Universidad ORT Uruguay

Escuela de Tecnología

Programación 3
Docente: Luis Dentone

Obligatorio 1

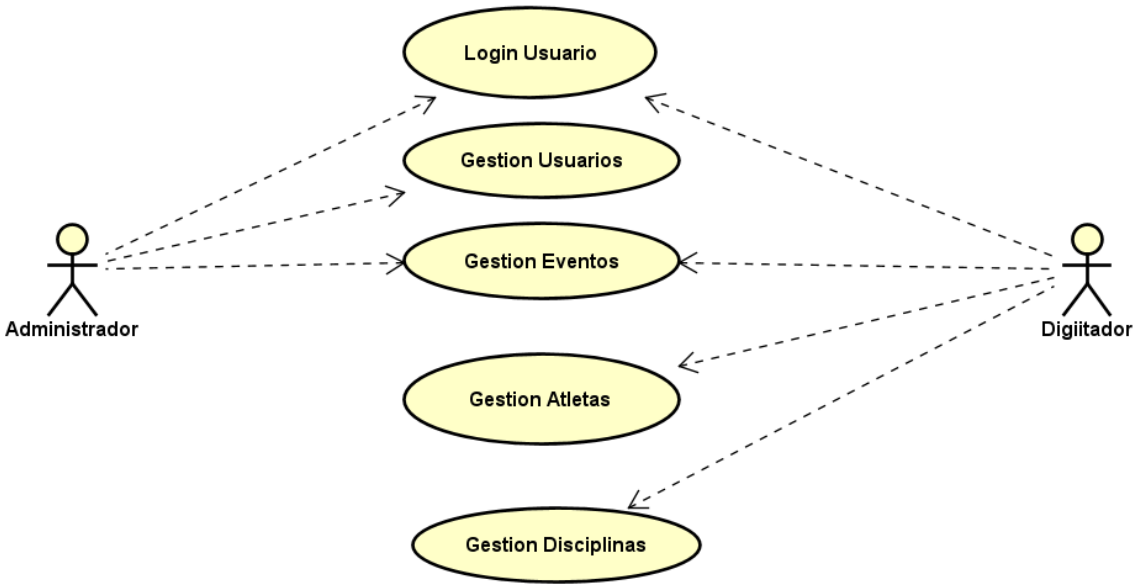
Grupo: N3B

Joaquin Acuña - 213406

JoaquinAcuna97

Índice

Diagrama de casos de uso



Descripción Narrativa de los casos de uso

Alta de Evento

Identificador: CU-01	**Nombre**:**Registrar un Evento**
Descripción: Permite registrar un nuevo evento en el sistema, dejándolo disponible para su visionado.	
Actor/es: Usuario registrado con rol administrador o digitador	
Precondiciones: Existen disciplinas cargadas y al menos 3 atletas de esa disciplina	
Pos condiciones: El evento quedó registrado en el catálogo con todos sus datos, se le asignó un código numérico y queda disponible para ser visualizado.	
Flujo Normal: 1. El usuario ingresa a agregar nuevo evento. 2. El sistema solicita Nombre, fecha de inicio, fecha de fin, Disciplina y atletas. 3. El actor introduce los datos solicitados y solicita continuar el ingreso. 4. El sistema verifica que no haya un evento con ese nombre y habilita el ingreso de los demás datos: la lista de atletas, se verifica que sean 3 de la misma disciplina 5. El sistema comprueba la validez de los datos, le asigna un código autonumérico, y redirecciona a la lista de eventos donde podemos ver el evento ingresado	

Flujo/s Alternativo/s:

3a. El usuario cancela el ingreso.

3a - 1. La evento no es guardado.

4a. No se ingresó nombre, fecha inicio o fin, o el rango de fechas es incorrecto

4a – 1. El sistema no solicita los datos adicionales y avisa que hay un error, marcando los campos erróneos con una descripción del error producido, quedando a la espera de su corrección.

4b. Ya existe una evento con ese nombre.

4b – 1. El sistema muestra un error indicando que el nombre esta repetido

5a. Los atletas seleccionados no son de la disciplina del evento

5b – 1. Si hay 3 o más atletas correctos el sistema crea el evento excluyendo a los atletas de otras disciplinas

5b – 2. Si no hay 3 atletas correctos el sistema muestra un error y no crea el evento

6a. El usuario cancela el ingreso.

