}

public void Update(Evento item)

{

\_dbContext.Eventos.Update(item);

\_dbContext.SaveChanges();

}

public List<Evento> GetEventosAtleta(int atletaId) =>

\_dbContext.Eventos

.Include(e => e.Disciplina)

.Where(e => e.LiPuntajes.Any(p => p.AtletaId == atletaId))

.OrderBy(e => e.Disciplina.Nombre.Valor)

.ToList();

public List<Evento> SelectByBusqueda

(int? disciplinaId, DateTime? fchInicio, DateTime? fchFin, string? nombreEvento, int? puntajeMin, int? puntajeMax)

{

IQueryable<Evento> eventos = \_dbContext.Eventos.AsQueryable();

if (disciplinaId.HasValue) eventos = eventos.Where(e => e.DisciplinaId == disciplinaId);

if (fchInicio.HasValue)

{

if (fchInicio == fchFin) eventos = eventos.Where(e => e.FchInicio == fchInicio || e.FchFin == fchInicio);

else eventos = eventos.Where(e => e.FchInicio >= fchInicio && e.FchFin <= fchFin);

}

if (!string.IsNullOrEmpty(nombreEvento)) eventos = eventos.Where(e => e.NombrePrueba.Contains(nombreEvento));

if (puntajeMin.HasValue)

eventos = eventos.Where(e => e.LiPuntajes.Any(pea => pea.Puntaje >= puntajeMin && pea.Puntaje <= puntajeMax));

return eventos.ToList();

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: RepositorioPais.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaAccesoDatos\Repositorios\RepositorioPais.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using LogicaNegocio.Entidades;

using LogicaNegocio.IRepositorios;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaAccesoDatos.Repositorios

{

public class RepositorioPais : IRepositorioPais

{

private readonly JuegosOlimpicosDBContext \_context;

public RepositorioPais(JuegosOlimpicosDBContext context)

{

\_context = context;

}

public void Add(Pais item)

{

throw new NotImplementedException();

}

public void Delete(Pais item)

{

throw new NotImplementedException();

}

public List<Pais> GetAll()

{

throw new NotImplementedException();

}

public Pais GetById(int id)

{

throw new NotImplementedException();

}

public void Update(Pais item)

{

throw new NotImplementedException();

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: RepositorioUsuario.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaAccesoDatos\Repositorios\RepositorioUsuario.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using LogicaNegocio.Entidades;

using LogicaNegocio.ExcepcionesEntidades;

using LogicaNegocio.IRepositorios;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaAccesoDatos.Repositorios

{

public class RepositorioUsuario : IRepositorioUsuario

{

private JuegosOlimpicosDBContext \_context;

public RepositorioUsuario(JuegosOlimpicosDBContext context)

{

\_context = context;

}

public void Add(Usuario item)

{

if (item == null)

{

throw new UsuarioException("El usuario no puede ser vacío");

}

\_context.Add(item);

\_context.SaveChanges();

}

public void Delete(Usuario item)

{

\_context.Remove(item);

\_context.SaveChanges();

}

public List<Usuario> GetAll()

{

return \_context.Usuarios.ToList();

}

public Usuario? GetById(int id) =>

\_context.Usuarios.SingleOrDefault(usuario => usuario.Id == id);

public Usuario? GetByEmail(string email) =>

\_context.Usuarios.SingleOrDefault(usuario => usuario.Email.Valor == email);

public void Update(Usuario item)

{

\_context.Usuarios.Update(item);

\_context.SaveChanges();

}

public Usuario? LoginUsuario(string email, string password)

{

List<Usuario> preRes = \_context.Usuarios

.Where(usuario => usuario.Email.Valor == email && usuario.Contrasena.Valor == password)

.ToList();

return preRes.FirstOrDefault(u => u.Email.Valor == email && u.Contrasena.Valor == password);

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: ValidarDisciplina.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaAplicacion\Validadores\ValidarDisciplina.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using LogicaNegocio.ExcepcionesEntidades;

using LogicaNegocio.ValueObjects.Disciplina;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaAplicacion.Validadores

{

public static class ValidarDisciplina

{

public static void Nombre(string nombre)

{

if (nombre.Length < 10) throw new DisciplinaException("El nombre de la disciplina debe contener al menos 10 caracteres");

if (nombre.Length > 50) throw new DisciplinaException("El nombre de la disciplina puede contener hasta 50 caracteres");

}

public static void Anio(int anio)

{

if (anio > DateTime.Now.Year) throw new DisciplinaException("El año no puede ser posterior al actual");

if(anio <= 0) throw new DisciplinaException("Año incorrecto");

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: ValidarEvento.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaAplicacion\Validadores\ValidarEvento.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using Compartido.DTOs.Eventos;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using LogicaNegocio.ExcepcionesEntidades;

using LogicaNegocio.Entidades;

namespace LogicaAplicacion.Validadores

{

public static class ValidarEvento

{

public static void CantidadAtletas(EventoInsertDTO dto)

{

if (dto.AtletasId != null ? dto.AtletasId.Count() < 3 : true)

throw new EventoException("Se requieren al menos 3 Atletas");

}

public static void AtletasRegistradosEnDisciplina(List<Atleta> atletas, int[] dtoIds)

{

int verificador = atletas.Where(a => dtoIds.Any(id => id == a.Id)).Count();

if (verificador != dtoIds.Length) throw new EventoException("Hay atletas no registrados en esta Disciplina");

}

public static void Fechas(EventoInsertDTO dto)

{

if (dto.FchInicio > dto.FchFin)

throw new EventoException("Fechas incorrectas");

}

public static void Puntajes(EventoUpdatePuntajesDTO eventoUpdatePuntajesDTO)

{

if (eventoUpdatePuntajesDTO.LiAtletas.Any(p => p.Puntaje < 1))

throw new EventoException("Los puntajes no pueden ser menores a 1");

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: ValidarUsuario.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaAplicacion\Validadores\ValidarUsuario.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using Compartido.DTOs.Usuarios;

using LogicaNegocio.Entidades;

using LogicaNegocio.Enums;

using LogicaNegocio.ExcepcionesEntidades;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Text.RegularExpressions;

using System.Threading.Tasks;

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

namespace LogicaAplicacion.Validadores

{

public static class ValidarUsuario

{

public static void Email(string email)

{

if (!Regex.IsMatch(email, @"^[a-zA-Z0-9]+@[a-zA-Z0-9]+\.[a-zA-Z0-9]+$"))

{

throw new UsuarioException("El formato del mail no es correcto");

}

if (email.Length > 255)

{

throw new UsuarioException("El email puede contener hasta 255 caracteres");

}

}

public static void Contrasena(string contrasena)

{

if (!Regex.IsMatch(contrasena, @"^(?=.\*[a-z])(?=.\*[A-Z])(?=.\*[0-9])(?=.\*[\.;,!]).{6,}$"))

{

throw new UsuarioException("La contraseña debe contener al menos 6 caracteres y al menos una mayúscula, minúscula, dígito y [. ; , !]");

}

}

public static void Rol(int rolId)

{

if (rolId < 0 || rolId > 1)

{

throw new UsuarioException("El rol solamente puede ser como Administrador o Digitador");

}

}

public static void Nombre(string nombre)

{

if (nombre != null && nombre.Length > 50)

{

throw new UsuarioException("El Nombre puede contener hasta 50 caracteres");

}

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: Atleta.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaNegocio\Entidades\Atleta.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using LogicaNegocio.Enums;

using LogicaNegocio.IEntidades;

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations.Schema;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaNegocio.Entidades

{

public class Atleta : IEntity

{

public int Id { get; set; }

public string Nombre { get; set; }

public string Apellido { get; set; }

public Sexo Sexo { get; set; }

[ForeignKey("Pais")]

public int PaisId { get; set; }

public Pais Pais { get; set; }

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

public Atleta() { }

public Atleta(int id, string nombre, string apellido, Sexo sexo, Pais pais)

{

Id = id;

Nombre = nombre;

Apellido = apellido;

Sexo = sexo;

Pais = pais;

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: Auditoria.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaNegocio\Entidades\Auditoria.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using LogicaNegocio.Enums;

using LogicaNegocio.IEntidades;

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations.Schema;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaNegocio.Entidades

{

[Table("Auditorias")]

public class Auditoria : IEntity

{

public int Id { get ; set; }

public DateTime Fecha { get; set; }

public Accion Accion { get; set; }

public string Entidad { get; set; }

public int EntidadId { get; set; }

public string EmailUsuario { get; set; }

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: Disciplina.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaNegocio\Entidades\Disciplina.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using LogicaNegocio.IEntidades;

using LogicaNegocio.ValueObjects.Disciplina;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaNegocio.Entidades

{

public class Disciplina : IEntity

{

public int Id { get; set; }

public RDisciplinaNombre Nombre { get; set; }

public int AnioIntegracion { get; set; }

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

public Disciplina() { }

public Disciplina(int id, string nombre, int anioIntegracion)

{

Id = id;

Nombre = new RDisciplinaNombre(nombre);

AnioIntegracion = anioIntegracion;

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: Evento.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaNegocio\Entidades\Evento.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using LogicaNegocio.IEntidades;

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations.Schema;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaNegocio.Entidades

{

public class Evento : IEntity

{

public int Id { get; set; }

public string NombrePrueba { get; set; }

public int DisciplinaId { get; set; }

public DateTime FchInicio { get; set; }

public DateTime FchFin { get; set; }

public List<PuntajeEventoAtleta> LiPuntajes { get; set; } = new List<PuntajeEventoAtleta>();

[ForeignKey("DisciplinaId")]

public Disciplina Disciplina { get; set; }

public Evento() { }

public Evento(int id, Disciplina disciplina, string nombrePrueba, DateTime fchInicio, DateTime fchFin)

{

Id = id;

Disciplina = disciplina;

NombrePrueba = nombrePrueba;

FchInicio = fchInicio;

FchFin = fchFin;

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: Pais.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaNegocio\Entidades\Pais.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using LogicaNegocio.IEntidades;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaNegocio.Entidades

{

public class Pais : IEntity

{

public int Id { get; set; }

public string Nombre { get; set; }

public int Habitantes { get; set; }

public string NombreDelegado { get; set; }

public string TelDelegado { get; set; }

public Pais() { }

Pais(int id, string nombre, int habitantes, string nombreDelegado, string telDelegado)

{

Id = id;

Nombre = nombre;

Habitantes = habitantes;

NombreDelegado = nombreDelegado;

TelDelegado = telDelegado;

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: PuntajeEventoAtleta.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaNegocio\Entidades\PuntajeEventoAtleta.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using Microsoft.EntityFrameworkCore;

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations.Schema;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaNegocio.Entidades

{

[PrimaryKey(nameof(AtletaId), nameof(EventoId))]

public class PuntajeEventoAtleta

{

public decimal Puntaje { get; set; }

public int AtletaId { get; set; }

public int EventoId { get; set; }

[ForeignKey("AtletaId")]

public Atleta Atleta { get; set; }

public PuntajeEventoAtleta() { }

public PuntajeEventoAtleta(Atleta atleta, decimal puntaje)

{

Atleta = atleta;

Puntaje = puntaje;

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: Usuario.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaNegocio\Entidades\Usuario.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using LogicaNegocio.Enums;

using LogicaNegocio.IEntidades;

using LogicaNegocio.ValueObjects.Usuario;

using Microsoft.EntityFrameworkCore;

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaNegocio.Entidades

{

public class Usuario : IEntity

{

public int Id { get; set; }

public RUsuarioEmail Email { get; set; }

public RUsuarioContrasena Contrasena { get; set; }

public Rol RolUsuario { get; set; }

public DateTime FechaRegistro { get; private set; } = DateTime.Now;

public int IdAdminRegistro { get; set; }

[MaxLength(50)]

public string? Nombre { get; set; }

public Usuario() { }

public Usuario(string email, string contrasena, Rol rolUsuario, int idAdminRegistro)

{

Email = new RUsuarioEmail(email);

Contrasena = new RUsuarioContrasena(contrasena);

RolUsuario = rolUsuario;

IdAdminRegistro = idAdminRegistro;

}

public Usuario(string email, string contrasena, Rol rolUsuario, int idAdminRegistro, string nombre)

{

Email = new RUsuarioEmail(email);

Contrasena = new RUsuarioContrasena(contrasena);

RolUsuario = rolUsuario;

IdAdminRegistro = idAdminRegistro;

Nombre = nombre;

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: Accion.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaNegocio\Enums\Accion.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaNegocio.Enums

{

public enum Accion

{

Create = 1,

Update = 2,

Delete = 3

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: Rol.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaNegocio\Enums\Rol.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaNegocio.Enums

{

public enum Rol

{

Administrador = 0,

Digitador = 1

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: Sexo.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaNegocio\Enums\Sexo.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaNegocio.Enums

{

public enum Sexo

{

Masculino = 0,

Femenino = 1

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: AtletaException.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaNegocio\ExcepcionesEntidades\AtletaException.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaNegocio.ExcepcionesEntidades

{

public class AtletaException : Exception

{

public AtletaException() { }

public AtletaException(string message) : base(message) { }

public AtletaException(string message, Exception innerException) : base(message, innerException) { }

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: AuditoriaException.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaNegocio\ExcepcionesEntidades\AuditoriaException.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaNegocio.ExcepcionesEntidades

{

public class AuditoriaException : Exception

{

public AuditoriaException() { }

public AuditoriaException(string message) : base(message) { }

public AuditoriaException(string message, Exception innerException) : base(message, innerException) { }

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: ConflictException.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaNegocio\ExcepcionesEntidades\ConflictException.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaNegocio.ExcepcionesEntidades

{

public class ConflictException:Exception

{

public ConflictException() { }

public ConflictException(string message) : base(message) { }

public ConflictException(string message, Exception innerException) : base(message, innerException) { }

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: DisciplinaException.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaNegocio\ExcepcionesEntidades\DisciplinaException.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaNegocio.ExcepcionesEntidades

{

public class DisciplinaException : Exception

{

public DisciplinaException() { }

public DisciplinaException(string message) : base(message) { }

public DisciplinaException(string message, Exception innerException) : base(message, innerException) { }

