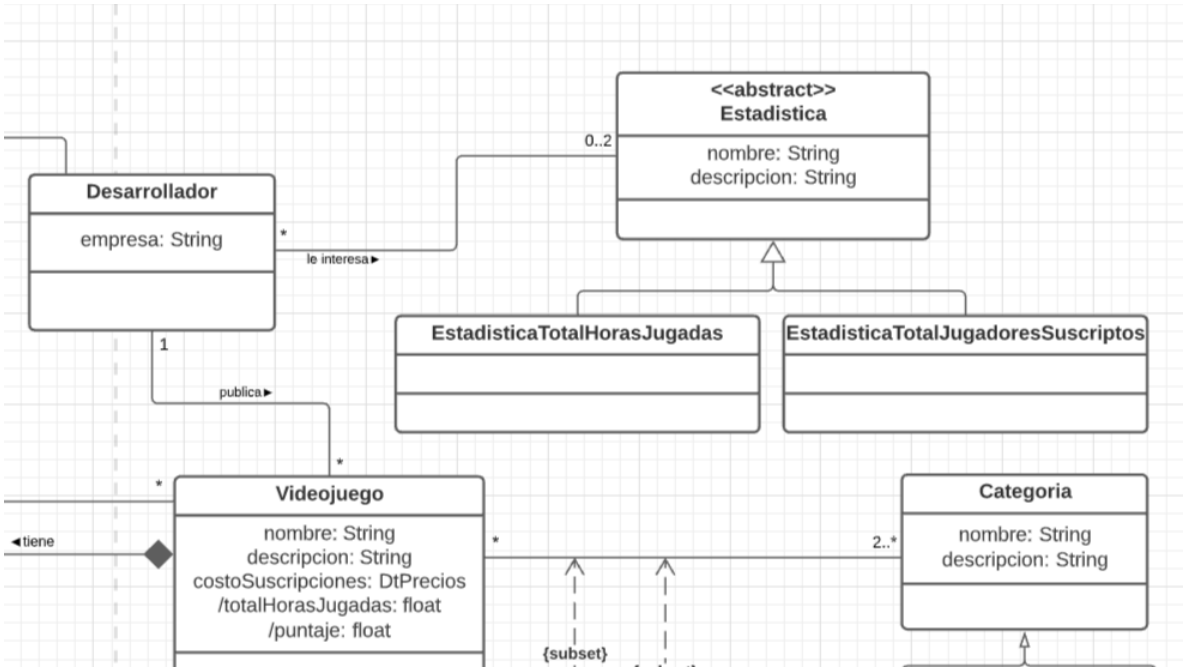
1 Nuevos Requerimientos

Los nuevos requerimientos pedidos en esta última tarea se basan en la inclusión de 3 nuevos casos de uso: Realizar Búsqueda, Seleccionar Estadísticas y Consultar Estadísticas. No se consideró necesario realizar modificaciones en los conceptos ni atributos anteriormente introducidos.

Para el Modelo de Dominio fue introducido el concepto "Estadística", el cual almacenará la información referida al nombre y descripción de la misma. Este concepto abstracto contendrá dos derivados representando cada una de las distintas estadísticas disponibles, las cuales el desarrollador estará interesado en conocer. Se decidió modelar de esta forma para garantizar flexibilidad al momento de incluir nuevas estadísticas al sistema.

En caso de ser necesario agregar una nueva estadística se deberá agregar un nuevo concepto derivado en "Estadística" y un nuevo atributo calculado dentro del concepto "Videojuego", incluyendo una sobrecarga para la operación de calcular las estadísticas.



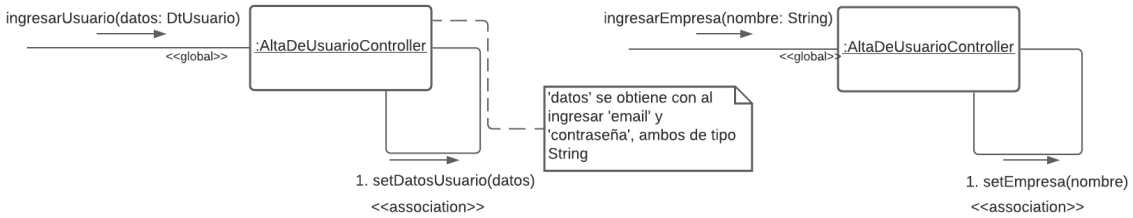
Como podemos ver, las estadísticas son independientes de los demás conceptos, a causa de esto no fue necesaria la modificación de los Diagramas de Secuencia de los Casos de Uso modelados en entregas anteriores.

En cuanto a la "Búsqueda", se aplicó el patrón de diseño "Composite". No fue pedida su inclusión en el modelo de dominio, por lo que no tuvo efecto sobre el mismo. Sólo se pide diseñar, aplicando patrones de diseño, una estructura de clases que sea adecuada y flexible para representar criterios de búsqueda simples y avanzados.

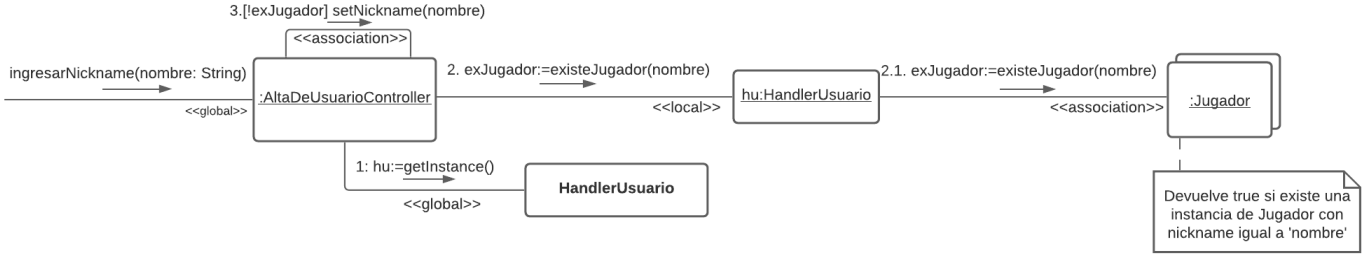
2 Realización de Casos de Uso

2.1. Alta de Usuario

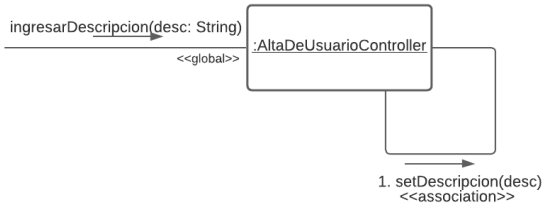
ingresarUsuario(DtUsuario)



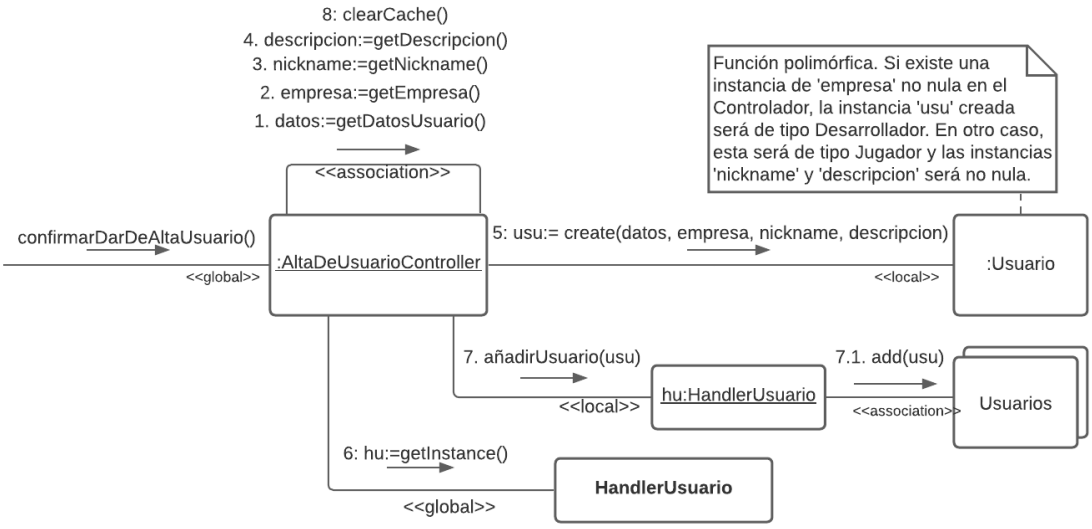
ingresarNickname(String)



ingresarDescripcion(String)

