1. Análisis del problema y requisitos del sistema

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

Usuario: puede ver toda la información sobre equipos, jugadores, torneos, partidas y premios.
Administrador: puede hacer todo lo del usuario y además, gestionar equipos, jugadores, torneos, partidas y premios.

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

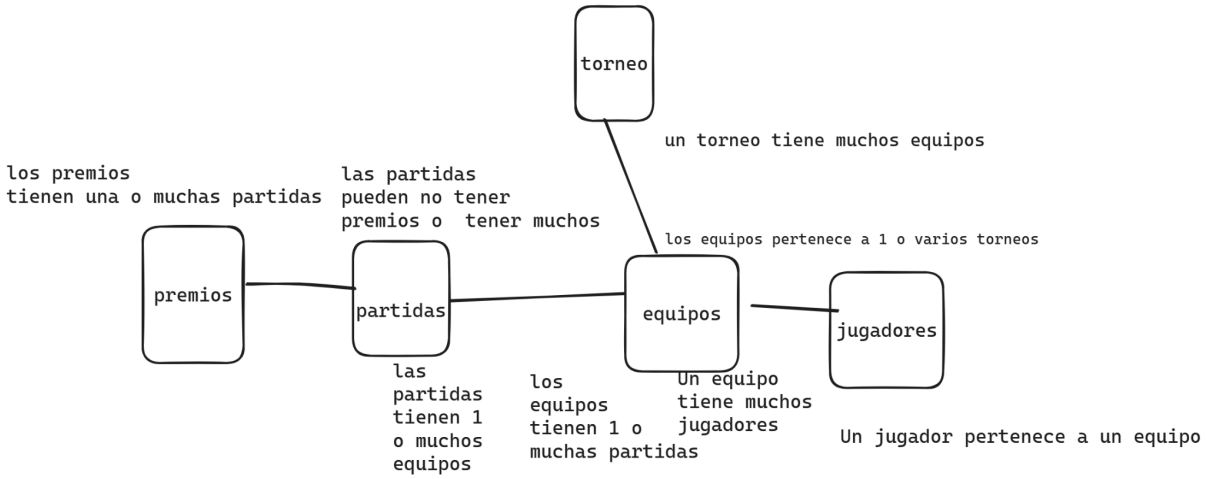
Administrador:

- ★ Registrar equipos.
- ★ Añadir jugadores a equipos.
- ★ Consultar equipos y jugadores.
- ★ Crear torneos.
- ★ Inscribir equipos en torneos.
- ★ Generar emparejamientos.
- ★ Registrar resultados de partidas.
- ★ Actualizar clasificaciones de torneos.
- ★ Asignar premios a ganadores.

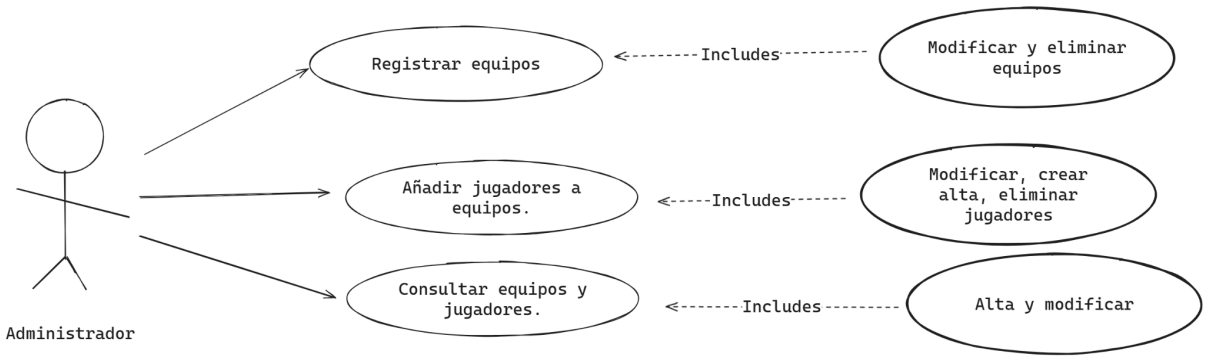
Usuarios:

- ★ Ver toda la información

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



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



3. Identificación de clases y relaciones

a) Identifica las clases principales en función de los casos de uso seleccionados.