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: EventoException.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaNegocio\ExcepcionesEntidades\EventoException.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaNegocio.ExcepcionesEntidades

{

public class EventoException : Exception

{

public EventoException() { }

public EventoException(string message) : base(message) { }

public EventoException(string message, Exception innerException) : base(message, innerException) { }

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: PaisException.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaNegocio\ExcepcionesEntidades\PaisException.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaNegocio.ExcepcionesEntidades

{

public class PaisException : Exception

{

public PaisException() { }

public PaisException(string message) : base(message) { }

public PaisException(string message, Exception innerException) : base(message, innerException) { }

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: UsuarioException.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaNegocio\ExcepcionesEntidades\UsuarioException.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaNegocio.ExcepcionesEntidades

{

public class UsuarioException : Exception

{

public UsuarioException() { }

public UsuarioException(string message) : base(message) { }

public UsuarioException(string message, Exception innerException) : base(message, innerException) { }

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: IEntity.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaNegocio\IEntidades\IEntity.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaNegocio.IEntidades

{

public interface IEntity

{

int Id { get; set; }

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: IRepositorio.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaNegocio\IRepositorios\IRepositorio.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaNegocio.IRepositorios

{

public interface IRepositorio<T>

{

List<T> GetAll();

T? GetById(int id);

void Add(T item);

void Update(T item);

void Delete(T item);

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: IRepositorioAtleta.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaNegocio\IRepositorios\IRepositorioAtleta.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using LogicaNegocio.Entidades;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaNegocio.IRepositorios

{

public interface IRepositorioAtleta : IRepositorio<Atleta>

{

public void GuardarCambios();

List<Atleta>? SelectByDisciplinaId(int disciplinaId);

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: IRepositorioAuditoria.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaNegocio\IRepositorios\IRepositorioAuditoria.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using LogicaNegocio.Entidades;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaNegocio.IRepositorios

{

public interface IRepositorioAuditoria: IRepositorio<Auditoria>

{

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: IRepositorioDisciplina.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaNegocio\IRepositorios\IRepositorioDisciplina.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using LogicaNegocio.Entidades;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaNegocio.IRepositorios

{

public interface IRepositorioDisciplina : IRepositorio<Disciplina>

{

Disciplina Insert(Disciplina item);

Disciplina? GetByNombre(string nombre);

List<Atleta> GetAtletasDisciplina(int idDisciplina);

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: IRepositorioEvento.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaNegocio\IRepositorios\IRepositorioEvento.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using LogicaNegocio.Entidades;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaNegocio.IRepositorios

{

public interface IRepositorioEvento : IRepositorio<Evento>

{

Evento GetByNombre(string nombre);

List<Evento> GetByFecha (DateTime fecha);

List<Evento> GetEventosAtleta(int atletaId);

List<Evento> SelectByBusqueda

(int? disciplinaId, DateTime? fchInicio, DateTime? fchFin, string? nombreEvento, int? puntajeMin, int? puntajeMax);

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: IRepositorioPais.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaNegocio\IRepositorios\IRepositorioPais.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using LogicaNegocio.Entidades;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaNegocio.IRepositorios

{

public interface IRepositorioPais : IRepositorio<Pais>

{

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: IRepositorioUsuario.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaNegocio\IRepositorios\IRepositorioUsuario.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using LogicaNegocio.Entidades;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaNegocio.IRepositorios

{

public interface IRepositorioUsuario : IRepositorio<Usuario>

{

Usuario? LoginUsuario(string email, string password);

Usuario? GetByEmail(string email);

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: AtletaController.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\WebApi\Controllers\AtletaController.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using Compartido.DTOs.Atletas;

using Compartido.DTOs.Eventos;

using LogicaAplicacion.ICasosDeUso.Atletas;

using LogicaNegocio.ExcepcionesEntidades;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

namespace WebApi.Controllers

{

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

[ApiController]

public class AtletaController : ControllerBase

{

private readonly ISelectByDisciplinaId \_selectByDisciplinaId;

public AtletaController(ISelectByDisciplinaId selectByDisciplinaId)

{

\_selectByDisciplinaId = selectByDisciplinaId;

}

/// <summary>

/// Permite obtener el listado de atletas de una disciplina

/// </summary>

/// <param name="disciplinaId"></param>

/// <returns></returns>

[ProducesResponseType(StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status400BadRequest)]

[ProducesResponseType(StatusCodes.Status404NotFound)]

[ProducesResponseType(StatusCodes.Status500InternalServerError)]

[HttpGet("{disciplinaId}")]

public IActionResult GetByDisciplina(int disciplinaId)

{

try

{

IEnumerable<AtletaListaDTO> res = \_selectByDisciplinaId.Ejecutar(disciplinaId);

return res == null

? NotFound("No existe disciplina con esa id")

: Ok(res);

}

catch (AtletaException aex)

{

return BadRequest(aex.Message);

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

return StatusCode(500, "Algo no salió correctamente");

}

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: DisciplinaController.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\WebApi\Controllers\DisciplinaController.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using Compartido.DTOs.Disciplinas;

using Compartido.DTOs.Auditorias;

using LogicaAplicacion.ICasosDeUso.Atletas;

using LogicaAplicacion.ICasosDeUso.Auditorias;

using LogicaAplicacion.ICasosDeUso.Disciplinas;

using LogicaNegocio.ExcepcionesEntidades;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using LogicaNegocio.Enums;

using LogicaNegocio.Entidades;

using System.Security.Claims;

namespace WebApi.Controllers

{

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

[ApiController]

public class DisciplinaController : ControllerBase

{

private readonly IFindAllDisciplinas \_finAllDisciplinas;

private readonly IInsertDisciplina \_altaDisciplina;

private readonly IDeleteDisciplina \_deleteDisciplina;

private readonly IDisciplinaUpdate \_disciplinaUpdate;

private readonly IDisciplinaSelectById \_disciplinaSelectById;

private readonly IAuditoriaInsert \_auditoriaInsert;

private readonly IDisciplinaSelectByNombre \_disciplinaSelectByNombre;

public DisciplinaController(

IFindAllDisciplinas findAllDisciplinas,

IInsertDisciplina altaDisciplina,

IDeleteDisciplina deleteDisciplina,

IDisciplinaSelectById disciplinaSelectById,

IAuditoriaInsert auditoriaInsert,

IDisciplinaSelectByNombre disciplinaSelectByNombre,

IDisciplinaUpdate disciplinaUpdate)

{

\_finAllDisciplinas = findAllDisciplinas;

\_altaDisciplina = altaDisciplina;

\_deleteDisciplina = deleteDisciplina;

\_disciplinaSelectById = disciplinaSelectById;

\_auditoriaInsert = auditoriaInsert;

\_disciplinaSelectByNombre = disciplinaSelectByNombre;

\_disciplinaUpdate = disciplinaUpdate;

}

/// <summary>

/// Permite obtener una disciplina por ID

/// </summary>

/// <param name="id"></param>

/// <returns></returns>

[ProducesResponseType(StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status400BadRequest)]

[ProducesResponseType(StatusCodes.Status404NotFound)]

[ProducesResponseType(StatusCodes.Status500InternalServerError)]

[Authorize(Roles = "Digitador")]

[HttpGet("{id}")]

public IActionResult Get(int id)

{

try

{

DisciplinaDTO? res = \_disciplinaSelectById.Ejecutar(id);

return res == null ? NotFound("No se encontró disciplina con ese Id") : Ok(res);

}

catch (DisciplinaException dex)

{

return BadRequest(dex.Message);

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

return StatusCode(500, "Algo no salió correctamente");

}

}

/// <summary>

/// Permite obtener una disciplina por nombre

/// </summary>

/// <param name="nombre"></param>

/// <returns></returns>

[ProducesResponseType(StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status400BadRequest)]

[ProducesResponseType(StatusCodes.Status404NotFound)]

[ProducesResponseType(StatusCodes.Status500InternalServerError)]

[Authorize(Roles = "Digitador")]

[HttpGet("Nombre/{nombre}")]

public IActionResult GetByNombre(string nombre)

{

try

{

DisciplinaDTO? res = \_disciplinaSelectByNombre.Ejecutar(nombre);

return res == null ? NotFound("No se encontró disciplina con ese nombre") : Ok(res);

}

catch (DisciplinaException dex)

{

return BadRequest(dex.Message);

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

return StatusCode(500, "Algo no salió correctamente");

}

}

/// <summary>

/// Permite dar de alta una disciplina

/// </summary>

/// <param name="disciplinaInsertDto"></param>

/// <returns></returns>

[ProducesResponseType(StatusCodes.Status201Created)]

[ProducesResponseType(StatusCodes.Status400BadRequest)]

[ProducesResponseType(StatusCodes.Status409Conflict)]

[ProducesResponseType(StatusCodes.Status500InternalServerError)]

[Authorize(Roles = "Digitador")]

[HttpPost]

public IActionResult Post([FromBody] DisciplinaInsertDTO disciplinaInsertDto)

{

try

{

int idDisciplina = \_altaDisciplina.Ejecutar(disciplinaInsertDto);

\_auditoriaInsert.Ejecutar(new AuditoriaInsertDTO

{

Accion = Accion.Create,

EmailUsuario = User.FindFirst(ClaimTypes.Email)?.Value,

Entidad = "Disciplina",

EntidadId = idDisciplina

});

return Created();

}

catch (ConflictException cex)

{

return Conflict(cex.Message);

}

catch (DisciplinaException dex)

{

return BadRequest(dex.Message);

}

catch (AuditoriaException aex)

{

// Esto es un tema interno de servidor, si llego hasta \_auditoriaInsert.Ejecutar ya hizo la inserción

Console.WriteLine(aex.Message);

return Created();

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

return StatusCode(500, "Algo no salió correctamente");

}

}

/// <summary>

/// Permite modificar una disciplina

/// </summary>

/// <param name="id"></param>

/// <param name="disciplinaUpdateDTO"></param>

/// <returns></returns>

[ProducesResponseType(StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status400BadRequest)]

[ProducesResponseType(StatusCodes.Status409Conflict)]

[ProducesResponseType(StatusCodes.Status500InternalServerError)]

[Authorize(Roles = "Digitador")]

[HttpPut("{id}")]

public IActionResult Put(int id, [FromBody] DisciplinaUpdateDTO disciplinaUpdateDTO)

{

DisciplinaDTO disciplinaDTO = new DisciplinaDTO();

try

{

disciplinaDTO = \_disciplinaUpdate.Ejecutar(id, disciplinaUpdateDTO);

\_auditoriaInsert.Ejecutar(new AuditoriaInsertDTO

{

Accion = Accion.Update,

EmailUsuario = User.FindFirst(ClaimTypes.Email)?.Value,

Entidad = "Disciplina",

EntidadId = id

});

return Ok(disciplinaDTO);

}

catch (ConflictException cex)

{

return Conflict(cex.Message);

}

catch (DisciplinaException dex)

{

return BadRequest(dex.Message);

}

catch (AuditoriaException aex)

{

Console.WriteLine(aex.Message);

return Ok(disciplinaDTO);

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

return StatusCode(500, "Algo no salió correctamente");

}

}

/// <summary>

/// Permite eliminar una disciplina

/// </summary>

/// <param name="id"></param>

/// <returns></returns>

[ProducesResponseType(StatusCodes.Status204NoContent)]

[ProducesResponseType(StatusCodes.Status400BadRequest)]

[ProducesResponseType(StatusCodes.Status409Conflict)]

[ProducesResponseType(StatusCodes.Status500InternalServerError)]

[Authorize(Roles = "Digitador")]

[HttpDelete("{id}")]

public IActionResult Delete(int id)

{

try

{

\_deleteDisciplina.Ejecutar(id);

\_auditoriaInsert.Ejecutar(new AuditoriaInsertDTO

{

Accion = Accion.Delete,

EmailUsuario = User.FindFirst(ClaimTypes.Email)?.Value,

Entidad = "Disciplina",

EntidadId = id

});

return NoContent();

}

catch (ConflictException cex)

{

return Conflict(cex.Message);

}

catch (DisciplinaException dex)

{

return BadRequest(dex.Message);

}

catch (AuditoriaException aex)

{

Console.WriteLine(aex.Message);

return NoContent();

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

return StatusCode(500, "Algo no salió correctamente");

}

}

/// <summary>

/// Permite obtener todas las disciplinas

/// </summary>

/// <returns></returns>

[ProducesResponseType(StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status500InternalServerError)]

//Ver en este el tema de la autenticación, es utilizado para obtener el listado para el anterior que es sin autenticación

[HttpGet]

public IActionResult GetAll()

{

try

{

IEnumerable<DisciplinaDTO> disciplinas = \_finAllDisciplinas.Ejecutar();

return Ok(disciplinas);

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

return StatusCode(500, "Algo no salió correctamente");

}

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: EventoController.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\WebApi\Controllers\EventoController.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using Compartido.DTOs.Eventos;

using LogicaAplicacion.ICasosDeUso.Eventos;

using LogicaNegocio.ExcepcionesEntidades;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

namespace WebApi.Controllers

{

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

[ApiController]

public class EventoController : ControllerBase

{

private readonly ISelectByBusqueda \_selectByBusqueda;

public EventoController(ISelectByBusqueda selectByBusqueda)

{

\_selectByBusqueda = selectByBusqueda;

}

/// <summary>

/// Permite obtener el listado de eventos de acuerdo a los filtros cargados.

