TORNEO ESPORTS UML – Samantha Mohedano

1. Análisis del problema y requisitos del sistema

Vamos a desarrollar un sistema de gestión de torneos eSports aplicado a POO respondiendo a una serie de cuestiones:

• ¿Quiénes son los actores que interactúan con el sistema?

Los actores que interactúan en este sistema de gestión son:

- Administrador: Maneja la gestión total sobre el sistema. Puede crear, eliminar, añadir, modificar, consultar, equipos, jugadores, torneos, partidas y premios y algunas cosas más.

- Jugador: Es el usuario que participa indirectamente en el sistema ya que sólo puede consultar información del equipo, premios, torneos, resultados, etc.

Hay que tener en cuenta que en UML las validaciones y el control de la integridad de datos son funcionalidades internas del sistema.

• ¿Cuáles son las acciones que cada actor puede realizar?

Tal y como se ha comentado anteriormente, las acciones que pueden realizar los actores son las siguientes:

Administrador:

- Eliminar equipos/jugadores/torneos/partidas/premios
- Registrar equipos/jugadores
- Crear torneos
- Modificar equipos/jugadores/torneos/partidas/premios
- Consultar equipos/jugadores/torneos/partidas/premios
- Registrar resultados
- Inscribir equipos en torneos
- Generar emparejamientos
- Asignar premios
- Crear partida

Jugador:

- Consultar equipos/jugadores/torneos/resultados/premios
- Consultar su perfil

• ¿Cómo se relacionan entre sí las entidades del sistema?

Las entidades presentes en este sistema de gestión de torneos son las siguientes:

Equipo, Jugador, Toreno, Partida, Premio. Y las relaciones existentes entre ellas, las cuales explican la lógica del sistema, son:

1) Equipo – Jugador
- Relación: Un equipo contiene uno o más jugadores.
- Tipo: Agregación ya que el jugador puede existir sin estar asignado a un equipo inicialmente.
- Cardinalidad: 1 equipo → * jugadores (1:N)
- Explicación: Cada jugador pertenece a un único equipo en un torneo, pero los jugadores pueden cambiar de equipo a lo largo del tiempo.

2) Torneo -Equipo
- Relación: Un torneo inscribe varios equipos.
- Tipo: Agregación
- Cardinalidad: 1 torneo → * equipos
- Explicación: Un torneo tiene varios equipos inscritos y los equipos participan en torneos.

3) Torneo - Partida
- Relación: Un torneo organiza partidas entre los equipos inscritos.
- Tipo: Composición
- Cardinalidad: 1 torneo → * partidas
- Explicación: Las partidas son generadas dentro del contexto de un torneo. Si el torneo se eliminase, las partidas asociadas a él también se eliminarían, ya que sin torneo no hay partidas.

4) Partida – Equipo
- Relación: Una partida es disputada por dos equipos.
- Tipo: Asociación bidireccional N:M
- Cardinalidad: 1 partida ↔ 2 equipos
- Explicación: Cada partida siempre tiene dos equipos que compiten, y los resultados se registran por separado para cada uno. Un equipo puede

participar en varias partidas y en una partida siempre van a estar involucrados dos equipos.

5) Torneo – Premio
- Relación: Un torneo asigna premios a equipos ganadores.
- Tipo: Composición
- Cardinalidad: 1 torneo → * premios
- Explicación: Los premios son definidos en el contexto del torneo. No tienen sentido sin el torneo que los otorga ya que sin torneo no hay premio y en este sistema un torneo puede asignar varios premios (1º posición, 2º posición, etc).

6) Premio – Equipo
- Relación: Un premio se asigna a un equipo.
- Tipo: Asociación bidireccional 1:N
- Cardinalidad: 1 Equipo → * premios / 1 premio → 1 equipo
- Explicación: El premio es entregado al equipo ganador, pero a su vez un equipo puede ganar varios premios si gana diferentes torneos.

