DISEÑO

Videojuego de Combate

Descripción breve

Diseño del Software que se desea desarrollar basado en el Documento de Análisis Funcional.

Iván Gutiérrez González [Analista Funcional][Jefe de proyecto]

Arturo Enrique Gutiérrez Mirandona [Analista Programador]

Jorge Andrés Echevarría [Ingeniero de desarrollo]

Víctor Bartolomé Letosa [QA]

0. Tabla de contenido

Contenido

Tabla de ilustraciones

1. Diagramas de clases

A continuación, presentamos los distintos diagramas de clases del proyecto, en ellos se podrá apreciar la distinta organización de los objetos que intervienen en el programa, así como las acciones que llevan a cabo en este, también se especificaran y explicaran las relaciones entre los distintos objetos.

1.1. Diagrama de clases 1

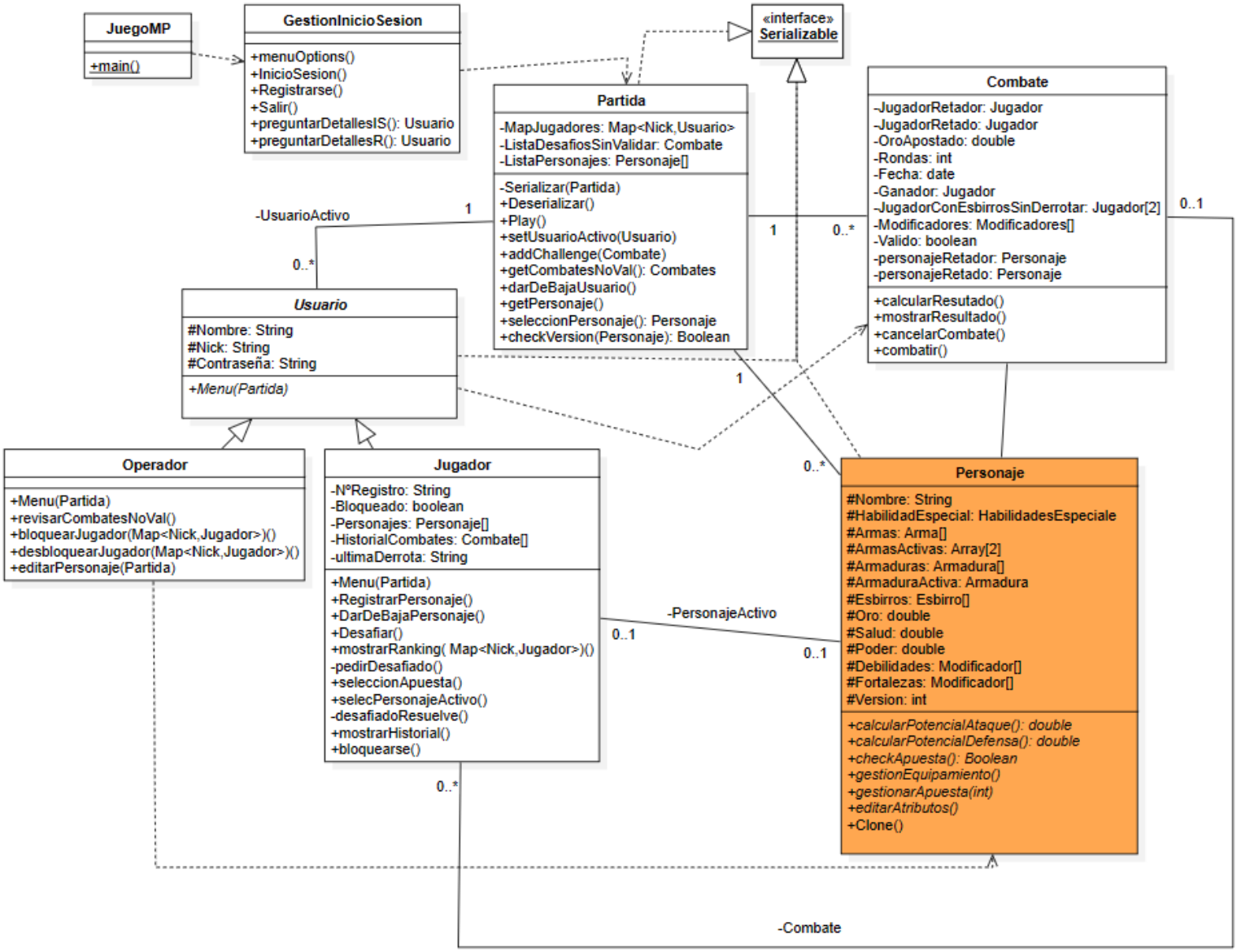

Imagen 1 Diagrama de clases 1

El Diagrama de clases principal contiene las clases más importantes que intervienen en el juego, con sus dependencias, herencias, asociaciones… entre ellas. A continuación, explicaremos individualmente cada clase que aparece en la ilustración I.

- *JuegoMP*: Clase encargada de iniciar el juego.
- *GestionInicioSesion*: Clase que se encarga de la gestión e inicio sesión de los usuarios, en ella se podrán registrar tanto los usuarios como los jugadores y por supuesto iniciar sesión para jugar, tiene acceso a ficheros con la información de los usuarios que conforman el juego.

- *Partida*: Esta clase gestiona el juego, contiene la información de la partida, tanto el usuario que se ha conectado, como la información de todos los jugadores existentes en el juego, también contiene una lista con los desafíos que faltan por ser validados por los operadores, esta clase se encarga de salvaguardar los datos principales del juego con cada partida iniciada. Esta clase llama al método **menú(partida)** de su Usuario activo.
- *Usuario*: Clase que contiene la información y de los distintos usuarios del juego, esta es la clase padre de 2 subclases que son *Operador* y *Jugador*, cada una contiene la información exclusiva de su tipo de usuario, así como las acciones que pueden realizar en el juego, recordado que el operador se encarga de gestionar el juego y el jugador de desafiar a otros jugadores. De los métodos de esta clase hay que destacar **menú(partida)**, que muestra todas las acciones que puede realizar el usuario en concreto, además la clase *Jugador* tiene métodos para calcular el ranking que siempre que quiera verlo se realizara la ordenación de todos los jugadores. Los *Jugadores* tiene un atributo combate que en caso de tener un desafío pendiente y validado por algún operador se guarda en dicho atributo, para que el *Jugador* lo resuelva al entrar a jugar.
- *Combate*: Contiene la información de los combates entre jugadores, y en ella por supuesto se realizan estos, esta clase también es manipulada por los operadores antes de que se produzca el combate. Esta clase llama a distintos métodos del personaje activo de cada jugador para poder realizar el combate.
- *Personaje*: Esta tiene la información de los personajes, y se asocia con los usuarios de tipo jugador, contiene los atributos de cada personaje, así como las acciones que se pueden realizar sobre ellos como puede ser la de elegir equipo (armas y armadura).
- *Interfaz serializable*: Mencionamos por último esta interfaz con la que se relaciona la partida, los usuarios y los personajes, esta interfaz es necesaria para hacer que la información del juego sea persistente entre una partida y otra.