/// </summary>

/// <param name="dto"></param>

/// <returns></returns>

[ProducesResponseType(StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status400BadRequest)]

[ProducesResponseType(StatusCodes.Status500InternalServerError)]

[HttpGet]

[Authorize(Roles = "Digitador")]

public IActionResult GetByBusqueda([FromQuery] EventoBusquedaDTO dto)

{

try

{

return Ok(\_selectByBusqueda.Ejecutar(dto));

}

catch (EventoException eex)

{

return BadRequest(eex.Message);

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

return StatusCode(500, "Algo no salió correctamente");

}

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: UsuarioController.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\WebApi\Controllers\UsuarioController.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using Compartido.DTOs.Usuarios;

using LogicaAplicacion.ICasosDeUso.Usuarios;

using LogicaNegocio.ExcepcionesEntidades;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using System.Security.Claims;

using WebApi.Tokens;

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

namespace WebApi.Controllers

{

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

[ApiController]

public class UsuarioController : ControllerBase

{

private readonly ILoginUsuario \_login;

private readonly IGetByIdUsuario \_getById;

private readonly IFindAllUsuarios \_getAll;

private readonly IAltaUsuario \_alta;

private readonly IUpdateUsuario \_update;

private readonly IDeleteUsuario \_delete;

private readonly IConfiguration \_config;

public UsuarioController(

ILoginUsuario login,

IAltaUsuario alta,

IUpdateUsuario update,

IGetByIdUsuario getById,

IFindAllUsuarios getAll,

IDeleteUsuario delete,

IConfiguration config)

{

\_login = login;

\_alta = alta;

\_update = update;

\_getById = getById;

\_getAll = getAll;

\_delete = delete;

\_config = config;

}

/// <summary>

/// Permite realizar el login a la página

/// </summary>

/// <param name="dto"></param>

/// <returns></returns>

[ProducesResponseType(StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status400BadRequest)]

[ProducesResponseType(StatusCodes.Status500InternalServerError)]

[HttpPost("iniciar-sesion")]

public IActionResult Login([FromBody] CredencialesDTO dto)

{

Token token = new Token(\_config);

try

{

UsuarioDTO usuario = \_login.Ejecutar(dto.Email, dto.Contrasena);

UsuarioLogueadoDTO usuarioRes = new UsuarioLogueadoDTO

{

Id = usuario.Id,

Rol = usuario.RolUsuario,

Token = token.Crear(usuario)

};

return Ok(usuarioRes);

}

catch (UsuarioException uex)

{

return BadRequest(uex.Message);

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

return StatusCode(500, ex.Message);

}

}

/// <summary>

/// Permite obtener el listado de usuarios

/// </summary>

/// <returns></returns>

// GET: api/<UsuarioController>

[HttpGet]

public IActionResult Get()

{

try

{

return Ok(\_getAll.Ejecutar());

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

return StatusCode(500, "Algo no salió correctamente");

}

}

/// <summary>

/// Permite obtener un usuario por id

/// </summary>

/// <param name="id"></param>

/// <returns></returns>

[ProducesResponseType(StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status400BadRequest)]

[ProducesResponseType(StatusCodes.Status500InternalServerError)]

[Authorize(Roles = "Administrador,Digitador")]

// GET api/<UsuarioController>/5

[HttpGet("{id}")]

public IActionResult Get(int id)

{

try

{

string idUsuario = User.FindFirst(ClaimTypes.NameIdentifier)?.Value;

string rol = User.FindFirst(ClaimTypes.Role)?.Value;

if (rol == "Digitador")

if (idUsuario != id.ToString()) return Unauthorized("No tiene permisos para acceder a estos recursos");

return Ok(\_getById.Ejecutar(id));

}

catch (UsuarioException uex)

{

return BadRequest(uex);

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

return StatusCode(500, "Algo no salió correctamente");

}

}

/// <summary>

/// Permite realizar el alta de un usuario

/// </summary>

/// <param name="usuarioInsert"></param>

/// <returns></returns>

[ProducesResponseType(StatusCodes.Status201Created)]

[ProducesResponseType(StatusCodes.Status400BadRequest)]

[ProducesResponseType(StatusCodes.Status409Conflict)]

[ProducesResponseType(StatusCodes.Status500InternalServerError)]

[Authorize(Roles = "Administrador")]

// POST api/<UsuarioController>

[HttpPost()]

public IActionResult Post([FromBody] UsuarioInsertDTO usuarioInsert)

{

try

{

\_alta.Ejecutar(usuarioInsert);

return Created();

}

catch (ConflictException cex)

{

return Conflict(cex.Message);

}

catch (UsuarioException uex)

{

return BadRequest(uex.Message);

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

return StatusCode(500, "Algo no salió correctamente");

}

}

/// <summary>

/// Permite modificar los datos de un usuario

/// </summary>

/// <param name="id"></param>

/// <param name="dtoUpdate"></param>

/// <returns></returns>

[ProducesResponseType(StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status400BadRequest)]

[ProducesResponseType(StatusCodes.Status409Conflict)]

[ProducesResponseType(StatusCodes.Status500InternalServerError)]

[Authorize(Roles = "Administrador,Digitador")]

[HttpPut("{id}")]

public IActionResult Put(int id, [FromBody] UsuarioUpdateDTO dtoUpdate)

{

try

{

string idUsuario = User.FindFirst(ClaimTypes.NameIdentifier)?.Value;

string rol = User.FindFirst(ClaimTypes.Role)?.Value;

if (rol == "Digitador")

if (idUsuario != id.ToString()) return Unauthorized("No tiene permisos para acceder a estos recursos");

return Ok(\_update.Ejecutar(id, dtoUpdate));

}

catch (ConflictException cex)

{

return Conflict(cex.Message);

}

catch (UsuarioException uex)

{

return BadRequest(uex);

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

return StatusCode(500, "Algo no salió correctamente");

}

}

/// <summary>

/// Permite modificar la contraseña de un usuario

/// </summary>

/// <param name="id"></param>

/// <param name="dto"></param>

/// <returns></returns>

[ProducesResponseType(StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status400BadRequest)]

[ProducesResponseType(StatusCodes.Status500InternalServerError)]

[Authorize(Roles = "Administrador,Digitador")]

[HttpPatch("{id}")]

public IActionResult Patch(int id, [FromBody] UpdateContrasenaDTO dto)

{

try

{

string idUsuario = User.FindFirst(ClaimTypes.NameIdentifier)?.Value;

string rol = User.FindFirst(ClaimTypes.Role)?.Value;

if (rol == "Digitador")

if (idUsuario != id.ToString()) return Unauthorized("No tiene permisos para acceder a estos recursos");

if (rol == "Digitador")

{

return Ok(\_update.Ejecutar(id, dto.ContrasenaNueva, dto.ContrasenaAnterior));

}

return Ok(\_update.Ejecutar(id, dto.ContrasenaNueva));

}

catch (UsuarioException uex)

{

return BadRequest(uex);

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

return StatusCode(500, "Algo no salió correctamente");

}

}

/// <summary>

/// Permite borrar un usuario

/// </summary>

/// <param name="id"></param>

/// <returns></returns>

[ProducesResponseType(StatusCodes.Status204NoContent)]

[ProducesResponseType(StatusCodes.Status400BadRequest)]

[ProducesResponseType(StatusCodes.Status500InternalServerError)]

[Authorize(Roles = "Administrador")]

[HttpDelete("{id}")]

public IActionResult Delete(int id)

{

try

{

\_delete.Ejecutar(id);

return NoContent();

}

catch (UsuarioException uex)

{

return BadRequest(uex);

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

return StatusCode(500, "Algo no salió correctamente");

}

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: Token.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\WebApi\Tokens\Token.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using Compartido.DTOs.Usuarios;

using Microsoft.IdentityModel.Tokens;

using System.IdentityModel.Tokens.Jwt;

using System.Security.Claims;

using System.Text;

namespace WebApi.Tokens

{

public class Token

{

private readonly string \_secret;

public Token(IConfiguration config)

{

\_secret = config.GetValue<string>("JWT:secret");

}

public string Crear(UsuarioDTO dto)

{

byte[] clave = Encoding.ASCII.GetBytes(\_secret);

JwtSecurityTokenHandler tokenHandler = new JwtSecurityTokenHandler();

//Se incluye un claim para el email

SecurityTokenDescriptor tokenDescriptor = new SecurityTokenDescriptor

{

Subject = new ClaimsIdentity(new Claim[]

{

new Claim(ClaimTypes.Email, dto.Email),

new Claim(ClaimTypes.Role, dto.RolUsuario),

new Claim(ClaimTypes.NameIdentifier, dto.Id.ToString())

}),

Expires = DateTime.UtcNow.AddMinutes(30),

SigningCredentials = new SigningCredentials(new SymmetricSecurityKey(clave),

SecurityAlgorithms.HmacSha256)

};

var token = tokenHandler.CreateToken(tokenDescriptor);

return tokenHandler.WriteToken(token);

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: AtletaDTO.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\Compartido\DTOs\Atletas\AtletaDTO.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using Compartido.DTOs.Disciplinas;

using LogicaNegocio.Entidades;

using LogicaNegocio.Enums;

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations.Schema;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Compartido.DTOs.Atletas

{

public class AtletaDTO

{

public int Id { get; set; }

public string Nombre { get; set; }

public string Apellido { get; set; }

public string Sexo { get; set; }

public string NombrePais { get; set; }

public IEnumerable<DisciplinaDTO> DisciplinasAtleta { get; set; }

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: AtletaInsertDTO.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\Compartido\DTOs\Atletas\AtletaInsertDTO.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Compartido.DTOs.Atletas

{

public class AtletaInsertDTO

{

public string Nombre { get; set; }

public string Apellido { get; set; }

public int Sexo { get; set; }

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: AtletaListaDTO.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\Compartido\DTOs\Atletas\AtletaListaDTO.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Compartido.DTOs.Atletas

{

public class AtletaListaDTO

{

public int Id { get; set; }

public string Nombre { get; set; }

public string Apellido { get; set; }

public string NombreCompleto { get; set; }

public string Sexo { get; set; }

public string NombrePais { get; set; }

}

//RF2 – Listado de Atletas por Disciplina(API Web + HttpClient) – Sin autenticación

//- Crear un listado de atletas filtrado por disciplina.Este endpoint permitirá consultar todos los atletas

//que están registrados en una disciplina dado su Id.El listado se retornará ordenado alfabéticamente

//por nombre completo de atleta.Se incluirán como mínimo su Id (o número, según haya utilizado), su

//nombre completo y el nombre de su país.

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: AuditoriaInsertDTO.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\Compartido\DTOs\Auditorias\AuditoriaInsertDTO.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using LogicaNegocio.Enums;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Compartido.DTOs.Auditorias

{

public class AuditoriaInsertDTO

{

public Accion Accion { get; set; }

public string Entidad { get; set; }

public int EntidadId { get; set; }

public string EmailUsuario { get; set; }

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: DisciplinaDTO.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\Compartido\DTOs\Disciplinas\DisciplinaDTO.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using LogicaNegocio.ValueObjects.Disciplina;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Compartido.DTOs.Disciplinas

{

public class DisciplinaDTO

{

public int Id { get; set; }

public string Nombre { get; set; }

public int AnioIntegracion { get; set; }

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: DisciplinaInsertDTO.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\Compartido\DTOs\Disciplinas\DisciplinaInsertDTO.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Compartido.DTOs.Disciplinas

{

public class DisciplinaInsertDTO

{

public string Nombre { get; set; }

public int AnioIntegracion { get; set; }

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: DisciplinaUpdateDTO.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\Compartido\DTOs\Disciplinas\DisciplinaUpdateDTO.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Compartido.DTOs.Disciplinas

{

public class DisciplinaUpdateDTO

{

public int Id { get; set; }

public string Nombre { get; set; }

public int AnioIntegracion { get; set; }

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: AtletaEventoListaDTO.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\Compartido\DTOs\Eventos\AtletaEventoListaDTO.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using LogicaNegocio.Entidades;

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations.Schema;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Compartido.DTOs.Eventos

{

public class AtletaEventoListaDTO

{

public int Id { get; set; }

public string NombrePrueba { get; set; }

public string NombreDisciplina { get; set; }

public DateTime FchInicio { get; set; }

public DateTime FchFin { get; set; }

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: EventoBusquedaDTO.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\Compartido\DTOs\Eventos\EventoBusquedaDTO.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Compartido.DTOs.Eventos

{

public class EventoBusquedaDTO

{

public int? DisciplinaId { get; set; }

public DateTime? FchInicio { get; set; }

public DateTime? FchFin { get; set; }

public string? NombreEvento { get; set; }

public int? PuntajeMin { get; set; }

public int? PuntajeMax { get; set; }

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: EventoDTO.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\Compartido\DTOs\Eventos\EventoDTO.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using LogicaNegocio.Entidades;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Compartido.DTOs.Eventos

{

public class EventoDTO

{

public int Id { get; set; }

public string NombrePrueba { get; set; }

public DateTime FchInicio { get; set; }

public DateTime FchFin { get; set; }

public IEnumerable<PuntajeEventoAtletaDTO> LiAtletas { get; set; }

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: EventoInsertDTO.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\Compartido\DTOs\Eventos\EventoInsertDTO.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Compartido.DTOs.Eventos

{

public class EventoInsertDTO

{

public int DisciplinaId { get; set; }

public string NombrePrueba { get; set; }

public DateTime FchInicio { get; set; }

public DateTime FchFin { get; set; }

public int[]? AtletasId { get; set; }

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: EventoListaDTO.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\Compartido\DTOs\Eventos\EventoListaDTO.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Compartido.DTOs.Eventos

{

public class EventoListaDTO

{

public int EventoId { get; set; }

public string NombrePrueba { get; set; }

public DateTime FchInicio { get; set; }

public DateTime FchFin { get; set; }

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: EventoUpdatePuntajesDTO.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\Compartido\DTOs\Eventos\EventoUpdatePuntajesDTO.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Compartido.DTOs.Eventos

{

public class EventoUpdatePuntajesDTO

{

public int Id { get; set; }

public IEnumerable<PEAUpdateDTO> LiAtletas { get; set; }

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: PEAUpdateDTO.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\Compartido\DTOs\Eventos\PEAUpdateDTO.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Compartido.DTOs.Eventos

{

public class PEAUpdateDTO

{

public int AtletaId { get; set; }

public decimal Puntaje { get; set; }

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: PuntajeEventoAtletaDTO.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\Compartido\DTOs\Eventos\PuntajeEventoAtletaDTO.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using Compartido.DTOs.Atletas;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Compartido.DTOs.Eventos

{

public class PuntajeEventoAtletaDTO

{

public AtletaDTO Atleta { get; set; }

public decimal Puntaje { get; set; }

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: PaisInsertDTO.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\Compartido\DTOs\Paises\PaisInsertDTO.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Compartido.DTOs.Paises

