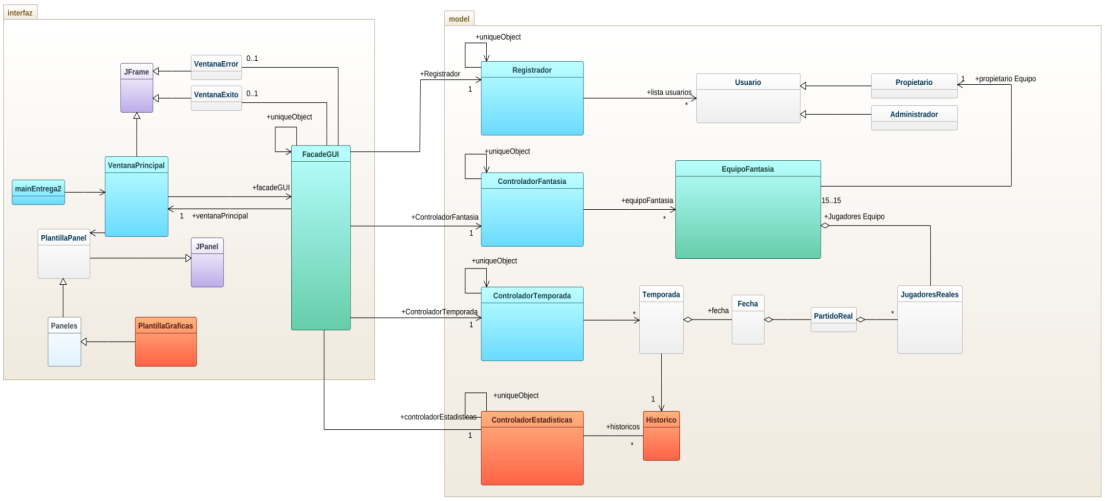
Diseño y Programación Orientado a Objetos
Proyecto 3 - Entrega 1: Diseño

Miguel Arturo Reina Rocabado - 202014739
Juan Esteban Coronel - 202111207

A continuación presentamos nuestro documento de diseño para cumplir con los requerimientos presentados en el proyecto 3 incluyendo los requerimientos acumulados de los otros requerimientos. Subrayamos en rojo y en notas lo que cambió con respecto a la última entrega para cumplir los nuevos requerimientos. Cabe mencionar que nuestra aplicación ya soportaba que un Propietario pudiese tener varios Equipos de Fantasía en una misma liga.

1. Diagrama de clases de alto nivel

Es importante añadir, que esta imagen se encuentra a detalle en nuestro repositorio de github del proyecto 3, donde podrá ser vista con una calidad más amplia. Ahora bien, a grandes rasgos, en este diagrama mostramos de una manera más compacta el horizonte que nuestro proyecto quiere alcanzar. En principio, decidimos que no era necesario mostrar en este diagrama todos los paneles que existen para mostrar las diversas funcionalidades de la interfaz gráfica, puesto que además estos se especifican en el diagrama de clases de bajo nivel.



De este diagrama, es necesario que expliquemos un poco la distribución de colores que hemos tomado puesto son colores que representan unas cualidades en específico para matizar un poco el panorama del UML. El color verde aguamarina representa las clases de mayor importancia en el diagrama, las clases vitales, sin las que no puede funcionar el sistema. Facade maneja toda la instanciación para unir las funcionalidades del modelo y de la interfaz. Equipo Fantasia

representa la composición principal en la que se basan los controladores, puesto mantiene y persiste toda la información necesaria para desarrollar el juego. El color azul son clases que catalogamos como funcionales y que desarrollan la lógica tanto del modelo como de la interfaz. El color morado representa simplemente las clases que pertenecen al framework de Java Swing. Por último, el color naranja es el que nos interesa en este proyecto, puesto representa los cambios y las clases que hemos añadido para implementar los nuevos requisitos que nos exigen. En conclusión, nuestro objetivo fue persistir el anterior diagrama de clases del anterior proyecto, realizando los cambios acorde a las nuevas modificaciones e implementaciones que debemos realizar para llevar a cabo este proyecto final, Equipos de Fantasía.

2. Glosario

En esta sección, daremos a detalle lo que es nuestro glosario de clases, es decir, la explicación y justificación de las clases que creamos para el desarrollo de nuestra aplicación, Equipos de Fantasía. No está demás, señalar que las clases que no salen en este glosario, es porque no lo consideramos necesario puesto su comportamiento es muy básico o son solo clases con atributos que no juegan un papel más allá de ser usadas por otras clases sin más. Es importante aclarar que el lenguaje para detallar estas explicaciones está dirigido al público que puede tener o no conocimientos avanzados sobre programación, es decir, es apto para cualquier interesado en el aplicativo con los conocimientos básicos en diseño de objetos.