Las clases principales son:

- ★ **Equipo**
- ★ **Jugador**

b) Distingue las clases de Entidad, Control e Interfaz para mantener una arquitectura modular

Clases de Entidad:

- ★ **Equipo**
- ★ **Jugador**

c) Define atributos y métodos para cada clase.

Clase Equipo

Atributos:

- idEquipo : int
- nombre : String
- listaJugadores : List<Jugador>

Métodos:

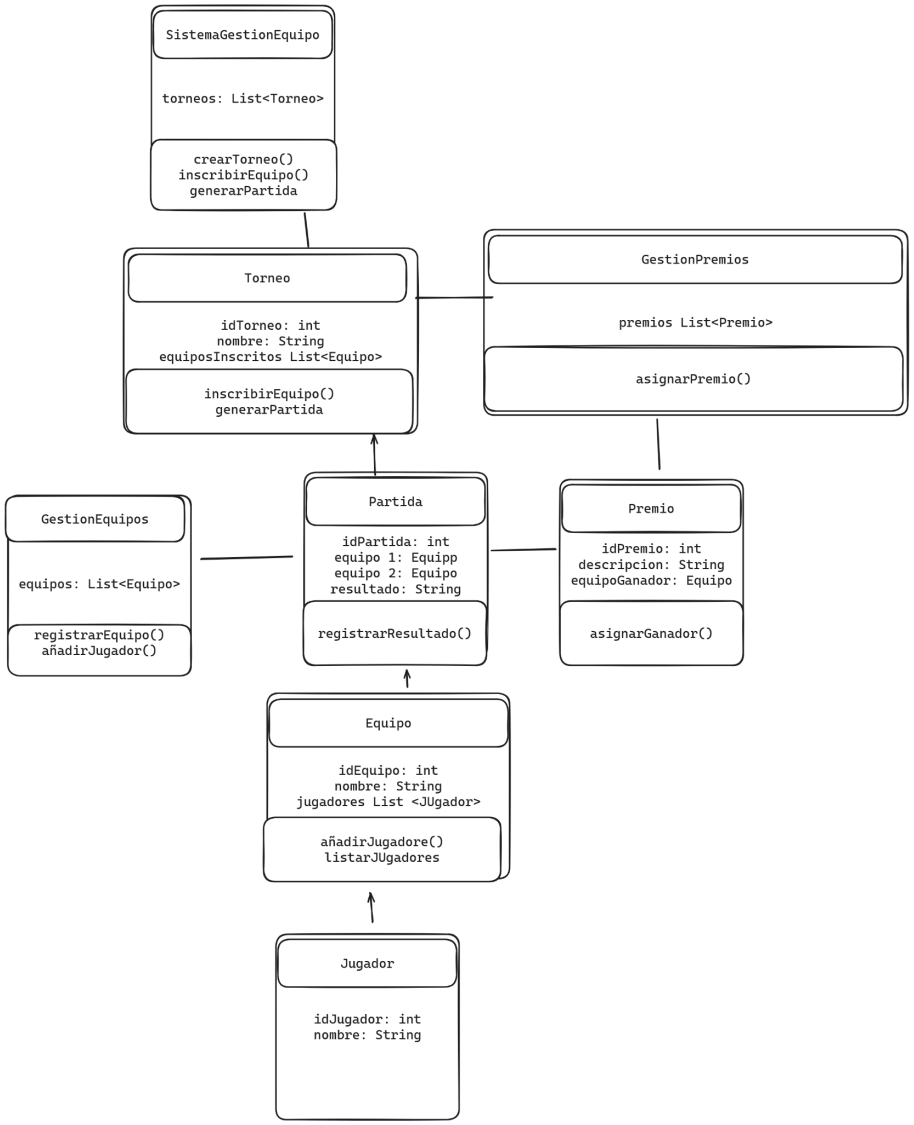
- añadirJugador(jugador: Jugador) : void

Clase Jugador

Atributos:

- idJugador : int
- nombre : String
- posición : String
- equipo : String

4. Establece relaciones entre clases, asegurando la correcta representación de asociaciones, agregaciones y composiciones.



Un equipo se compone de jugadores y una partida se compone de equipos, que a su vez componen un torneo. Los equipos sin jugadores no existen, las partidas sin equipos tampoco y sin equipos no hay torneo.