{

public class PaisInsertDTO

{

public string Nombre { get; set; }

public int Habitantes { get; set; }

public string NombreDelegado { get; set; }

public string TelDelegado { get; set; }

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: CredencialesDTO.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\Compartido\DTOs\Usuarios\CredencialesDTO.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Compartido.DTOs.Usuarios

{

public class CredencialesDTO

{

public string Email { get; set; }

public string Contrasena { get; set; }

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: UpdateContrasenaDTO.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\Compartido\DTOs\Usuarios\UpdateContrasenaDTO.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Compartido.DTOs.Usuarios

{

public class UpdateContrasenaDTO

{

public string ContrasenaNueva { get; set; }

public string ContrasenaAnterior { get; set; }

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: UsuarioDTO.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\Compartido\DTOs\Usuarios\UsuarioDTO.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using LogicaNegocio.Enums;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Compartido.DTOs.Usuarios

{

public class UsuarioDTO

{

public int Id { get; set; }

public string Email { get; set; }

public string RolUsuario { get; set; }

public string? Nombre { get; set; }

public string FechaRegistro { get; set; }

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: UsuarioInsertDTO.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\Compartido\DTOs\Usuarios\UsuarioInsertDTO.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Compartido.DTOs.Usuarios

{

public class UsuarioInsertDTO

{

public string Email { get; set; }

public string Contrasena { get; set; }

public int RolUsuario { get; set; }

public int IdAdminRegistro { get; set; }

public string? Nombre { get; set; }

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: UsuarioLogueadoDTO.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\Compartido\DTOs\Usuarios\UsuarioLogueadoDTO.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Compartido.DTOs.Usuarios

{

public class UsuarioLogueadoDTO

{

public int Id { get; set; }

public string Rol { get; set; }

public string Token { get; set; }

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: UsuarioUpdateDTO.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\Compartido\DTOs\Usuarios\UsuarioUpdateDTO.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Compartido.DTOs.Usuarios

{

public class UsuarioUpdateDTO

{

public string Email { get; set; }

public string Nombre { get; set; }

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: AgregarDisciplina.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaAplicacion\CasosDeUso\Atletas\AgregarDisciplina.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using LogicaAplicacion.ICasosDeUso.Atletas;

using LogicaNegocio.Entidades;

using LogicaNegocio.ExcepcionesEntidades;

using LogicaNegocio.IRepositorios;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaAplicacion.CasosDeUso.Atletas

{

public class AgregarDisciplina : IAgregarDisciplina

{

private readonly IRepositorioAtleta \_repositorioAtleta;

private readonly IRepositorioDisciplina \_repositorioDisciplina;

public AgregarDisciplina(IRepositorioAtleta repositorioAtleta, IRepositorioDisciplina repositorioDisciplina)

{

\_repositorioAtleta = repositorioAtleta;

\_repositorioDisciplina = repositorioDisciplina;

}

public void Ejecutar(int id, int? idDisciplina)

{

if (id == 0)

{

throw new AtletaException("Id de atleta incorrecto");

}

if (idDisciplina == null || idDisciplina == 0)

{

throw new AtletaException("Id de disciplina incorrecto");

}

Atleta atleta = \_repositorioAtleta.GetById(id);

Disciplina tieneDisciplina = atleta.LiDisciplinas.FirstOrDefault(d => d.Id == idDisciplina);

if (tieneDisciplina != null)

{

throw new AtletaException("El atleta ya está registrado en esta disciplina");

};

Disciplina disciplinaAgregar = \_repositorioDisciplina.GetById(idDisciplina.Value);

atleta.LiDisciplinas.Add(disciplinaAgregar);

\_repositorioAtleta.GuardarCambios();

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: AltaAtleta.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaAplicacion\CasosDeUso\Atletas\AltaAtleta.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using Compartido.DTOs.Atletas;

using LogicaAccesoDatos;

using LogicaAplicacion.ICasosDeUso.Atletas;

using LogicaNegocio.IRepositorios;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaAplicacion.CasosDeUso.Atletas

{

public class AltaAtleta : IAltaAtleta

{

private readonly IRepositorioAtleta \_repositorioAtleta;

public AltaAtleta(IRepositorioAtleta repositorioAtleta)

{

\_repositorioAtleta = repositorioAtleta;

}

public void Ejecutar(AtletaInsertDTO atletaInsertDTO)

{

throw new NotImplementedException();

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: EventosAtleta.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaAplicacion\CasosDeUso\Atletas\EventosAtleta.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using Compartido.DTOs.Eventos;

using Compartido.Mappers;

using LogicaAplicacion.ICasosDeUso.Atletas;

using LogicaNegocio.IRepositorios;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaAplicacion.CasosDeUso.Atletas

{

public class EventosAtleta : IEventosAtleta

{

private readonly IRepositorioEvento \_repositorioEvento;

public EventosAtleta(IRepositorioEvento repositorioEvento)

{

\_repositorioEvento = repositorioEvento;

}

public IEnumerable<AtletaEventoListaDTO> Ejecutar(int atletaId)

=> EventoMapper.EventoToListaAtletaEventoDTO(\_repositorioEvento.GetEventosAtleta(atletaId));

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: FindAllAtletas.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaAplicacion\CasosDeUso\Atletas\FindAllAtletas.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using Compartido.DTOs.Atletas;

using Compartido.Mappers;

using LogicaAplicacion.ICasosDeUso.Atletas;

using LogicaNegocio.IRepositorios;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaAplicacion.CasosDeUso.Atletas

{

public class FindAllAtletas : IFindAllAtletas

{

private readonly IRepositorioAtleta \_repositorioAtleta;

public FindAllAtletas(IRepositorioAtleta repositorioAtleta)

{

\_repositorioAtleta = repositorioAtleta;

}

public IEnumerable<AtletaListaDTO> Ejectuar()

{

return AtletaMapper.AtletasToListaDTO(\_repositorioAtleta.GetAll());

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: GetByIdAtleta.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaAplicacion\CasosDeUso\Atletas\GetByIdAtleta.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using Compartido.DTOs.Atletas;

using Compartido.Mappers;

using LogicaAplicacion.ICasosDeUso.Atletas;

using LogicaNegocio.Entidades;

using LogicaNegocio.IRepositorios;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaAplicacion.CasosDeUso.Atletas

{

public class GetByIdAtleta : IGetByIdAtleta

{

private readonly IRepositorioAtleta \_repositorioAtleta;

public GetByIdAtleta(IRepositorioAtleta repositorioAtleta)

{

\_repositorioAtleta = repositorioAtleta;

}

public AtletaDTO Ejecutar(int id) => AtletaMapper.AtletaToDTO(\_repositorioAtleta.GetById(id));

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: SelectByDisciplinaId.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaAplicacion\CasosDeUso\Atletas\SelectByDisciplinaId.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using Compartido.DTOs.Atletas;

using Compartido.Mappers;

using LogicaAplicacion.ICasosDeUso.Atletas;

using LogicaNegocio.Entidades;

using LogicaNegocio.ExcepcionesEntidades;

using LogicaNegocio.IRepositorios;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaAplicacion.CasosDeUso.Atletas

{

public class SelectByDisciplinaId : ISelectByDisciplinaId

{

private readonly IRepositorioAtleta \_repositorioAtleta;

public SelectByDisciplinaId(IRepositorioAtleta repositorioAtleta)

{

\_repositorioAtleta = repositorioAtleta;

}

public IEnumerable<AtletaListaDTO> Ejecutar(int disciplinaId)

{

if (disciplinaId <= 0) throw new AtletaException("Id incorrecto");

List<Atleta>? resdb = \_repositorioAtleta.SelectByDisciplinaId(disciplinaId);

return resdb == null

? null

: AtletaMapper.AtletasToListaDTO(resdb).OrderBy(a => a.NombreCompleto);

}

//RF2 – Listado de Atletas por Disciplina(API Web + HttpClient) – Sin autenticación

//- Crear un listado de atletas filtrado por disciplina.Este endpoint permitirá consultar todos los atletas

//que están registrados en una disciplina dado su Id.El listado se retornará ordenado alfabéticamente

//por nombre completo de atleta.Se incluirán como mínimo su Id (o número, según haya utilizado), su

//nombre completo y el nombre de su país.

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: AuditoriaInsert.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaAplicacion\CasosDeUso\Auditorias\AuditoriaInsert.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using Compartido.DTOs.Auditorias;

using Compartido.Mappers;

using LogicaAplicacion.ICasosDeUso.Auditorias;

using LogicaNegocio.ExcepcionesEntidades;

using LogicaNegocio.IRepositorios;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaAplicacion.CasosDeUso.Auditorias

{

public class AuditoriaInsert:IAuditoriaInsert

{

private readonly IRepositorioAuditoria \_repositorioAuditoria;

public AuditoriaInsert(IRepositorioAuditoria repositorioAuditoria)

{

\_repositorioAuditoria = repositorioAuditoria;

}

public void Ejecutar(AuditoriaInsertDTO dto)

{

if (dto == null) throw new AuditoriaException("Error al registrar la auditoría");

\_repositorioAuditoria.Add(AuditoriaMapper.DtoInsertToAuditoria(dto));

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: DeleteDisciplina.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaAplicacion\CasosDeUso\Disciplinas\DeleteDisciplina.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using LogicaAplicacion.ICasosDeUso.Disciplinas;

using LogicaNegocio.Entidades;

using LogicaNegocio.IRepositorios;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using LogicaNegocio.ExcepcionesEntidades;

namespace LogicaAplicacion.CasosDeUso.Disciplinas

{

public class DeleteDisciplina : IDeleteDisciplina

{

private readonly IRepositorioDisciplina \_repositorioDisciplina;

public DeleteDisciplina(IRepositorioDisciplina repositorioDisciplina)

{

\_repositorioDisciplina = repositorioDisciplina;

}

public void Ejecutar(int id)

{

if (id <= 0) throw new DisciplinaException("Id incorrecto");

Disciplina buscarDisciplina = \_repositorioDisciplina.GetById(id)

?? throw new DisciplinaException($"No se encontró una Disciplina con id: {id}");

if (buscarDisciplina.LiAtletas.Count > 0)

throw new ConflictException("No se puede eliminar una disciplina con atletlas registrados");

\_repositorioDisciplina.Delete(buscarDisciplina);

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: DisciplinaSelectById.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaAplicacion\CasosDeUso\Disciplinas\DisciplinaSelectById.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using Compartido.DTOs.Disciplinas;

using LogicaAplicacion.ICasosDeUso.Disciplinas;

using LogicaNegocio.Entidades;

using LogicaNegocio.ExcepcionesEntidades;

using LogicaNegocio.IRepositorios;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaAplicacion.CasosDeUso.Disciplinas

{

public class DisciplinaSelectById: IDisciplinaSelectById

{

private readonly IRepositorioDisciplina \_repositorioDisciplina;

public DisciplinaSelectById(IRepositorioDisciplina repositorioDisciplina)

{

\_repositorioDisciplina = repositorioDisciplina;

}

public DisciplinaDTO? Ejecutar(int id)

{

if (id <= 0) throw new DisciplinaException("Id Incorrecto");

Disciplina? res = \_repositorioDisciplina.GetById(id);

return res == null ? null : new DisciplinaDTO

{

Id = id,

AnioIntegracion = res.AnioIntegracion,

Nombre = res.Nombre.Valor

};

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: DisciplinaSelectByNombre.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaAplicacion\CasosDeUso\Disciplinas\DisciplinaSelectByNombre.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using Compartido.DTOs.Disciplinas;

using LogicaAplicacion.ICasosDeUso.Disciplinas;

using LogicaNegocio.Entidades;

using LogicaNegocio.ExcepcionesEntidades;

using LogicaNegocio.IRepositorios;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaAplicacion.CasosDeUso.Disciplinas

{

public class DisciplinaSelectByNombre : IDisciplinaSelectByNombre

{

private readonly IRepositorioDisciplina \_repositorioDisciplina;

public DisciplinaSelectByNombre(IRepositorioDisciplina repositorioDisciplina)

{

\_repositorioDisciplina = repositorioDisciplina;

}

public DisciplinaDTO? Ejecutar(string nombre)

{

if (string.IsNullOrEmpty(nombre)) throw new DisciplinaException("Se requieren un Nombre");

Disciplina? res = \_repositorioDisciplina.GetByNombre(nombre);

return res == null ? null : new DisciplinaDTO

{

Id = res.Id,

AnioIntegracion = res.AnioIntegracion,

Nombre = nombre

};

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: DisciplinaUpdate.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaAplicacion\CasosDeUso\Disciplinas\DisciplinaUpdate.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using Compartido.DTOs.Disciplinas;

using Compartido.Mappers;

using LogicaAplicacion.ICasosDeUso.Disciplinas;

using LogicaAplicacion.Validadores;

using LogicaNegocio.Entidades;

using LogicaNegocio.ExcepcionesEntidades;

using LogicaNegocio.IRepositorios;

using LogicaNegocio.ValueObjects.Disciplina;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaAplicacion.CasosDeUso.Disciplinas

{

public class DisciplinaUpdate : IDisciplinaUpdate

{

private readonly IRepositorioDisciplina \_repositorioDisciplina;

public DisciplinaUpdate(IRepositorioDisciplina repositorioDisciplina)

{

\_repositorioDisciplina = repositorioDisciplina;

}

public DisciplinaDTO Ejecutar(int id, DisciplinaUpdateDTO dto)

{

ValidarDisciplina.Nombre(dto.Nombre);

ValidarDisciplina.Anio(dto.AnioIntegracion);

Disciplina Existe = \_repositorioDisciplina.GetById(id)

?? throw new DisciplinaException("Disciplina no encontrada");

Disciplina? ExisteNombre = \_repositorioDisciplina.GetByNombre(dto.Nombre);

if(ExisteNombre != null)

{

if (dto.Nombre == ExisteNombre.Nombre.Valor && id != ExisteNombre.Id)

throw new ConflictException("Ya existe una disciplina con ese nombre");

}

Existe.Nombre = new RDisciplinaNombre( dto.Nombre);

Existe.AnioIntegracion = dto.AnioIntegracion;