2. Identificación de los casos de uso y elaboración del diagrama

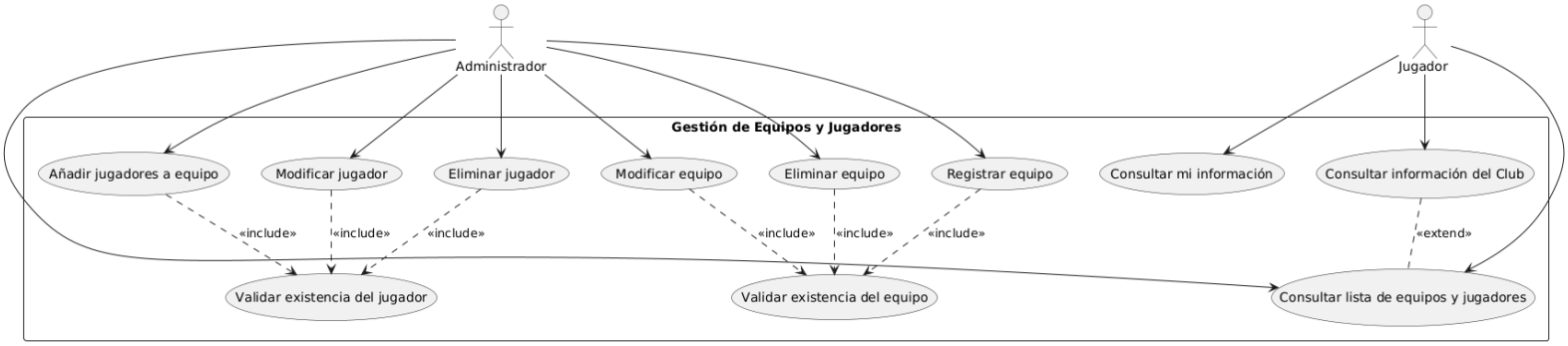
Para la explicación de este sistema de gestión hemos realizado diagramas de uso por las diferentes interacciones existentes. Cabe mencionar, que para la realización de todos los diagramas UML hemos utilizado la herramienta PlantUML donde se le ha insertado el código y la herramienta ha generado el diagrama.

1. Gestión de equipos y jugadores

Para la gestión de equipos y juegos hemos establecido los siguientes casos de uso por actor:

- Administrador:
    - Añadir/modificar/eliminar jugador
    - Registrar/modificar/eliminar equipo
    - Consultar lista de equipos y jugadores

- Jugador
    - Consultar lista de equipos y jugadores
    - Consultar información del club
    - Consultar su perfil.

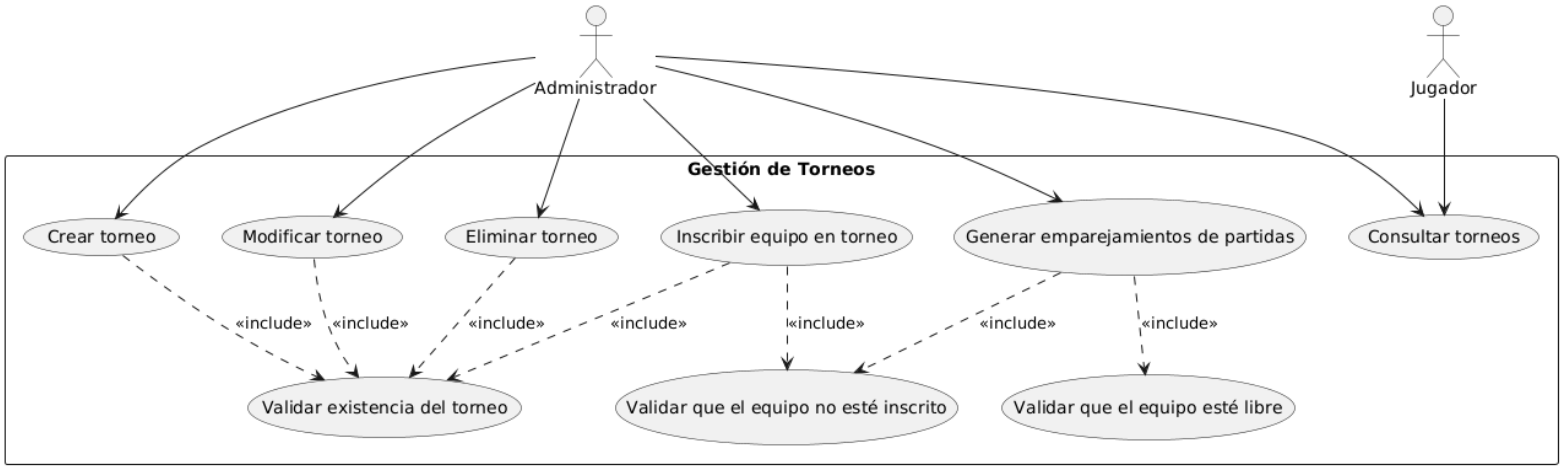


2. Gestión de torneos

Para la gestión de torneos hemos establecido los siguientes casos de uso por actor:

- Administrador:
    - Crear/modificar/eliminar torneo
    - Inscribir equipo en torneo
    - Generar emparejamientos de partidas
    - Consultar torneos

- Jugador
    - Consultar torneos



3. Gestión de partidas y resultados

Para la gestión de partidas y resultados hemos establecido los siguientes casos de uso por actor:

- Administrador:
    - Registrar resultado de partida
    - Modificar/eliminar resultado
    - Consultar resultados
    - Actualizar clasificación
    - Crear partida