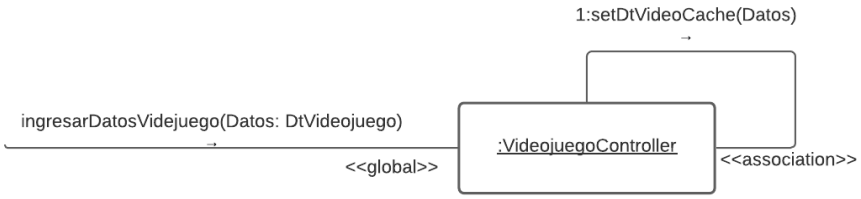


confirmarDarDeAltaUsuario()

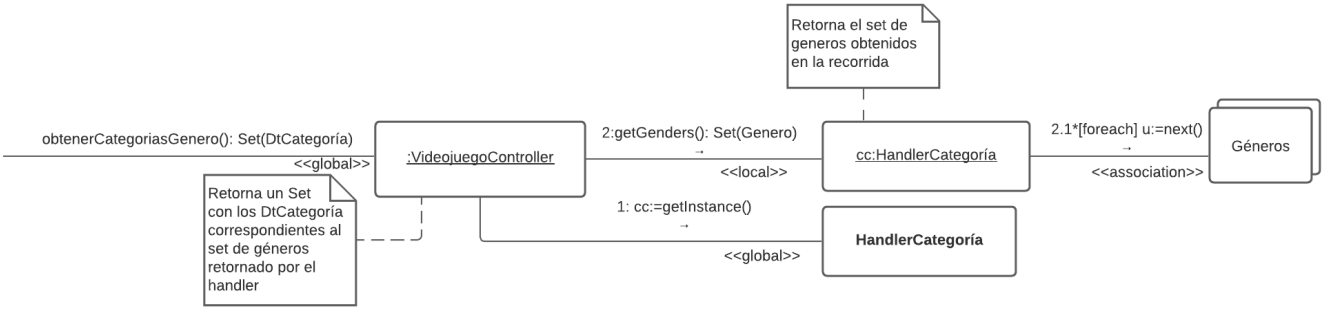


2.2. Publicar Videojuego

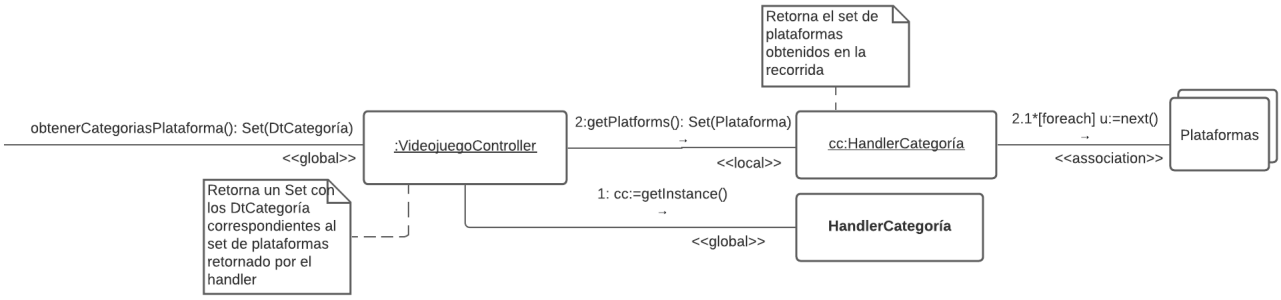
ingresarDatosVideojuego(DtVideojuego)



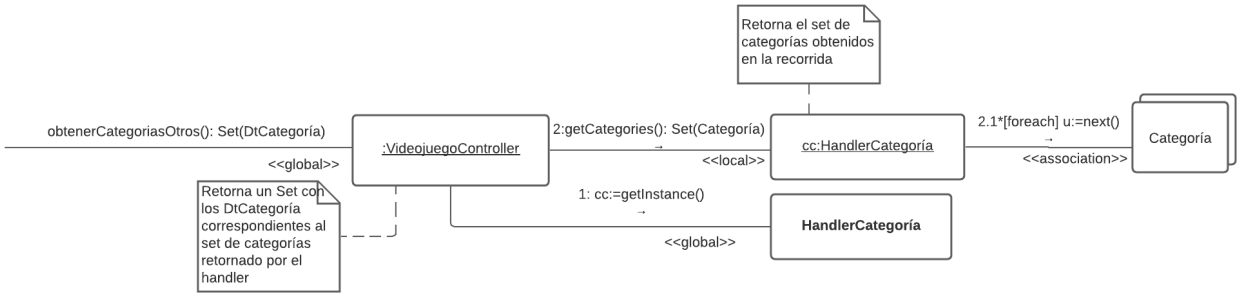
obtenerCategoriasGenero(): Set(DtCategoria)



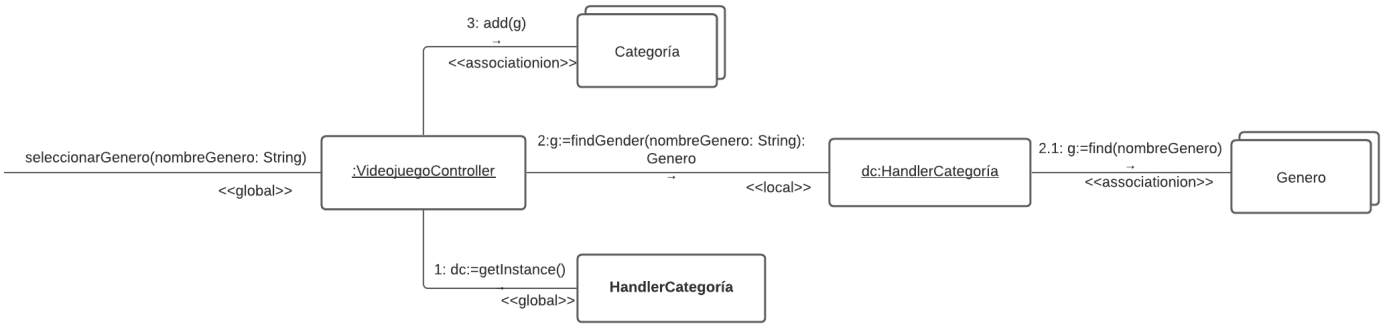
obtenerCategoriasPlataforma(): Set(DtCategoria)



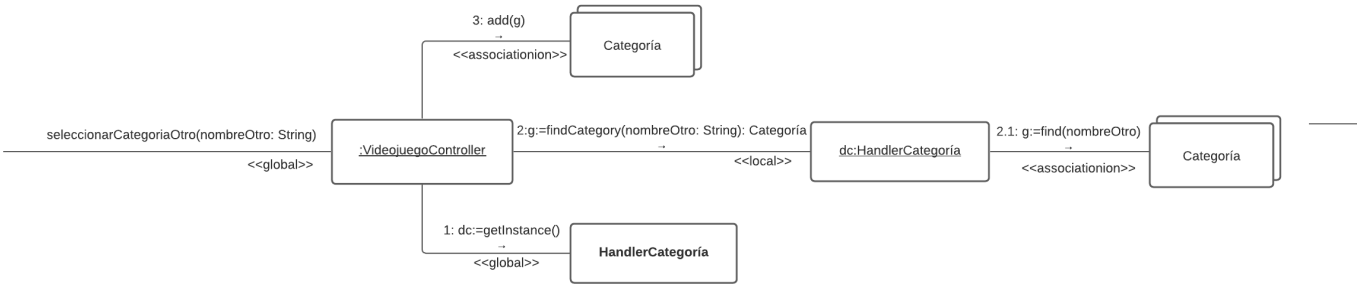
obtenerCategoriasOtro(): Set(DtCategoria)