\_repositorioDisciplina.Update(Existe);

return DisciplinaMapper.DisciplinaToDTO(Existe);

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: FindAllDisciplinas.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaAplicacion\CasosDeUso\Disciplinas\FindAllDisciplinas.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using Compartido.DTOs.Disciplinas;

using Compartido.Mappers;

using LogicaAccesoDatos;

using LogicaAplicacion.ICasosDeUso.Disciplinas;

using LogicaNegocio.IRepositorios;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaAplicacion.CasosDeUso.Disciplinas

{

public class FindAllDisciplinas : IFindAllDisciplinas

{

private readonly IRepositorioDisciplina \_repositorioDisciplina;

public FindAllDisciplinas(IRepositorioDisciplina repositorioDisciplina)

{

\_repositorioDisciplina = repositorioDisciplina;

}

public IEnumerable<DisciplinaDTO> Ejecutar() => DisciplinaMapper.DisciplinasToListaDTO(\_repositorioDisciplina.GetAll());

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: FindAtletasDisciplina.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaAplicacion\CasosDeUso\Disciplinas\FindAtletasDisciplina.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using Compartido.DTOs.Atletas;

using Compartido.Mappers;

using LogicaAplicacion.ICasosDeUso.Disciplinas;

using LogicaNegocio.ExcepcionesEntidades;

using LogicaNegocio.IRepositorios;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaAplicacion.CasosDeUso.Disciplinas

{

public class FindAtletasDisciplina : IFindAtletasDisciplina

{

private IRepositorioDisciplina \_repositorioDisciplina;

public FindAtletasDisciplina(IRepositorioDisciplina repositorioDisciplina)

{

\_repositorioDisciplina = repositorioDisciplina;

}

public IEnumerable<AtletaListaDTO> Ejecutar(int idDisciplina)

{

if (idDisciplina == 0)

{

throw new DisciplinaException("Id Incorrecto");

}

return AtletaMapper.AtletasToListaDTO(\_repositorioDisciplina.GetAtletasDisciplina(idDisciplina));

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: InsertDisciplina.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaAplicacion\CasosDeUso\Disciplinas\InsertDisciplina.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using Compartido.DTOs.Disciplinas;

using Compartido.Mappers;

using LogicaAccesoDatos;

using LogicaAplicacion.ICasosDeUso.Disciplinas;

using LogicaAplicacion.Validadores;

using LogicaNegocio.Entidades;

using LogicaNegocio.ExcepcionesEntidades;

using LogicaNegocio.IRepositorios;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaAplicacion.CasosDeUso.Disciplinas

{

public class InsertDisciplina : IInsertDisciplina

{

private IRepositorioDisciplina \_repositorioDisciplina;

public InsertDisciplina(IRepositorioDisciplina repositorioDisciplina)

{

\_repositorioDisciplina = repositorioDisciplina;

}

public int Ejecutar(DisciplinaInsertDTO disciplinaInsertDTO)

{

ValidarDisciplina.Nombre(disciplinaInsertDTO.Nombre);

ValidarDisciplina.Anio(disciplinaInsertDTO.AnioIntegracion);

Disciplina? BuscarSiExiste = \_repositorioDisciplina.GetByNombre(disciplinaInsertDTO.Nombre);

if (BuscarSiExiste != null) throw new ConflictException("Ya existe una disciplina con ese nombre");

Disciplina disciplina = \_repositorioDisciplina.Insert(DisciplinaMapper.InsertDTOToDisciplina(disciplinaInsertDTO));

return disciplina.Id;

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: AltaEvento.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaAplicacion\CasosDeUso\Eventos\AltaEvento.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using Compartido.DTOs.Eventos;

using LogicaAccesoDatos;

using LogicaAccesoDatos.Repositorios;

using LogicaAplicacion.ICasosDeUso.Eventos;

using LogicaNegocio.IRepositorios;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using LogicaNegocio.ExcepcionesEntidades;

using LogicaAplicacion.Validadores;

using Compartido.Mappers;

namespace LogicaAplicacion.CasosDeUso.Eventos

{

public class AltaEvento : IAltaEvento

{

private readonly IRepositorioEvento \_repositorioEvento;

private readonly IRepositorioDisciplina \_repositorioDisciplina;

public AltaEvento(IRepositorioEvento repositorioEvento, IRepositorioDisciplina repositorioDisciplina)

{

\_repositorioEvento = repositorioEvento;

\_repositorioDisciplina = repositorioDisciplina;

}

public void Ejecutar(EventoInsertDTO eventoInsertDTO)

{

if (eventoInsertDTO == null)

throw new EventoException("El Evento DTO se encuentra vacío");

if (\_repositorioEvento.GetByNombre(eventoInsertDTO.NombrePrueba) != null)

throw new EventoException("Ya existe un evento con ese nombre");

ValidarEvento.CantidadAtletas(eventoInsertDTO);

ValidarEvento.AtletasRegistradosEnDisciplina

(\_repositorioDisciplina.GetAtletasDisciplina(eventoInsertDTO.DisciplinaId), eventoInsertDTO.AtletasId);

ValidarEvento.Fechas(eventoInsertDTO);

\_repositorioEvento.Add(EventoMapper.DtoToEvento(eventoInsertDTO));

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: CargarPuntajes.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaAplicacion\CasosDeUso\Eventos\CargarPuntajes.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using Compartido.DTOs.Eventos;

using Compartido.Mappers;

using LogicaAplicacion.ICasosDeUso.Eventos;

using LogicaNegocio.Entidades;

using LogicaNegocio.IRepositorios;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using LogicaNegocio.ExcepcionesEntidades;

using LogicaAplicacion.Validadores;

namespace LogicaAplicacion.CasosDeUso.Eventos

{

public class CargarPuntajes : ICargarPuntajes

{

private readonly IRepositorioEvento \_repositorioEvento;

public CargarPuntajes(IRepositorioEvento repositorioEvento)

{

\_repositorioEvento = repositorioEvento;

}

public EventoDTO Ejecutar(EventoUpdatePuntajesDTO eventoUpdatePuntajesDTO)

{

// Validar puntajes >= 0

ValidarEvento.Puntajes(eventoUpdatePuntajesDTO);

// Validar que existe el Evento

Evento eventoAModificar = \_repositorioEvento.GetById(eventoUpdatePuntajesDTO.Id);

if (eventoAModificar == null) throw new EventoException("No se encontró un evento con ese Id");

// Update

eventoAModificar.LiPuntajes = EventoMapper.DtoListaModificadaToListaPuntaje(eventoUpdatePuntajesDTO.LiAtletas);

\_repositorioEvento.Update(eventoAModificar);

return EventoMapper.EventoToDTO(eventoAModificar);

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: FindById.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaAplicacion\CasosDeUso\Eventos\FindById.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using Compartido.DTOs.Eventos;

using Compartido.Mappers;

using LogicaAplicacion.ICasosDeUso.Eventos;

using LogicaNegocio.ExcepcionesEntidades;

using LogicaNegocio.IRepositorios;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaAplicacion.CasosDeUso.Eventos

{

public class FindById : IFindById

{

private readonly IRepositorioEvento \_repositorioEvento;

public FindById(IRepositorioEvento repositorioEvento)

{

\_repositorioEvento = repositorioEvento;

}

public EventoDTO Ejecutar(int id)

{

if (id < 1) throw new EventoException("No existe atleta con ese id");

return EventoMapper.EventoToDTO(\_repositorioEvento.GetById(id));

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: FindEventosFecha.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaAplicacion\CasosDeUso\Eventos\FindEventosFecha.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using Compartido.DTOs.Eventos;

using Compartido.Mappers;

using LogicaAplicacion.ICasosDeUso.Eventos;

using LogicaNegocio.IRepositorios;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaAplicacion.CasosDeUso.Eventos

{

public class FindEventosFecha : IFindEventosFecha

{

private readonly IRepositorioEvento \_repositorioEvento;

public FindEventosFecha(IRepositorioEvento repositorioEvento)

{

\_repositorioEvento = repositorioEvento;

}

public IEnumerable<EventoListaDTO> Ejecutar(DateTime fecha) => EventoMapper.EventosToListaDTO(\_repositorioEvento.GetByFecha(fecha));

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: SelectByBusqueda.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaAplicacion\CasosDeUso\Eventos\SelectByBusqueda.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using Compartido.DTOs.Eventos;

using Compartido.Mappers;

using LogicaAplicacion.ICasosDeUso.Eventos;

using LogicaNegocio.ExcepcionesEntidades;

using LogicaNegocio.IRepositorios;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaAplicacion.CasosDeUso.Eventos

{

public class SelectByBusqueda : ISelectByBusqueda

{

private readonly IRepositorioEvento \_repositorioEvento;

public SelectByBusqueda(IRepositorioEvento repositorioEvento)

{

\_repositorioEvento = repositorioEvento;

}

public IEnumerable<EventoListaDTO> Ejecutar(EventoBusquedaDTO busqueda)

{

// Validaciones de la búsqueda

if(busqueda.DisciplinaId <= 0)

throw new EventoException("Id incorrecto");

if (

(busqueda.FchInicio != null && busqueda.FchFin == null) ||

(busqueda.FchFin != null && busqueda.FchInicio == null) ||

(busqueda.PuntajeMin != null && busqueda.PuntajeMax == null) ||

(busqueda.PuntajeMax != null && busqueda.PuntajeMin == null)

) throw new EventoException("Si ingresa una fecha debe ingresar ambas, lo mismo para los puntajes");

if (busqueda.FchInicio > busqueda.FchFin) throw new EventoException("Rango de fechas incorrecto");

if (busqueda.PuntajeMax < busqueda.PuntajeMin) throw new EventoException("Rango de puntajes incorrecto");

return EventoMapper.EventosToListaDTO(\_repositorioEvento.SelectByBusqueda(

busqueda.DisciplinaId,

busqueda.FchInicio,

busqueda.FchFin,

busqueda.NombreEvento,

busqueda.PuntajeMin,

busqueda.PuntajeMax

));

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: AltaPais.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaAplicacion\CasosDeUso\Paises\AltaPais.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using Compartido.DTOs.Paises;

using LogicaAccesoDatos;

using LogicaAplicacion.ICasosDeUso.Paises;

using LogicaNegocio.IRepositorios;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaAplicacion.CasosDeUso.Paises

{

public class AltaPais : IAltaPais

{

private readonly IRepositorioPais \_repositorioPais;

public AltaPais(IRepositorioPais repositorioPais)

{

\_repositorioPais = repositorioPais;

}

public void Ejecutar(PaisInsertDTO paisInsertDTO)

{

throw new NotImplementedException();

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: AltaUsuario.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaAplicacion\CasosDeUso\Usuarios\AltaUsuario.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using Compartido.DTOs.Usuarios;

using Compartido.Mappers;

using LogicaAccesoDatos;

using LogicaAplicacion.ICasosDeUso.Usuarios;

using LogicaAplicacion.Validadores;

using LogicaNegocio.Entidades;

using LogicaNegocio.ExcepcionesEntidades;

using LogicaNegocio.IRepositorios;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Text.RegularExpressions;

using System.Threading.Tasks;

namespace LogicaAplicacion.CasosDeUso.Usuarios

{

public partial class AltaUsuario : IAltaUsuario

{

private readonly IRepositorioUsuario \_repositorioUsuario;

public AltaUsuario(IRepositorioUsuario repoUsuario)

{

\_repositorioUsuario = repoUsuario;

}

public void Ejecutar(UsuarioInsertDTO usuarioInsertDTO)

{

if (usuarioInsertDTO == null)

{

throw new UsuarioException("El usuario no puede ser vacío");

}

Usuario? buscarSiExiste = \_repositorioUsuario.GetByEmail(usuarioInsertDTO.Email);

if (buscarSiExiste != null)

{

throw new ConflictException("Ya se registró un usuario con ese email");

}

// Validaciones

var (email, contrasena, rol, nombre) =

(usuarioInsertDTO.Email, usuarioInsertDTO.Contrasena, usuarioInsertDTO.RolUsuario, usuarioInsertDTO.Nombre);

ValidarUsuario.Email(email);

ValidarUsuario.Contrasena(contrasena);

ValidarUsuario.Rol(rol);

ValidarUsuario.Nombre(nombre);

\_repositorioUsuario.Add(UsuarioMapper.InsertDTOToUsuario(usuarioInsertDTO));

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: DeleteUsuario.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaAplicacion\CasosDeUso\Usuarios\DeleteUsuario.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using LogicaAplicacion.ICasosDeUso.Usuarios;

using LogicaNegocio.Entidades;

using LogicaNegocio.IRepositorios;

using LogicaNegocio.ExcepcionesEntidades;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaAplicacion.CasosDeUso.Usuarios

{

public class DeleteUsuario : IDeleteUsuario

{

private readonly IRepositorioUsuario \_repositorioUsuario;

public DeleteUsuario(IRepositorioUsuario repositorioUsuario)

{

\_repositorioUsuario = repositorioUsuario;

}

public void Ejecutar(int id)

{

Usuario resDB = \_repositorioUsuario.GetById(id);

if (resDB == null)

{

throw new UsuarioException("Usuario no encontrado por ese ID");

}

\_repositorioUsuario.Delete(resDB);

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: FindAllUsuarios.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaAplicacion\CasosDeUso\Usuarios\FindAllUsuarios.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using Compartido.DTOs.Usuarios;

using Compartido.Mappers;

using LogicaAplicacion.ICasosDeUso.Usuarios;

using LogicaNegocio.Entidades;

using LogicaNegocio.IRepositorios;

namespace LogicaAplicacion.CasosDeUso.Usuarios

{

public class FindAllUsuarios : IFindAllUsuarios

{

private readonly IRepositorioUsuario \_repositorioUsuario;

public FindAllUsuarios(IRepositorioUsuario repoUsuario)

{

\_repositorioUsuario = repoUsuario;

}

public IEnumerable<UsuarioDTO> Ejecutar()

{

List<Usuario> usuarios = \_repositorioUsuario.GetAll();

return UsuarioMapper.ListaUsuariosToDTOListaUsuarios(usuarios);