1.2. Diagrama de clases 2

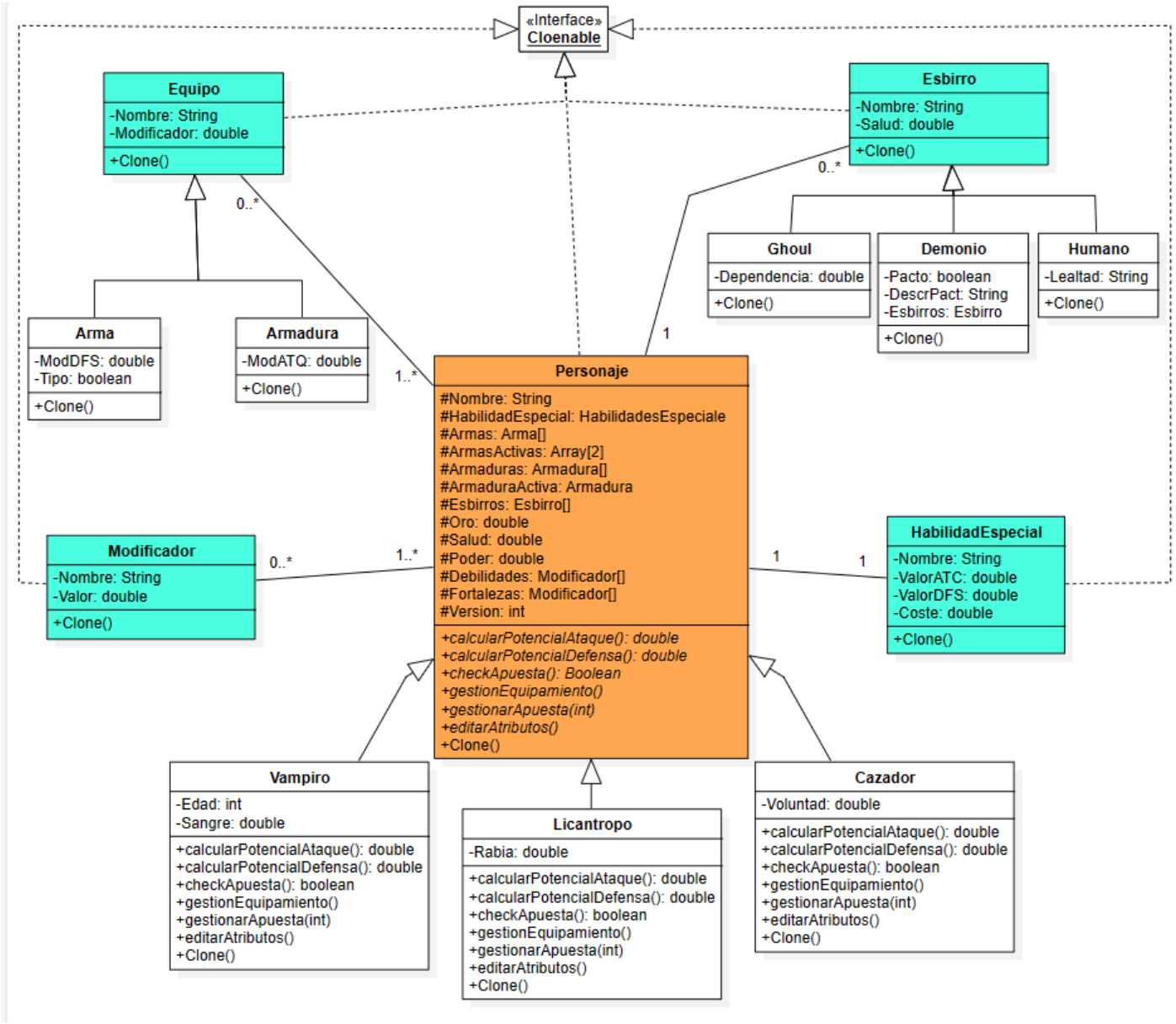

Imagen 2 Diagrama de clases 2

En el diagrama de clases que aparece en la ilustración 2 presentamos la relaciones que tiene la clase *Personaje* mencionada anteriormente con otras clases que únicamente tiene relación con esta y no con las clases antes mencionadas, nuevamente explicamos detalladamente clase a clase.

- *Personajes*: Clase que contiene la información y acciones de los personajes y que además de esta se heredan otras 3 clases como son la clase *Vampiro*, *Licántropo* y *Cazador*, que son tipos de personajes cada uno con acciones y características diferentes entre sí pero que comparten todos los atributos y métodos que aparecen en la clase padre, al tener características distintas algunos métodos como calcular los potenciales de ataque y defensa deben ser diferentes entre tipos de personajes.
- *HabilidadEspecial*: Cada *Personaje* tiene una, esta cuenta con una serie de atributos que ayudan al *Personaje* en los combates.

- *Esbirro*: Cada personaje tiene un número indefinido de ellos, se pueden diferenciar tres tipos, los *Demonios*, los *Humanos* y los *Ghouls* que son diferentes clases, todas heredadas de *Esbirro*.
- *Modificador*: Cada *Personaje* tiene varios modificadores unos buenos y otros malos, estos afectan a los atributos de los personajes a la hora de los combates.
- *Equipo*: Clase padre de la que se heredan 2 clases, las *Armas* y las *Armaduras*, cada *Personaje* cuenta con un conjunto de ambas y puede elegir la que desea para tener un distinto desempeño en los combates.

- *Interface cloneable*: Usamos esta clase para poder clonar a cada jugador su correspondiente personaje, en caso de que el operador modifique algún aspecto del personaje este se clone en vez de modificarlo (siempre antes de clonar guardamos los atributos individuales de cada relación jugador-personaje).

2. Diagramas de actividad

Para entender bien el funcionamiento en ciertas partes clave de la aplicación hace falta explicarlas mediante diagramas de actividad. Estas partes clave son el funcionamiento a grandes rasgos de la aplicación, iniciar sesión, registrarse, retar a otro jugador y los combates.

2.1. Operativa del programa

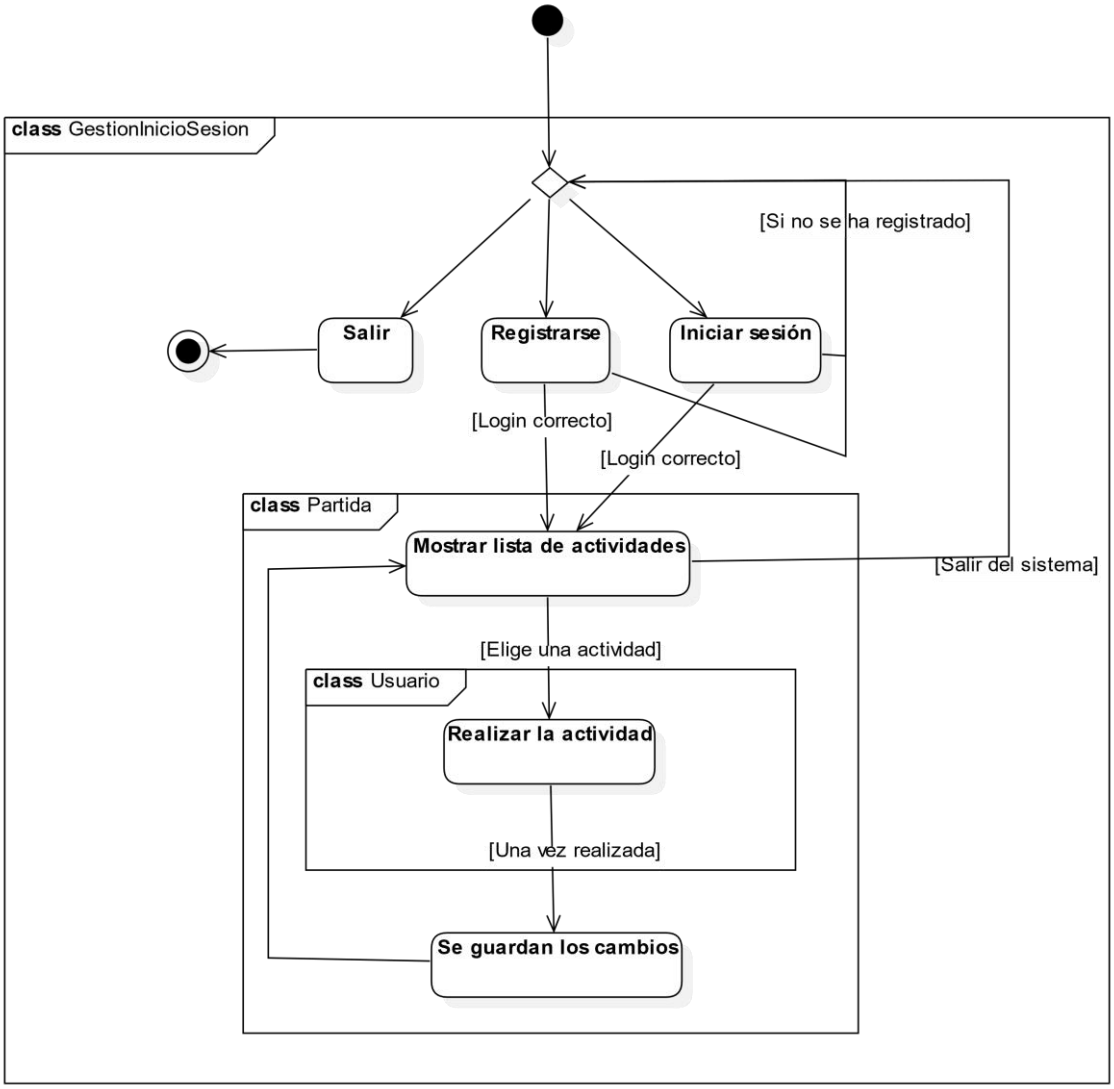

Imagen 3 Diagrama de actividad de operativa del programa

Este diagrama de actividad representa como va a ser en general el programa en su conjunto. Primero se le mostrará un menú al usuario donde ha de iniciar sesión o registrarse y en caso de que no consiga identificarse de manera correcta o se haya equivocado en la elección del primer menú se podrá volver hacia atrás. Una vez hecho el login correcto se le mostrará una serie de actividades, distintas en función de si se es jugador u operador. Cuando elige la actividad que desea realizar se lleva a cabo y se guardan los cambios. Y así hasta que el usuario decida salir del sistema y en ese caso se cerrará sesión y volverá a la pantalla de inicio para volver a iniciar sesión o registrarse con otra cuenta o cerrar el programa.