FacadeGui: Escogimos esta clase puesto que nos basamos en el patrón de diseño de Facade. Decidimos escoger este patrón de diseño puesto que queríamos reducir la complejidad que estaba teniendo el problema en sus subsistemas. Así pudimos minimizar las comunicaciones y las dependencias entre las clases. En pocas palabras, esta clase se encarga de instanciar los controladores, manejarlos, y de igual manera, manipular las interfaces. De esta manera, cada vez que necesitamos llamar una función que está dentro de los controladores, o hacer cambios en nuestras interfaces, hacemos uso del Facade que almacena estas comunicaciones entre los paquetes, para tener un acoplamiento mínimo y una implementación más eficaz.

VentanaPrincipal: Esta clase es la que se encarga del manejo de las interfaces que usamos y mostramos al usuario. Tiene un tamaño predeterminado con parámetros estáticos (x,y), tiene paneles que se muestran dependiendo del requisito funcional que seleccione el usuario, tales paneles son: panel Propietario, panel Administrador, panel de usuario, panel para crear equipo, panel para registrarse como administrador o como usuario, panel de espera, etc. Así pues, todos estos componentes son paneles que versan sobre la ventana principal, y cualquier apertura o cierre de estos, siempre llevará hacia a la ventana principal, o saldrá de esta. Cabe recalcar, que para que la ventana principal traiga la lógica del modelo y gestione cambios, tiene la instancia única del facade que le permitirá establecer todas estas conexiones en el programa.

mainEntrega2: Ejecuta todo el proyecto, creando la ventana principal e iniciando los diversos los controladores. Se encuentra en la interfaz (vista) del código.

Registrador: El registrador y los demás controladores, están serializados para que sean únicos y manipulables por el facade. Esta clase se encarga principalmente de realizar toda la lógica y persistencia de los registros de usuarios, de administrador, y guarda esta información en un mapa de usuarios.

ControladorFantasia: El controlador fantasía, desarrolla toda la lógica relacionada con la creación de los equipos, la manipulación de estos, la administración de compra y venta de jugadores, entre otras, que son de suma relevancia para llevar a cabo el desarrollo del juego. Este controlador tiene un parámetro map donde se almacenan y se persisten todos los equipos de fantasía, el cual es muy importante puesto será un parámetro que utilizan mucho las otras clases.

EquipoFantasia: Un equipo de fantasía se compone por 15 jugadores reales, es decir 15 objetos del tipo JugadorReal, tiene un único propietario, tiene puntos, tiene una alineación que la define el usuario y siempre se comprueba si es válida o no. El único método importante de esta clase es el que se encarga de definir y extraer los puntos que el equipo va obteniendo conforme transcurre la temporada.

ControladorTemporada: El controlador de temporada se encarga de desarrollar toda la lógica del modelo relacionada con las funciones del administrador, es decir, solo el administrador puede acceder a estas funcionalidades. La clase facilita la creación de una nueva temporada, su manipulación, conseguir registros, modificar registros, definir partidos, fechas, jugadores de la temporada, e ir evidenciando los puntos que van ganando los jugadores. Su función principal es persistir la temporada en un map y brindar información sumamente importante a los otros controladores.

ControladorEstadisticas: El controlador de estadísticas, mediante un registro modificable y actualizable, se encarga de desarrollar la persistencia y lógica de las estadísticas para mostrar el desarrollo de la temporada a los usuarios, claro está, mediante el facade que lleva la información a el panel de estadísticas para imprimir esta información mediante una gráfica visible.

Historico: El registro modificable y actualizable del que hablamos anteriormente, es esta clase. Es el histórico que guarda la información de todos los equipos, los resultados, y como fueron los desempeños de los jugadores durante las diversas temporadas que se carguen en el csv.

3. Justificación del diseño

En esta parte definiremos algunos aspectos que usamos para el diseño de nuestra aplicación, justificamos el porqué del diseño y las implicaciones que esto trae consigo mismo.

¿Por qué las interfaces no están divididas en nuevas ventanas si no en paneles?

Escogimos diseñar paneles para cada funcionalidad que escoja el usuario, puesto así es más fácil visualizar el track de la información que va almacenando el usuario, en qué panel está ubicado, que quiere buscar y facilita el devolverse a la ventana principal en caso de salir de los paneles. Eventualmente, parece que a simple vista es más fácil simplemente ir creando ventanas para cada ejecución, lo cual puede ser cierto, sin embargo sería más complicado persistir la información que el usuario va ingresando en la aplicación. El código se complejiza un poco, pero así tenemos un registro correcto de la relación del usuario con la interfaz.

¿Por qué el facade es el comunicador para realizar acciones entre la interfaz y el modelo?

El facade, que en este caso también es un singleton, es una clase de alto acoplamiento que sirve como controlador y mediador entre las interfaces y el modelo, para así mediar las comunicaciones, reducir el acoplamiento de las otras clases, persistir información, y hacer que las clases de mayor nivel no dependan de clases de menor nivel.

¿Por qué controladores para manipular el desarrollo de la lógica del programa?

Los controladores, entre ellos el registrador, pertenecen al paquete del modelo, es decir, su supuesta función es desarrollar todos los procesos lógicos del programa. Así pues, es más sencillo modularizar la lógica en diversos controladores para cada funcionalidad de una naturaleza en específico, respetando así el single responsability principle. Como podemos evidenciar, la clase registro se encarga de la funcionalidad que tenga que ver con el registro de usuarios, así mismo, las clases controladoras para equipos, para ventas, para estadísticas, tienen su nombre definido puesto es el papel que desarrollan en las funcionalidades del software. Además, así reducimos el acoplamiento en las clases.