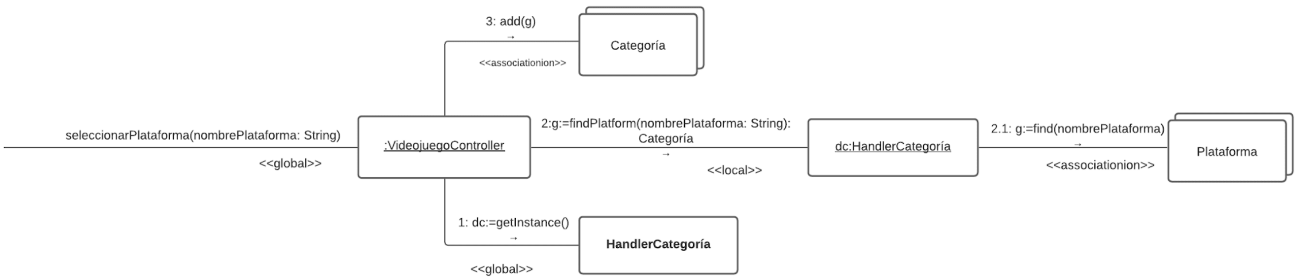
seleccionarGenero(String)



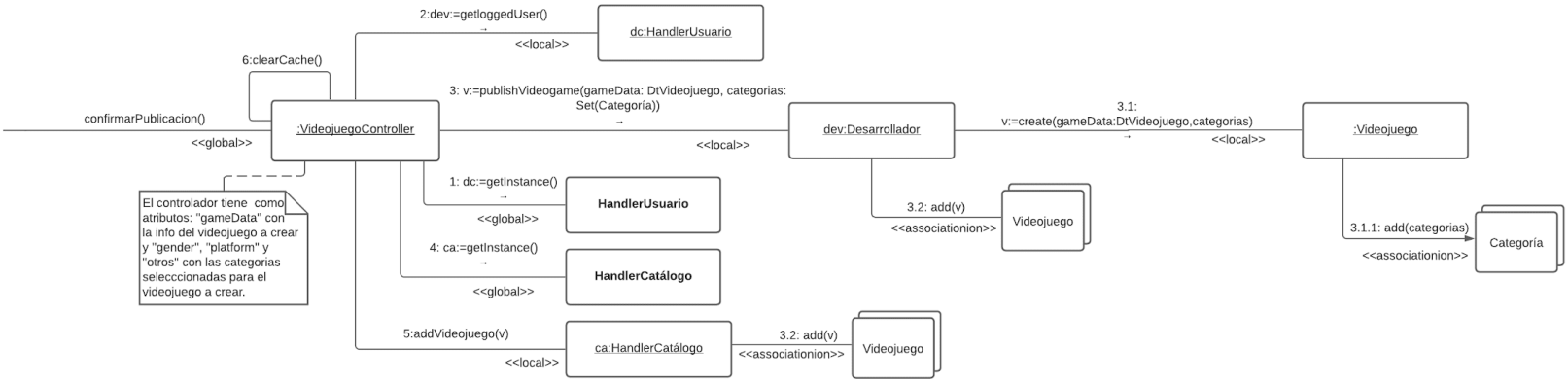
seleccionarGeneroOtro(String)



seleccionarGeneroPlataforma(String)

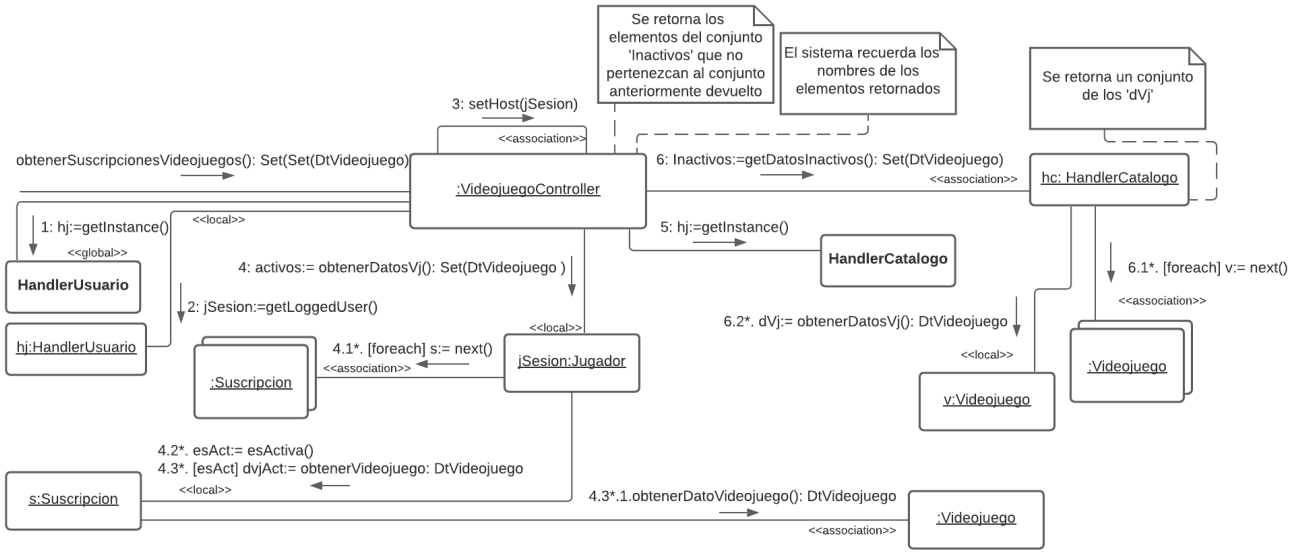


confirmarPublicacion()

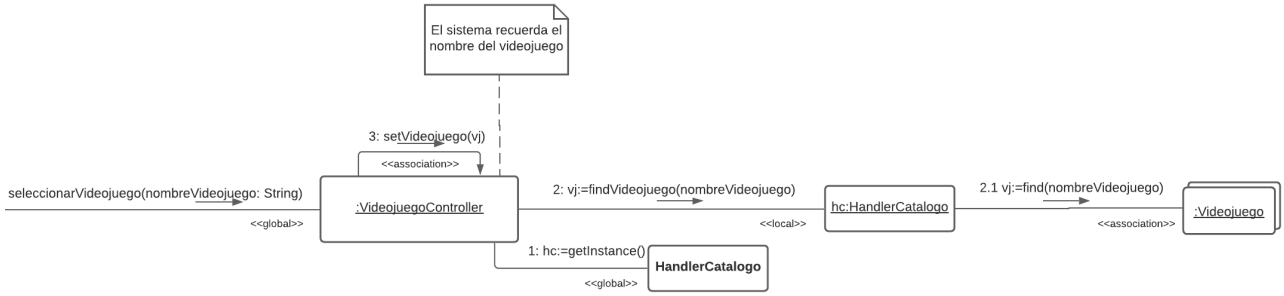


2.3. Suscribirse a Videojuego

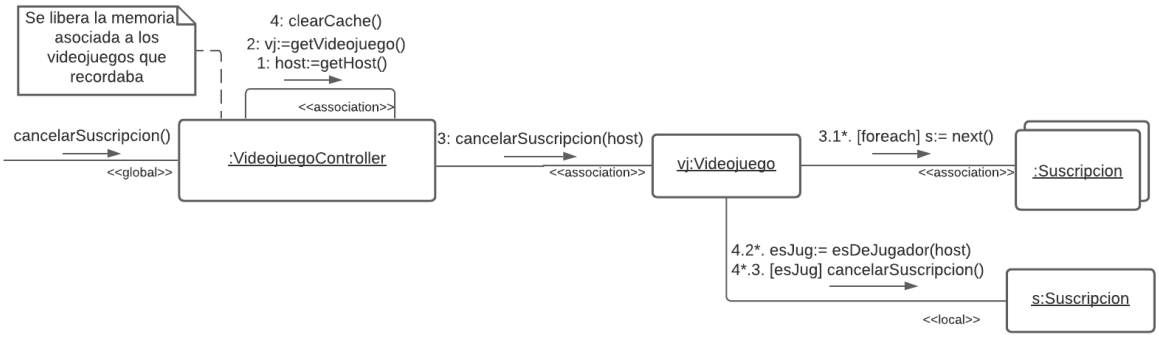
obtenerSuscripcionesVideojuegos(): Set(Set(DtVideojuego))



