**Documento de Diseño**

**Elementos relacionados al dominio del problema**

* Usuario
* Equipo Fantasía
* Equipo Real
* Partido
* Jugador
* Usuario

**Elementos relacionados con lo que hace el sistema**

* Buscar Usuarios
* Buscar Equipos
* Registrar datos de los partidos
* Asignar precios a los jugadores
* Aumentar la puntuación del jugador
* Clasificar equipos fantasía

**Cosas que fluyen dentro del Sistema**

* Cargar datos de equipo
* Validación del usuario

**Agrupadores**

* Temporada

**Estereotipos**

* Equipo Real <Estructurador>
* Usuario<Coordinador>
* Cargar Datos<Estructurador>
* Jugador<Controlador>
* Partido<Controlador>
* Equipo Fantasía<Controlador>
* Temporada<Controlador>

|  |  |  |
| --- | --- | --- |
| N° | Responsabilidad | Componente |
| 1 | Cargar Usuarios | Cargar Datos |
| 2 | Cargar Temporada | Cargar Datos |
| 3 | Cargar Partidos Jugados | Cargar Datos |
| 4 | Cargar Equipos | Cargar Datos |
| 5 | Validar un usuario | Usuario |
| 6 | Registrar un nuevo usuario | Usuario |
| 7 | Crear nuevo equipo | Equipo de Fantasía |
| 8 | Comprar Jugador | Equipo de Fantasía |
| 9 | Vender Jugador | Equipo de Fantasía |
| 10 | Cambiar alineación | Equipo de Fantasía |
| 11 | Registrar partidos | Partido Jugado |
| 12 | Sumar puntos | Equipo Fantasía / Jugador |
| 13 | Dar ranking de mejores equipos fantasía según su puntuación | Temporada |
| 14 | Dar ranking de mejores jugadores según la puntuación de cada jugador | Temporada |
| 15 | Dar ranking de los jugadores con menor puntuación | Temporada |
| 16 | Retornar mejor equipo fantasía | Temporada |

Con lo anterior podemos comenzar a definir nuestras clases, atributos y métodos

**Clases**

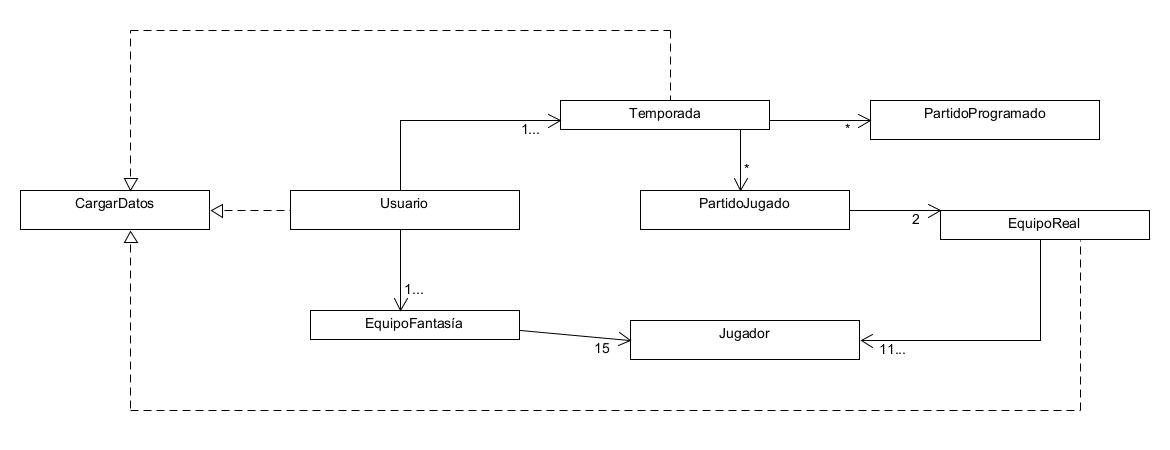
* CargarDatos
* Usuario
* EquipoFantasía
* PartidoJugado
* Jugador
* Temporada
* PartidoProgramado
* EquipoReal

**Atributos de cada Clase**

* **CargarDatos**
* “Por el momento no tendrá ningún atributo, no es necesario que los tenga, solamente tendrá métodos que se encargar de leer los datos de la temporada, equipos, jugadores y usuarios”
* **Usuarios**
* nombreUsuario: String
* contraseña: String
* administrador: boolean “Indicara si el usuario es administrador o no lo es, en tal caso de serlo tendrá la posibilidad de registrar los resultados de cada partido”
* **EquipoFantasia**
* nombre: String
* presupuesto: double
* puntosEquipo: int
* alineación: ArrayList<Jugador> “Es la alineación del equipo”
* **PartidoJugado**
* fecha: String
* resultado: String “Por el momento se manejará como un String, es posible que en la implementación de la aplicación esto cambie”
* **Temporada**
* **“**No tendrá atributos inicialmente**”**
* **EquipoReal**
* nombreEquipo: String
* **PartidoProgramado**
* fecha: String
* local: String
* visitante: String

**Relaciones entre las clases**

Para esto usuraremos un diagrama de clases de alto nivel para entender mejor como se van a relacionar las clases

 Figura 1. Diagrama de clases de alto nivel

**Métodos de cada clase**

* **CargarDatos**
* cargarUsuarios
* cargarTemporada
* cargarPartidosJugados
* cargarEquipos
* **Usuarios**
* registrarNuevoUsuario : void
* **EquipoFantasia**
* crearNuevoEquipo: void
* comprarJugador: void
* venderJugador: void
* configurarAlineación: void
* **PartidoJugado**
* registrarResultados: void
* **Temporada**
* rankingEquipoFantasia: ArrayList<EquipoFantasia>
* rankingJugadoresMasPuntos: ArrayList<Jugador>
* rankingJugadoresMenosPuntos: ArrayList<Jugador>
* mejorEquipoFantasiaPorFecha: EquipoFantasia
* **EquipoReal**
* “No tiene métodos”
* **PartidoProgramado**
* “No tiene métodos”

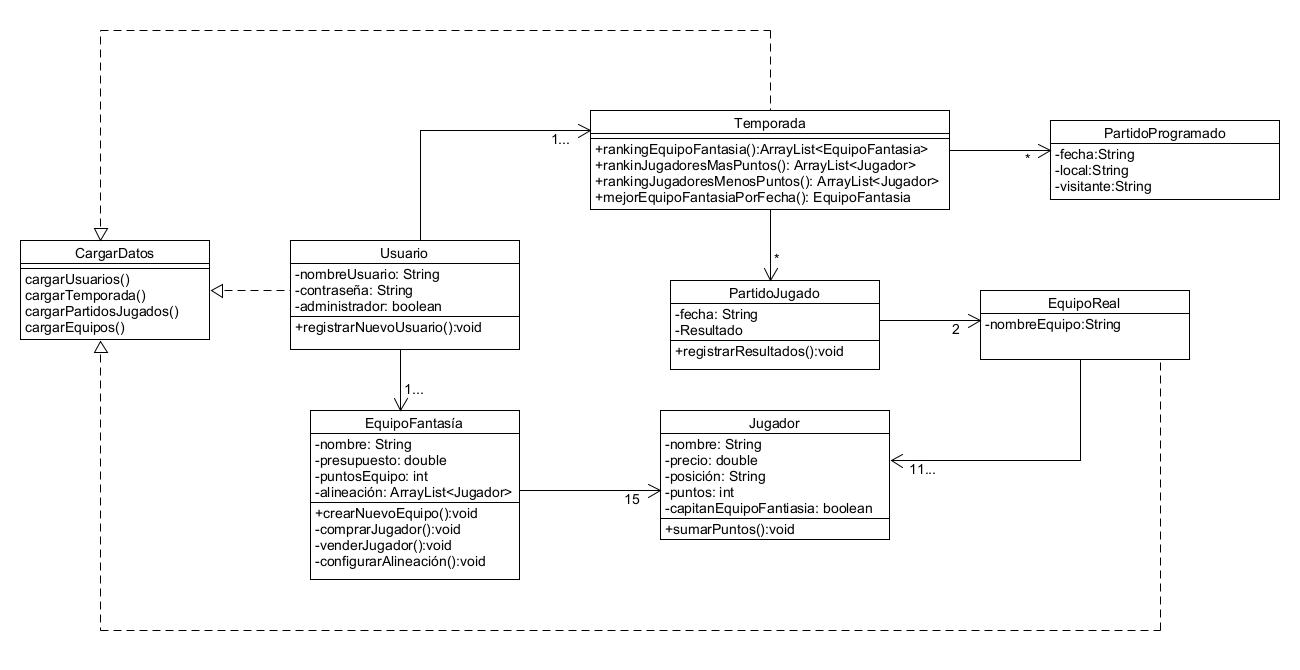
Con lo anterior ya podemos crear el diagrama UML con sus clases, atributos, métodos y relaciones:

Figura2. Diagrama UML