¿Por qué no hay interfaces?

No hay interfaces principalmente porque no encontramos ninguna clase que pueda abstraer funciones en común para otras clases hijas, ya que todo el diseño está tan modularizado y tienen funciones sumamente específicas, no vimos la necesidad de usar interfaces ni patrones que tengan que ver con estas abstracciones. El juego en sí es muy lineal y los objetos están definidos claramente para todas instancias del modelo.

4. Requisitos Funcionales (actualización)

A continuación presentamos los requisitos funcionales actualizados teniendo en cuenta el cambio en el enunciado.

4.1 Usuario

Nombre	RF1. Registrarse en la aplicación
Resumen	El usuario ingresa sus datos necesarios para poder ingresar correctamente a la aplicación
Entradas	Apertura de la aplicación por primera vez en el dispositivo. Botón “Registrarme” Información que ingresa el usuario ● Usuario ● Contraseña
Resultados	El usuario es ingresado a la plataforma de la aplicación y sus datos son almacenados en una base de datos para llevar registro de sus interacciones dentro de las aplicaciones
Historia de Usuario Relacionada	Yo como usuario, quiero poder ingresar y registrarme, para que de esta manera pueda empezar a hacer uso de la aplicación sin que otras personas puedan modificar mi equipo.

Nombre	RF2. Crear un equipo de Fantasía
Resumen	El usuario creara un equipo de fantasía dentro la aplicación con ciertos parámetros como lo son el presupuesto y la cantidad de jugadores

Entradas	Tener en cuenta el presupuesto ya que no puede gastar más dinero de la que posee para poder comprar jugadores. Tener en cuenta los jugadores ya que un equipo de fantasía siempre debe tener 15 jugadores en el equipo, en todo momento. Información para crear el equipo ● Nombre del equipo ● Compra de 2 porteros de equipo real ● Compra de 5 defensores de equipo real ● Compra de 5 mediocampistas de equipo real ● Compra de 3 delanteros de equipo real (Las compras de jugadores siempre menor al presupuesto)
Resultados	El usuario tiene su equipo de futbol de fantasía creado correctamente y este ya puede ser utilizado para comenzar la liga con otros equipos de futbol
Historia de Usuario Relacionada	Yo como usuario, quiero poder crear un equipo de fantasía, para que de esta manera pueda empezar jugar con mi equipo dentro de la aplicación

Nombre	RF3. Configurar alineación antes de una fecha (seleccionar jugadores titulares)
Resumen	El usuario configura su equipo de fantasía dentro la aplicación con ciertos parámetros como lo son la cantidad de jugadores por posición
Entradas	Tener en cuenta los parámetros de jugadores por posición (Ejemplo, ‘un equipo no puede tener dos arqueros jugando al mismo tiempo o dos capitanes’) Tener en cuenta los parámetros de tiempo ya que la configuración del equipo solo se puede hacer hasta la media noche el día anterior al partido Información para crear el equipo ● Colocar 1 arquero dentro de la alineación ● Colocar 4 defensas dentro de la alineación ● Colocar 4 mediocampistas dentro de la alineación ● Colocar 2 delanteros dentro de la alineación ● Seleccionar a uno de los 11 titulares como capitán

	• Colocar a los 4 suplentes del equipo
Resultados	El usuario es ingresado a la plataforma de la aplicación y sus datos son almacenados en una base de datos para llevar registro de sus interacciones dentro de las aplicaciones
Historia de Usuario Relacionada	Yo como usuario, quiero poder configurar mi equipo antes de cada fecha para que de esta manera tenga más oportunidades de ganar el partido

Nombre	RF4. Comprar jugadores
Resumen	El usuario hace la compra de un jugador por el precio establecido del ofertante. El precio acordado por el jugador es descontado del presupuesto de su equipo y el jugador por el cual negociaron es inscrito en el club del demandante.
Entradas	Información para comprar jugador • Indicar el jugador que se quiere comprar • Indicar el precio de compra
Resultados	El usuario compra un jugador por el 100% de su valor y este es agregado a su equipo de fantasía
Historia de Usuario Relacionada	Yo como usuario, quiero poder comprar jugadores, para que de esta manera pueda tener el mejor equipo posible con mi dinero y con los parámetros de cantidad de jugadores por equipo

Nombre	RF5. Vender jugadores
Resumen	El usuario hace la venta de un jugador por un precio establecido por él. El jugador pierde el 3% del precio de la venta del jugador y el resto de la plata es ingresada al club para que de esta manera se pueda comprar un jugador. Aparte, el jugador por el cual hubo la negociación es quitado del club del que lo vendió y es pasado al club que lo compró.
Entradas	Información para vender jugador • Indicar el jugador que se quiere vender • Indicar el precio de venta
Resultados	El usuario vende su jugador, pero la cantidad de dinero que será entregada al club será el 97% del precio acordado por la venta del determinado jugador.