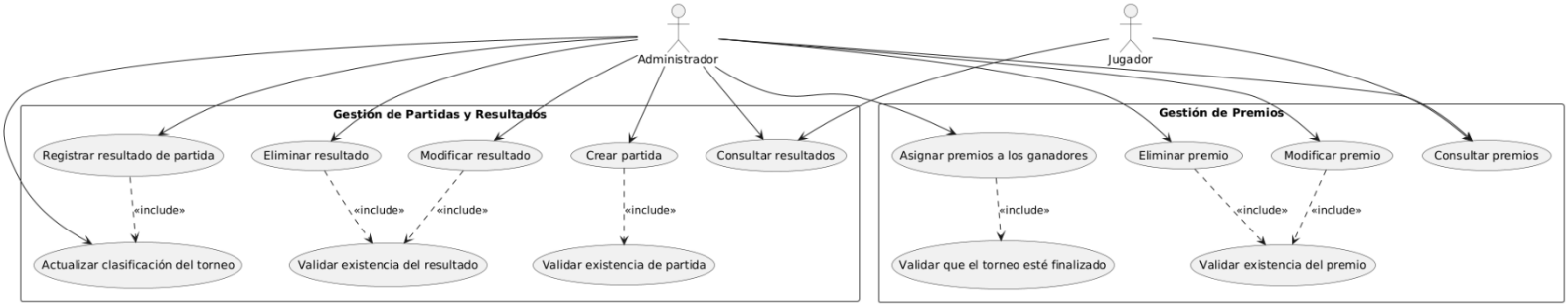
- Jugador
    - Consultar resultados

4. Gestión de premios

Para la gestión de premios hemos establecido los siguientes casos de uso por actor:

- Administrador:
    - Asignar premio a los ganadores
    - Modificar/eliminar premio
    - Consultar premios

- Jugador
    - Consultar premios



3. Identificación de clases y relaciones

A partir de los casos de uso establecidos, podemos identificar las siguientes clases con sus respectivos atributos y métodos:

- Clases de Entidad

Estas clases representan la información principal del sistema ya que modelan los elementos del dominio del problema.

- Equipo

Atributos	id, nombre, jugadores, partidas, premios
Métodos	Constructor, getters y setters, equals, hashCode, toString, calcularPuntosTotales()

- Jugador

Atributos	id, nombre, apodo, puntuacion
Métodos	Constructor, getters y setters, equals, hashCode, toString, validarNombre()

- Torneo

Atributos	id, nombre, equipos, partidas, premios
Métodos	Constructor, getters y setters, equals, hashCode, toString, cerrarTorneo()

- Partida

Atributos	equipo1, equipo2, resultadoEquipo1, resultadoEquipo2
Métodos	Constructor, getters y setters, equals, hashCode, toString

- Premio

Atributos	id, descripcion, valor
Métodos	Constructor, getters y setters, equals, hashCode, toString, asignarAEquipo()

- Clases de Control

Las clases del control en este sistema se encargan de la lógica del negocio y gestionan el flujo de trabajo entre las entidades del sistema y la interfaz. Los gestores agregan a las entidades.

- GestorEquipos

Atributos	List<Equipo>equipos, List<Jugador>jugadores
Funciones	Crear, modificar, eliminar equipos, gestionar jugadores, consulta

- GestorTorneos

Atributos	List<Torneo>torneos
Funciones	Crear, modificar, eliminar torneos, inscribir equipos, emparejamientos, consulta

- GestorPartidas

Atributos	List<Partida>partidas
Funciones	Registrar, modificar, eliminar resultados, crear partida, actualizar clasificación, consulta

- GestorPremios

Atributos	List<Premio>premios
Funciones	Asignar, modificar, eliminar premios, consulta

- Clases Interfaz

Definen los métodos que las vistas utilizarán para interactuar con el sistema y mostrar la información al usuario.

Las vistas implementan la interfaz ESports y a su vez dependen de los gestores para ejecutar las acciones del negocio y acceder a los datos de las entidades.