2.2. Registrarse

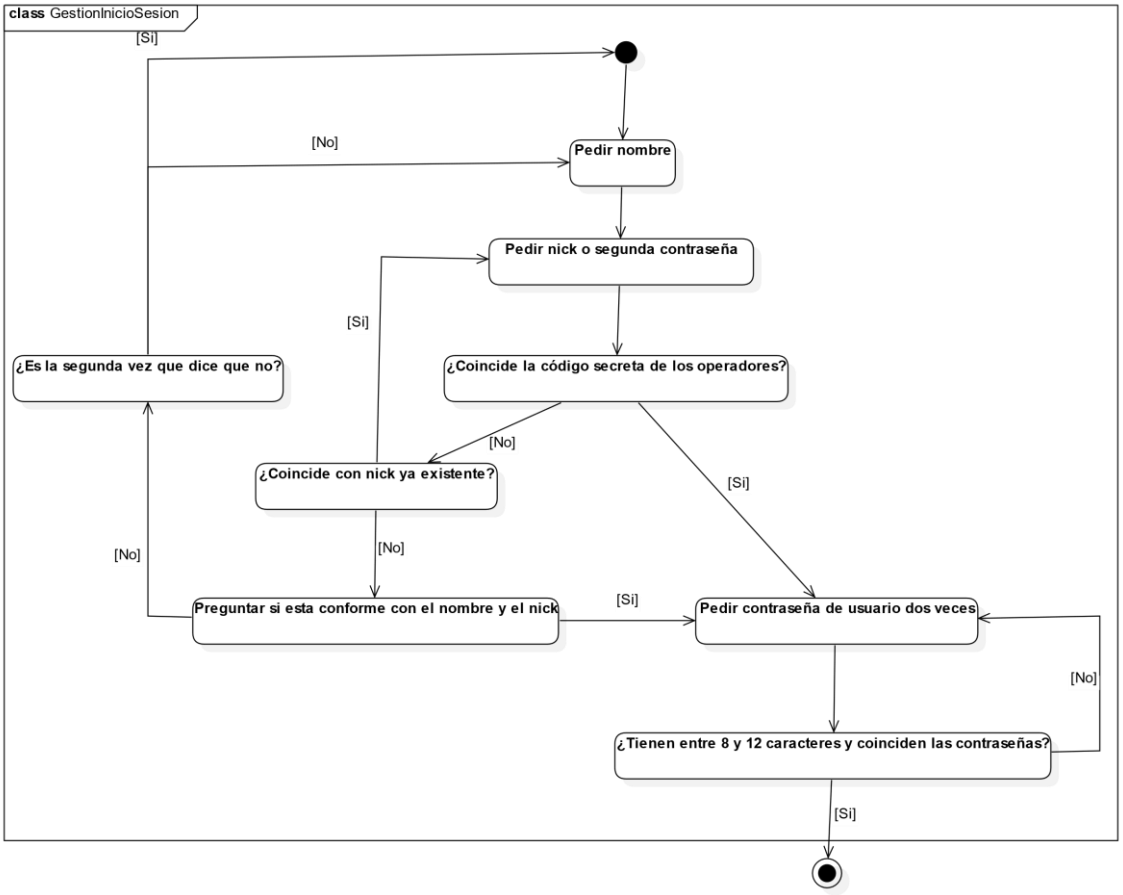

Imagen 4 Diagrama de actividad de registrarse

Ahora pasamos a ver el diagrama de actividad que muestra cómo se realizará el registrarse por primera vez un usuario. Primero se pedirá el nombre y después se pedirá el nick o el código secreto que se habilitará para operadores, en caso de que no coincida con este código se supondrá que es un jugador y los caracteres introducidos serán su nick. Se deberá comprobar si ese nick ya existe y en tal caso volver a pedir uno válido y si no se pedirá confirmación al jugador sobre si está conforme con su nombre y nick elegido. En caso de que se haya preguntado dos veces por la confirmación de nick y nombre se entenderá que el usuario quiere salir y se le volverá a mostrar el menú inicial. Por último, tanto para jugadores como para operadores, se les pedirá una contraseña y confirmar esa contraseña. En caso de que no cumpla con la condición de tener entre 8 y 12 caracteres o que no coincidan las dos contraseñas se volverán a pedir hasta que cumplan las condiciones.

2.3. Iniciar sesión

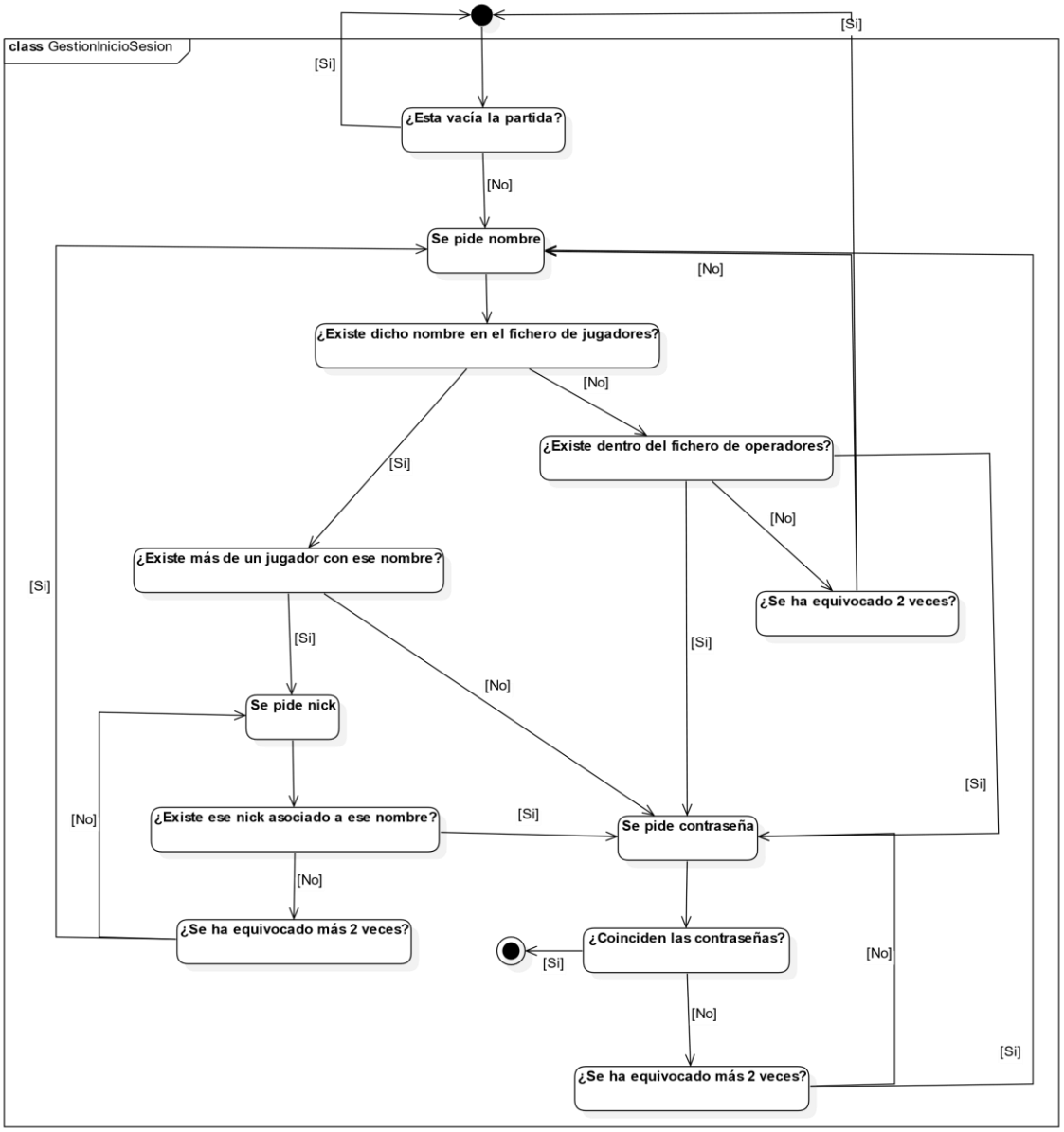

Imagen 5 Diagrama de actividad de iniciar sesión

Además de explicar cómo registrarse explicamos cómo iniciar sesión. Primero habrá que comprobar si la instancia de partida que serializamos se ha guardado o por el contrario es la primera vez que se ha entrado al juego y entonces no será posible iniciar sesión. Después al igual que al registrarse se pedirá primero el nombre y se comprobará si existe en el fichero que guarda los datos de los jugadores (se guarda en el mapa de jugadores de la instancia de partida que se serializa), si no está aquí se buscará en el fichero de texto de los datos de los operadores. En caso de que no exista en ninguno de los dos se comprobará si es la segunda vez que se equivoca, si es que si se le mostrará el menú inicial, y si no volverá a introducir su nombre. Cuando introduzca un nombre válido como jugador se comprobará si hay dos jugadores con el mismo nombre y se le pedirá el nick para distinguirlos. Una vez identificado el jugador u operador se le pedirá la contraseña para poder loguearse correctamente. Tanto cuando se pide el nick o la contraseña si se equivoca dos veces se volverá a pedir el nombre y si no se volverá a pedir nick o contraseña.