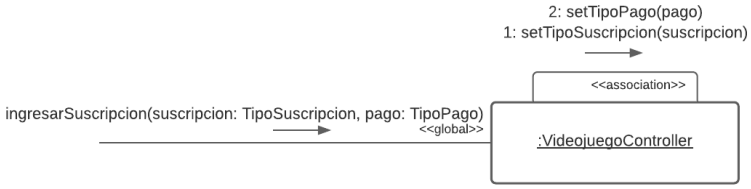
seleccionarVideojuego(String)



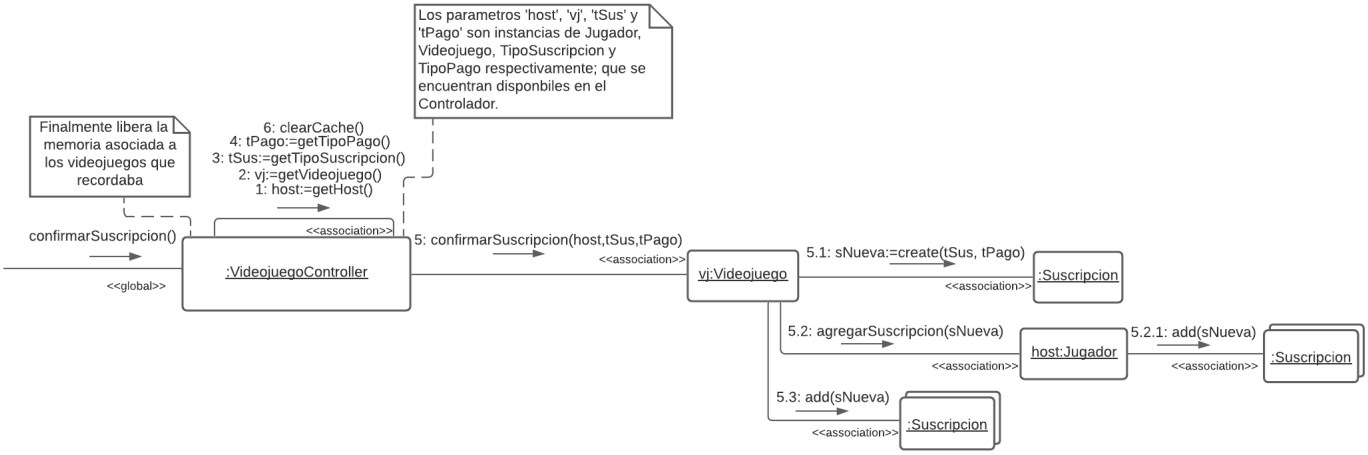
cancelarSuscripcion()



ingresarSuscripcion(TipoSuscripcion, TipoPago)

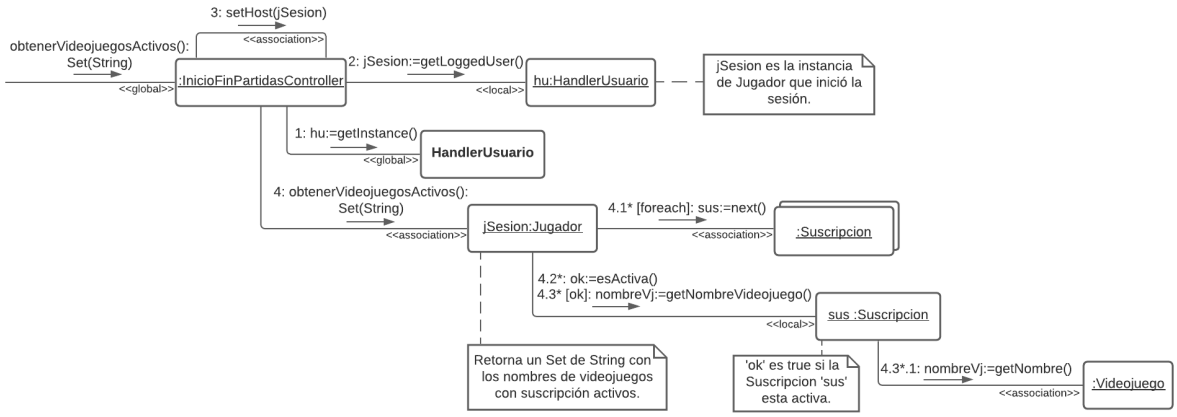


confirmarSuscripcion()



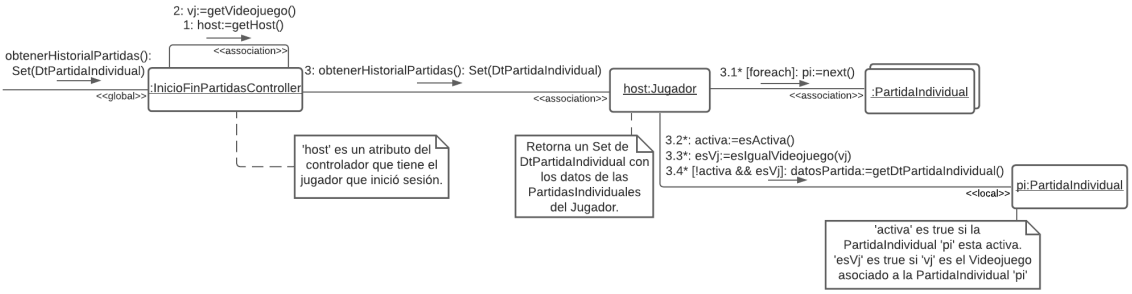
2.4. Iniciar Partida

obtenerVideojuegosActivos(): Set(String)

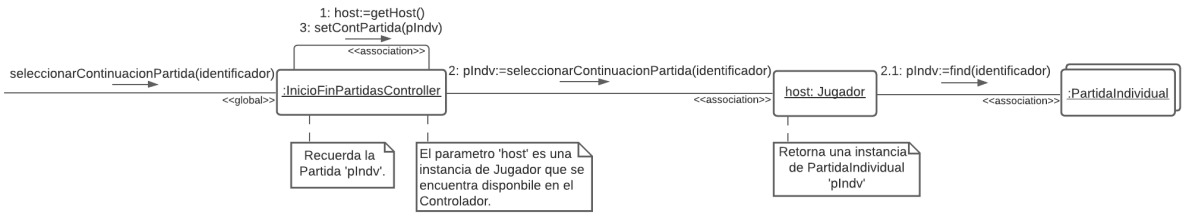


seleccionarVideojuegos(String)

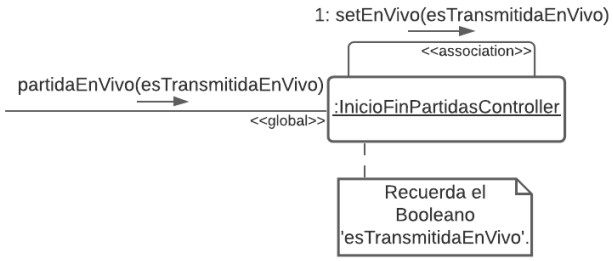
obtenerHistorialPartidas(): Set(DtPartidaIndividual)