	El usuario ya no tendrá el jugador dentro de su equipo de fantasía.
Historia de Usuario Relacionada	Yo como usuario, quiero poder vender jugadores, para que de esta manera pueda tener más dinero y así, poder comprar mejores jugadores para el equipo

4.2 Administrador

Nombre	RF1. Registrarse en la aplicación
Resumen	El administrador ingresa sus datos necesarios para poder ingresar correctamente a la aplicación
Entradas	Apertura de la aplicación por primera vez en el dispositivo. Botón “Registrarme” Información que ingresa el administrador ● Usuario ● Contraseña
Resultados	El administrador es ingresado a la plataforma de la aplicación y sus datos son almacenados en una base de datos para llevar registro de sus interacciones dentro de las aplicaciones
Historia de Usuario Relacionada	Yo como administrador, quiero poder ingresar y registrarme, para que de esta manera pueda empezar a hacer uso de la aplicación

Nombre	RF2. Planeación de la Temporada
Resumen	El administrador crea la temporada para los equipos de fantasía y también genera el cronograma fecha a fecha de toda la temporada.
Entradas	Información que ingresa el administrador ● Equipos reales de la temporada ● Conjunto de partidos próximos, con la información de a qué fecha (día y hora) pertenecen, qué equipo es el local, qué equipo es el visitante
Resultados	La creación de la temporada es creada y toda la información persiste en el sistema. En particular, cada uno de los jugadores tiene la información de todas las fechas. Dentro de esta información esta que equipo

	juega contra quien, que equipo es el local, la hora del encuentro, entre otras.
Historia de Usuario Relacionada	Yo como administrador, quiero poder hacer la planificación de la temporada antes de que esta empiece, para que de esta manera cada uno de los equipos tenga la facilidad de obtener esta información la cual es importante antes de empezar la temporada

Nombre	RF3. Registro de nóminas reales
Resumen	El administrador ingresa los datos de las nóminas de las plantillas reales. Los parámetros ingresados de los equipos reales se mantienen constantes durante toda la temporada
Entradas	La información que se obtendrá de cada jugador del equipo real será: Información que ingresa el administrador ● Nombre del jugador ● Posición del jugador ● Precio del jugador
Resultados	Los jugadores del equipo real quedan inscritos dentro de la aplicación para que de esta manera los equipos de fantasía se puedan crear
Historia de Usuario Relacionada	Yo como administrador, quiero poder inscribir a los jugadores reales dentro de la aplicación, para que de esta manera los usuarios puedan empezar a crear cada uno de sus equipos

Nombre	RF4. Registrar resultados de partidos
Resumen	El administrador ingresa los resultados de los partidos reales para que de esta manera se pueda actualizar los puntajes dentro de los equipos de fantasía
Entradas	Información que ingresa el administrador ● Resultado del partido real dentro de la aplicación
Resultados	La aplicación tendrá actualizada los puntajes de los equipos debido al registro de los resultados por parte del administrador
Historia de Usuario Relacionada	Yo como administrador, quiero poder ingresar los resultados de los partidos, para que de esta manera la aplicación se actualice con los respectivos puntos para los equipos de fantasía

Nombre	RF5. Registrar desempeño de los jugadores

Resumen	Teniendo en cuenta los parámetros de puntajes de los jugadores por partido y del desempeño de los jugadores reales, se tendrá actualizada la tabla de puntajes en los equipos de fantasía
Entradas	Información que ingresa el administrador ● Desempeño de los jugadores
Resultados	Cada jugador tendrá su puntaje obtenido dependiendo de su desempeño del partido y este será actualizado dentro de la aplicación.
Historia de Usuario Relacionada	Yo como administrador, quiero poder ingresar el desempeño de los jugadores para que de esta manera la aplicación se actualice con los respectivos puntos para los equipos de fantasía

Nombre	RF6. Mostrar Estadísticas
Resumen	El administrador muestra estadísticas de la temporada actual dependiendo de los puntajes que cada equipo ha ido obteniendo y de las calificaciones que cada uno de los jugadores ha obtenido
Entradas	Información que ingresa el administrador ● Tabla de clasificación de la temporada ● Tabla de clasificación de puntajes de jugadores ● Puntajes de equipos por cada fecha
Resultados	La aplicación tendrá la posibilidad de ver en cualquier momento estadísticas como el ranking del equipo, jugador con mejor y peor puntaje, el equipo con mejor puntaje por fecha, entre otros.
Historia de Usuario Relacionada	Yo como administrador, quiero poder mostrar estadísticas de la temporada, para que de esta manera los usuarios puedan acceder y tomar decisiones frente a estas como lo puede ser comprar nuevos jugadores o vender jugadores, entre otros.

Es evidente entonces que tenemos un nuevo requerimiento funcional de mostrar las estadísticas dentro de las ligas actuales. Además, el requerimiento de tener varios equipos de fantasía, ya se cumplía desde el proyecto 2.