2.4. Retar otros jugadores

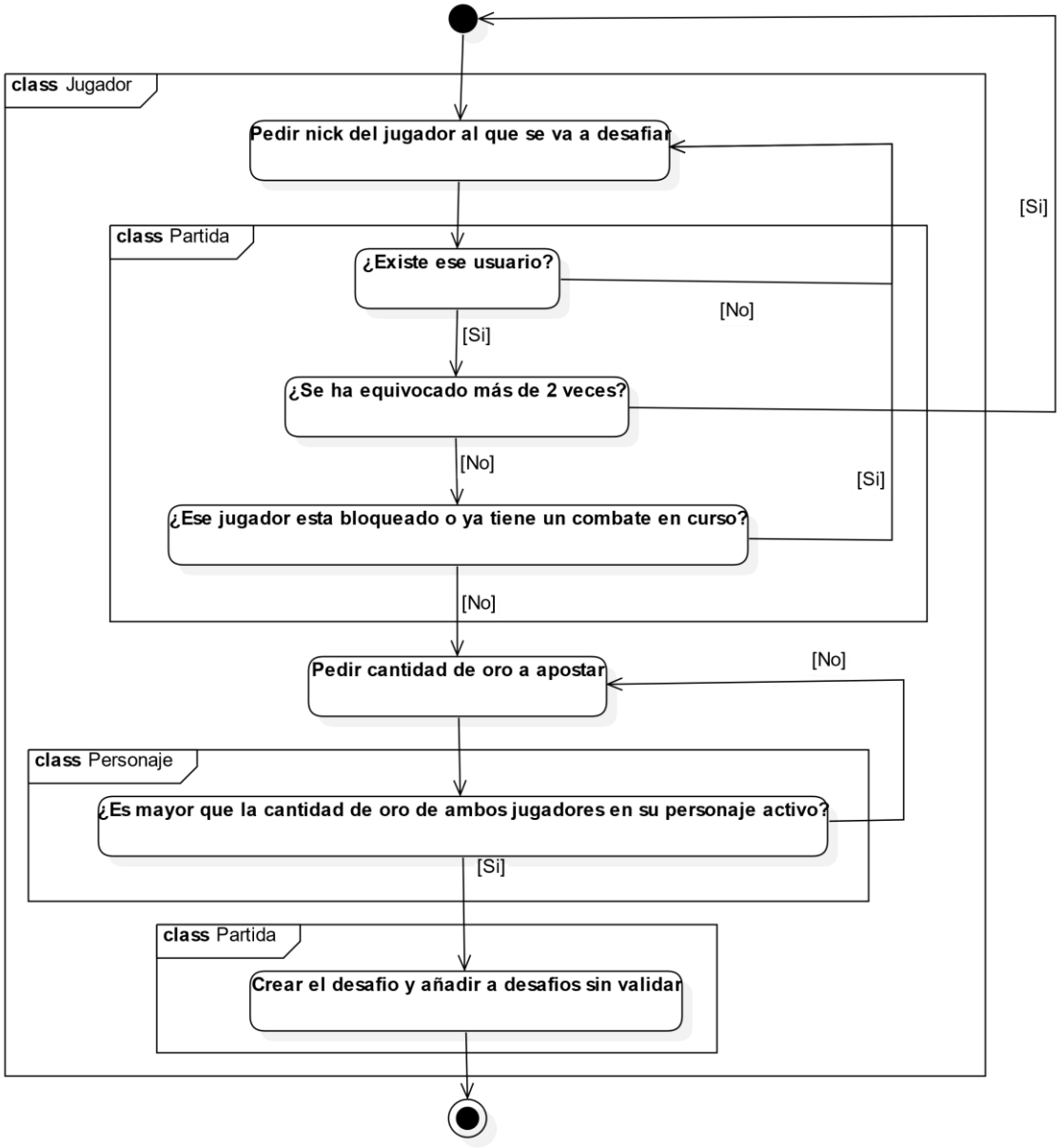

Imagen 6 Diagrama de actividad de retar a otro jugador

Entrando en la operativa de las actividades del jugador está la de retar a otro jugador. Se empieza pidiendo el nick del jugador al que se va a desafiar y se comprueba si existe. En caso de que no exista se comprobará si es la segunda vez que se equivoca si es que si se volverá al menú del jugador y si no se volverá a pedir el nick. En caso de que exista el nick se verificará si está bloqueado o tiene ya un combate en curso. Si es que no se pedirá otro nick y si es que si se le pedirá al jugador desafiante una cantidad de oro que quiera apostar. Después se comprobará que los personajes principales de ambos jugadores tengan esa o una mayor cantidad de oro, volviendo a pedir la cantidad de oro en caso de que no se lo puedan permitir. Y por último se creará el desafío con las características recogidas y se añadirá a la lista de desafíos sin validar a la espera de que un operador lo valide.

2.5. Combates

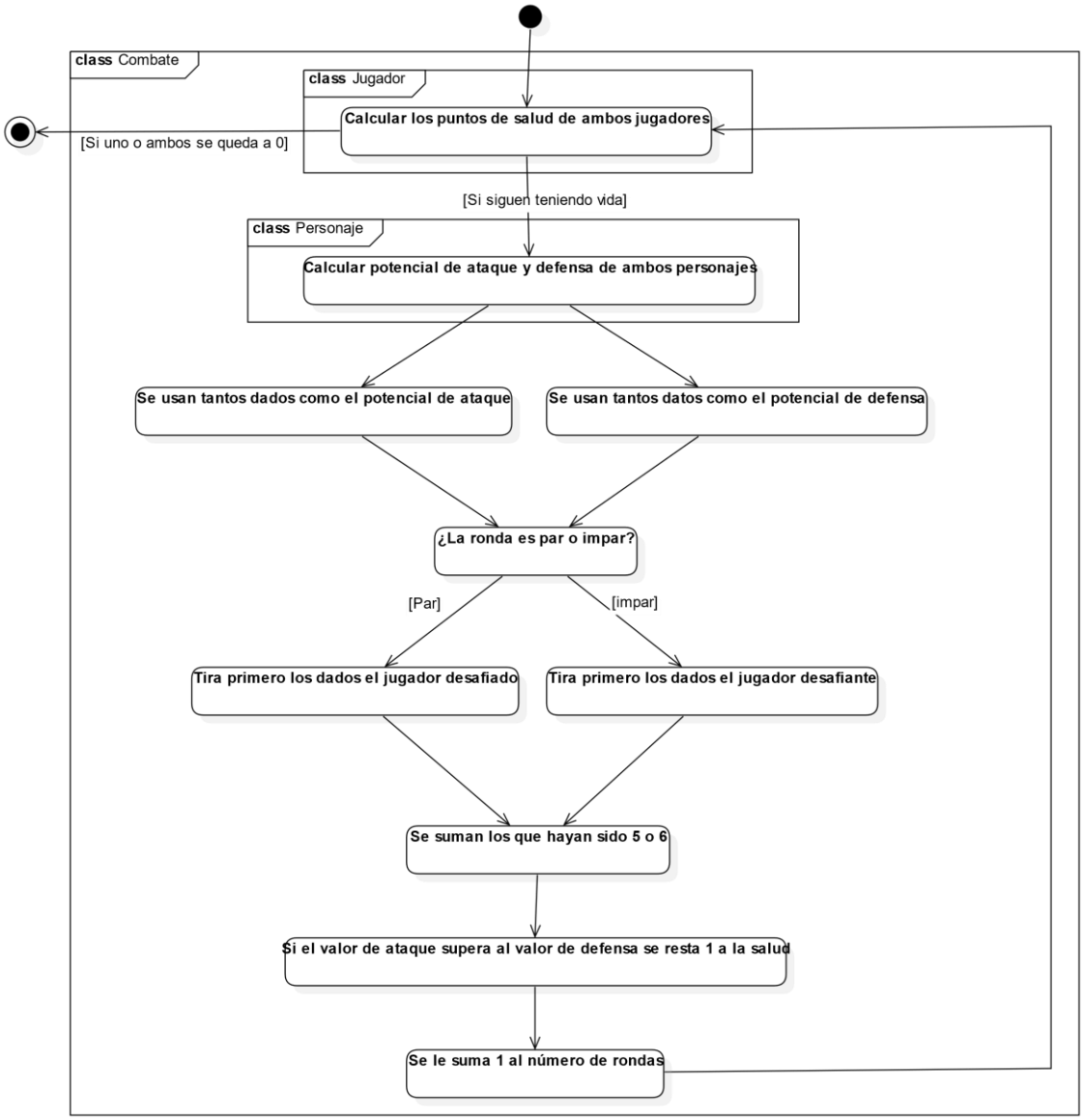

Imagen 7 Diagrama de actividad de los combates

En el último diagrama de actividad veremos cómo se realizarán los combates. Los combates consistirán en un bucle que comprueba la vida total de los personajes (solo los personajes porque primero se resta a los esbirros por lo que se quedarán sin vida antes que los personajes), si se queda un jugador o ambos sin vida en su personaje se termina el combate. Si aún tienen vida se calcula el potencial de ataque y defensa de sus personajes más el de los esbirros si quedan. Si es una ronda par primero tira los dados el jugador desafiado y si es impar empieza el jugador desafiante. Después se tiran tantos dados como potencial de ataque y defensa de ambos personajes y los que salgan 5 o 6 se consideran éxito sumando su valor. Como da igual el orden al atacar empieza el jugador desafiante y después el desafiado, y si el valor de ataque supera al de defensa se le resta uno de salud a los esbirros si quedan y si no a los personajes. Y por último se aumenta en uno la ronda y se vuelve a iniciar el bucle.