- ESports

Define todos los métodos visibles para administrador/general

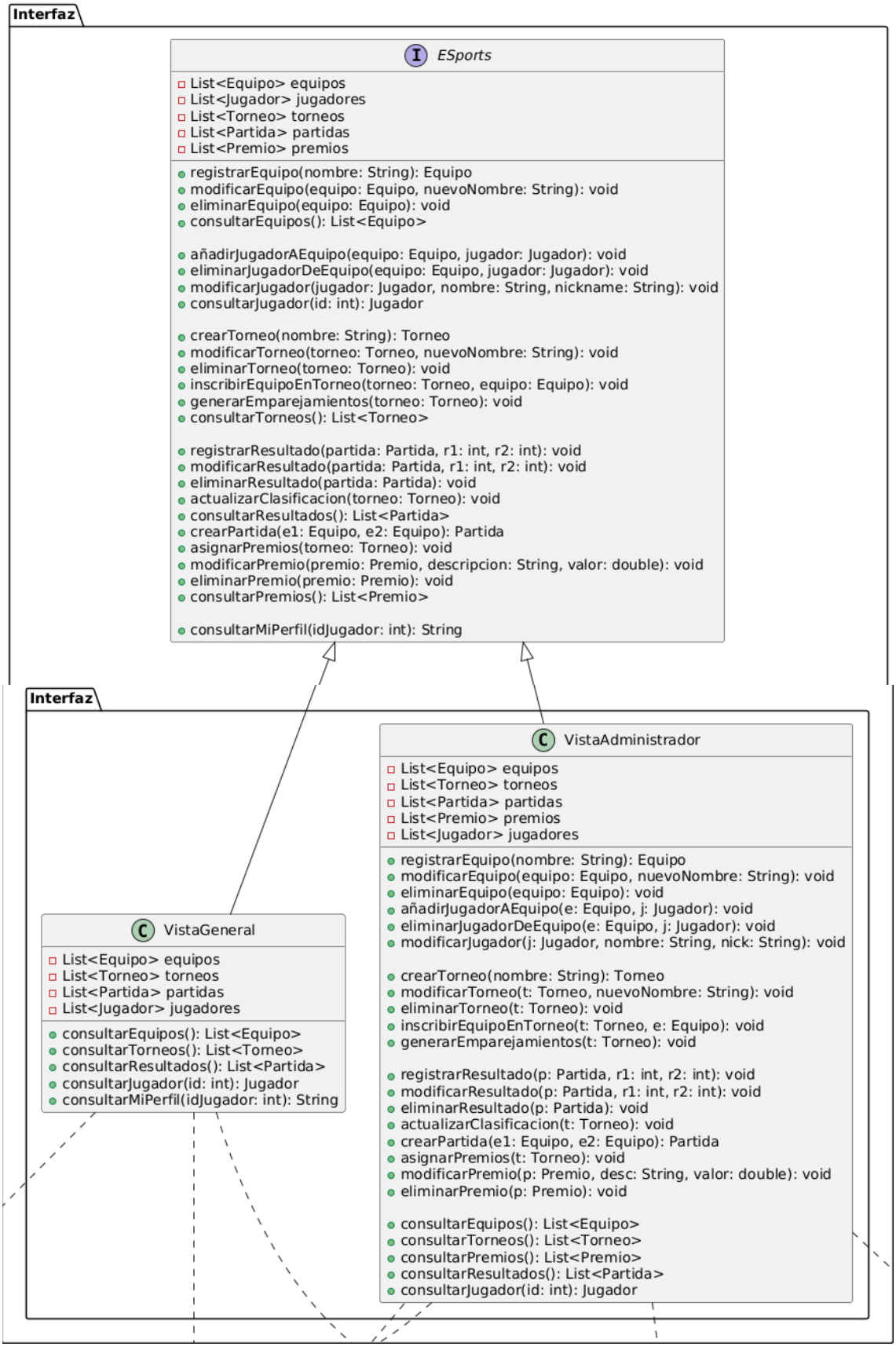
- VistaAdministrador

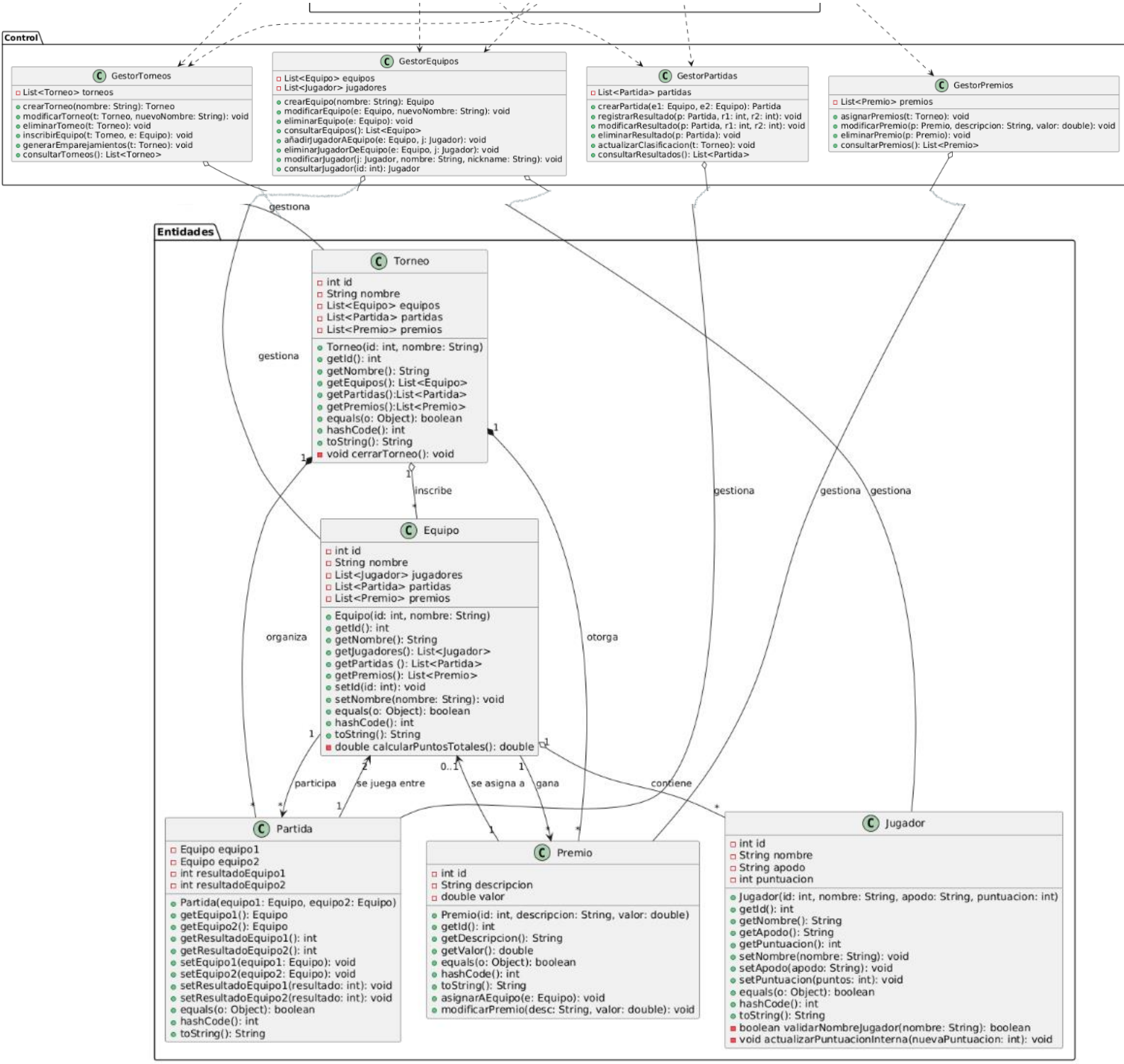
Utilizada para todos los casos de uso administrativos.