5. Requerimientos no funcionales

Los requerimientos no funcionales se conservan del proyecto 2, por lo que aquí únicamente por completitud los volvemos a mencionar y contextualizar al respecto. Dado el contexto del proyecto, que se irá a mayor detalle en las restricciones, se sabe que algunos atributos de calidad son deseables con la solución que se planteará en un futuro cercano. En primer lugar, debe ser **usable** o fácilmente entendible para el administrador y los usuarios que la vayan a utilizar: no debería involucrar conocimientos específicos de programación, más allá de usar Git y VSCode,

para poder usarla: estos tipos de usuarios no deberían saber cómo está diseñada la aplicación para moderarla o para jugarla. Tiene que ser **rápida** para que haya una interacción eficaz con los tipos de usuario, no debería demorarse excesivamente para iniciar sesión, comprar jugadores, etc. Ha de ser **segura** para que los usuarios no pierdan su información por personas mal intencionadas, pero esto está más allá de los límites del proyecto tal y como se dice en el enunciado. Ha de ser **escalable** en el sentido de que múltiples usuarios podrán jugar al fútbol de fantasía y debería poder ser eficiente para ingresar más usuarios a la solución sin mayores costos.

6. Restricciones:

1. Un propietario puede tener un número ilimitado de equipos de fantasía en una misma liga
2. El equipo de fantasía debe tener 15 jugadores.
3. Los jugadores del equipo de fantasía tienen que estar compuesto por jugadores presentes en la liga real.
4. Los equipos no están obligados a gastar su saldo presupuesto.
5. Los usuarios deben configurar su alineación del equipo de fantasía antes de la fecha del campeonato.
6. Toda la información debe ser persistente durante todo proceso, es decir, las listas de equipos y jugadores, la conformación de los equipos de fantasía, los resultados de los partidos, el desempeño de los jugadores, los puntos asignados a cada equipo, etc., deben ser consistentes y sin adulterar.
7. La carpeta donde se guardarán los textos de la información de los equipos y demás, no puede ser la misma carpeta donde se encuentre el código fuente de la aplicación.
8. La sincronización al usar GIT no tiene que hacerse desde la aplicación: los usuarios deben encargarse de sincronizar la carpeta usando algún cliente de GIT.
9. Tanto el administrador como los usuarios “normales” del sistema deben usar la misma aplicación: dependiendo del tipo de usuario, las opciones que se muestren deben ser diferentes.
10. Vamos a suponer que todos los usuarios del sistema, así como los partidos, utilizan la misma zona horaria. Es decir que las fechas y horas no necesariamente deben tener en cuenta la zona horaria (ej. GMT -5).
11. Si llega a ocurrir que más de uno de los jugadores titulares que ocupan la misma posición se quedan sin jugar, sólo uno será sustituido por el suplente.
12. Para actualizar los puntajes, sólo se deberán tener en cuenta los jugadores que hayan sido designados como titulares para la fecha, a menos de que el jugador no haya jugado ni siquiera un minuto en esa fecha, en cuyo caso, será reemplazado por el suplente correspondiente.
13. La aplicación debe estar hecha en Java y la interfaz debe estar basada en consola.

7. Responsabilidades

A continuación presentamos la asignación de las responsabilidades identificadas en el enunciado

frente a qué componente del sistema estará encargado de cumplirlas.

Tabla 1: Asignación de responsabilidades

No.	Responsabilidad	Tipo de usuario	Componente
1	Registrar Usuario	Usuario	Registrador
2	Crear equipo de fantasía	Propietario	ControladorFantasía
3	Vender jugador	Propietario	ControladorFantasía
4	Comprar jugador	Propietario	ControladorFantasía
5.	Ver histórico del equipo por fecha	Propietario	ControladorFantasía
6.	Ver desempeño de los jugadores por posición por fecha	Propietario	ControladorFantasía
7.	Cambiar alineación del equipo de fantasía	Propietario	ControladorFantasía
8.	Ver reportes y gráficas del avance de la liga	Propietario	ControladorEstadísticas
9.	Planear temporada (y toda su info. asociada)	Administrador	ControladorTemporada
10.	Registrar resultado de un partido	Administrador	ControladorTemporada
11.	Establecer una temporada actual	Administrador	ControladorTemporada
12.	Establecer una fecha como actual	Administrador	ControladorTemporada