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: GetByIdUsuario.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaAplicacion\CasosDeUso\Usuarios\GetByIdUsuario.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using Compartido.DTOs.Usuarios;

using LogicaAplicacion.ICasosDeUso.Usuarios;

using LogicaNegocio.IRepositorios;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using LogicaNegocio.ExcepcionesEntidades;

using Compartido.Mappers;

namespace LogicaAplicacion.CasosDeUso.Usuarios

{

public class GetByIdUsuario : IGetByIdUsuario

{

private readonly IRepositorioUsuario \_repositorioUsuario;

public GetByIdUsuario(IRepositorioUsuario repositorioUsuario)

{

\_repositorioUsuario = repositorioUsuario;

}

public UsuarioDTO Ejecutar(int id)

{

LogicaNegocio.Entidades.Usuario? res = \_repositorioUsuario.GetById(id);

if (res == null)

{

throw new UsuarioException("Usuario no encontrado por ID");

}

return UsuarioMapper.UsuarioToDTO(res);

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: LoginUsuario.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaAplicacion\CasosDeUso\Usuarios\LoginUsuario.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using LogicaAccesoDatos;

using LogicaNegocio.Entidades;

using LogicaNegocio.IRepositorios;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using LogicaNegocio.Entidades;

using LogicaAplicacion.ICasosDeUso.Usuarios;

using Compartido.DTOs.Usuarios;

using Compartido.Mappers;

using LogicaNegocio.ExcepcionesEntidades;

namespace LogicaAplicacion.CasosDeUso.Usuarios

{

public class LoginUsuario : ILoginUsuario

{

private readonly IRepositorioUsuario \_repositorioUsuario;

public LoginUsuario(IRepositorioUsuario repositorioUsuario)

{

\_repositorioUsuario = repositorioUsuario;

}

public UsuarioDTO Ejecutar(string email, string password)

{

Usuario resBD = \_repositorioUsuario.LoginUsuario(email, password)

?? throw new UsuarioException("Usuario o contraseña incorrectos");

return UsuarioMapper.UsuarioToDTO(resBD);

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: UpdateUsuario.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaAplicacion\CasosDeUso\Usuarios\UpdateUsuario.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using Compartido.DTOs.Usuarios;

using LogicaAplicacion.ICasosDeUso.Usuarios;

using LogicaAplicacion.Validadores;

using LogicaNegocio.Entidades;

using LogicaNegocio.IRepositorios;

using LogicaNegocio.ExcepcionesEntidades;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Compartido.Mappers;

using LogicaNegocio.ValueObjects.Usuario;

using LogicaNegocio.Enums;

namespace LogicaAplicacion.CasosDeUso.Usuarios

{

public class UpdateUsuario : IUpdateUsuario

{

private readonly IRepositorioUsuario \_repositorioUsuario;

public UpdateUsuario(IRepositorioUsuario repositorioUsuario)

{

\_repositorioUsuario = repositorioUsuario;

}

public UsuarioUpdateDTO Ejecutar(int id, UsuarioUpdateDTO usuarioUpdateDTO)

{

var (email, nombre) = (usuarioUpdateDTO.Email, usuarioUpdateDTO.Nombre);

ValidarUsuario.Email(email);

ValidarUsuario.Nombre(nombre);

Usuario actualizarUsuario = \_repositorioUsuario.GetById(id)

?? throw new UsuarioException("No se encontró el usuario que intenta actualizar");

if (\_repositorioUsuario.GetByEmail(email) != null)

throw new ConflictException("Este email ya pertenece a un usuario");

actualizarUsuario.Email = new RUsuarioEmail(email);

actualizarUsuario.Nombre = nombre;

\_repositorioUsuario.Update(actualizarUsuario);

return UsuarioMapper.UsuarioToUpdateDTO(actualizarUsuario);

}

public UsuarioUpdateDTO Ejecutar(int id, string contrasena)

{

ValidarUsuario.Contrasena(contrasena);

Usuario actualizarUsuario = \_repositorioUsuario.GetById(id)

?? throw new UsuarioException("No se encontró el usuario que intenta actualizar");

actualizarUsuario.Contrasena = new RUsuarioContrasena(contrasena);

\_repositorioUsuario.Update(actualizarUsuario);

return UsuarioMapper.UsuarioToUpdateDTO(actualizarUsuario);

}

public UsuarioUpdateDTO Ejecutar(int id, string contrasena, string contrasenaAnterior)

{

ValidarUsuario.Contrasena(contrasena);

Usuario actualizarUsuario = \_repositorioUsuario.GetById(id)

?? throw new UsuarioException("No se encontró el usuario que intenta actualizar");

if (contrasenaAnterior != actualizarUsuario.Contrasena.Valor)

throw new UsuarioException("La contraseña actual que ingresó es incorrecta");

if (contrasenaAnterior == contrasena)

throw new UsuarioException("La nueva contraseña que intenta ingresar es igual a la actual");

actualizarUsuario.Contrasena = new RUsuarioContrasena(contrasena);

\_repositorioUsuario.Update(actualizarUsuario);

return UsuarioMapper.UsuarioToUpdateDTO(actualizarUsuario);

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: IAgregarDisciplina.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaAplicacion\ICasosDeUso\Atletas\IAgregarDisciplina.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaAplicacion.ICasosDeUso.Atletas

{

public interface IAgregarDisciplina

{

void Ejecutar(int id, int? idDisciplina);

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: IAltaAtleta.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaAplicacion\ICasosDeUso\Atletas\IAltaAtleta.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using Compartido.DTOs.Atletas;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaAplicacion.ICasosDeUso.Atletas

{

public interface IAltaAtleta

{

void Ejecutar(AtletaInsertDTO atletaInsertDTO);

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: IEventosAtleta.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaAplicacion\ICasosDeUso\Atletas\IEventosAtleta.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using Compartido.DTOs.Eventos;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaAplicacion.ICasosDeUso.Atletas

{

public interface IEventosAtleta

{

IEnumerable<AtletaEventoListaDTO> Ejecutar(int atletaId);

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: IFindAllAtletas.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaAplicacion\ICasosDeUso\Atletas\IFindAllAtletas.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using LogicaNegocio.Entidades;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Compartido.DTOs.Atletas;

namespace LogicaAplicacion.ICasosDeUso.Atletas

{

public interface IFindAllAtletas

{

IEnumerable<AtletaListaDTO> Ejectuar();

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: IGetByIdAtleta.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaAplicacion\ICasosDeUso\Atletas\IGetByIdAtleta.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using Compartido.DTOs.Atletas;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaAplicacion.ICasosDeUso.Atletas

{

public interface IGetByIdAtleta

{

AtletaDTO Ejecutar(int id);

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: ISelectByDisciplinaId.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaAplicacion\ICasosDeUso\Atletas\ISelectByDisciplinaId.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using Compartido.DTOs.Atletas;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaAplicacion.ICasosDeUso.Atletas

{

public interface ISelectByDisciplinaId

{

IEnumerable<AtletaListaDTO> Ejecutar(int disciplinaId);

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: IAuditoriaInsert.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaAplicacion\ICasosDeUso\Auditorias\IAuditoriaInsert.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using Compartido.DTOs.Auditorias;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaAplicacion.ICasosDeUso.Auditorias

{

public interface IAuditoriaInsert

{

void Ejecutar(AuditoriaInsertDTO dto);

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: IDeleteDisciplina.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaAplicacion\ICasosDeUso\Disciplinas\IDeleteDisciplina.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaAplicacion.ICasosDeUso.Disciplinas

{

public interface IDeleteDisciplina

{

void Ejecutar(int id);

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: IDisciplinaSelectById.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaAplicacion\ICasosDeUso\Disciplinas\IDisciplinaSelectById.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using Compartido.DTOs.Disciplinas;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaAplicacion.ICasosDeUso.Disciplinas

{

public interface IDisciplinaSelectById

{

DisciplinaDTO? Ejecutar(int id);

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: IDisciplinaSelectByNombre.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaAplicacion\ICasosDeUso\Disciplinas\IDisciplinaSelectByNombre.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using Compartido.DTOs.Disciplinas;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaAplicacion.ICasosDeUso.Disciplinas

{

public interface IDisciplinaSelectByNombre

{

DisciplinaDTO? Ejecutar(string nombre);

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: IDisciplinaUpdate.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaAplicacion\ICasosDeUso\Disciplinas\IDisciplinaUpdate.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using Compartido.DTOs.Disciplinas;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaAplicacion.ICasosDeUso.Disciplinas

{

public interface IDisciplinaUpdate

{

DisciplinaDTO Ejecutar(int id,DisciplinaUpdateDTO dto);

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: IFindAllDisciplinas.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaAplicacion\ICasosDeUso\Disciplinas\IFindAllDisciplinas.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using Compartido.DTOs.Disciplinas;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaAplicacion.ICasosDeUso.Disciplinas

{

public interface IFindAllDisciplinas

{

IEnumerable<DisciplinaDTO> Ejecutar();

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: IFindAtletasDisciplina.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaAplicacion\ICasosDeUso\Disciplinas\IFindAtletasDisciplina.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using Compartido.DTOs.Atletas;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaAplicacion.ICasosDeUso.Disciplinas

{

public interface IFindAtletasDisciplina

{

IEnumerable<AtletaListaDTO> Ejecutar(int idDisciplina);

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: IInsertDisciplina.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaAplicacion\ICasosDeUso\Disciplinas\IInsertDisciplina.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using Compartido.DTOs.Disciplinas;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaAplicacion.ICasosDeUso.Disciplinas

{

public interface IInsertDisciplina

{

int Ejecutar(DisciplinaInsertDTO disciplinaInsertDTO);

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: IAltaEvento.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaAplicacion\ICasosDeUso\Eventos\IAltaEvento.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using Compartido.DTOs.Eventos;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaAplicacion.ICasosDeUso.Eventos

{

public interface IAltaEvento

{

void Ejecutar(EventoInsertDTO eventoInsertDTO);

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: ICargarPuntajes.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaAplicacion\ICasosDeUso\Eventos\ICargarPuntajes.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using Compartido.DTOs.Eventos;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaAplicacion.ICasosDeUso.Eventos

{

public interface ICargarPuntajes

{

EventoDTO Ejecutar(EventoUpdatePuntajesDTO eventoUpdatePuntajesDTO);

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: IFindById.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaAplicacion\ICasosDeUso\Eventos\IFindById.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using Compartido.DTOs.Eventos;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaAplicacion.ICasosDeUso.Eventos

{

public interface IFindById

{

EventoDTO Ejecutar(int id);

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: IFindEventosFecha.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaAplicacion\ICasosDeUso\Eventos\IFindEventosFecha.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Compartido.DTOs.Eventos;

namespace LogicaAplicacion.ICasosDeUso.Eventos

{

public interface IFindEventosFecha

{

IEnumerable<EventoListaDTO> Ejecutar(DateTime fecha);

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: ISelectByBusqueda.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaAplicacion\ICasosDeUso\Eventos\ISelectByBusqueda.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using Compartido.DTOs.Eventos;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaAplicacion.ICasosDeUso.Eventos

{

public interface ISelectByBusqueda

{

IEnumerable<EventoListaDTO> Ejecutar(EventoBusquedaDTO busqueda);

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: IAltaPais.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaAplicacion\ICasosDeUso\Paises\IAltaPais.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using Compartido.DTOs.Paises;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaAplicacion.ICasosDeUso.Paises

{

public interface IAltaPais

{

void Ejecutar(PaisInsertDTO paisInsertDTO);

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: IAltaUsuario.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaAplicacion\ICasosDeUso\Usuarios\IAltaUsuario.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using Compartido.DTOs.Usuarios;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaAplicacion.ICasosDeUso.Usuarios

{

public interface IAltaUsuario

{

void Ejecutar(UsuarioInsertDTO usuarioInsertDTO);

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: IDeleteUsuario.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaAplicacion\ICasosDeUso\Usuarios\IDeleteUsuario.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaAplicacion.ICasosDeUso.Usuarios

{

public interface IDeleteUsuario

{

void Ejecutar(int id);

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: IFindAllUsuarios.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaAplicacion\ICasosDeUso\Usuarios\IFindAllUsuarios.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using Compartido.DTOs.Usuarios;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaAplicacion.ICasosDeUso.Usuarios

{

public interface IFindAllUsuarios

{

IEnumerable<UsuarioDTO> Ejecutar();

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: IGetByIdUsuario.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaAplicacion\ICasosDeUso\Usuarios\IGetByIdUsuario.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using Compartido.DTOs.Usuarios;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaAplicacion.ICasosDeUso.Usuarios

{

public interface IGetByIdUsuario

{

UsuarioDTO Ejecutar(int id);

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: ILoginUsuario.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaAplicacion\ICasosDeUso\Usuarios\ILoginUsuario.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using Compartido.DTOs.Usuarios;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaAplicacion.ICasosDeUso.Usuarios

{

public interface ILoginUsuario

{

UsuarioDTO Ejecutar(string email, string password);

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: IUpdateUsuario.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaAplicacion\ICasosDeUso\Usuarios\IUpdateUsuario.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using Compartido.DTOs.Usuarios;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaAplicacion.ICasosDeUso.Usuarios

{

public interface IUpdateUsuario

{

UsuarioUpdateDTO Ejecutar(int id, UsuarioUpdateDTO usuarioUpdateDTO);

UsuarioUpdateDTO Ejecutar(int id, string contrasena, string contrasenaAnterior);

UsuarioUpdateDTO Ejecutar(int id, string contrasena);

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: RDisciplinaNombre.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaNegocio\ValueObjects\Disciplina\RDisciplinaNombre.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaNegocio.ValueObjects.Disciplina

{

[ComplexType]

public record RDisciplinaNombre

{

[Required]

[StringLength(50, MinimumLength = 10)]

public string Valor { get; init; }

public RDisciplinaNombre(string valor)

{

Valor = valor;

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: RUsuarioContrasena.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaNegocio\ValueObjects\Usuario\RUsuarioContrasena.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaNegocio.ValueObjects.Usuario

{

[ComplexType]

public record RUsuarioContrasena

{

[Required]

public string Valor { get; init; }

public RUsuarioContrasena(string valor)

{

Valor = valor;

