# **1. Análisis del problema y requisitos del sistema**

**Sistema**: Gestión de Torneos de eSports

**¿Quiénes son los actores?**

* **Administrador** (gestiona equipos, jugadores, torneos).
* **Jugador** (forma parte de un equipo).
* **Sistema** (automatiza emparejamientos, clasificación, premios).

**¿Qué acciones puede hacer cada actor?**

* Administrador:  
  + Registrar un equipo.
  + Añadir jugadores a un equipo.
  + Consultar la lista de equipos y jugadores.

**¿Relaciones principales?**

* Un **equipo** puede tener varios **jugadores**.
* Un **jugador** pertenece a un único **equipo**.

**2. Diagrama de Casos de Uso**

**los casos de uso serían:**

Registrar equipo (Administrador)

Añadir jugador a equipo (Administrador)

Consultar lista de equipos y jugadores (Administrador)

**Con relaciones tipo:**

Registrar equipo <<include>> Añadir jugador (porque al crear un equipo podría ser obligatorio añadir jugadores).

# **3. Identificación de Clases y Relaciones**

**Claes:**

* **Entidad:**
  + **Clase Equipo:**
    - **Atributos principales: nombre (String), listaJugadores (List<Jugador>)**
    - **Métodos principales: agregarJugador(Jugador)**
  + **Clase Jugador:**
    - **Atributos principales: nombre (String), nickname (String), edad (int)**
    - **Métodos principales: No tiene métodos principales definidos.**
* **Control:**
  + **Clase GestorEquipos:**
    - **Atributos principales: listaEquipos (List<Equipo>)**
    - **Métodos principales:**
      * **registrarEquipo(String)**
      * **añadirJugadorAEquipo(String, Jugador)**
      * **consultarEquipos()**
* **Interfaz:**
  + **Clase VistaConsola:**
    - **Atributos principales: No tiene.**
    - **Métodos principales:**
      * **mostrarMenu()**
      * **mostrarEquipos()**

**Relaciones:**

* **Equipo tiene una lista de Jugador (Composición).**
* **GestorEquipos gestiona varios Equipo (Agregación).**

**4. Diagrama de Clases UML**