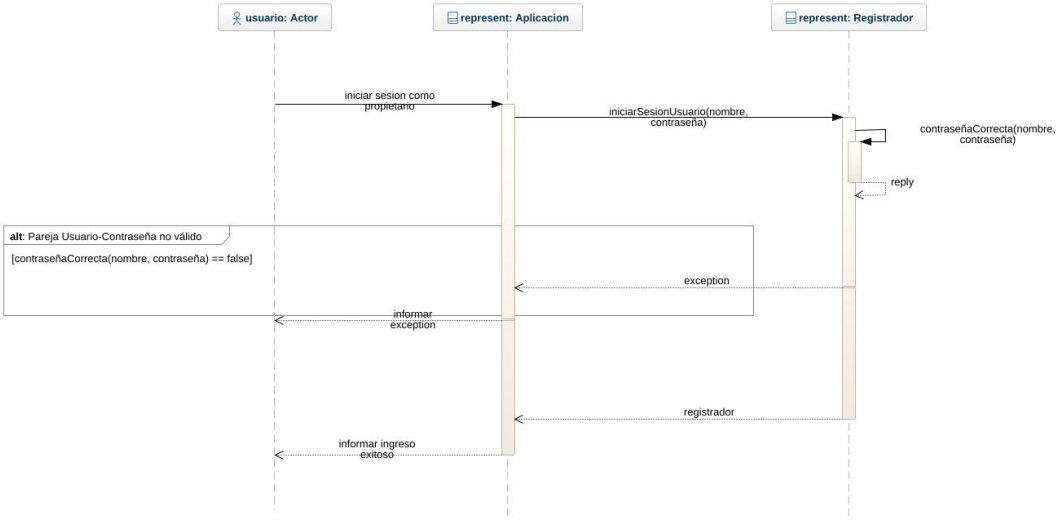
De esta forma, las responsabilidades principalmente asociadas con planear, establecer una temporada o fecha actual estarán agrupadas en la clase ControladorTemporada ya que así la caracterización y estado de la temporada están agrupadas en una sola clase. De manera similar, ControladorFantasía se encargará de lidiar con las modificaciones dentro de los equipos de fantasía (interactúa principalmente con los pedidos del propietario). Mientras que el registrador únicamente está encargado del login y registro de nuevos usuarios ya sean propietarios o administradores.

Por este diseño, se espera que ControladorTemporada colabore con clases como: Temporada, Fecha, JugadorReal mientras que ControladorFantasía colabore con clases como: EquipoFantasía y JugadorReal y por último Registrador colabore con clases como: Propietario y Administrador mediante la interfaz de Usuario.

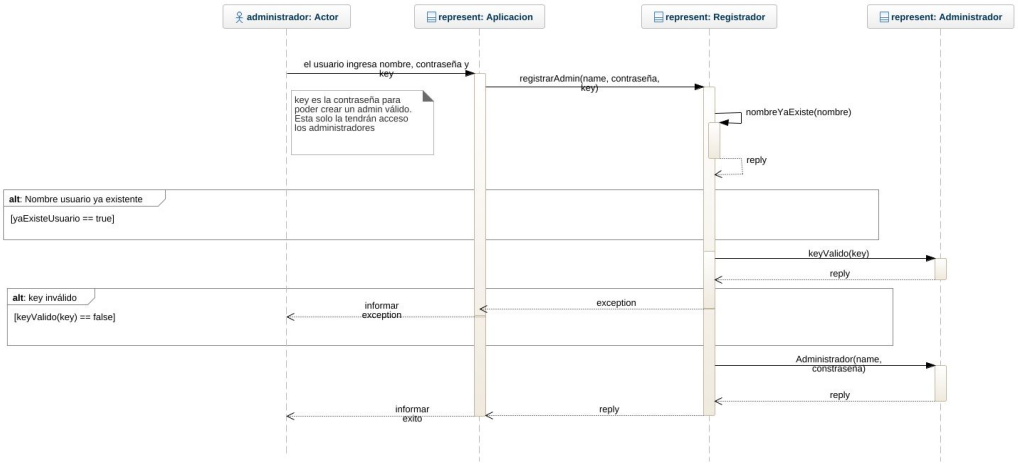
8. Diagramas de secuencia

A continuación, se presentan los diagramas de secuencia de los requerimientos críticos identificados del enunciado. Todos los diagramas de secuencia se pueden encontrar en el repositorio de GitHub.

