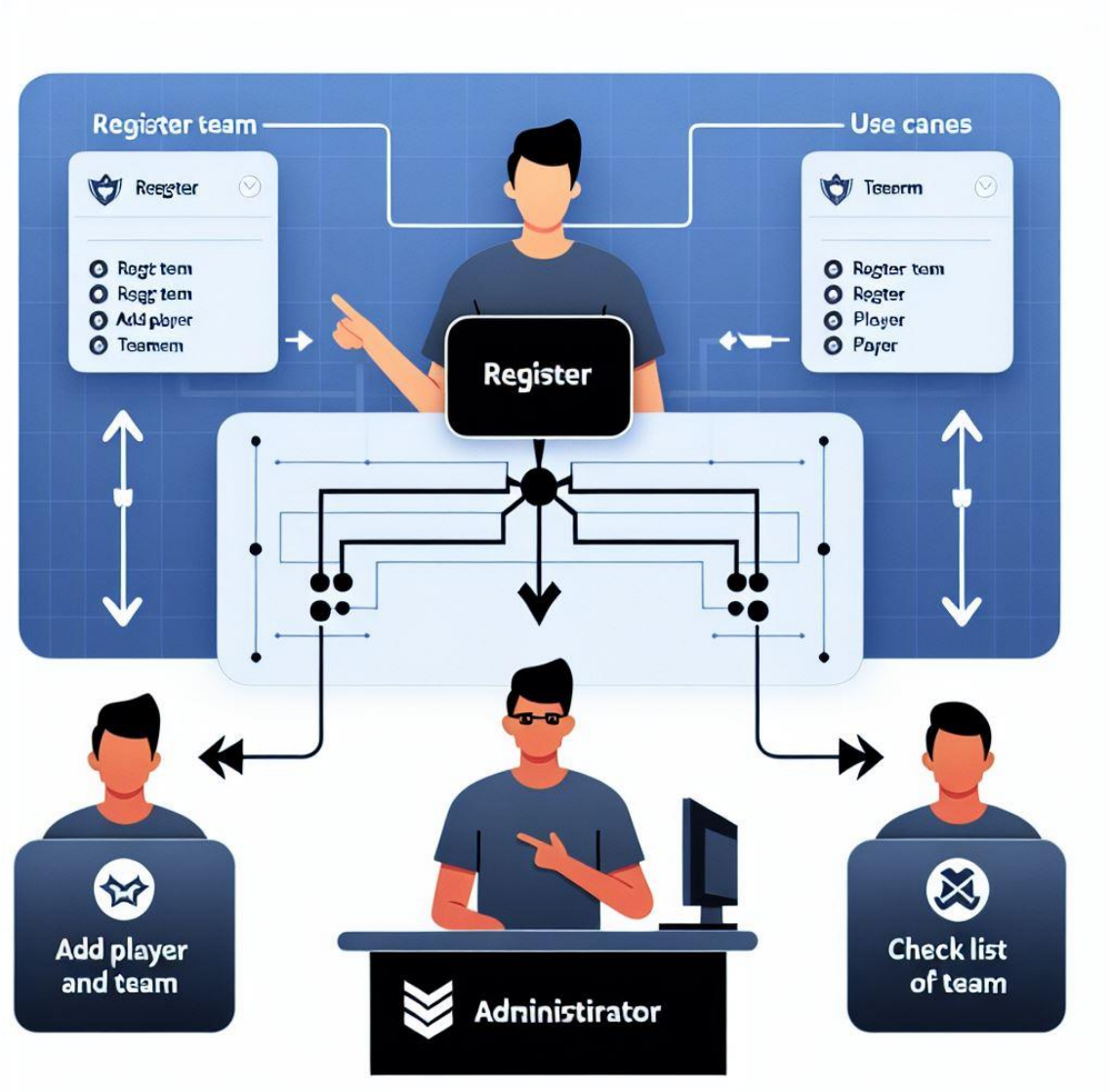
AD-3: Diagramas UML - Sistema de Gestión de Torneos de eSports



Asignatura	Datos del alumno
ENTORNOS DE DESARROLLO	**Apellidos:** *Veloso Ferreiro*
AD-3: Diagramas UM	**Nombre:** *José Carlos*

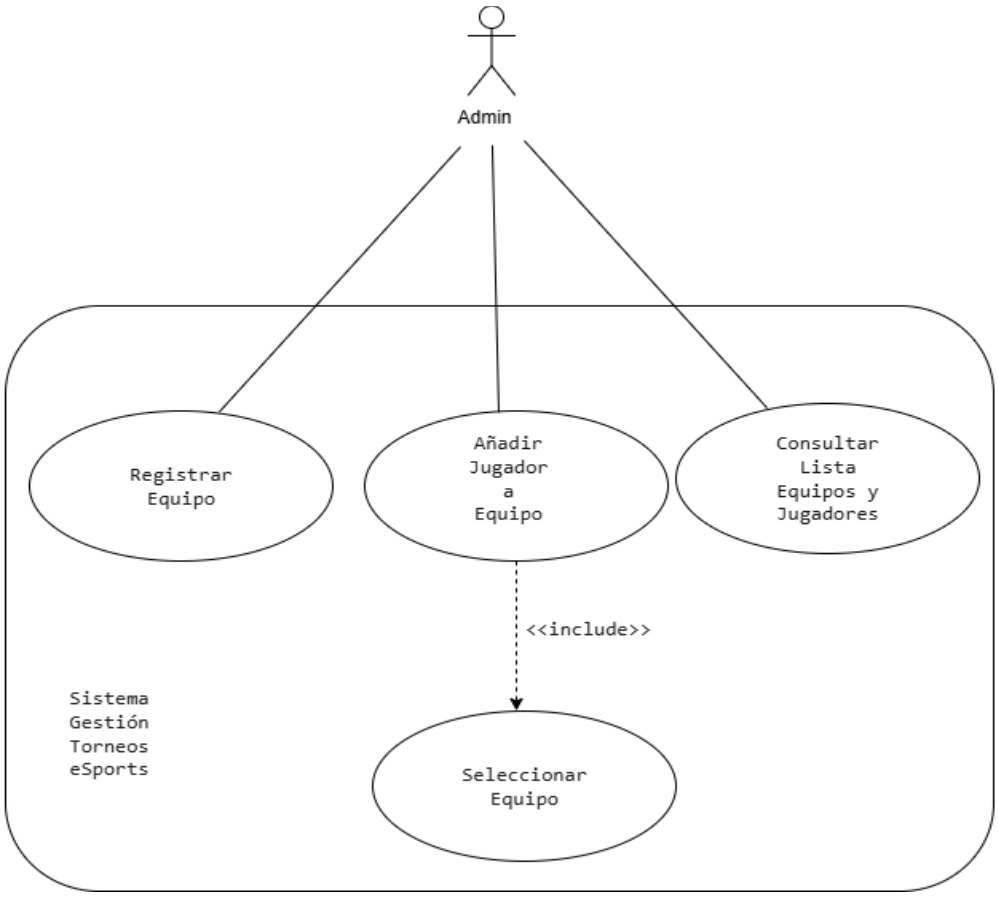
1. Análisis del problema y requisitos del sistema

El objetivo principal es diseñar un **sistema de gestión de torneos de eSports** utilizando UML para modelar su estructura (diagrama de clases) y funcionalidad (diagrama de casos de uso). Aunque el proyecto completo incluye la implementación en Java, **nos centraremos exclusivamente en los pasos de modelado UML requeridos para la entrega**, que según el documento

Los casos de uso modelados son:

- Registrar Equipo
- Añadir Jugador a Equipo
- Consultar Lista Equipos y Jugadores

Se incluye el caso de uso Seleccionar Equipo como una funcionalidad obligatoria (<<include>>) invocada por Añadir Jugador a Equipo, ya que siempre es necesario identificar el equipo antes de añadirle un jugador.



3. Identificación de Clases y Relaciones

1. Clases Principales:

- **Entidades:**
    - Equipo
    - Jugador
- **Control:**
    - GestorEquipos
- **Interfaz:**
    - VistaAdmin

2. Atributos y Métodos Esenciales:

- **Jugador (Entidad):**
    - Atributos: - nickname: String (Dato identificativo)
    - Métodos: + getNickname(): String (Para obtenerlo)
- **Equipo (Entidad):**
    - Atributos: - nombre: String, - jugadores: List<Jugador> (Es clave la relación)
    - Métodos: + getNombre(): String, + agregarJugador(jugador: Jugador): void, + getJugadores(): List<Jugador>
- **GestorEquipos (Control):**
    - Métodos (Directamente relacionados con los casos de uso):
        - + registrarEquipo(nombre: String): void
        - + añadirJugadorAEquipo(nombreEquipo: String, nicknameJugador: String): void
        - + consultarEquiposYJugadores(): String (Devuelve una representación simple, ej. texto)
        - + buscarEquipo(nombreEquipo: String): Equipo (Necesario para Seleccionar Equipo y Añadir Jugador)

        - + buscarJugador(nickname: String): Jugador (Necesario para Añadir Jugador)

- **VistaAdmin (Interfaz):**
    - Atributos: - gestor: GestorEquipos (Necesita la referencia al control)
    - Métodos (Solo para iniciar acciones):
        - + accionRegistrarEquipo(): void (Internamente llama al gestor)
        - + accionAgregarJugador(): void (Internamente llama al gestor)
        - + accionConsultarListas(): void (Internamente llama al gestor)

3. Relaciones Principales :

- **VistaAdmin ---> GestorEquipos (Dependencia/Uso):** La vista usa el gestor.
- **GestorEquipos ---> Equipo (Dependencia/Uso):** El gestor maneja equipos.
- **GestorEquipos ---> Jugador (Dependencia/Uso):** El gestor maneja jugadores (para buscarlos/referenciarlos).
- **Equipo o---- Jugador (Agregación):** Un equipo tiene jugadores. Especificaremos cardinalidad: 1 (Equipo) a 0..* (Jugadores) - Un equipo existe, puede tener 0 o más jugadores. Un jugador puede existir sin equipo.

4: Creación del diagrama de clases UML.