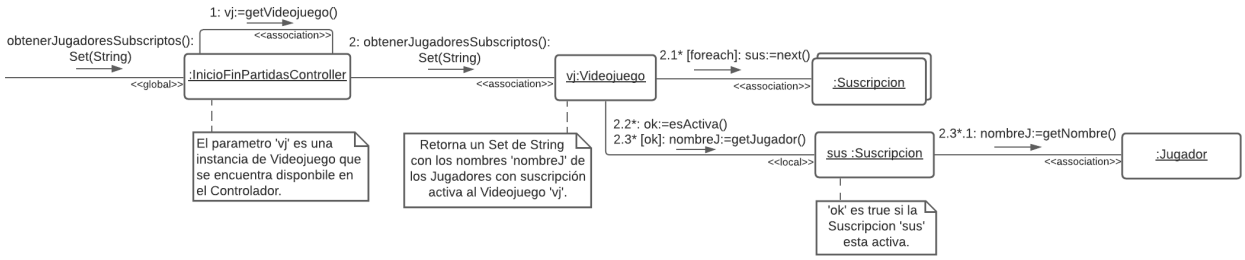
seleccionarContinuacionPartida(int)



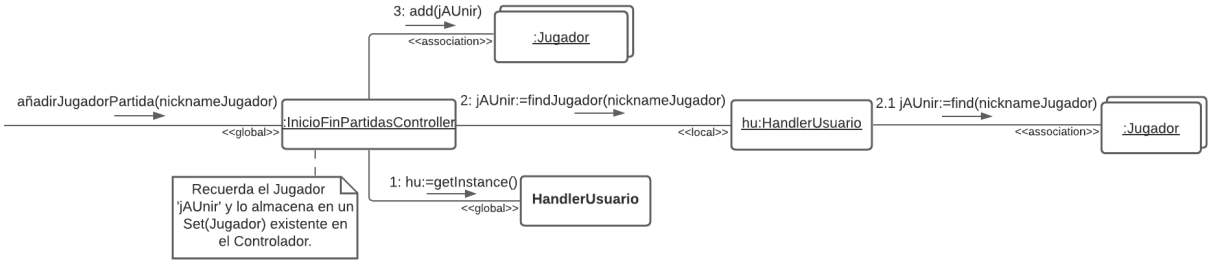
partidaEnVivo(bool)



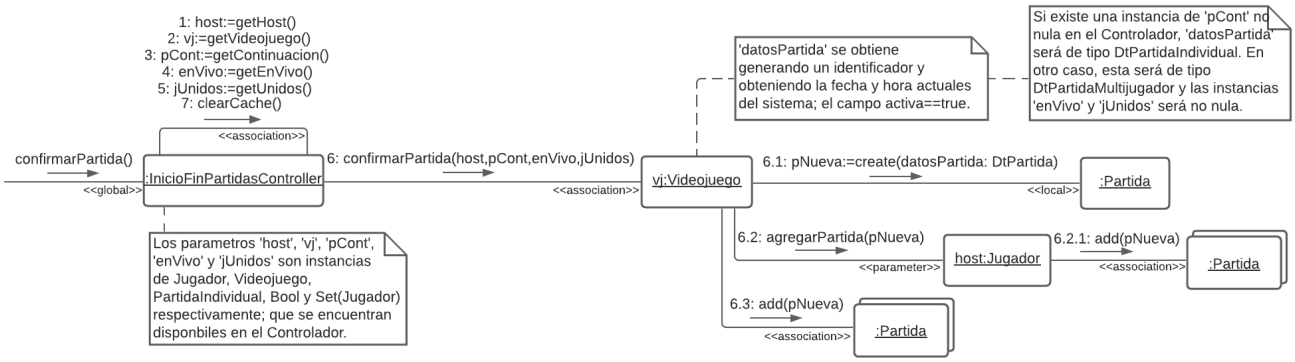
obtenerJugadoresSubscriptos(): Set(String)

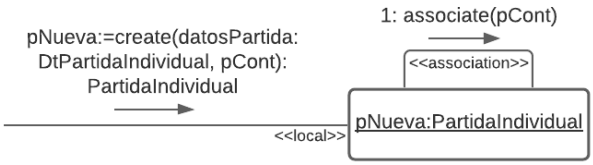


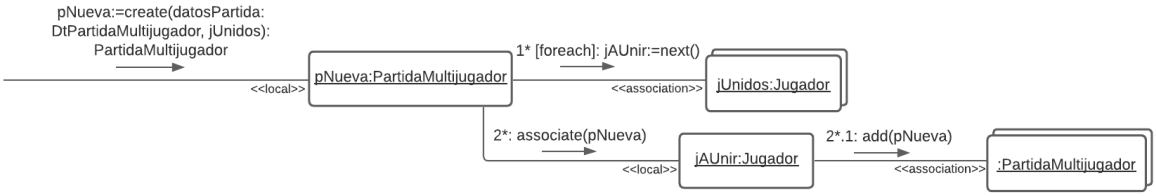
añadirJugadorPartida(String)



confirmarPartida()