3. Diagramas de secuencia

En la presente sección, se presentarán diagramas secuenciales con el propósito de esclarecer inquietudes y orientar el análisis detallado de la interacción entre clases. Estos diagramas, esenciales en el desarrollo de software, nos ofrecen una representación gráfica precisa del flujo de los métodos y la comunicación entre objetos. Con el objetivo de proporcionar una guía y disipar cualquier ambigüedad, se abordarán varios ejemplos de diagramas secuenciales. A través de imágenes visuales y textos de apoyo, se delineará el camino lógico de ejecución, permitiendo una comprensión de las relaciones entre clases.

3.1. Diagrama de secuencia 1

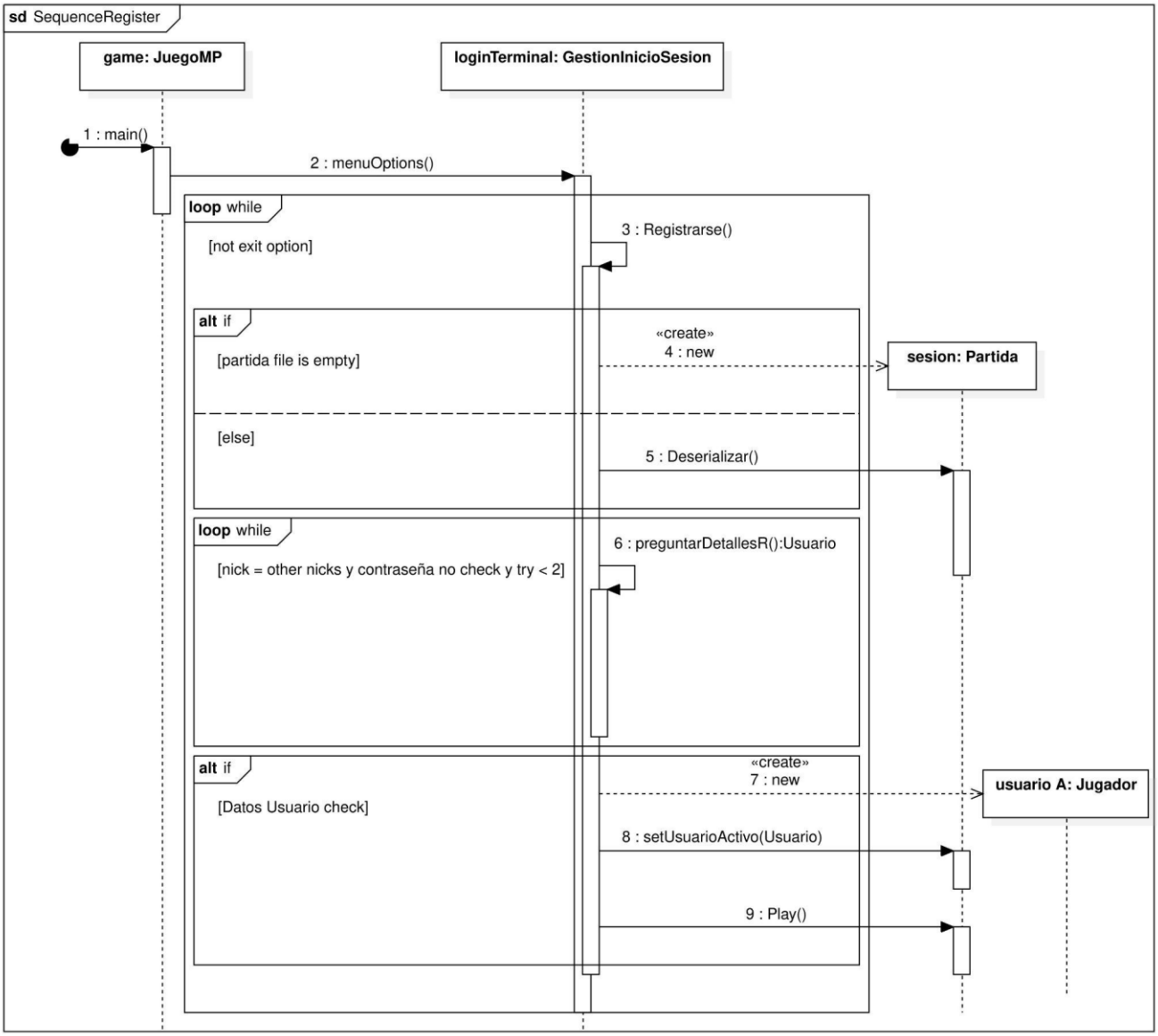

Imagen 8 Diagrama de registro

En el diagrama presente se muestra el flujo que se sigue a la hora de registrar un nuevo usuario al sistema. Hay que destacar que en esta operación tanto operadores como jugadores comparten la gran mayoría del recorrido, exceptuando la verificación de la contraseña adicional para los operadores, para asegurar la seguridad de los privilegios en el juego.

Podemos observar una llamada inicial al método ***menuOptions()*** perteneciente a la clase ***GestionInicioSesion***. En esta llamada se hará un display de un menú similar a la *ilustración 22*. Donde, los usuarios al escoger la opción de registrarse, llamarán al método que gestiona esta operación (***registrarse()***). Dentro de este método en función de si es la primera vez que se abre el juego o no, se creará la instancia de partida y se inicializará o se deserializará una partida previamente guardada. Posteriormente, se invocará el método ***preguntarDetallesR()*** que tendrá la responsabilidad de recaudar todos los datos para crear un jugador o un operador nuevo, con los datos específicos que cada uno requiere para su creación. Destacar que como se muestra en el

bucle, no se permiten nicks repetidos ya que esta es la identificación de cada usuario. Por último, si todos los datos han sido introducidos acorde a los requisitos, se creará un nuevo objeto del tipo que se haya escogido para registrarse (en este caso un jugador). Se seleccionará como usuario activo en la partida y se iniciará el método ***play()*** que encamina al menú de posibilidades una vez iniciada sesión.

3.2. Diagrama de secuencia 2

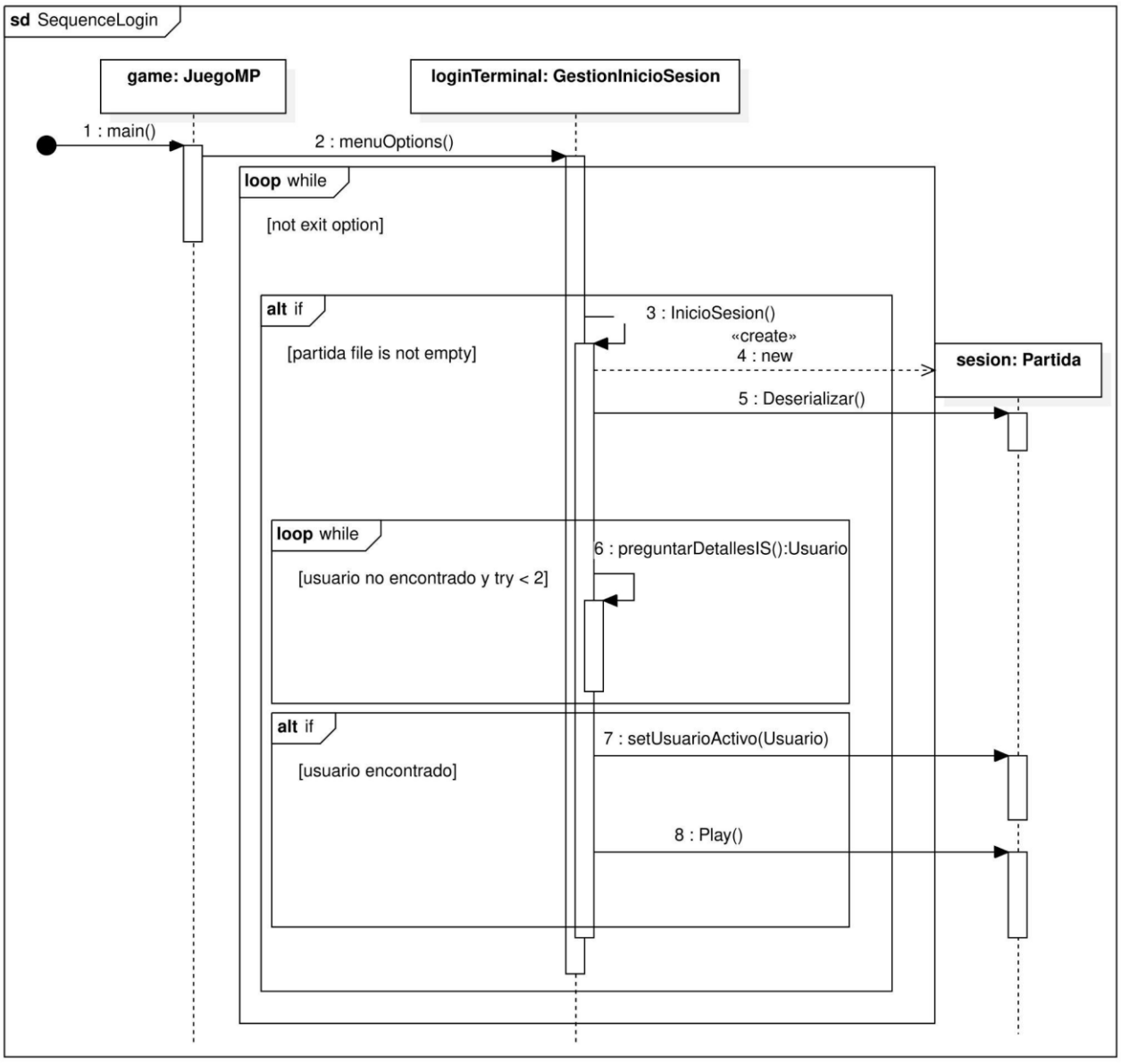

Imagen 9 Diagrama de inicio de sesión

En este diagrama se muestra en profundidad la opción alternativa al apartado anterior, el inicio de sesión. Destacar que no será posible iniciar sesión si se está bloqueado ni ejecutar esta acción en el caso de que no existan usuarios a los cuales iniciar sesión, es decir, si "usuarios" se encuentra vacío.

