# AD3\_UML

## Análisis del problema y requisitos del sistema

Comenzamos a analizar los usos y aplicaciones de un sistema gestor de torneos de eSports. El enfoque es el de una aplicación con la que se organizan, automatizan y se gestionan las características de un torneo de deportes online. Con ella interactúan tanto personas como otros sistemas.

## Actores

Todas las personas que hacen uso de la aplicación de una u otra manera se catalogan como **usuarios**. Estos pueden darse de alta, iniciar/cerrar sesión y gestionar la configuración de su perfil. Los usuarios, según el papel que desempeñen una vez dentro del sistema, tienen diferentes roles: estos pueden ser principalmente jugadores, administradores de torneos o espectadores.

* Los jugadores son usuarios que participan en los torneos de forma individual o como parte de un **equipo** en función del deporte y la categoría. Por tanto, entre las acciones que realiza un jugador están: configurar su perfil de jugador determinando el deporte, la categoría u otros parámetros importantes, unirse a un equipo y consultar resultados.
* Los **administradores** de torneos, por su parte, se encargan de crear torneos, gestionar las inscripciones, actualizar los resultados y establecer las reglas específicas del torneo.
* Además, los usuarios con rol de **espectador** pueden ver estadísticas y resultados, además de los esquemas de emparejamientos entre equipos, es decir, los brackets.
* Cada equipo debe tener un usuario con rol de **manager**, para que pueda registrar a su equipo en los torneos, componer un equipo, gestionar a los jugadores y ver los resultados y estadísticas de este.

Además de los usuarios, puede haber otros sistemas interactuando con la propia aplicación que mencionamos. En este caso, considerando que un sistema como estos puede ser de pago o gratuito, pero admitir suscripciones, determinamos que una **plataforma de pago** puede ser un actor. En este caso, la plataforma de pago gestiona suscripciones, y puede realizar pagos en caso de que haya un sistema de premios.

Para visualizar los partidos o torneos y notificar el calendario de un torneo puede haber una **plataforma de streaming** conectada al sistema (como Twitch).

## Acciones que cada actor puede realizar

A continuación, un breve esquema:

* User
  + signUp()
  + logIn()
  + profileSettings()
  + findTeam()
  + findPlayer()
  + findTournament()
  + checkStatistics()
* Player
  + joinTeam()
  + joinTournament()
  + joinGame() //si es un partido individual
* Team
  + registerTeam()
  + addToTournament()
  + manageLineUp()
  + addPlayer()
* Admin
  + createTournament()
  + bracketManagement()
  + resultsUpdate()
  + closeTournamnet()
  + disputeManagement()
* Team Manager
  + registerTeam()
  + addToTournament()
  + manageLineUp()
  + addPlayer()
* Viewer
  + subscription()
  + watchMatch()
  + followTeam()
  + followPlayer()
* payment Platform
  + subscriptionManagement()
  + payOutPrices()
* streaming platform
  + liveShow()
  + calendarShow()

## Entidades y relaciones entre sí

En cuanto a las entidades del sistema, distinguimos varias que almacenarán datos o información con las que el sistema trabaja. Inicialmente, distinguimos las siguientes:

* Jugador
* Torneo
* Partido
* Equipo
* Manager
* Admin
* Bracket
* Resultado

Antes de profundizar en el diagrama, podemos observar varias relaciones entre las entidades principales:

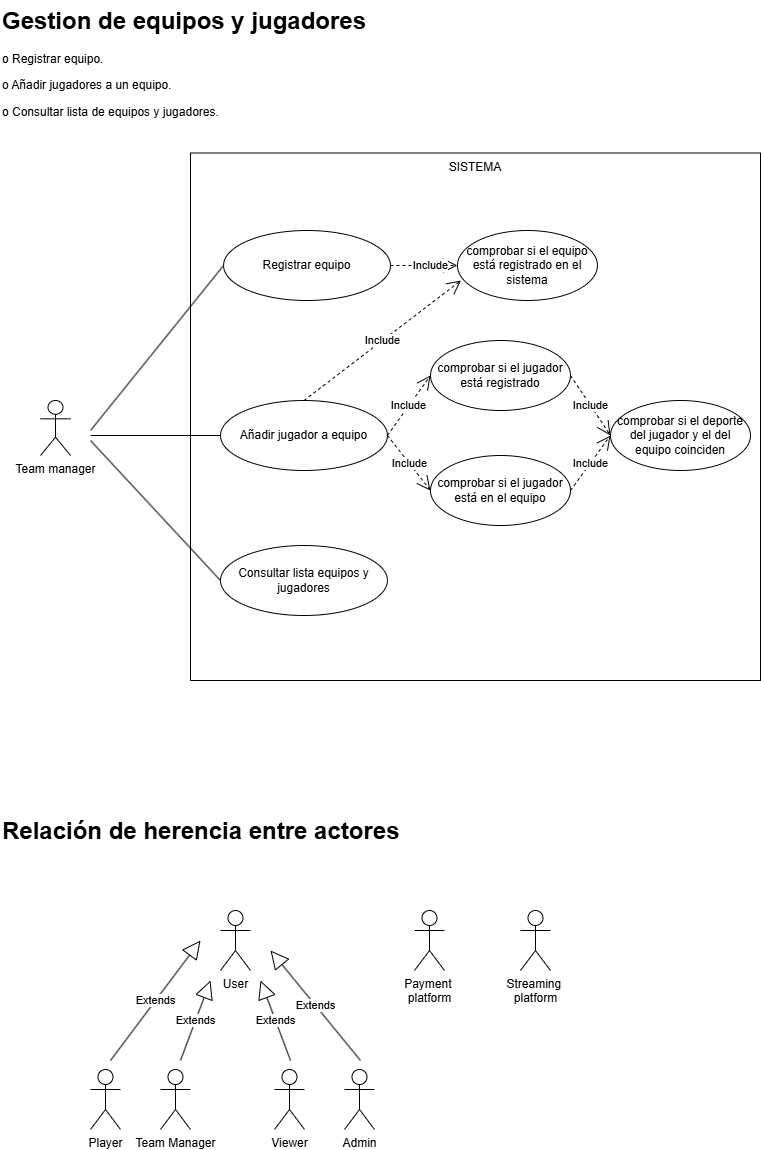
* Todos los actores individuales son en realidad usuarios con atributos (nombre, Nick, email, etc.). Por lo que establecemos usuario como una clase abstracta, fuera de las entidades internas del sistema.
* Los jugadores heredan atributos y métodos de usuario. Además, tendrán atributos propios como “deporte”, “ranking”, “estadísticas”, etc. Un jugador puede participar en uno o más torneos y pertenecer a 0 o 1 equipo.
* Un torneo es organizado por 1 administrador. Un torneo contiene n partidas, 1 bracket, n jugadores, n equipos, n managers y n resultados.
* En un partido participan n jugadores, n equipos, n managers, hay 1 admin y 1 resultado.
* Un equipo contiene n jugadores, 1 manager, puede participar en 1 o n torneos y jugar en 1 o n partidos, así como recibir 1 o n resultados.
* Un manager gestiona 1 equipo, n jugadores, juega en n torneos y n partidos.
* Un administrador gestiona 1 o varios torneos, recibe y actualiza los Brackets y los resultados.
* Un bracket contiene varios equipos y varios partidos.
* Un resultado es común a varios equipos y partidos.

## Identificación de los casos de uso y elaboración del diagrama

### Gestión de equipos y jugadores

* **Registrar equipo**: esta interacción la realiza el manager con el sistema. Para este caso de uso, debemos “**comprobar si el equipo ya está registrado en el sistema**”.
* **Añadir jugadores a un equipo**: el manager del equipo incluye jugadores en su equipo. Para añadir jugadores a un equipo, tenemos que comprobar si el equipo al que vamos a añadir el jugador ya está registrado así como si el propio **jugador ya está registrado** o si **el jugador ya está en el equipo** dentro del sistema.
* Consultar listas de equipos y jugadores: este último caso de uso puede enfocarse desde el punto de vista de cualquier tipo de usuario, sin embargo, nos enfocamos en los casos de uso desde el punto de vista del usuario “team manager”.

A continuación, el diagrama de casos de uso para la gestión de equipos y jugadores:



## Identificación de clases y relaciones

Las clases principales según los casos de uso mencionados arriba (gestión de equipos y jugadores) serían:

* Equipo
* Jugador
* Team manager

Estas tres clases se identifican como entidades que representan objetos del mundo real con datos que encapsulan la lógica de negocio.

Las relaciones entre las tres entidades serían las siguientes:

* **Equipo – Jugador**: Es una relación de composición, un jugador forma parte de un equipo y si el equipo desapareciera, los jugadores perderían su significado. Un equipo contiene 1-n jugadores, mientras que 1 jugador pertenece a 1 solo equipo.
* **Equipo – team manager**: Relación de agregación. Un entrenador está asociado a un equipo, pero podría existir fuera del equipo y gestionar otro. Similar a la composición, pero más débil. Un equipo tiene 0 o 1 team manager, al mismo tiempo, un entrenador puede estar en 0 o 1 team manager.
* **Jugador – team manager**: relación de asociación. Ambos están relacionados a través del equipo

En el punto 1 hemos identificado las entidades posibles y qué acciones pueden realizar

## Creación del diagrama de clases UML

Dado que aún no podemos contemplar las clases a todo detalle, especificamos con puntos suspensivos que podría haber más atributos y métodos.

La clase usuario es una clase abstracta cuyos atributos están protegidos. Las clases Player y Team Manager tienen atributos propios y heredan de User. Tanto las cardinalidades como las