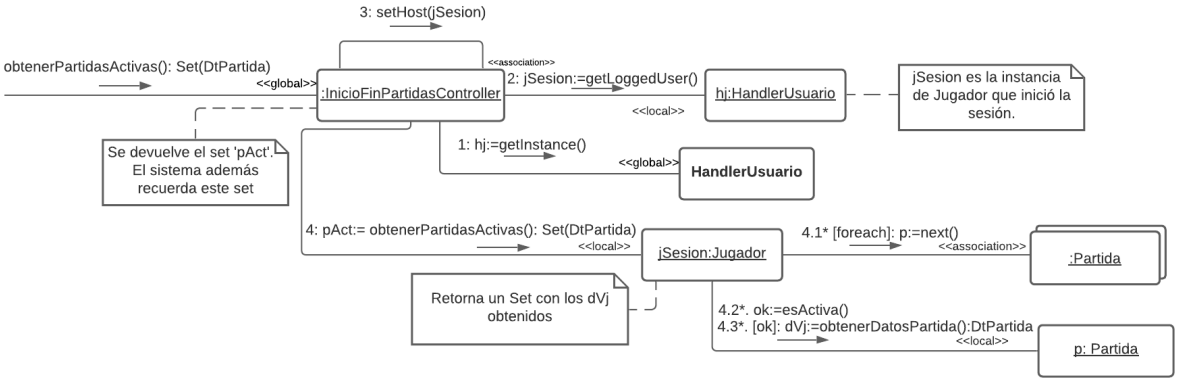




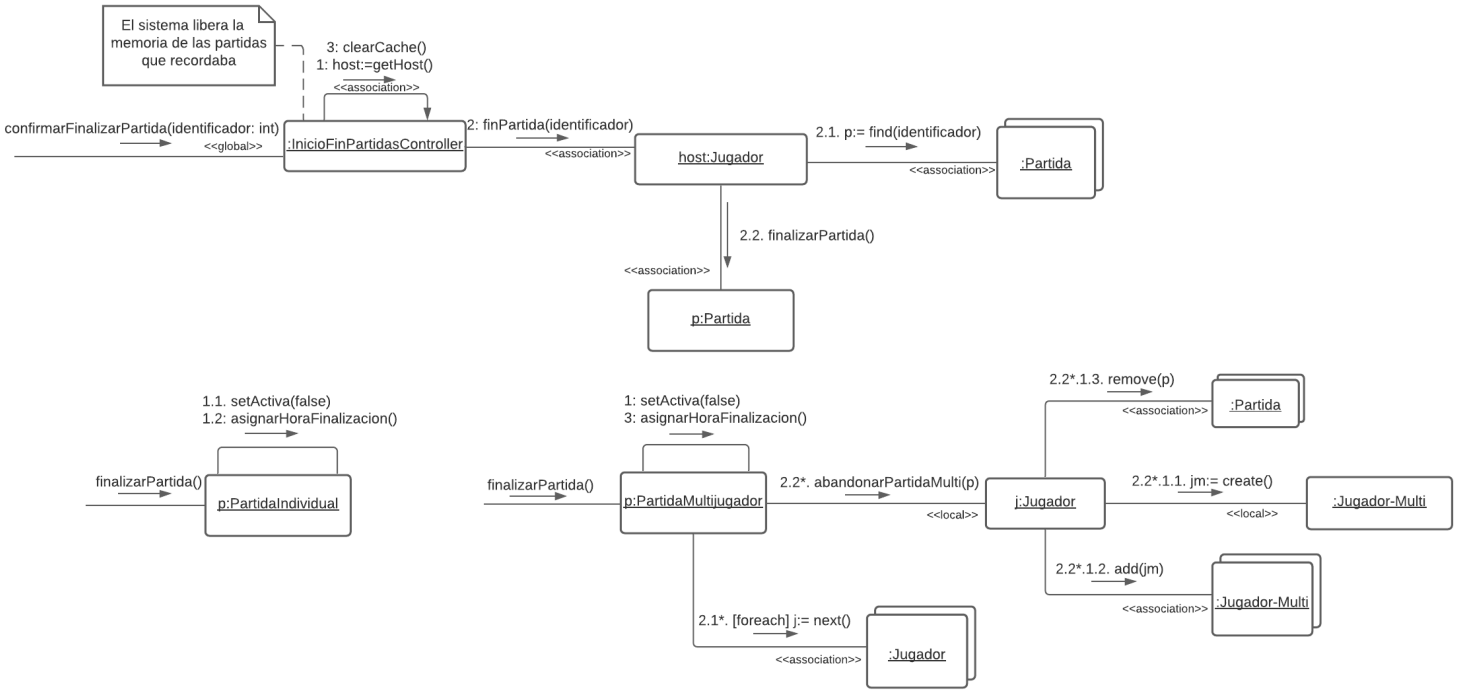


2.5. Finalizar Partida

obtenerPartidasActivas(): Set(DtPartida)

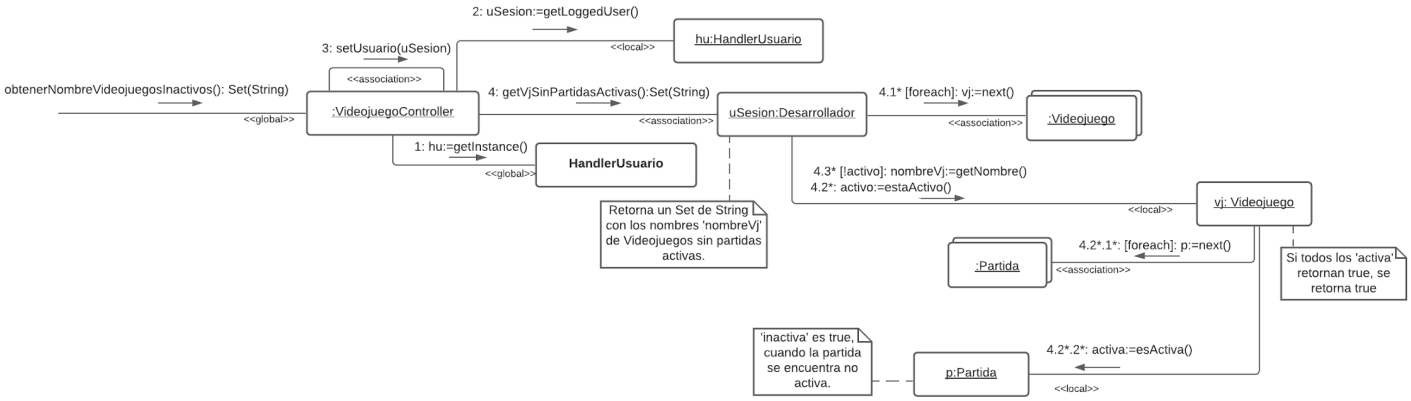


confirmarFinalizarPartida(int)

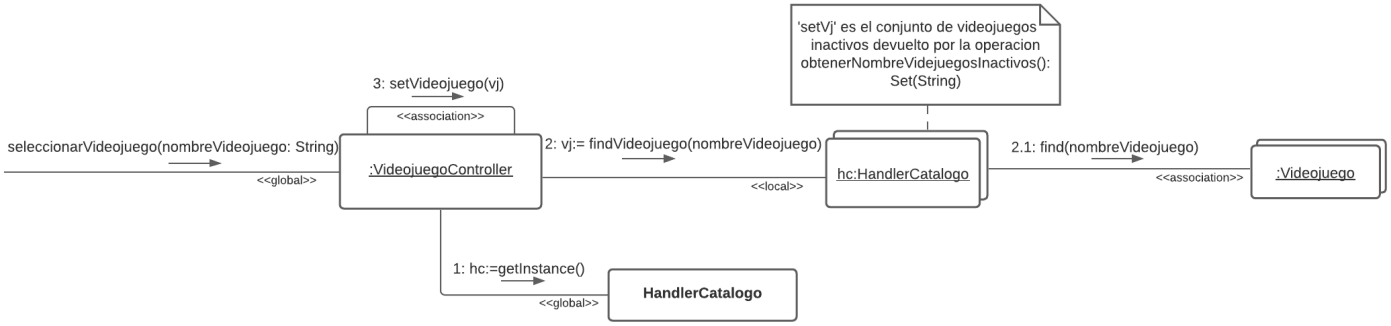


2.6 Eliminar Videojuego

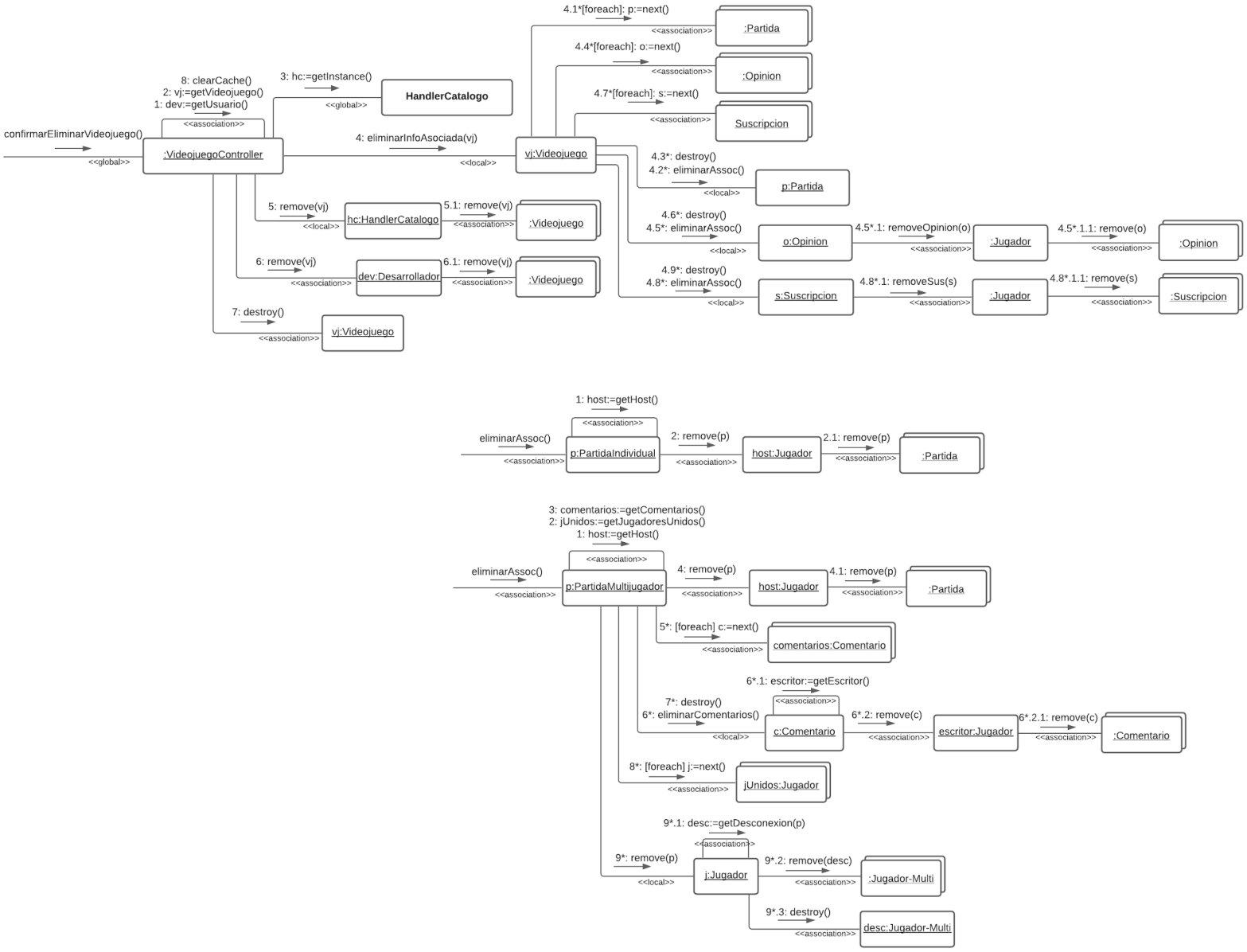
obtenerNombresVideojuegosInactivos(): Set(String)



seleccionarVideojuego(String)