De lo contrario, si un usuario desea iniciar sesión se deserializará la partida cargando así los datos de los usuarios. Posteriormente, se preguntará al usuarios las credenciales de inicio de sesión únicas de su cuenta en el método ***preguntarDetallesIS()*** y si se ha encontrado un usuario en menos de ciertos intentos. Como en el caso de registrarse se seleccionará como usuario activo en la

partida y se iniciará el método ***play()*** que encamina al menú de posibilidades una vez iniciada sesión. Dicho método revisará en los atributos del jugador si tiene un combate ya realizado en su ausencia y si es el caso mostrará el resultado de este.

3.3. Diagrama de secuencia 3

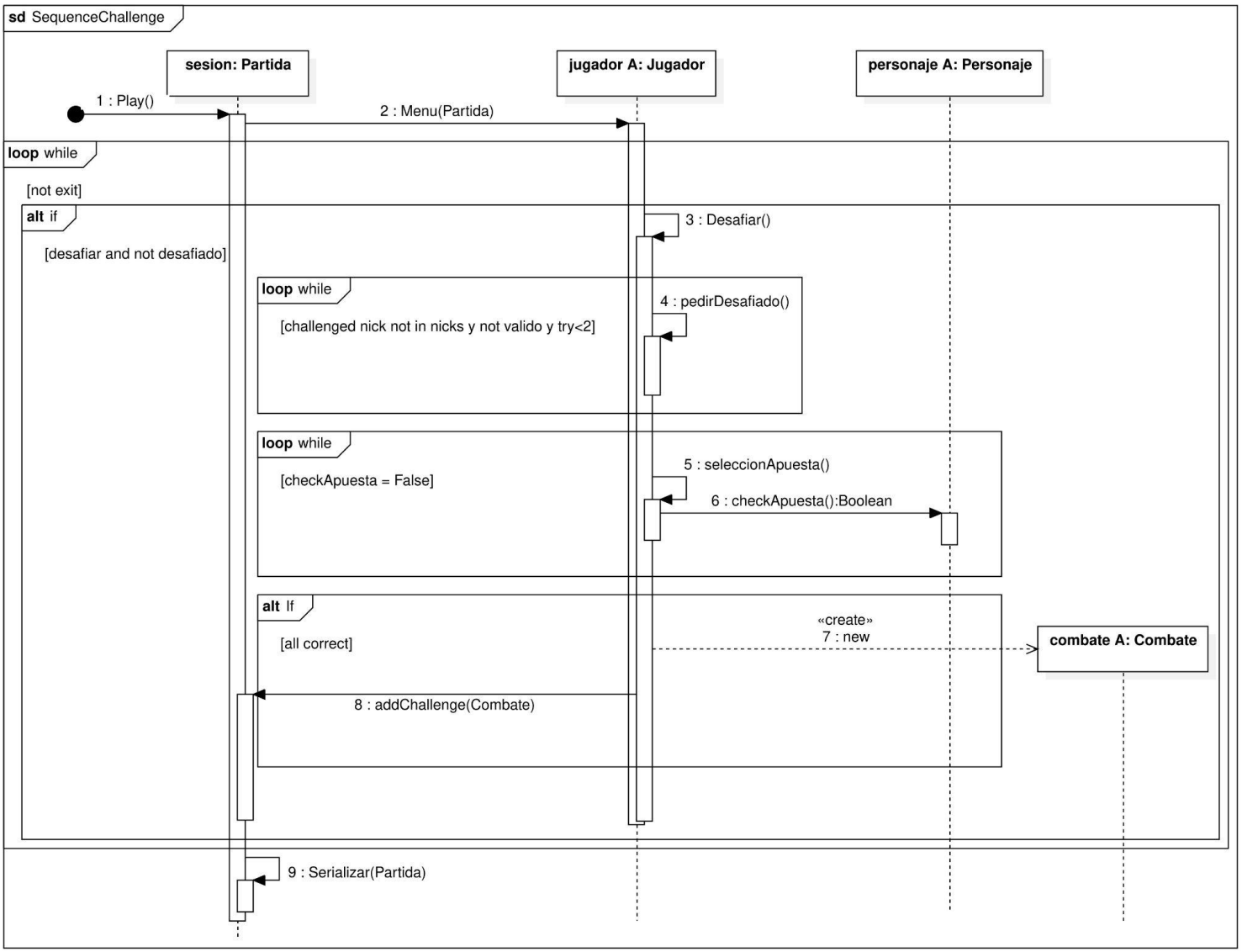

Imagen 10 Diagrama de desafiar

En la ***ilustración 10*** se muestra la secuencia de acciones a la hora de que un jugador rete a otro en combate. Destacar que el método ***menu()*** juega un crucial papel en el programa ya que es el encargado de encauzar a los usuarios por su correcto flujo en función de la acción escogida.

Recordar, tras haber realizado el proceso de inicio de sesión, se llamará al método ***play()*** que a su vez llamará al menú de actividades disponibles. Se entrará en un bucle que permite al usuario realizar cualquier actividad hasta que decida la opción de salir de él. En este ejemplo, se escoge la opción de desafiar (se llama al método ***desafiar()***). Una vez aquí, se preguntará por el nick del jugador a desafiar en ***pedirDesafiado()***, se comprobará que el otro jugador no esté bloqueado ni que haya perdido hace menos de 24 horas. A continuación, se intentará determinar una cantidad a apostar. En este tramo, se escoge una apuesta monetaria, esta cantidad será comprobada en el personaje activo (***checkApuesta()***) en caso de no ser posible se repetirá este tramo. De lo

contrario, se seguirá el flujo creando una nueva instancia de combate referente al desafío, a la espera de validación por parte de un operador.

Remarcar, que tras acabar el bucle de ***menu(),*** es decir, una vez seleccionada la opción de salir, se serializará automáticamente la partida para cumplir con el compromiso de fiabilidad ante caídas del sistema y evitar la pérdida de datos.