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: RUsuarioEmail.cs

Carpeta: C:\Users\cri\_1\OneDrive\Escritorio\Obligatorio1\_Prog3\ObligatorioP3v2\Entrega Obligatorio 2\SolObligatorioP3\LogicaNegocio\ValueObjects\Usuario\RUsuarioEmail.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicaNegocio.ValueObjects.Usuario

{

[ComplexType]

public record RUsuarioEmail

{

[Required]

[MaxLength(255)]

public string Valor { get; init; }

public RUsuarioEmail(string valor)

{

Valor = valor;

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: AtletaController.cs

Carpeta: C:\Users\cri\_1\Downloads\COPIAMVC\MVC\Controllers\AtletaController.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using MVC.Models.Atleta;

using MVC.Models.Disciplina;

using MVC.Utils;

using Newtonsoft.Json;

using System.Net.Http.Headers;

namespace MVC.Controllers

{

public class AtletaController : Controller

{

private readonly string \_url;

public AtletaController(IConfiguration config)

{

\_url = config.GetConnectionString("API");

}

public async Task<ActionResult> ListAtletasDisciplina(int? disciplinaId)

{

try

{

(string, HttpResponseMessage) disciplinas =

await ConexionServidor.ClientSinBody(\_url + "/api/Disciplina", "GET");

if (disciplinas.Item2.IsSuccessStatusCode)

{

IEnumerable<DisciplinaVM> listaDiscplinas = JsonConvert.DeserializeObject<IEnumerable<DisciplinaVM>>(disciplinas.Item1) ??

throw new Exception("Fallo en obtener listado de disciplinas");

ViewBag.Disciplinas = listaDiscplinas;

}

else

{

TempData["ErrorMessage"] = disciplinas.Item1;

return View();

}

if (disciplinaId != null)

{

(string, HttpResponseMessage) atletas =

await ConexionServidor.ClientSinBody(\_url + "/api/Atleta/" + disciplinaId, "GET");

if (atletas.Item2.IsSuccessStatusCode)

{

IEnumerable<AtletaListaVM> listaAtletasvm = JsonConvert.DeserializeObject<IEnumerable<AtletaListaVM>>(atletas.Item1) ??

throw new Exception("Fallo en obtener listado de atletas");

return View(listaAtletasvm);

}

else

{

TempData["ErrorMessage"] = atletas.Item1;

}

}

return View();

}

catch (Exception ex)

{

TempData["ErrorMessage"] = ex.Message;

}

return View();

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: DisciplinaController.cs

Carpeta: C:\Users\cri\_1\Downloads\COPIAMVC\MVC\Controllers\DisciplinaController.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using MVC.Models.Disciplina;

using MVC.Utils;

using Newtonsoft.Json;

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

namespace MVC.Controllers

{

public class DisciplinaController : Controller

{

private readonly string \_url;

public DisciplinaController(IConfiguration config)

{

\_url = config.GetConnectionString("API");

}

public async Task<ActionResult> Index()

{

if (ManejoSession.GetRolLogueado(HttpContext) == "Digitador")

{

try

{

(string, HttpResponseMessage) disciplinas =

await ConexionServidor.ClientSinBody(\_url + "/api/Disciplina", "GET");

if (disciplinas.Item2.IsSuccessStatusCode)

{

IEnumerable<DisciplinaVM> listaDiscplinas = JsonConvert.DeserializeObject<IEnumerable<DisciplinaVM>>(disciplinas.Item1) ??

throw new Exception("Error al intentar obtener listado de disciplinas");

return View(listaDiscplinas);

}

else

{

throw new Exception(disciplinas.Item1);

}

}

catch (Exception ex)

{

return RedirectToAction("Index", "Error", new { code = 500, message = ex.Message });

}

}

else

{

return RedirectToAction("Index", "Error", new { code = 401, message = "No tiene permisos para ver esta información" });

}

}

public async Task<ActionResult> Details(int id)

{

if (id == 0)

{

TempData["ErrorMessage"] = "Id incorrecto";

return View();

}

if (ManejoSession.GetRolLogueado(HttpContext) == "Digitador")

{

try

{

string token = ManejoSession.GetToken(HttpContext)

?? throw new Exception("Fallo en la obtención del token");

(string, HttpResponseMessage) disciplinas =

await ConexionServidor.ClientSinBody(\_url + "/api/Disciplina/" + id.ToString(), "GET", token);

if (disciplinas.Item2.IsSuccessStatusCode)

{

DisciplinaVM res = JsonConvert.DeserializeObject<DisciplinaVM>(disciplinas.Item1);

return View(res);

}

else if ((int)disciplinas.Item2.StatusCode == StatusCodes.Status500InternalServerError)

{

throw new Exception(disciplinas.Item1);

}

else

{

TempData["ErrorMessage"] = disciplinas.Item1;

return View();

}

}

catch (Exception ex)

{

return RedirectToAction("Index", "Error", new { code = 500, message = ex.Message });

}

}

else

{

return RedirectToAction("Index", "Error", new { code = 401, message = "No tiene permisos para ver esta información" });

}

}

[HttpGet]

public async Task<ActionResult> Buscar(int? tipoBusqueda, int? iId, string? iNa)

{

if (ManejoSession.GetRolLogueado(HttpContext) == "Digitador")

{

try

{

if (tipoBusqueda is null || tipoBusqueda <= 0)

{

return View();

}

else if (tipoBusqueda == 1)

{

if(iId is null || iId <= 0)

{

TempData["ErrorMessage"] = "Error: Id incorrecto";

return View();

}

string token = ManejoSession.GetToken(HttpContext)

?? throw new Exception("Fallo en la obtención del token");

(string, HttpResponseMessage) disciplinas =

await ConexionServidor.ClientSinBody(\_url + "/api/Disciplina/" + iId, "GET", token);

if (disciplinas.Item2.IsSuccessStatusCode)

{

DisciplinaVM res = JsonConvert.DeserializeObject<DisciplinaVM>(disciplinas.Item1);

return View(res);

}

else if ((int)disciplinas.Item2.StatusCode == StatusCodes.Status500InternalServerError)

{

throw new Exception(disciplinas.Item1);

}

else

{

TempData["ErrorMessage"] = disciplinas.Item1;

return RedirectToAction("Buscar");

}

}

else

{

if(string.IsNullOrEmpty(iNa))

{

TempData["ErrorMessage"] = "Error: Nombre vacío";

return View();

}

iNa = iNa.Trim();

string token = ManejoSession.GetToken(HttpContext)

?? throw new Exception("Fallo en la obtención del token");

(string, HttpResponseMessage) disciplinas =

await ConexionServidor.ClientSinBody(\_url + "/api/Disciplina/Nombre/" + iNa, "GET", token);

if (disciplinas.Item2.IsSuccessStatusCode)

{

DisciplinaVM res = JsonConvert.DeserializeObject<DisciplinaVM>(disciplinas.Item1);

return View(res);

}

else if ((int)disciplinas.Item2.StatusCode == StatusCodes.Status500InternalServerError)

{

throw new Exception(disciplinas.Item1);

}

else

{

TempData["ErrorMessage"] = disciplinas.Item1;

return View();

}

}

}

catch (Exception ex)

{

return RedirectToAction("Index", "Error", new { code = 500, message = ex.Message });

}

}

else

{

return RedirectToAction("Index", "Error", new { code = 401, message = "No tiene permisos para ver esta información" });

}

}

[HttpGet]

public async Task<ActionResult> Create()

{

if (ManejoSession.GetRolLogueado(HttpContext) == "Digitador")

{

return View();

}

else

{

return RedirectToAction("Index", "Error", new { code = 401, message = "No tiene permisos para ver esta información" });

}

}

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<ActionResult> Create(DisciplinaInsertVM disciplinaInsertVM)

{

if (ManejoSession.GetRolLogueado(HttpContext) == "Digitador")

{

try

{

string token = ManejoSession.GetToken(HttpContext)

?? throw new Exception("Fallo en la obtención del token");

(string, HttpResponseMessage) disciplinas =

await ConexionServidor.ClientConBody(\_url + "/api/Disciplina/", "POST", disciplinaInsertVM, token);

if (disciplinas.Item2.IsSuccessStatusCode)

{

TempData["Message"] = "Disciplina creada correctamente";

return RedirectToAction(nameof(Create));

}

else if ((int)disciplinas.Item2.StatusCode == StatusCodes.Status500InternalServerError)

{

throw new Exception(disciplinas.Item1);

}

else

{

TempData["ErrorMessage"] = disciplinas.Item1;

return View();

}

}

catch (Exception ex)

{

return RedirectToAction("Index", "Error", new { code = 500, message = ex.Message });

}

}

else

{

return RedirectToAction("Index", "Error", new { code = 401, message = "No tiene permisos para ver esta información" });

}

}

public async Task<ActionResult> Edit(int id)

{

if (id == 0)

{

TempData["ErrorMessage"] = "Id incorrecto";

return View();

}

if (ManejoSession.GetRolLogueado(HttpContext) == "Digitador")

{

try

{

string token = ManejoSession.GetToken(HttpContext)

?? throw new Exception("Fallo en la obtención del token");

(string, HttpResponseMessage) disciplinas =

await ConexionServidor.ClientSinBody(\_url + "/api/Disciplina/" + id.ToString(), "GET", token);

if (disciplinas.Item2.IsSuccessStatusCode)

{

DisciplinaUpdateVM res = JsonConvert.DeserializeObject<DisciplinaUpdateVM>(disciplinas.Item1);

return View(res);

}

else if ((int)disciplinas.Item2.StatusCode == StatusCodes.Status500InternalServerError)

{

throw new Exception(disciplinas.Item1);

}

else

{

TempData["ErrorMessage"] = disciplinas.Item1;

return View();

}

}

catch (Exception ex)

{

return RedirectToAction("Index", "Error", new { code = 500, message = ex.Message });

}

}

else

{

return RedirectToAction("Index", "Error", new { code = 401, message = "No tiene permisos para ver esta información" });

}

}

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<ActionResult> Edit(int id, DisciplinaUpdateVM disciplinaUpdate)

{

if (id == 0)

{

TempData["ErrorMessage"] = "Id incorrecto";

return View();

}

if (ManejoSession.GetRolLogueado(HttpContext) == "Digitador")

{

try

{

disciplinaUpdate.Nombre = disciplinaUpdate.Nombre.Trim();

string token = ManejoSession.GetToken(HttpContext)

?? throw new Exception("Fallo en la obtención del token");

(string, HttpResponseMessage) updated =

await ConexionServidor.ClientConBody(\_url + "/api/Disciplina/" + id.ToString(), "PUT", disciplinaUpdate, token);

if (updated.Item2.IsSuccessStatusCode)

{

DisciplinaUpdateVM res = JsonConvert.DeserializeObject<DisciplinaUpdateVM>(updated.Item1);

TempData["Message"] = "Disciplina actualizada correctamente";

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

}

else if ((int)updated.Item2.StatusCode == StatusCodes.Status500InternalServerError)

{

throw new Exception(updated.Item1);

}

else

{

TempData["ErrorMessage"] = updated.Item1;

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

}

}

catch (Exception ex)

{

return RedirectToAction("Index", "Error", new { code = 500, message = ex.Message });

}

}

else

{

return RedirectToAction("Index", "Error", new { code = 401, message = "No tiene permisos para ver esta información" });

}

}

public async Task<ActionResult> DeleteDesdeBuscar(int id)

{

if (ManejoSession.GetIdLogueado(HttpContext) != null && ManejoSession.GetRolLogueado(HttpContext) == "Digitador")

{

try

{

string token = ManejoSession.GetToken(HttpContext)

?? throw new Exception("Fallo en la obtención del token");

(string, HttpResponseMessage) disciplinas =

await ConexionServidor.ClientSinBody(\_url + "/api/Disciplina/" + id.ToString(), "DELETE", token);

if (disciplinas.Item2.IsSuccessStatusCode)

{

TempData["Message"] = "Disciplina eliminada correctamente";

return RedirectToAction("Buscar");

}

else if ((int)disciplinas.Item2.StatusCode == StatusCodes.Status500InternalServerError)

{

throw new Exception(disciplinas.Item1);

}

else

{

TempData["ErrorMessage"] = disciplinas.Item1;

return RedirectToAction("Buscar");

}

}

catch (Exception ex)

{

return RedirectToAction("Index", "Error", new { code = 500, message = ex.Message });

}

}

else

{

return RedirectToAction("Index", "Error", new { code = 401, message = "No tiene permisos para ver esta información" });

}

}

public async Task<ActionResult> Delete(int id)

{

if (ManejoSession.GetIdLogueado(HttpContext) != null && ManejoSession.GetRolLogueado(HttpContext) == "Digitador")

{

try

{

string token = ManejoSession.GetToken(HttpContext)

?? throw new Exception("Fallo en la obtención del token");

(string, HttpResponseMessage) disciplinas =

await ConexionServidor.ClientSinBody(\_url + "/api/Disciplina/" + id.ToString(), "DELETE", token);

if (disciplinas.Item2.IsSuccessStatusCode)

{

TempData["Message"] = "Disciplina eliminada correctamente";

return RedirectToAction("Index");

}

else if ((int)disciplinas.Item2.StatusCode == StatusCodes.Status500InternalServerError)

{

throw new Exception(disciplinas.Item1);

}

else

{

TempData["ErrorMessage"] = disciplinas.Item1;

return RedirectToAction("Index");

}

}

catch (Exception ex)

{

return RedirectToAction("Index", "Error", new { code = 500, message = ex.Message });

}

}

else

{

return RedirectToAction("Index", "Error", new { code = 401, message = "No tiene permisos para ver esta información" });

}

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: ErrorController.cs

Carpeta: C:\Users\cri\_1\Downloads\COPIAMVC\MVC\Controllers\ErrorController.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using Microsoft.AspNetCore.Mvc;

namespace MVC.Controllers

{

public class ErrorController : Controller

{

public IActionResult Index(int? code, string? message)

{

if (code != null)

{

ViewBag.Message = new { code, message };

}

return View();

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: EventoController.cs

Carpeta: C:\Users\cri\_1\Downloads\COPIAMVC\MVC\Controllers\EventoController.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using MVC.Models.Atleta;

using MVC.Models.Disciplina;

using MVC.Models.Evento;

using MVC.Utils;

using Newtonsoft.Json;

namespace MVC.Controllers

{

public class EventoController : Controller

{

private readonly string \_url;

public EventoController(IConfiguration config)

{

\_url = config.GetConnectionString("API");

}

[HttpGet]

public async Task<ActionResult> Busqueda(EventoBusquedaVM? vm)

{

if (ManejoSession.GetRolLogueado(HttpContext) == "Digitador")

{

// En teoría vm nunc debería ser null, pero por las dudas

if (vm is null || vm.GetType().GetProperties().All(prop => prop.GetValue(vm) == null))

{

return View();

}

else

{

try

{

// Validaciones del VM

if (vm.DisciplinaId <= 0)

{

ViewBag.ErrorMessage = "Id incorrecto";

return View();

}

if (

(vm.FchInicio != null && vm.FchFin == null) ||

(vm.FchFin != null && vm.FchInicio == null) ||

(vm.PuntajeMin != null && vm.PuntajeMax == null) ||

(vm.PuntajeMax != null && vm.PuntajeMin == null)

)

{

ViewBag.ErrorMessage = "Si ingresa una fecha debe ingresar ambas, lo mismo para los puntajes";

return View();

}

if (vm.FchInicio > vm.FchFin)

{

ViewBag.ErrorMessage = "Rango de fechas incorrecto";

return View();

}

if (vm.PuntajeMax < vm.PuntajeMin ||

vm.PuntajeMin < 0 ||

vm.PuntajeMin > 10 ||

vm.PuntajeMax < 0 ||

vm.PuntajeMax > 10

)

{

ViewBag.ErrorMessage = "Rango de puntajes incorrecto";

return View();

}

// Reflección para generar URL dinámicamente:

List<string> queries = vm.GetType()

.GetProperties()

.Where(p => p.GetValue(vm) != null)

.Select(p =>

{

if (p.PropertyType == typeof(DateTime?))