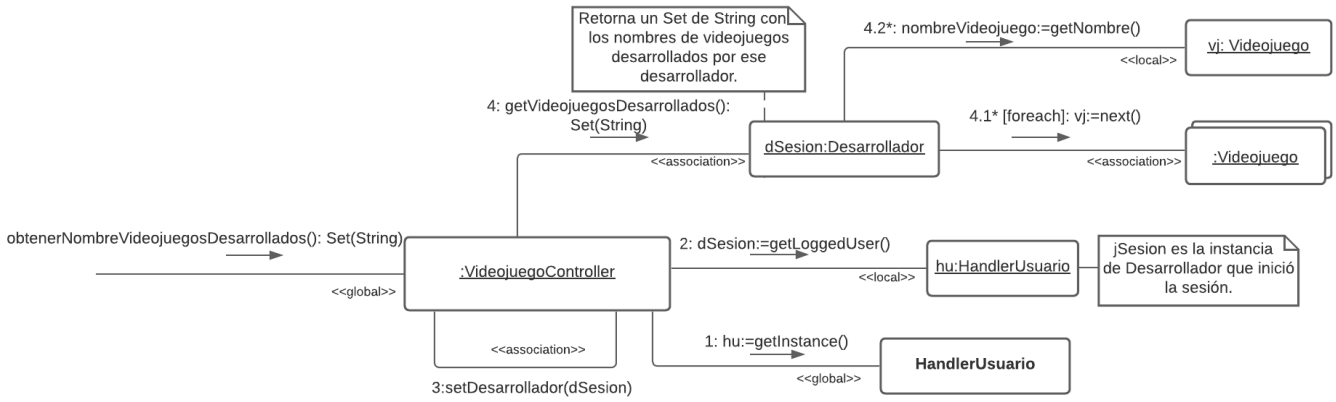


confirmarEliminarVideojuego()

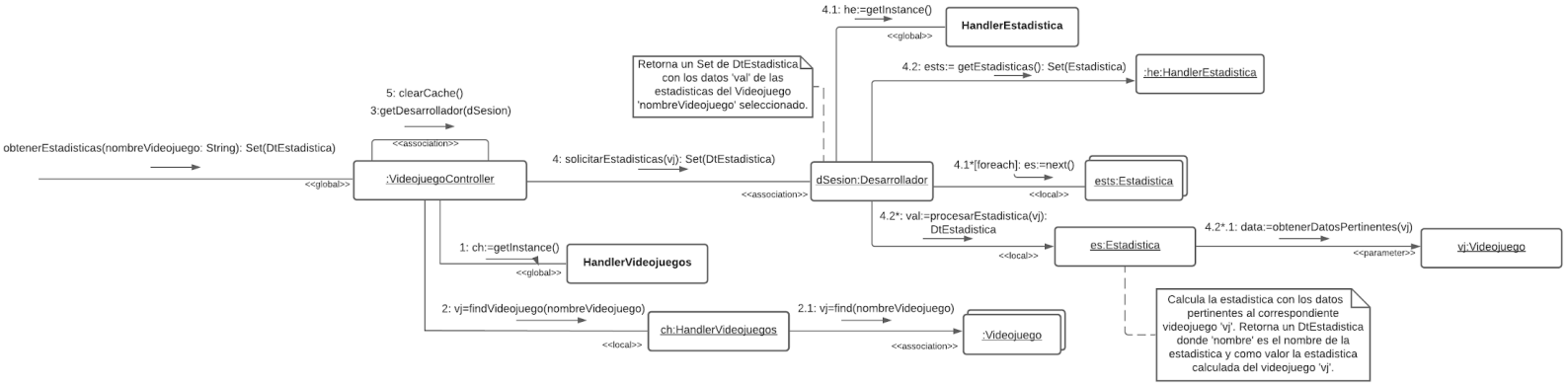


2.7. Consultar Estadísticas

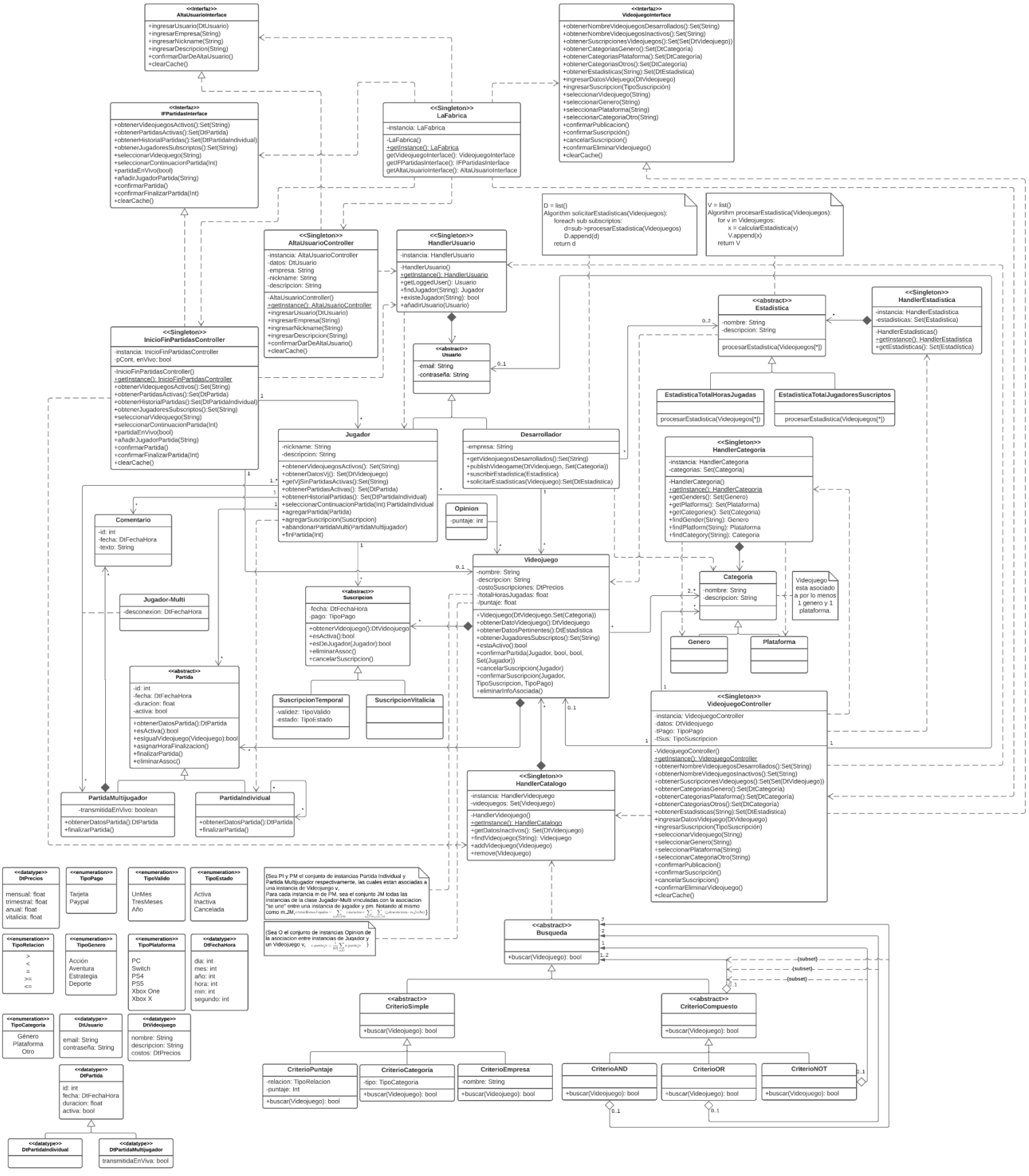
obtenerNombreVideojuegosDesarrollados(): Set(String)



obtenerEstadisticas(String): Set(DtEstadisticas)