3.4. Diagrama de secuencia 4

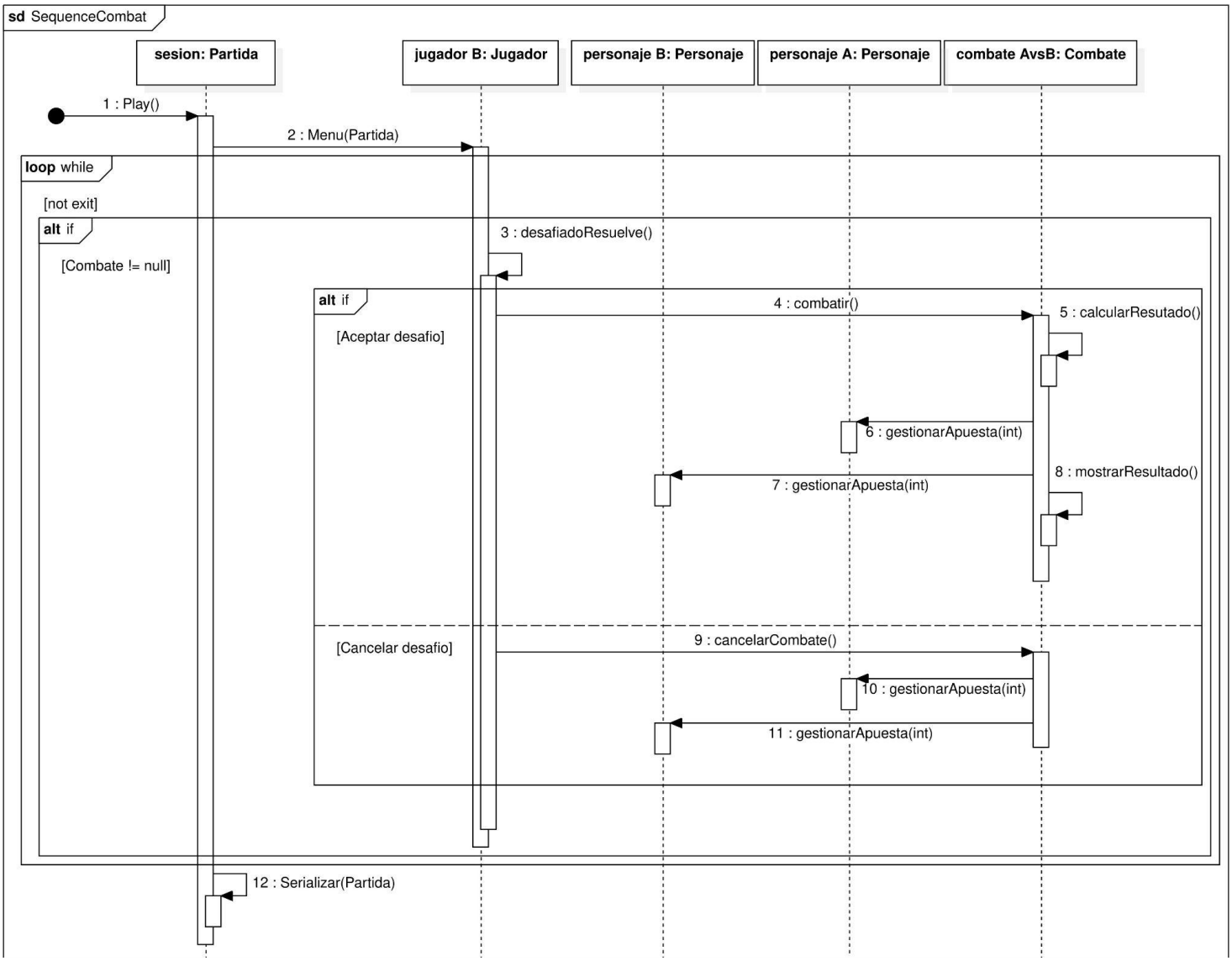

Imagen 11 Diagrama de combate

En este subapartado trataremos la operación de combatir. Quiero recordar que esta acción sólo ocurre si un jugador es desafiado. Este estará forzado a no realizar ninguna otra acción antes del combate.

En primer lugar, volvemos a encontrar ***menu()***. Que, como mencionado anteriormente, en este caso encaminará al jugador para resolver el combate que tiene pendiente (ya validado por un operador). Por lo que, se llamará al método encargado de esto en el objeto **Jugador**, ***desafiadoResuelve()***. Este dará la posibilidad de aceptar/rechazar el desafío.

En caso de aceptarlo, se llamará al método responsable del combate en la clase **Combate** que llamará a otro método que calculará el resultado del combate siguiendo el diagrama de la ***ilustración 7.*** Se llamará al método ***gestionarApuesta()*** que se encargará de sumar o restar la cantidad de oro correspondiente a cada personaje según su resultado en el combate. Por último, se mostrarán los resultados del combate por pantalla.

En caso de rechazarlo, se llamará al método encargado de borrar ese combate, cobrar y pagar las partes correspondientes a cada personaje por huir del desafío.

Destacar de nuevo, que tras acabar el bucle de ***menu()*** se serializará automáticamente la partida para cumplir con el compromiso de fiabilidad ante caídas del sistema y evitar la pérdida de datos.

3.5. Diagrama de secuencia 5

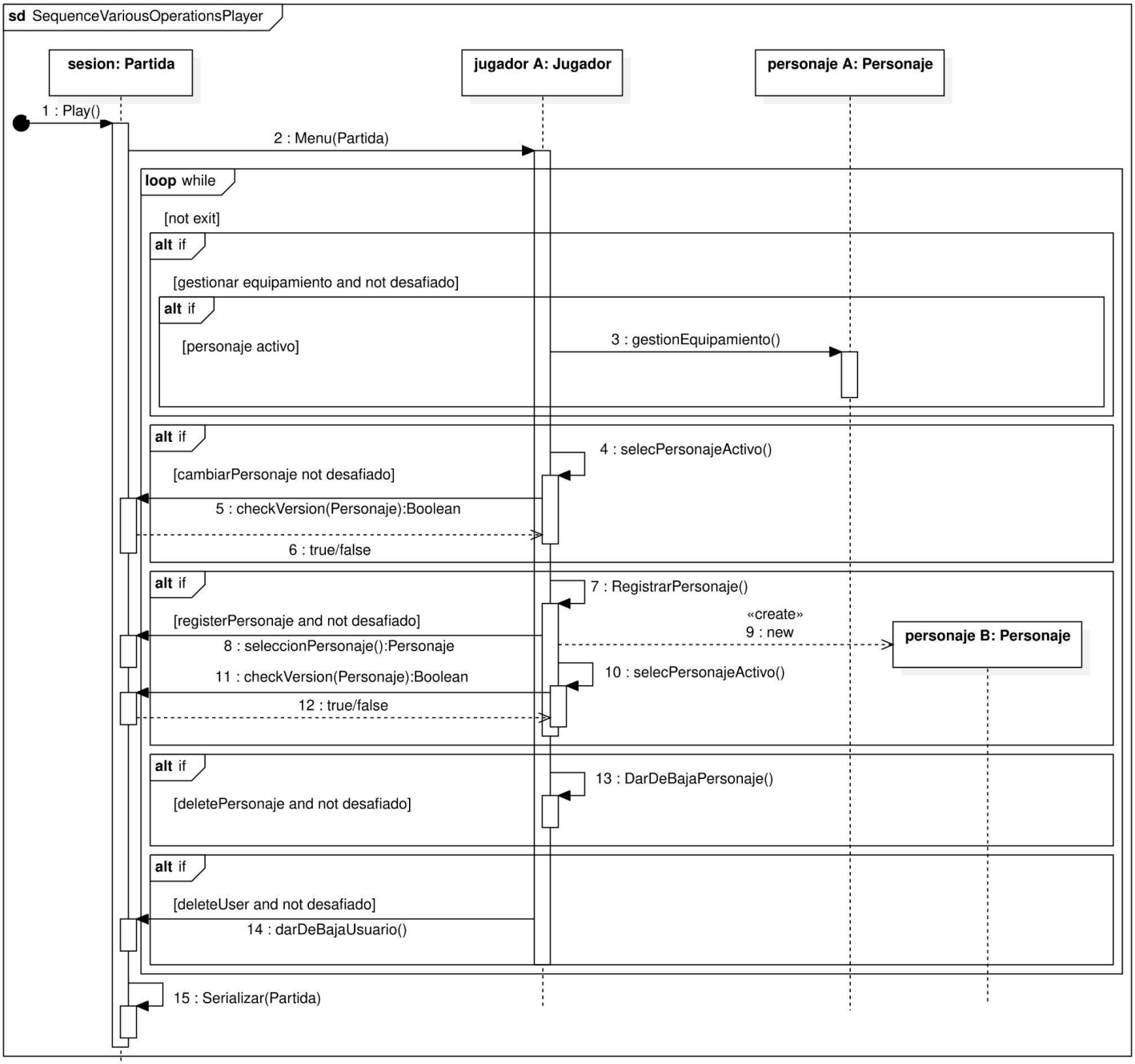

Imagen 12 Diagrama de operaciones adicionales de jugador

En la ilustración superior se muestran los diversos caminos que se siguen a la hora de realizar diferentes operaciones de menor tamaño por parte del jugador. Destacar de nuevo, que todas ellas divergen de ***menu()*** al igual que las operaciones anteriores. Por otro lado, remarcar que como se puede observar en sus condicionales, no está permitido realizarlas sin haber resuelto un desafío pendiente en caso de tenerlo.

En primer lugar, encontramos la posibilidad de gestionar equipamiento del personaje activo, está, llamará al método ***gestionEquipamiento()*** que tiene la responsabilidad de posibilitar el cambio de armas y armaduras dentro de cada personaje.

A continuación, se muestra la posibilidad de cambiar de personaje activo (***selecPersonajeActivo()***), destacar que este método eligirá automáticamente el personaje en caso de tener solo uno en el jugador. Observar la comprobación de las versiones de los personajes para asegurar la última actualización.

En tercer lugar, se permite al jugador registrar un personaje nuevo. Destacar, que se le pide a partida que muestre las distintas posibilidades de personaje puestas a disposición por los operadores. De esta, el jugador escogerá una y se creará un duplicado de ese objeto con la versión actual del personaje para posteriormente asegurar que la versión del personaje es la última. Posteriormente se permitirá elegir o no ese personaje como personaje activo.

En cuarto lugar, se permitirá dar de baja un personaje si es requerido en el método ***darDeBajaPersonaje()***.

Por último, se dará la posibilidad de dar de baja un jugador. Destacar la interacción con partida para borrar ese jugador de la lista de usuarios (***darDeBajaUsuario()***). Se añadirá en implementación doble confirmación para evitar posibles problemas o despistes.

En este bucle ***menu()*** se encontrarán otras posibilidades no ilustradas como ***mostrarRanking()*** que pasándole la lista de los jugadores obtenida de partida generará un ranking en función del oro obtenido. O ***mostrarHistorial()*** que mostrará el historial completo de los combates del jugador y sus resultados.

Destacar de nuevo, que tras acabar el bucle de ***menu()*** se serializará automáticamente la partida para cumplir con el compromiso de fiabilidad ante caídas del sistema y evitar la pérdida de datos.