6a- 1. Evento no es guardado.

Flujo/s Excepcionales/s:

La comunicación con el servidor se corta durante el registro. El evento es descartado en su totalidad.

Ingreso de puntaje de atleta

Identificador: CU-02	**Nombre**:**Puntaje de atleta**

Descripción:
Permite ingresar el puntaje del atleta en el sistema

Actor/es:
Usuario registrado con rol administrador o digitador

Precondiciones:
Existe un evento cargado correctamente en el sistema

Pos condiciones:
Los datos del puntaje del atleta son ingresados al sistema.

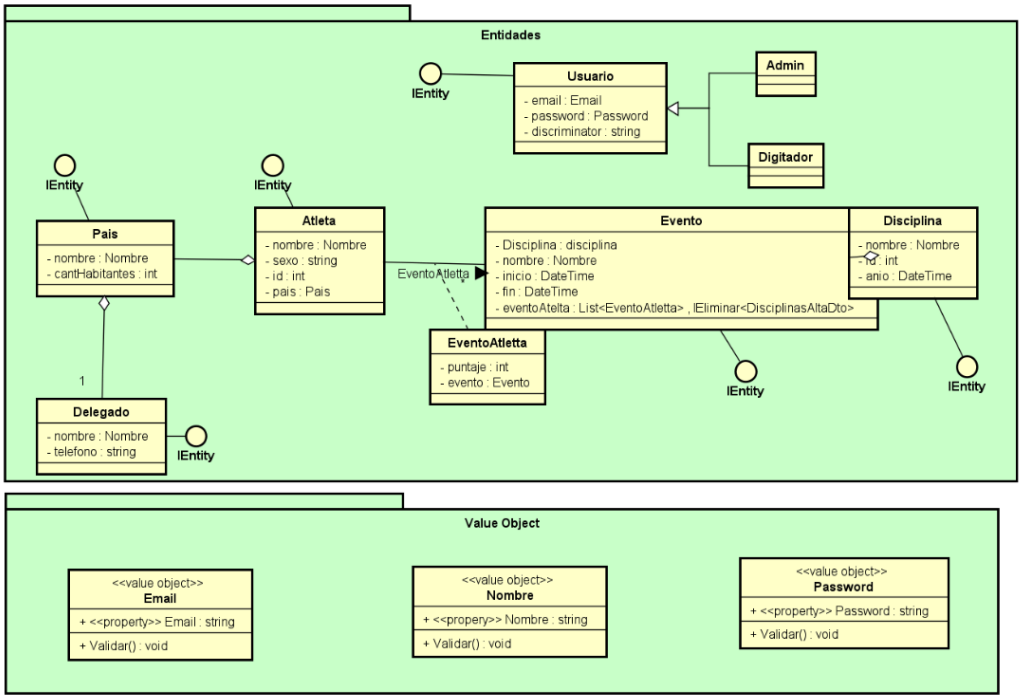
Flujo Normal:

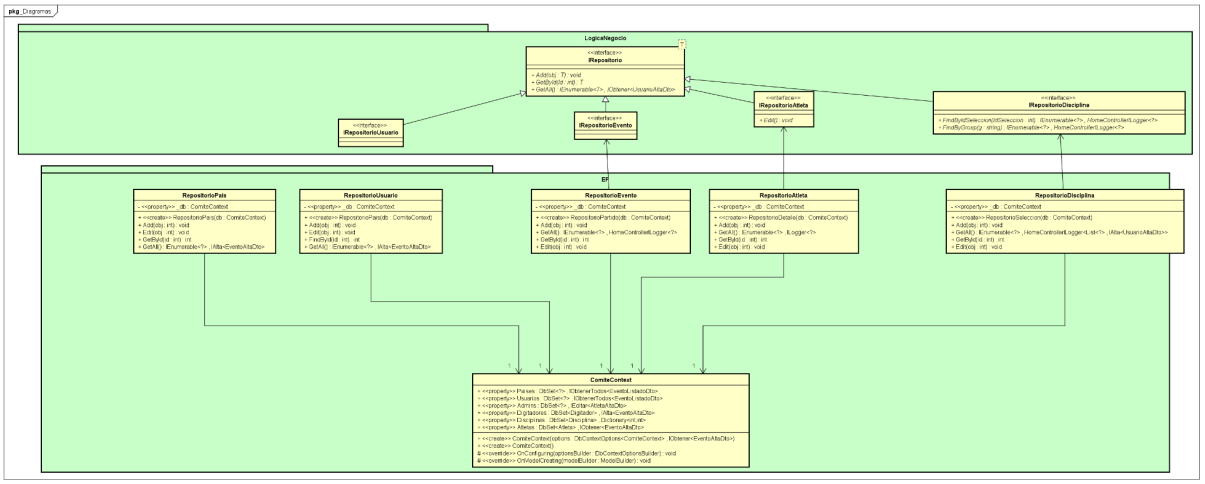
1. El usuario ingresa a lista de eventos, y selecciona agregar puntaje.
2. El sistema muestra una lista de los atletas del evento y un input numerico para ingresar el puntaje.
3. El usuario introduce los datos solicitados y solicitaguardar puntos.

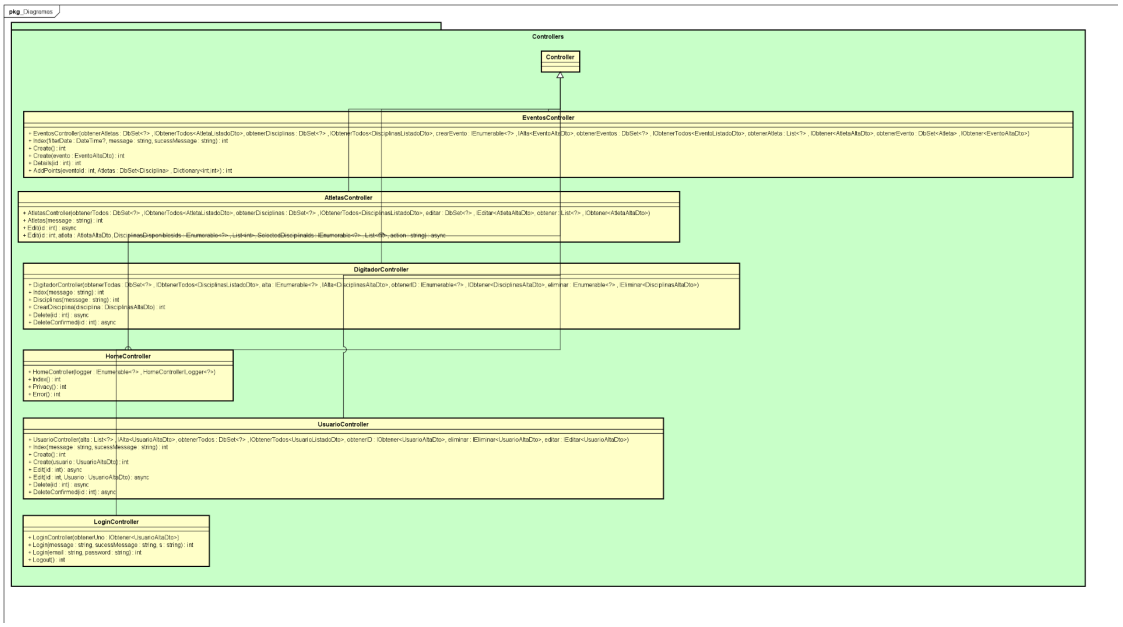
Flujo/s Excepcionales/s:

La comunicación con el servidor se corta durante el registro. La película es descartada en su totalidad.