3 Diagrama de Clases de Diseño



4 Criterios Generales

4.1. Criterios GRASP

Durante esta etapa de diseño, se tuvo en cuenta de forma general el uso de todos los criterios GRASP para la elaboración de los diagramas. Los siguientes párrafos muestran algunas puestas a uso destacadas:

Para cada caso de uso, diseñamos un manejador artificial de todas sus operaciones. Debido a la elevada cantidad de casos de uso solicitados, optamos por diseñar clases que implementan varios casos de uso relacionados. Por ejemplo, VideojuegoController contiene los casos de uso de publicar, eliminar, obtener estadísticas, suscribirse y desuscribirse a un videojuego. De esta manera mantenemos un balance entre la cohesión del controlador, realizando operaciones que tratan exclusivamente sobre las acciones a realizar sobre un videojuego y un bajo acoplamiento del controlador, ya que este tiene dependencia solamente con los manejadores (handlers) de videojuegos, estadísticas y usuarios. Todo lo anterior es una aplicación del criterio controller.

Se empleó el criterio de expert para asignar subresponsabilidades, ya sea desde un controlador a una clase más particular. Como ejemplo, tenemos el caso de que al querer iniciar una nueva partida en el sistema para la cual el controlador designado de este caso, IniciarFinPartidaController, relega la operación de crear la nueva instancia de Partida al objeto Videojuego (al cual se quiere iniciar), ya que este es el experto en determinar si esta puede ser iniciada, gracias a la información que posee.

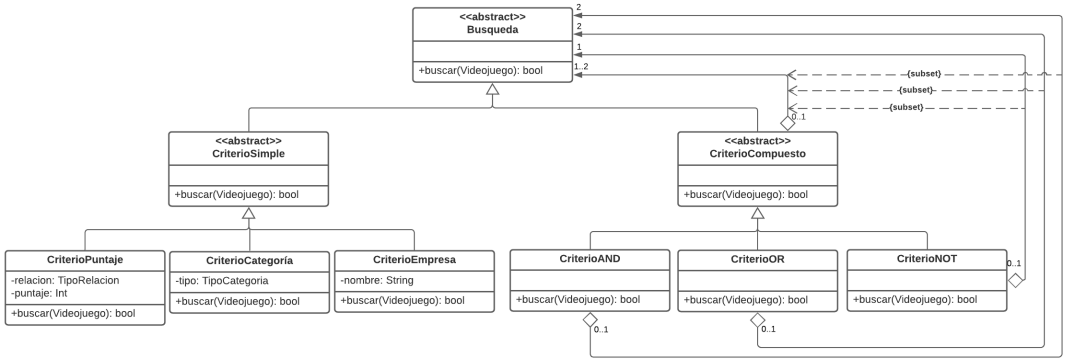
Utilizamos el criterio creator para definir quién es el responsable de crear los objetos del sistema. Por ejemplo, es el desarrollador el responsable de crear las instancias de videojuego ya que es el desarrollador el experto en información para ello, ya que para crear el videojuego se necesita asociar al mismo con el desarrollador que lo publicó. Otro caso es el de las instancias de videojuego, quien crea las instancias de partida. Como las partidas están relacionadas bajo composición con videojuego, el criterio creator sugiere que sea el anterior el responsable de crear las instancias de partida, lo cual es lo que efectivamente hicimos.

Para cumplir con el criterio de bajo acoplamiento, se tuvo en cuenta la implementación de manejadores y, como se mencionó anteriormente, que los controladores solamente dependan de estos.

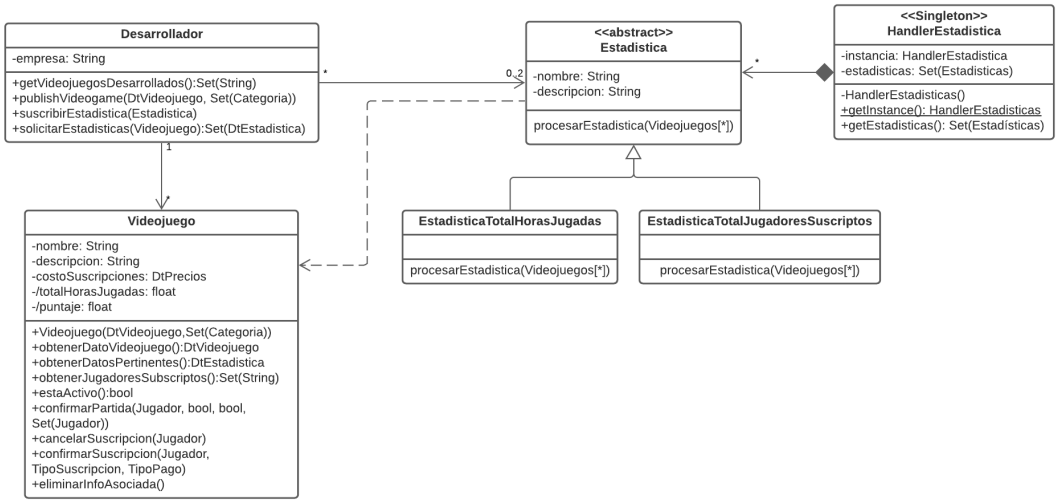
4.2. Design Patterns Utilizados

Para diseñar la funcionalidad de Búsqueda para Videojuegos se optó por aplicar el patrón de diseño “Composite”. Esto se debe gracias a la estructura arborescente que presentan los operadores lógicos AND; OR; y NOT, que coincide con el problema tipo del patrón. De esta manera podemos aplicarla a los criterios de búsqueda compuestos requeridos por el usuario, tomando a las búsquedas compuestas como componentes y a los criterios simples como hojas. Esto permite a los clientes tratar las búsquedas compuestas y simples de manera uniforme.

Fueron incluidas las etiquetas “subset” para lograr simular el hecho de que los operadores AND y OR son binarios, mientras que el operador NOT es unario. Esta solución cumple los requerimientos no funcionales al permitir añadir nuevos criterios simples con facilidad: Crear un criterio simple no es más que crear una nueva clase hija de Criterio Simple. Esto es análogo para nuevos criterios compuestos.



El comportamiento del desarrollador con las estadísticas fue modelado siguiendo el patrón Strategy. Permitiendo modelar el comportamiento de que un Desarrollador (Context) contenga una colección de estadísticas (Strategies) y cuando el desarrollador quiere conocer las estadísticas a las que él mismo seleccionó invoca la operación procesarEstadisticas de cada una de las estadísticas solicitando que calculen las estadísticas de los videojuegos correspondientes. Esta solución permite añadir a futuro nuevas estadísticas (Strategies) con facilidad. Solamente es necesario crear nuevas clases derivadas de estadística, cumpliendo con los requisitos no funcionales. Notar que agregar una nueva estadística no implica ningún cambio para las clases Jugador ni Videojuego (Suponiendo que videojuego ya tiene los datos necesarios para calcular la estadística)



Para asegurarnos que los controladores, los handler de las colecciones del sistema y la fábrica sean únicos utilizamos el patrón Singleton. Ya que debe haber una única instancia de cada uno de los anteriores y estas deben ser globalmente accesibles.

Finalmente utilizamos el patrón Factory para implementar la frontera entre la capa lógica y la de presentación del sistema. Junto a las interfaces de los controladores, el cliente utiliza la fábrica para obtener dichas interfaces sin la necesidad de depender de los controladores en sí mismos.