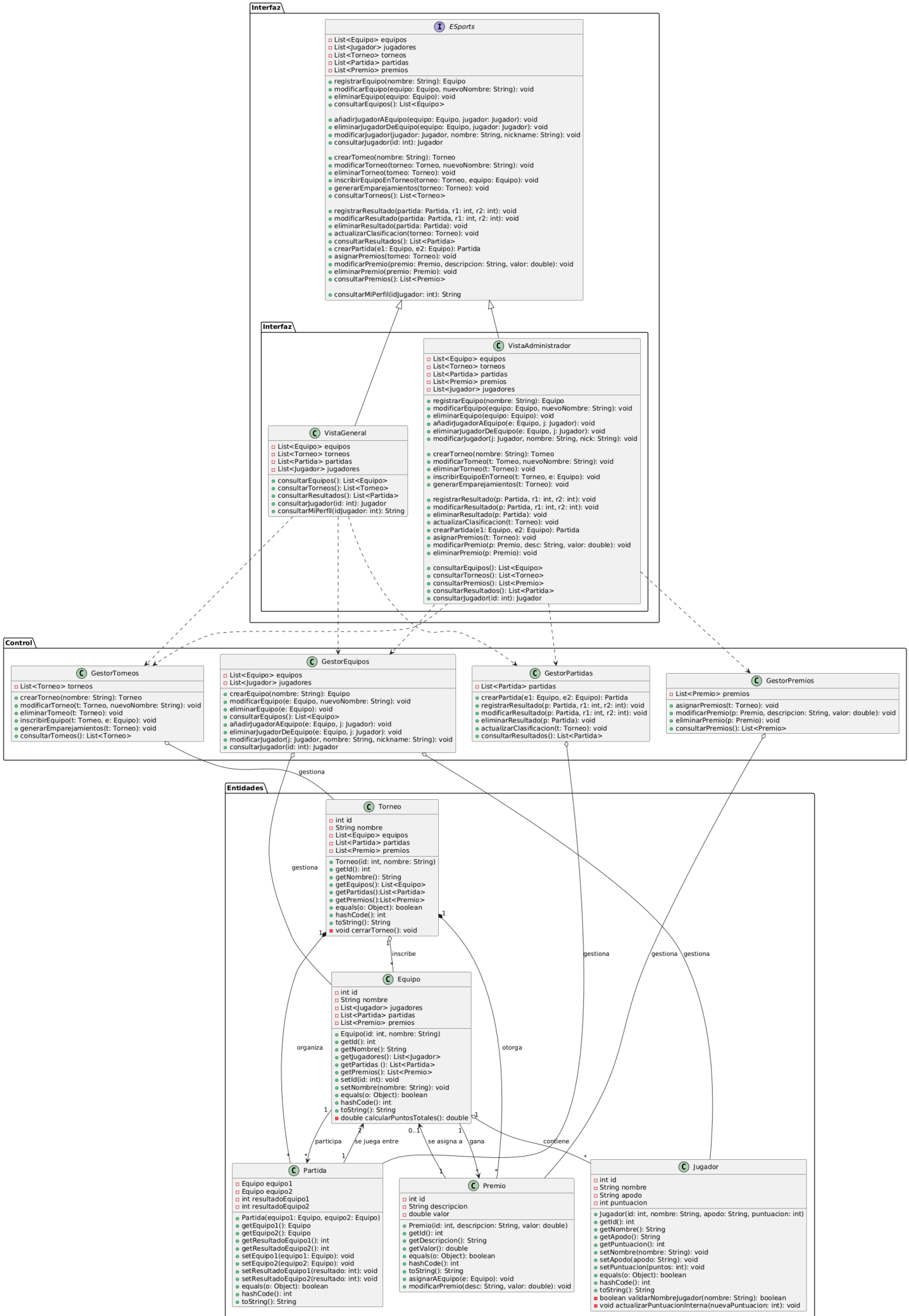
- VistaGeneral

Utilizada para consultas básicas para público.

4. Creación del diagrama de clases UML




Control
GestorTorneos
- List<Torneo> torneos
+ crearTorneo(nombre: String): Torneo
+ modificarTorneo(t: Torneo, nuevoNombre: String): void
+ eliminarTorneo(t: Torneo): void
+ inscribirEquipo(t: Torneo, e: Equipo): void
+ generarEmparejamientos(t: Torneo): void
+ consultarTorneos(): List<Torneo>
GestorEquipos
- List<Equipo> equipos
- List<Jugador> jugadores
+ crearEquipo(nombre: String): Equipo
+ modificarEquipo(e: Equipo, nuevoNombre: String): void
+ eliminarEquipo(e: Equipo): void
+ consultarEquipos(): List<Equipo>
+ añadirJugadorAEquipo(e: Equipo, j: Jugador): void
+ eliminarJugadorDeEquipo(e: Equipo, j: Jugador): void
+ modificarJugador(j: Jugador, nombre: String, nickname: String): void
+ consultarJugador(id: int): Jugador
GestorPartidas
- List<Partida> partidas
+ crearPartida(e1: Equipo, e2: Equipo): Partida
+ registrarResultado(p: Partida, r1: int, r2: int): void
+ modificarResultado(p: Partida, r1: int, r2: int): void
+ eliminarResultado(p: Partida): void
+ actualizarClasificacion(t: Torneo): void
+ consultarResultados(): List<Partida>
GestorPremios
- List<Premio> premios
+ asignarPremios(t: Torneo): void
+ modificarPremio(p: Premio, descripcion: String, valor: double): void
+ eliminarPremio(p: Premio): void
+ consultarPremios(): List<Premio>
gestiona
Entidades
Torneo
- int id
- String nombre
- List<Equipo> equipos
- List<Partida> partidas
- List<Premio> premios
+ Torneo(id: int, nombre: String)
+ getId(): int
+ getNombre(): String
+ getEquipos(): List<Equipo>
+ getPartidas():List<Partida>
+ getPremios():List<Premio>
+ equals(o: Object): boolean
+ hashCode(): int
+ toString(): String
- void cerrarTorneo(): void
gestiona
inscribe
Equipo