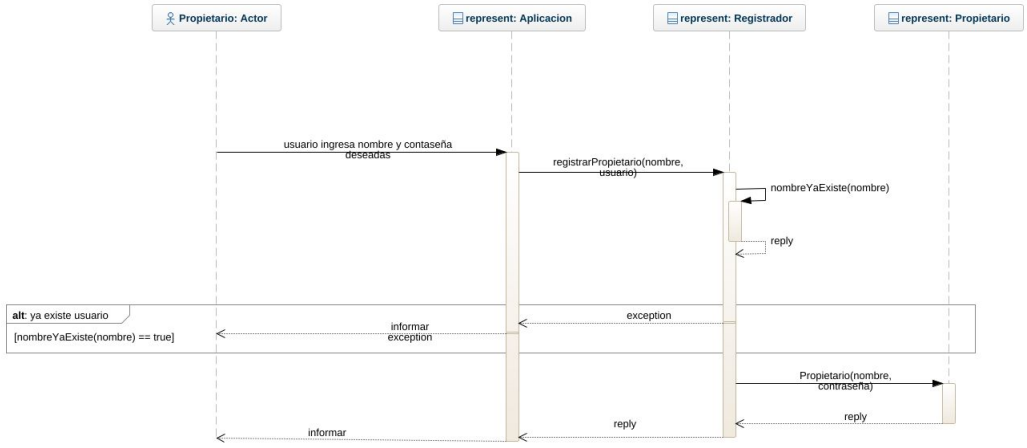
Inicio de sesión - usuario



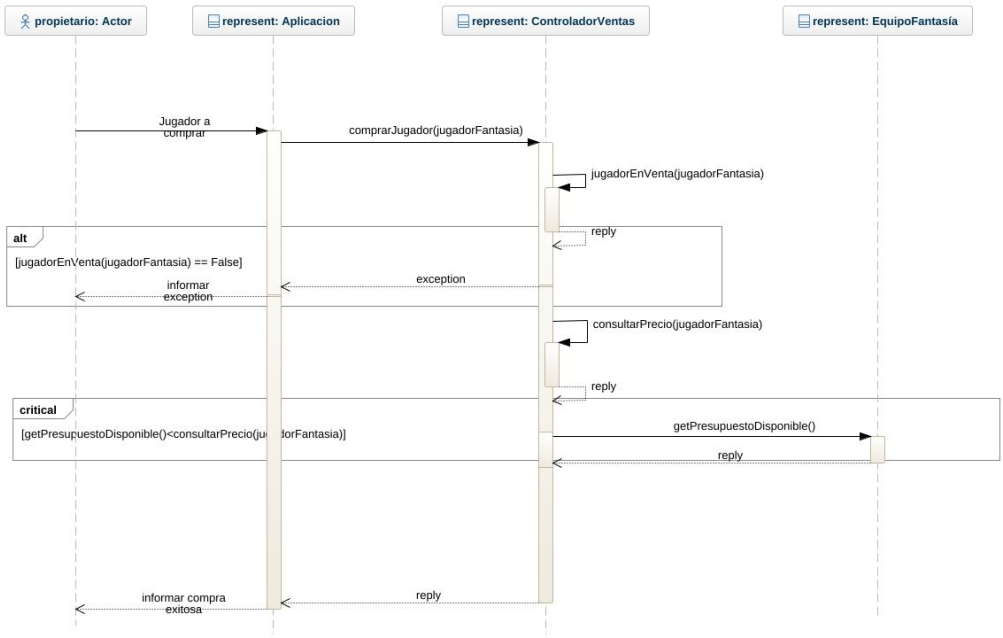
Registrar administrador



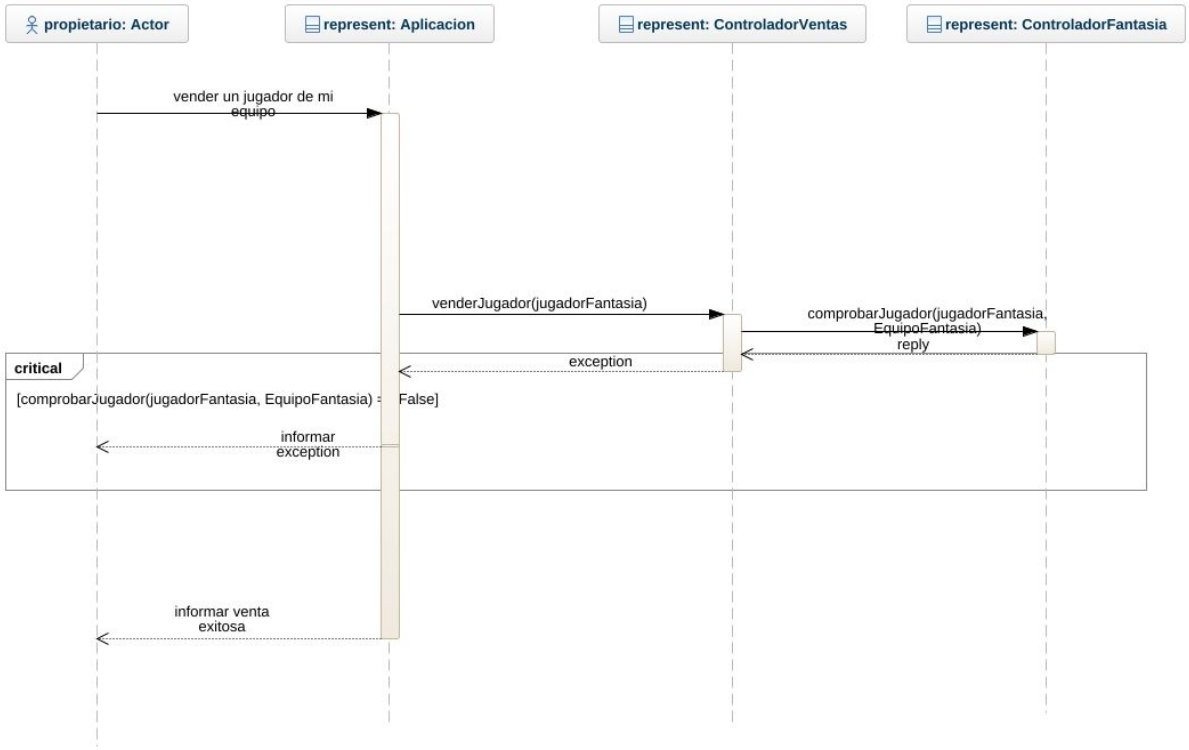
Registrar propietario



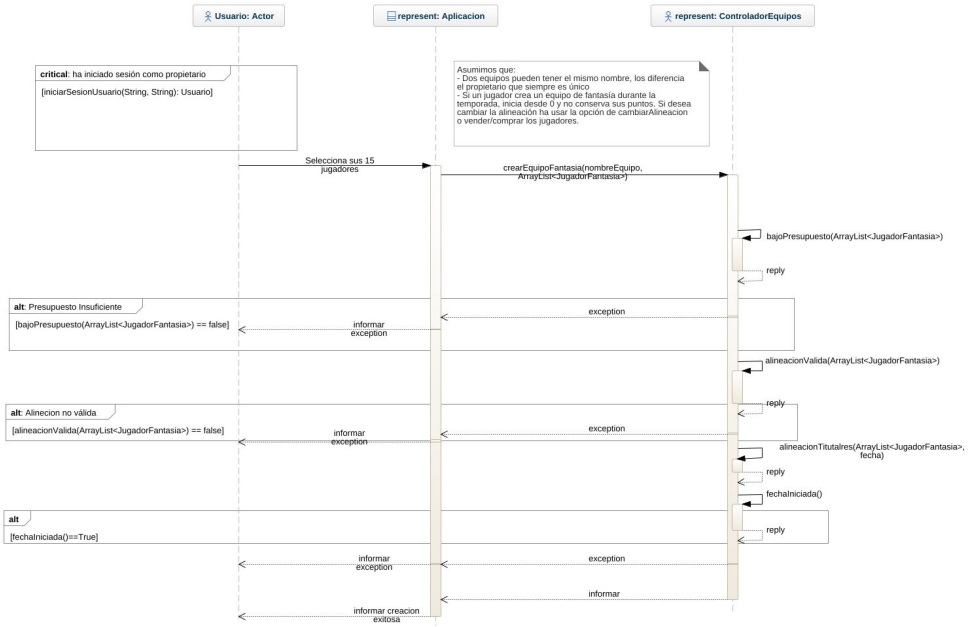
