# TOMA DE DECISIONES

Empezaremos trabajando con los modelos.

Los partidos se componen de dos jugadores que estarán introducidos en un array de jugadores en vez de tener dos atributos separados. Además, contarán con un arbitro registrado que será el único que podrá *cantar* los puntos en ese partido. Por último, para contar los puntos tendrá un marcador que se compone de los sets que ha ganado cada jugador y el juego actual que se está disputando. Puede haber dos tipos de juegos, si se está en un juego regular se contarán los puntos como en un partido normal de tenis gracias a un enum (0,15,30…) pero, si están en un tiebrake se contarán cómo los números ordinarios(0,1,2,…,6). Además, no se podrá actualizar los partidos una vez hayan terminado aunque, lo intente el propio árbitro que lo ha registrado

El siguiente modelo a discutir será el de los usuarios que están registrados en este sistema, los árbitros que contarán con su nombre y su contraseña. Además, al iniciar sesión se tendrán que introducir el resto de datos (email y apellido). Por otro lado, los jugadores tendrán simplemente el nombre y la edad.

El último grupo de modelos que hay que mencionar serán los repositorios que los jugadores, árbitros y partidos tendrán uno separado que contará con un id único para así poder encontrarlos.

Continuaremos con los controladores.

Los controladores serán servicios que se usarán para realizar las peticiones de los comandos introducidos en la terminal. Los servicios de los partidos se han usado como variables locales la interfaz de los partidos y de los dos usuarios (jugadores y árbitros). Después, está los jugadores que en su servicio se podrá crear y mostrar los jugadores de diferentes maneras, aún así, en el comando de crear los jugadores solo se daba el nombre y como te pedían también la fecha, he creado las dos funcionalidades. Al final comentar el servicio de los árbitros que servirá para crear los árbitros.

El último será la vista de los comandos.

Este repositorio se encuentran cuatro clases. La primera es un enum con todos los comandos que se necesitan para que el sistema funciona como nos han pedido en el enunciado. El interprete de comandos que recogerá los comandos que se introduzcan

en la terminal, comprobar los datos introducidos y pasar la responsabilidad al controlador más oportuno de dicho comando. Existe una clase para que en todo momento, la terminal este pidiendo comandos y que no termine su ejecución y que salten los errores que se puedan producir al ejecutarlos. Por último mencionar una clase para que cada mensaje de la terminal tenga un color propio.