- int id
- String nombre
- List<Jugador> jugadores
- List<Partida> partidas
- List<Premio> premios
+ Equipo(id: int, nombre: String)
+ getId(): int
+ getNombre(): String
+ getJugadores(): List<Jugador>
+ getPartidas (): List<Partida>
+ getPremios(): List<Premio>
+ setId(id: int): void
+ setNombre(nombre: String): void
+ equals(o: Object): boolean
+ hashCode(): int
+ toString(): String
- double calcularPuntosTotales(): double
organiza
otorga
participa
se juega entre
se asigna a
gana
contiene
gestiona
gestiona
gestiona
Partida
- Equipo equipo1
- Equipo equipo2
- int resultadoEquipo1
- int resultadoEquipo2
+ Partida(equipo1: Equipo, equipo2: Equipo)
+ getEquipo1(): Equipo
+ getEquipo2(): Equipo
+ getResultadoEquipo1(): int
+ getResultadoEquipo2(): int
+ setEquipo1(equipo1: Equipo): void
+ setEquipo2(equipo2: Equipo): void
+ setResultadoEquipo1(resultado: int): void
+ setResultadoEquipo2(resultado: int): void
+ equals(o: Object): boolean
+ hashCode(): int
+ toString(): String
Premio
- int id
- String descripcion
- double valor
+ Premio(id: int, descripcion: String, valor: double)
+ getId(): int
+ getDescripcion(): String
+ getValor(): double
+ equals(o: Object): boolean
+ hashCode(): int
+ toString(): String
+ asignarAEquipo(e: Equipo): void
+ modificarPremio(desc: String, valor: double): void
Jugador
- int id
- String nombre
- String apodo
- int puntuacion
+ Jugador(id: int, nombre: String, apodo: String, puntuacion: int)
+ getId(): int
+ getNombre(): String
+ getApodo(): String
+ getPuntuacion(): int
+ setNombre(nombre: String): void
+ setApodo(apodo: String): void
+ setPuntuacion(puntos: int): void
+ equals(o: Object): boolean
+ hashCode(): int
+ toString(): String
- boolean validarNombreJugador(nombre: String): boolean
- void actualizarPuntuacionInterna(nuevaPuntuacion: int): void