Compra de jugador



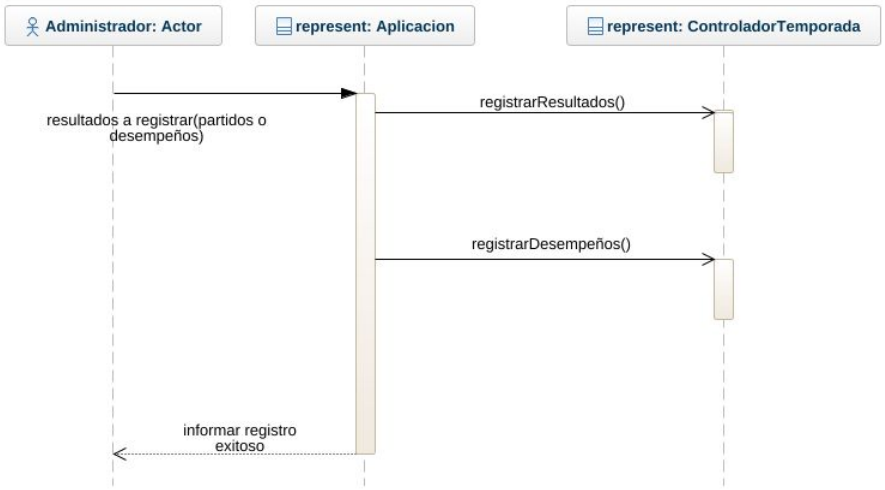
Venta de jugador



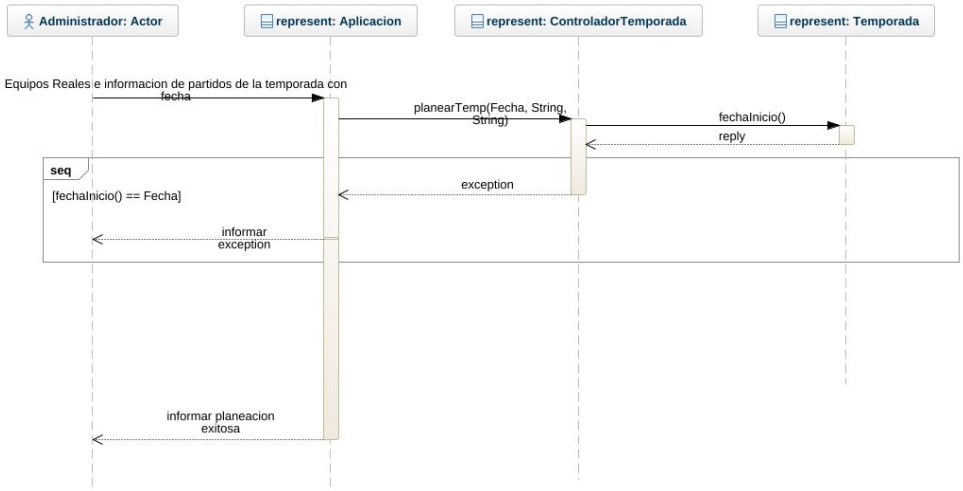
Creación de un equipo de fantasía



Registrar resultados de un partido real



Planear temporada



Registrar estadísticas