Diagrama de Clases UML




pkg_Diagramas
LogicaNegocio
<<interface>>
IRepositorio
<<interface>>
IRepositorioUsuario
<<interface>>
IRepositorioEvento
<<interface>>
IRepositorioAtleta
<<interface>>
IRepositorioDisciplina
EF
RepositorioPais
RepositorioUsuario
RepositorioEvento
RepositorioAtleta
RepositorioDisciplina
ComiteContext


pkg_Diagramas
Controllers
Controller
EventosController
AtletasController
DigitadorController
HomeController
UsuarioController
LoginController

Código fuente de las clases del dominio

```
namespace ComiteLogicaNegocio.Entidades
{
    public abstract class Usuario
    {
        private string email;
        private string password;

        public int ID { get; set; }
        public Email Email { get; set; }
        public Password Password { get; set; }
        public string Discriminator { get; set; }
        public DateTime fecRegistro { get; set; }
        protected Usuario() { }
        public Usuario(int id,
               Email email,
               Password password)
        {
            ID = id;
            Email = email;
            Password = password;
            fecRegistro = DateTime.Now;
            Validar();
        }
        private void Validar()
        {
        }

    }
}
namespace ComiteLogicaNegocio.Entidades
{
    public class Admin : Usuario
    {
        protected Admin()
        {
        }
```

```
        public Admin(int id, string email, string password) : base(id, new
Email(email), new Password(password))
        {
        }
    }
}


namespace ComiteLogicaNegocio.Entidades
{
    public class Atleta
    {
        public int ID { get; set; }
        public string Nombre { get; set; }
        public string Sexo { get; set; }

        public Pais Pais { get; set; }

        public int PaisId { get; set; }

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


        public Atleta() { }

        public Atleta(int iD, string nombre, string sexo, int paisId,
List<int> disciplinasIds)
        {
            ID = iD;
            Nombre = nombre;
            Sexo = sexo;
            PaisId = paisId;
            DisciplinasIds = disciplinasIds;
            Disciplinas = new List<Disciplina>();
        }
    }
}
```

```
namespace ComiteLogicaNegocio.Entidades
{
    public class Delegado
    {
    }
}

namespace ComiteLogicaNegocio.Entidades
{
    public class Digitador : Usuario
    {
      //  public Admin adminRegistro { get; set; }
        protected Digitador()
        {
        }

        public Digitador(int id, string email, string password) : base(id, new
Email(email), new Password(password))
        {

        }
    }
}

namespace ComiteLogicaNegocio.Entidades
{
    public class Disciplina
    {
        public int ID { get; set; }
        public string Nombre { get; set; }
        public int Year { get; set; }

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

        public Disciplina() { }

        public Disciplina(int id, string nombre, int year)
        {
            ID = id;
```

```
            Nombre = nombre;
            Year = year;
        }
    }
}


namespace ComiteLogicaNegocio.Entidades
{
    public class Evento
    {
        public int ID { get; set; }
        public Disciplina Disciplina { get; set; }

        public int DisciplinaId { get; set; }
        public string Nombre { get; set; }
        public DateTime Inicio { get; set; }
        public DateTime Fin { get; set; }

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

        protected Evento() { }

        public Evento(int iD, int disciplina, string nombre, DateTime inicio,
DateTime fin)
        {
            ID = iD;
            DisciplinaId = disciplina;
            Nombre = nombre;
            Inicio = inicio;
            Fin = fin;
            atletas = new List<Atleta>();
        }
    }
}

namespace ComiteLogicaNegocio.Entidades
{
    public class EventoAtleta
```

```
    {
        public int ID { get; set; }
        public Evento Evento { get; set; }
        public int EventoId { get; set; }
        public Atleta Atleta { get; set; }
        public int AtletaId { get; set; }
        public decimal Puntaje { get; set; }

        protected EventoAtleta() { }

        public EventoAtleta(int iD, int evento, int atleta, decimal puntaje)
        {
            ID = iD;
            EventoId = evento;
            AtletaId = atleta;
            Puntaje = puntaje;
        }
    }
}

namespace ComiteLogicaNegocio.Entidades
{
    public class Pais
    {
        public int ID { get; set; }
        public string NombrePais { get; set; }

        public int CantidadHabitantes { get; set; }

        public string NombreContacto { get; set; }
        public string TelefonoContacto { get; set; }

        public Pais() { }
        public  Pais(int iD, string nombrePais, int cantidadHabitantes, string
nombreContacto, string telefonoContacto)
        {
            ID = iD;
            NombrePais = nombrePais;
            CantidadHabitantes = cantidadHabitantes;
```

```
            NombreContacto = nombreContacto;
            TelefonoContacto = telefonoContacto;
        }
    }
}
```