Interfaz
ESports
- List<Equipo> equipos
- List<Jugador> jugadores
- List<Torneo> torneos
- List<Partida> partidas
- List<Premio> premios
+ registrarEquipo(nombre: String): Equipo
+ modificarEquipo(equipo: Equipo, nuevoNombre: String): void
+ eliminarEquipo(equipo: Equipo): void
+ consultarEquipos(): List<Equipo>
+ añadirJugadorAEquipo(equipo: Equipo, jugador: Jugador): void
+ eliminarJugadorDeEquipo(equipo: Equipo, jugador: Jugador): void
+ modificarJugador(jugador: Jugador, nombre: String, nickname: String): void
+ consultarJugador(id: int): Jugador
+ crearTorneo(nombre: String): Torneo
+ modificarTorneo(torneo: Torneo, nuevoNombre: String): void
+ eliminarTorneo(torneo: Torneo): void
+ inscribirEquipoEnTorneo(torneo: Torneo, equipo: Equipo): void
+ generarEmparejamientos(torneo: Torneo): void
+ consultarTorneos(): List<Torneo>
+ registrarResultado(partida: Partida, r1: int, r2: int): void
+ modificarResultado(partida: Partida, r1: int, r2: int): void
+ eliminarResultado(partida: Partida): void
+ actualizarClasificacion(torneo: Torneo): void
+ consultarResultados(): List<Partida>
+ crearPartida(e1: Equipo, e2: Equipo): Partida
+ asignarPremios(torneo: Torneo): void
+ modificarPremio(premio: Premio, descripcion: String, valor: double): void
+ eliminarPremio(premio: Premio): void
+ consultarPremios(): List<Premio>
+ consultarMiPerfil(idJugador: int): String
Interfaz
VistaGeneral
- List<Equipo> equipos
- List<Torneo> torneos
- List<Partida> partidas
- List<Jugador> jugadores
+ consultarEquipos(): List<Equipo>
+ consultarTorneos(): List<Torneo>
+ consultarResultados(): List<Partida>
+ consultarJugador(id: int): Jugador
+ consultarMiPerfil(idJugador: int): String
VistaAdministrador
- List<Equipo> equipos
- List<Torneo> torneos
- List<Partida> partidas
- List<Premio> premios
- List<Jugador> jugadores
+ registrarEquipo(nombre: String): Equipo
+ modificarEquipo(equipo: Equipo, nuevoNombre: String): void
+ eliminarEquipo(equipo: Equipo): void
+ añadirJugadorAEquipo(e: Equipo, j: Jugador): void
+ eliminarJugadorDeEquipo(e: Equipo, j: Jugador): void
+ modificarJugador(j: Jugador, nombre: String, nick: String): void
+ crearTorneo(nombre: String): Torneo
+ modificarTorneo(t: Torneo, nuevoNombre: String): void
+ eliminarTorneo(t: Torneo): void
+ inscribirEquipoEnTorneo(t: Torneo, e: Equipo): void
+ generarEmparejamientos(t: Torneo): void
+ registrarResultado(p: Partida, r1: int, r2: int): void
+ modificarResultado(p: Partida, r1: int, r2: int): void
+ eliminarResultado(p: Partida): void
+ actualizarClasificacion(t: Torneo): void
+ crearPartida(e1: Equipo, e2: Equipo): Partida
+ asignarPremios(t: Torneo): void
+ modificarPremio(p: Premio, desc: String, valor: double): void
+ eliminarPremio(p: Premio): void
+ consultarEquipos(): List<Equipo>
+ consultarTorneos(): List<Torneo>
+ consultarPremios(): List<Premio>
+ consultarResultados(): List<Partida>
+ consultarJugador(id: int): Jugador
Control
GestorTorneos
- List<Torneo> torneos
+ crearTorneo(nombre: String): Torneo
+ modificarTorneo(t: Torneo, nuevoNombre: String): void
+ eliminarTorneo(t: Torneo): void
+ inscribirEquipo(t: Torneo, e: Equipo): void
+ generarEmparejamientos(t: Torneo): void