{

DateTime? otroP = p.GetValue(vm) as DateTime?;

return $"{p.Name}={otroP.Value.ToString("yyyy-MM-dd")}";

}

else

{

return $"{p.Name}={p.GetValue(vm)}";

}

})

.ToList();

// Ej. api/Evento?DisciplinaId=2&NombreEvento=Salto | Si no hay filtro que traiga todos

string queryString = queries.Any() ? "?" + string.Join("&", queries) : "";

string token = ManejoSession.GetToken(HttpContext)

?? throw new Exception("Fallo en la obtención del token");

(string, HttpResponseMessage) disciplinas =

await ConexionServidor.ClientSinBody(\_url + "/api/Evento" + queryString, "GET", token);

if (disciplinas.Item2.IsSuccessStatusCode)

{

var res = JsonConvert.DeserializeObject<IEnumerable<EventoListaVM>>(disciplinas.Item1)

?? throw new Exception("Fallo en obtener listado de disciplinas");

return View(res);

}

else if ((int)disciplinas.Item2.StatusCode == StatusCodes.Status500InternalServerError)

{

throw new Exception(disciplinas.Item1);

}

else

{

ViewBag.ErrorMessage = disciplinas.Item2;

return View();

}

}

catch (Exception ex)

{

return RedirectToAction("Index", "Error", new { code = 500, message = ex.Message });

}

}

}

else

{

return RedirectToAction("Index", "Error", new { code = 401, message = "No tiene permisos para ver esta información" });

}

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: HomeController.cs

Carpeta: C:\Users\cri\_1\Downloads\COPIAMVC\MVC\Controllers\HomeController.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using Microsoft.AspNetCore.Mvc;

namespace MVC.Controllers

{

public class HomeController : Controller

{

public IActionResult Index()

{

return View();

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: UsuarioController.cs

Carpeta: C:\Users\cri\_1\Downloads\COPIAMVC\MVC\Controllers\UsuarioController.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using MVC.Models.Usuario;

using Microsoft.VisualStudio.Web.CodeGenerators.Mvc.Templates.BlazorIdentity.Pages.Manage;

using MVC.Utils;

using System.Net.Http.Headers;

using System.Security.Policy;

using Newtonsoft.Json;

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

using Microsoft.VisualStudio.Web.CodeGenerators.Mvc.View;

namespace MVC.Controllers

{

public class UsuarioController : Controller

{

private readonly string \_url;

public UsuarioController(IConfiguration config)

{

\_url = config.GetConnectionString("API");

}

public ActionResult Index()

{

int? usuarioId = ManejoSession.GetIdLogueado(HttpContext);

if (usuarioId != null) return RedirectToAction("Index", "Home");

return View();

}

[HttpPost]

public async Task<ActionResult> Index(string email, string password)

{

try

{

if(string.IsNullOrEmpty(email) || string.IsNullOrEmpty(password))

{

ViewBag.ErrorMessage = "Debe completar todos los campos";

return View();

}

CredencialesVM cred = new CredencialesVM

{

Email = email,

Contrasena = password

};

(string, HttpResponseMessage) res =

await ConexionServidor.ClientConBody<CredencialesVM>(\_url + "/api/Usuario/iniciar-sesion", "POST", cred);

if (res.Item2.IsSuccessStatusCode)

{

UsuarioLogueadoVM vm = JsonConvert.DeserializeObject<UsuarioLogueadoVM>(res.Item1) ??

throw new Exception("Fallo en iniciar-sesion de API");

HttpContext.Session.SetInt32("idLogueado", vm.Id);

HttpContext.Session.SetString("rolLogueado", vm.Rol);

HttpContext.Session.SetString("token", vm.Token);

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

}

else

{

ViewBag.ErrorMessage = res.Item1;

}

}

catch (Exception ex)

{

ViewBag.ErrorMessage = ex.Message;

}

return View();

}

[HttpGet]

public ActionResult CerrarSesion()

{

HttpContext.Session.Clear();

return RedirectToAction("Index");

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: ErrorViewModel.cs

Carpeta: C:\Users\cri\_1\Downloads\COPIAMVC\MVC\Models\ErrorViewModel.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

namespace MVC.Models

{

public class ErrorViewModel

{

public string? RequestId { get; set; }

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

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: ConexionServidor.cs

Carpeta: C:\Users\cri\_1\Downloads\COPIAMVC\MVC\Utils\ConexionServidor.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using Microsoft.VisualStudio.Web.CodeGenerators.Mvc.Templates.BlazorIdentity.Pages.Manage;

using MVC.Models.Usuario;

using System.Net.Http.Headers;

using System.Security.Cryptography;

using System.Security.Policy;

namespace MVC.Utils

{

public static class ConexionServidor

{

public static async Task<(string, HttpResponseMessage)> ClientSinBody(string url, string method)

{

HttpClient cliente = new HttpClient();

HttpResponseMessage respuesta;

switch (method)

{

case "GET":

respuesta = await cliente.GetAsync(url);

break;

case "DELETE":

respuesta = await cliente.DeleteAsync(url);

break;

default:

throw new Exception("Method incorrecto");

}

string resBody = await respuesta.Content.ReadAsStringAsync();

return (resBody, respuesta);

}

public static async Task<(string, HttpResponseMessage)> ClientSinBody(string url, string method, string token)

{

HttpClient cliente = new HttpClient();

cliente.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", token);

HttpResponseMessage respuesta;

switch (method)

{

case "GET":

respuesta = await cliente.GetAsync(url);

break;

case "DELETE":

respuesta = await cliente.DeleteAsync(url);

break;

default:

throw new Exception("Method incorrecto");

}

string resBody = await respuesta.Content.ReadAsStringAsync();

return (resBody, respuesta);

}

public static async Task<(string, HttpResponseMessage)> ClientConBody<T>(string url, string method, T VM)

{

HttpClient cliente = new HttpClient();

HttpResponseMessage respuesta;

switch (method)

{

case "POST":

respuesta = await cliente.PostAsJsonAsync(url, VM);

break;

case "PUT":

respuesta = await cliente.PutAsJsonAsync(url, VM);

break;

case "PATCH":

respuesta = await cliente.PatchAsJsonAsync(url, VM);

break;

default:

throw new Exception("Method incorrecto");

}

string resBody = await respuesta.Content.ReadAsStringAsync();

return (resBody, respuesta);

}

public static async Task<(string, HttpResponseMessage)> ClientConBody<T>(string url, string method, T VM, string token)

{

HttpClient cliente = new HttpClient();

cliente.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", token);

HttpResponseMessage respuesta;

switch (method)

{

case "POST":

respuesta = await cliente.PostAsJsonAsync(url, VM);

break;

case "PUT":

respuesta = await cliente.PutAsJsonAsync(url, VM);

break;

case "PATCH":

respuesta = await cliente.PatchAsJsonAsync(url, VM);

break;

default:

throw new Exception("Method incorrecto");

}

string resBody = await respuesta.Content.ReadAsStringAsync();

return (resBody, respuesta);

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: ManejoSession.cs

Carpeta: C:\Users\cri\_1\Downloads\COPIAMVC\MVC\Utils\ManejoSession.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

namespace MVC.Utils

{

public static class ManejoSession

{

public static int? GetIdLogueado(HttpContext httpContext)

{

return httpContext.Session.GetInt32("idLogueado");

}

public static string? GetRolLogueado(HttpContext httpContext)

{

return httpContext.Session.GetString("rolLogueado");

}

public static string? GetToken(HttpContext httpContext)

{

return httpContext.Session.GetString("token");

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: AtletaListaVM.cs

Carpeta: C:\Users\cri\_1\Downloads\COPIAMVC\MVC\Models\Atleta\AtletaListaVM.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System.ComponentModel;

using System.ComponentModel.DataAnnotations.Schema;

namespace MVC.Models.Atleta

{

public class AtletaListaVM

{

public int Id { get; set; }

public string Nombre { get; set; }

public string Apellido { get; set; }

[DisplayName("Nombre Completo")]

public string NombreCompleto { get; set; }

public string Sexo { get; set; }

[DisplayName("País")]

public string NombrePais { get; set; }

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: DisciplinaInsertVM.cs

Carpeta: C:\Users\cri\_1\Downloads\COPIAMVC\MVC\Models\Disciplina\DisciplinaInsertVM.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System.ComponentModel;

using System.ComponentModel.DataAnnotations;

namespace MVC.Models.Disciplina

{

public class DisciplinaInsertVM

{

[DisplayName("Nombre de disciplina")]

[Length(10,50,ErrorMessage = "El nombre de la disciplina debe tener entre 10 y 50 caracteres")]

[Required(ErrorMessage = "El nombre es requerido")]

public string Nombre { get; set; }

[DisplayName("Año de integración")]

[Required(ErrorMessage = "El año es requerido")]

[Range(1, 2024, ErrorMessage = "Año fuera del rango")]

public int AnioIntegracion { get; set; }

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: DisciplinaUpdateVM.cs

Carpeta: C:\Users\cri\_1\Downloads\COPIAMVC\MVC\Models\Disciplina\DisciplinaUpdateVM.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

namespace MVC.Models.Disciplina

{

public class DisciplinaUpdateVM

{

public int Id { get; set; }

public string Nombre { get; set; }

public int AnioIntegracion { get; set; }

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: DisciplinaVM.cs

Carpeta: C:\Users\cri\_1\Downloads\COPIAMVC\MVC\Models\Disciplina\DisciplinaVM.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System.ComponentModel;

namespace MVC.Models.Disciplina

{

public class DisciplinaVM

{

public int Id { get; set; }

public string Nombre { get; set; }

[DisplayName("Año integración")]

public int AnioIntegracion { get; set; }

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: EventoBusquedaVM.cs

Carpeta: C:\Users\cri\_1\Downloads\COPIAMVC\MVC\Models\Evento\EventoBusquedaVM.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System.ComponentModel.DataAnnotations;

using System.ComponentModel;

namespace MVC.Models.Evento

{

public class EventoBusquedaVM

{

[DisplayName("Disciplina Id")]

public int? DisciplinaId { get; set; }

[DisplayName("Fecha Inicio")]

[DataType(DataType.Date)]

public DateTime? FchInicio { get; set; }

[DisplayName("Fecha Final")]

[DataType(DataType.Date)]

public DateTime? FchFin { get; set; }

public string? NombreEvento { get; set; }

[DisplayName("Puntaje Mínimo")]

[Range(0, 10, ErrorMessage = "Los puntajes van desde 0 a 10")]

public int? PuntajeMin { get; set; }

[DisplayName("Puntaje Máximo")]

[Range(0, 10, ErrorMessage = "Los puntajes van desde 0 a 10")]

public int? PuntajeMax { get; set; }

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: EventoListaVM.cs

Carpeta: C:\Users\cri\_1\Downloads\COPIAMVC\MVC\Models\Evento\EventoListaVM.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using MVC.Models.Atleta;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel;

namespace MVC.Models.Evento

{

public class EventoListaVM

{

public int EventoId { get; set; }

[DisplayName("Nombre de la Prueba")]

public string NombrePrueba { get; set; }

[DisplayName("Fecha Inicio")]

[DataType(DataType.Date)]

public DateTime FchInicio { get; set; }

[DataType(DataType.Date)]

[DisplayName("Fecha Finalización")]

public DateTime FchFin { get; set; }

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: CredencialesVM.cs

Carpeta: C:\Users\cri\_1\Downloads\COPIAMVC\MVC\Models\Usuario\CredencialesVM.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

namespace MVC.Models.Usuario

{

public class CredencialesVM

{

public string Email { get; set; }

public string Contrasena { get; set; }

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Archivo: UsuarioLogueadoVM.cs

Carpeta: C:\Users\cri\_1\Downloads\COPIAMVC\MVC\Models\Usuario\UsuarioLogueadoVM.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

namespace MVC.Models.Usuario

{

public class UsuarioLogueadoVM

{

public int Id { get; set; }

public string Rol { get; set; }

public string Token { get; set; }

}

}

# Prompts y datos.

Listado de disciplina:

Prompt:

Una captura de pantalla de un celular con letras

Descripción generada automáticamente con confianza media

Respuesta

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente