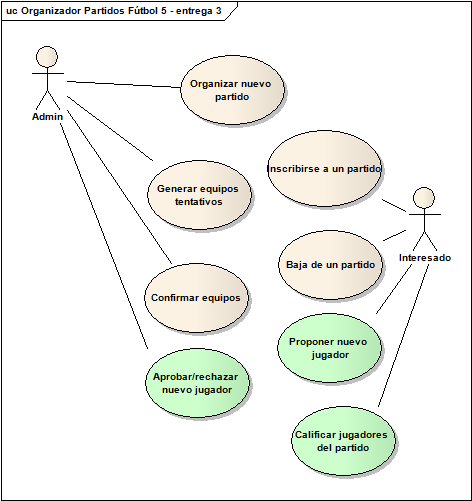
**La idea de este documento es ir planteando las ideas, opiniones o diferencias que tengamos, hasta llegar a un consenso y unificarlo despues en el doc de justificacion. Si quieren modificar o agregar algo en el medio, o al final, por favor háganlo, en otro color.**

Este diagrama de casos de uso, del enunciado, nos muestra dos cosas: en un momento T1, el interesado es quien propone nuevos jugadores. Y luego en otro momento T2, el admin toma la propuesta, y la aprueba; o la rechaza.



Según yo, el paso de mensajes para proponer (y aceptar) podría ser así:

En el momento T1, la persona inicia la propuesta indicando el amigo, y la modalidad en que quiere inscribirlo. Despues le pasa la propuesta al admin, quien la almacena en una coleccion de propuestas pendientes.

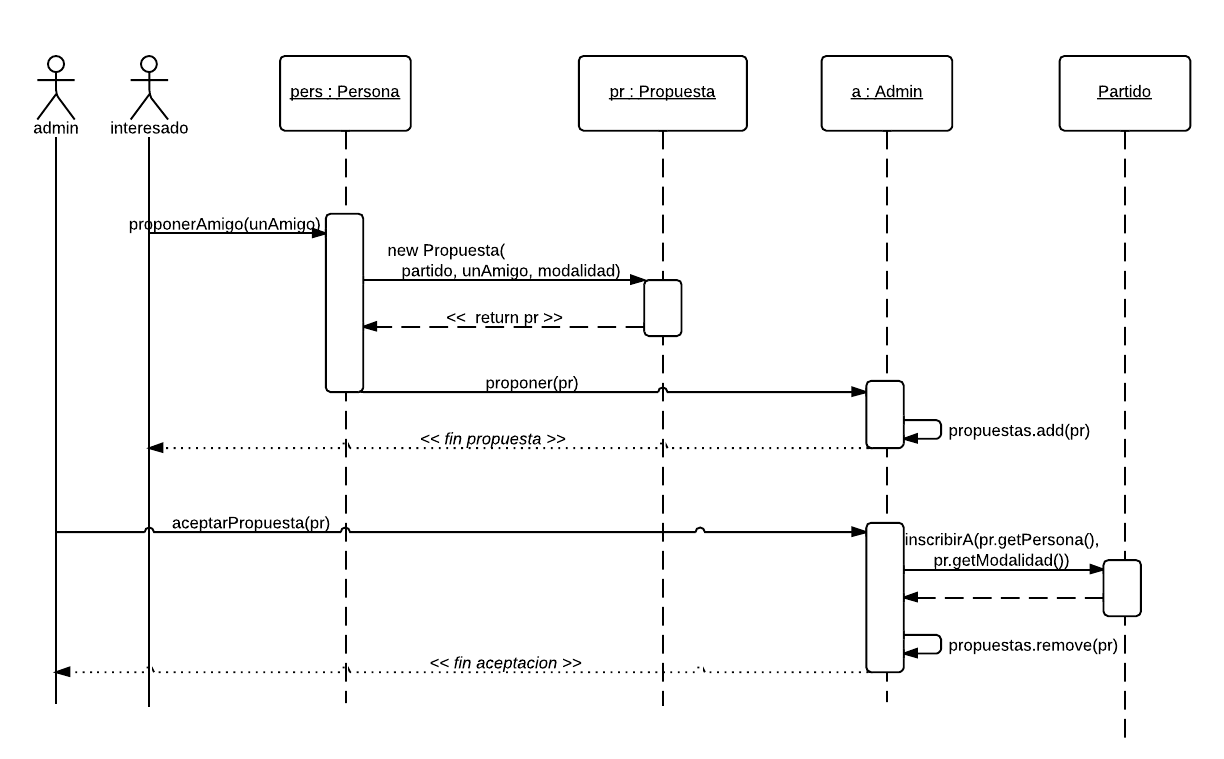
Luego, en el momento T2, el admin selecciona la propuesta sacandola de la lista de las propuestas pendientes, y la acepta. Finalmente, el hecho de aceptarla implica inscribirlo al partido en que fue propuesto.

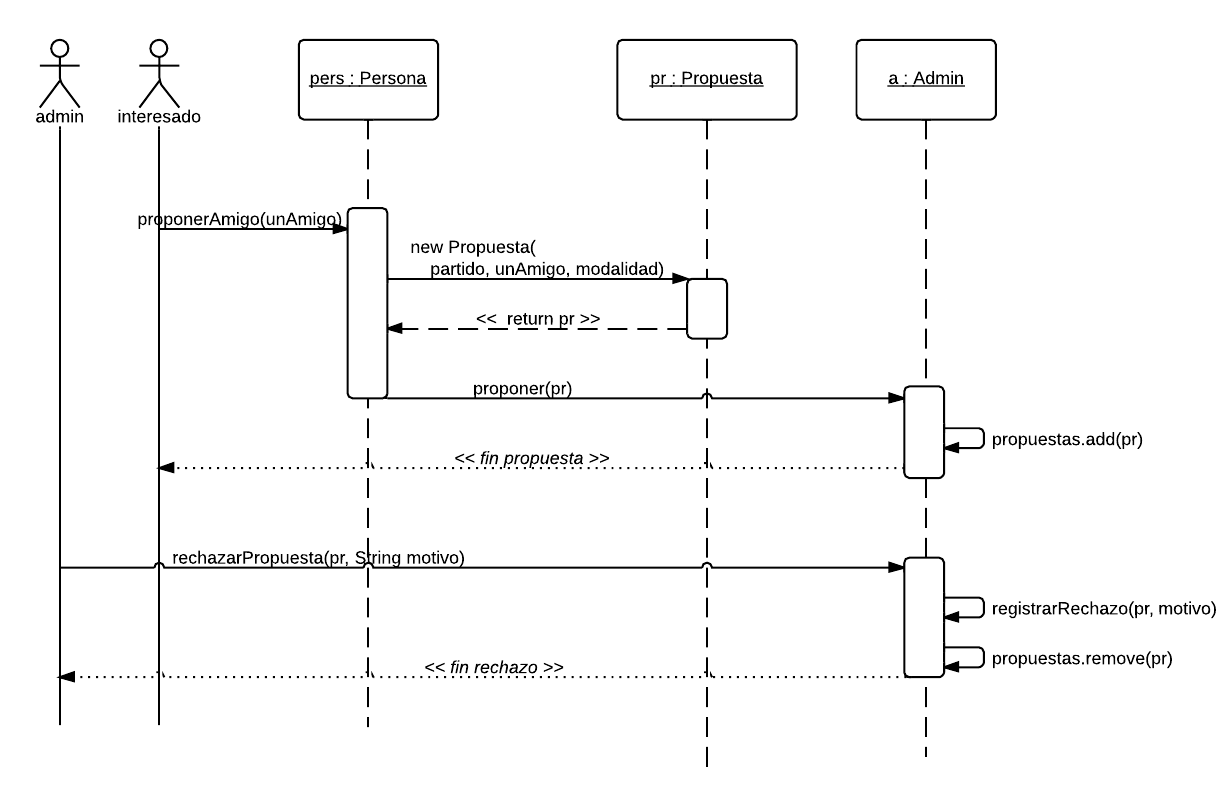
Un ejemplo del paso de mensajes puede verse en los diagramas de secuencia abajo (sino, pueden ver el archivo directamente, en la misma carpeta que está el presente doc). **No sé si está bien**, porque algunos puntos del enunciado son medio ambiguos o poco claros, pero en este momento es lo que me parece que podría cerrar.

(**NOTA**: en el grupo de google [deje consultas al respecto antes de escribir todo esto, y ya me las respondieron](https://groups.google.com/forum/#!topic/dds-jv-cursada/AH_3j0G6J7A))

**Actualización**: en base a las respuestas que me dieron, creo que la idea cierra mucho más. Por favor revisen y diganme qué les parece.

Secuencia de propuesta y **aceptacion** (v1.0):



Secuencia de propuesta y **rechazo**:

**Actualización 2:**

Me di cuenta que en el caso de aceptación de propuesta, estoy haciendo que el admin agarre de la propuesta el partido, la persona, y el modo. Y despues lo inscribe, o sea, en *pseudocódigo*:

**Admin** >> aceptarPropuesta(Propuesta unaProp)

{

Partido partido = unaProp.getPartido();

partido.inscribirA(unaProp.getPersona(), unaProp.getModalidad());

/\* hacer otras cosas, como borrar unaProp de lista de propuestas \*/

}

Esto seria un code smell, **feature envy**, ya que le estoy pidiendo demasiadas cosas a la propuesta. Aparte, la propuesta en este momento no tiene comportamiento, sólo almacena datos (otro smell, *data object*). Entonces, por qué no delegarselo al objeto propuesta, y que éste se encargue?

Quedaria:

**Admin** >> aceptarPropuesta(Propuesta unaProp)

{

unaProp.aceptar();

/\* hacer otras cosas, como borrar unaProp de lista de propuestas \*/

}

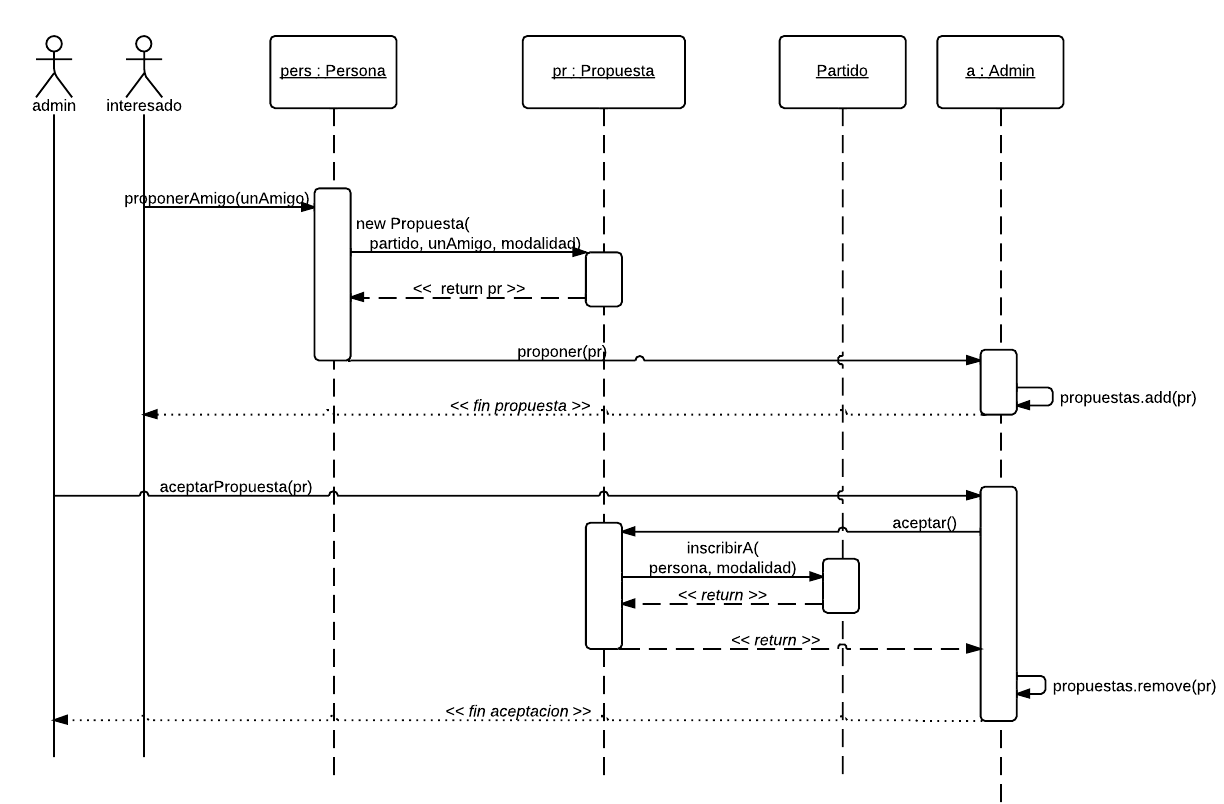
**Propuesta** >> aceptar()

{

partido.inscribirA(persona, modalidad); //partido, persona y modalidad son atributos de Propuesta

}

**Y el diagrama de secuencia v2.0 queda:***(ver abajo)*



Notar cómo el admin ya no interactúa con el partido, ni se encarga de hacer la inscripción, pues todo esto fue delegado a la propuesta. El admin sólo sabe que acepta la propuesta, y la propuesta es quien sabe manejar esa aceptación.

ACTUALIZACION 3:

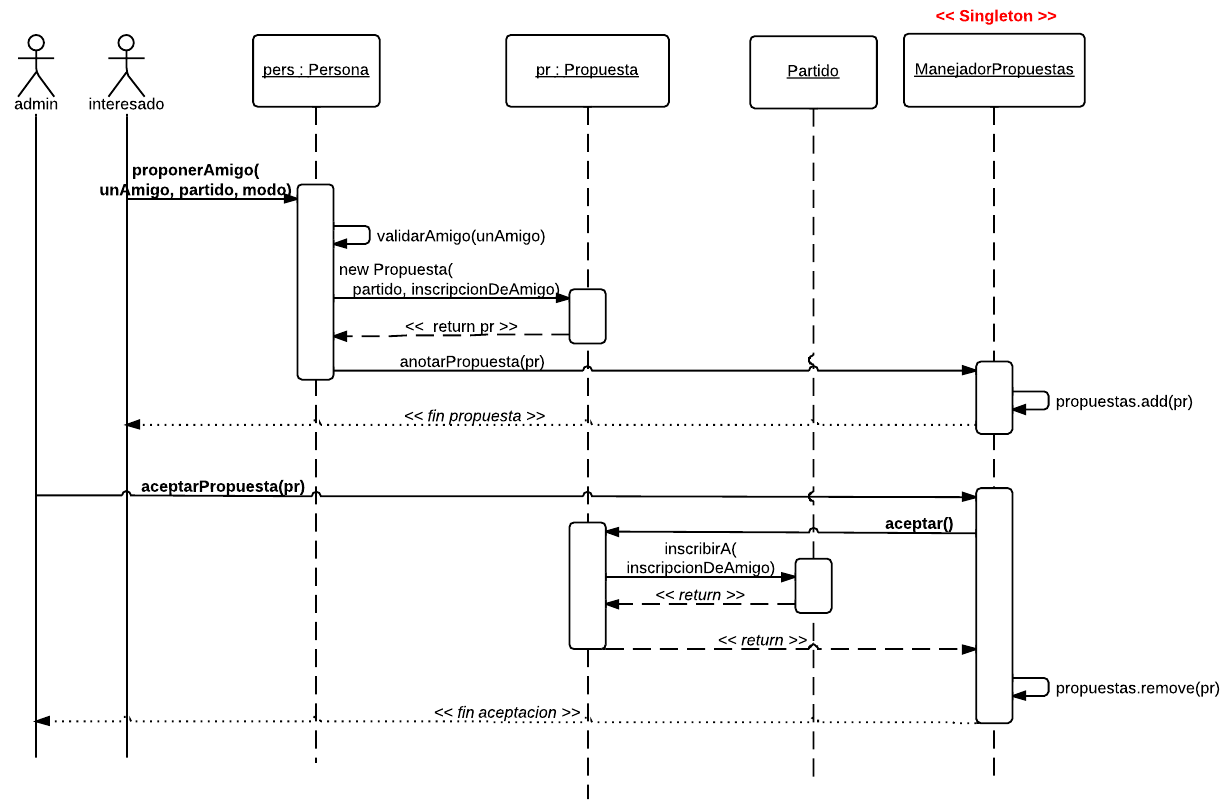
*(parte de la respuesta al item 3 acá)*

Nos dimos cuenta que al hacerlo de la forma planteada en 2.0, la persona tiene que conocer al objeto admin (para poder mandarle el mensaje proponer(pr), que luego llamamos anotarPropuesta(pr)). Esto no es correcto por dos motivos: la persona NO conoce al admin, no tiene por qué hacerlo; y a parte si quisieramos forzar esto el admin no tendría por qué ser unico, es decir cualquier admin debería poder resolver las propuestas.

Entonces, dijimos de pasar la responsabilidad de manejar las propuestas, al partido. Pero nos dimos cuenta que entonces el partido pasaría a ser un God Object (es decir, tendría demasiadas responsabilidades, algunas que no le corresponden necesariamente).

Finalmente, para **desacoplar** este comportamiento, tomamos la decision de crear un Singleton, un objeto ManejadorPropuestas que se encarge de estas tareas. Por ser Singleton, resulta ser global y puede ser accedido desde cualquier punto del sistema. Así, no ensuciamos el código del partido, y no obligamos al objeto admin a ser único ni global.

La secuencia queda:



***En el caso de que el partido quede confirmado, todas las propuestas para ese partido deberian borrarse???***