+ consultarTorneos(): List<Torneo>
GestorEquipos
- List<Equipo> equipos
- List<Jugador> jugadores
+ crearEquipo(nombre: String): Equipo
+ modificarEquipo(e: Equipo, nuevoNombre: String): void
+ eliminarEquipo(e: Equipo): void
+ consultarEquipos(): List<Equipo>
+ añadirJugadorAEquipo(e: Equipo, j: Jugador): void
+ eliminarJugadorDeEquipo(e: Equipo, j: Jugador): void
+ modificarJugador(j: Jugador, nombre: String, nickname: String): void
+ consultarJugador(id: int): Jugador
GestorPartidas
- List<Partida> partidas
+ crearPartida(e1: Equipo, e2: Equipo): Partida
+ registrarResultado(p: Partida, r1: int, r2: int): void
+ modificarResultado(p: Partida, r1: int, r2: int): void
+ eliminarResultado(p: Partida): void
+ actualizarClasificacion(t: Torneo): void
+ consultarResultados(): List<Partida>
GestorPremios
- List<Premio> premios
+ asignarPremios(t: Torneo): void
+ modificarPremio(p: Premio, descripcion: String, valor: double): void
+ eliminarPremio(p: Premio): void
+ consultarPremios(): List<Premio>
gestiona
Entidades
Torneo
- int id
- String nombre
- List<Equipo> equipos
- List<Partida> partidas
- List<Premio> premios
+ Torneo(id: int, nombre: String)
+ getId(): int
+ getNombre(): String
+ getEquipos(): List<Equipo>
+ getPartidas():List<Partida>
+ getPremios():List<Premio>
+ equals(o: Object): boolean
+ hashCode(): int
+ toString(): String
- void cerrarTorneo(): void
gestiona
inscribe
Equipo
- int id
- String nombre
- List<Jugador> jugadores
- List<Partida> partidas
- List<Premio> premios
+ Equipo(id: int, nombre: String)
+ getId(): int
+ getNombre(): String
+ getJugadores(): List<Jugador>
+ getPartidas (): List<Partida>
+ getPremios(): List<Premio>
+ setId(id: int): void
+ setNombre(nombre: String): void
+ equals(o: Object): boolean
+ hashCode(): int
+ toString(): String
- double calcularPuntosTotales(): double
organiza
otorga
participa
se juega entre
se asigna a
gana
contiene
Partida
- Equipo equipo1
- Equipo equipo2
- int resultadoEquipo1
- int resultadoEquipo2
+ Partida(equipo1: Equipo, equipo2: Equipo)
+ getEquipo1(): Equipo
+ getEquipo2(): Equipo
+ getResultadoEquipo1(): int
+ getResultadoEquipo2(): int
+ setEquipo1(equipo1: Equipo): void
+ setEquipo2(equipo2: Equipo): void
+ setResultadoEquipo1(resultado: int): void
+ setResultadoEquipo2(resultado: int): void
+ equals(o: Object): boolean
+ hashCode(): int
+ toString(): String
Premio
- int id
- String descripcion
- double valor
+ Premio(id: int, descripcion: String, valor: double)
+ getId(): int
+ getDescripcion(): String
+ getValor(): double
+ equals(o: Object): boolean
+ hashCode(): int
+ toString(): String
+ asignarAEquipo(e: Equipo): void
+ modificarPremio(desc: String, valor: double): void
Jugador
- int id
- String nombre
- String apodo
- int puntuacion
+ Jugador(id: int, nombre: String, apodo: String, puntuacion: int)
+ getId(): int
+ getNombre(): String
+ getApodo(): String
+ getPuntuacion(): int
+ setNombre(nombre: String): void
+ setApodo(apodo: String): void
+ setPuntuacion(puntos: int): void
+ equals(o: Object): boolean
+ hashCode(): int
+ toString(): String
- boolean validarNombreJugador(nombre: String): boolean
- void actualizarPuntuacionInterna(nuevaPuntuacion: int): void