3.6. Diagrama de secuencia 6

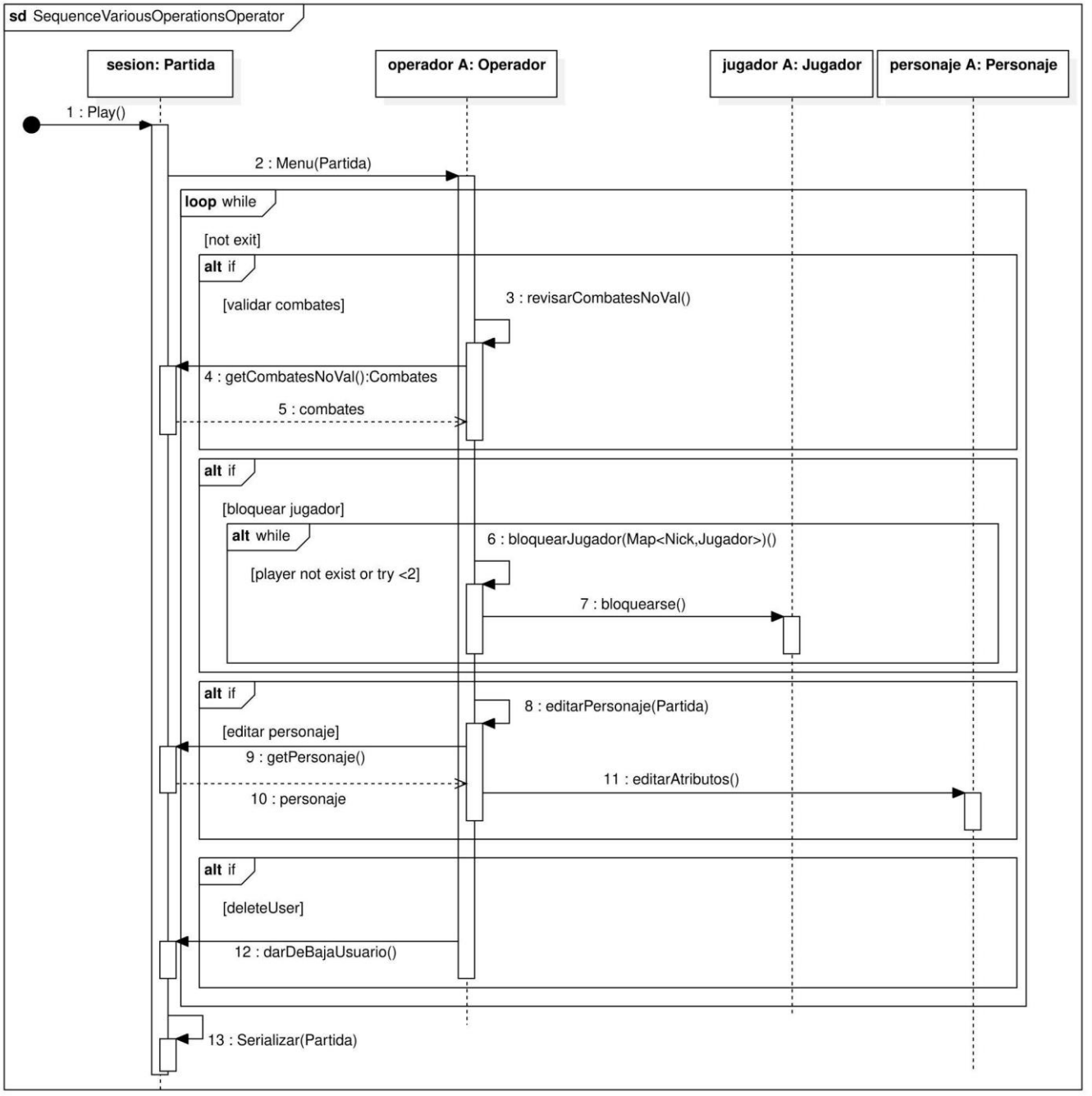

Imagen 13 Diagrama de operaciones adicionales de operador

En este último diagrama de secuencia, se mostrarán las operaciones relacionadas con los operadores. Destacar que se emplea el mismo ***menu()*** solo que debido a la herencia presentará otra implementación para satisfacer los requisitos de los operadores, aunque en esencia se presenta la misma estructura.

En primer lugar, se materializa la opción de validar combates. Esto se realizará pidiendo a partida en ***getCombatesNoVal()*** la lista completa de combates no validados y se irán presentando uno a uno al operador para dar su visto bueno.

En segunda instancia, se observa la posibilidad de bloquear un jugador que se salga de los estándares establecidos por la organización. Esto se realizará buscando en la lista de usuarios obtenida de partida el jugador por su nick y bloqueandolo. De forma simétrica ocurrirá en caso de querer desbloquear un jugador. El propio método ***bloquearse()*** gestionará las dos posibilidades distintas.

En tercer lugar, se posibilitará a los operadores la opción de editar aspectos de los personajes disponibles o crear nuevos de los tipos disponibles. Esto se realizará obteniendo el personaje exacto a cambiar de la lista de personajes en partida (***getPersonaje()***) y una vez obtenida la referencia se llamará al método ***editarAtributos()*** que es responsable de cubrir todas las posibilidades de personalización requeridas. Destacar que será de suma importancia incrementar el atributo de versión asociado a cada personaje puesto que luego se comprobará que todos los personajes, al ser seleccionados como personaje activo de los jugadores, tengan la última versión de su modelo.

Por último, se dará la posibilidad de dar de baja un operador. Destacar la interacción con partida para borrar ese operador de la lista de usuarios (***darDeBajaUsuario()***). Se añadirá en implementación doble confirmación para evitar posibles problemas o despistes.

Destacar de nuevo, que tras acabar el bucle de ***menu()*** se serializará automáticamente la partida para cumplir con el compromiso de fiabilidad ante caídas del sistema y evitar la pérdida de datos.

4. Diagramas de estados

Con la finalidad de entender a detalle el funcionamiento del sistema, se proporcionarán diagramas de estado ya que estos nos proporcionan una representación visual de los diferentes estados que pueden experimentar los elementos del sistema, así como las transiciones entre estos estados, lo que facilita la comprensión del flujo de interacción y las posibles ramificaciones en cada etapa.

4.1. Diagrama de estado del registro

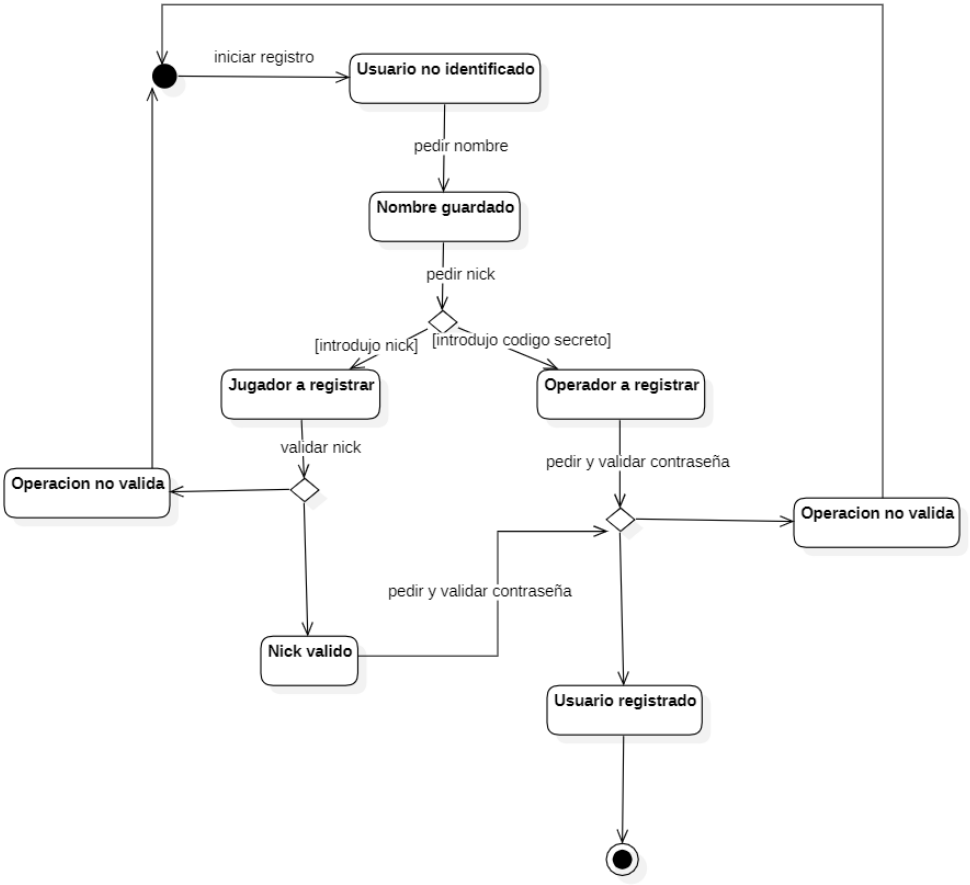

Imagen 14 Diagrama de estado del registro

Al iniciar el proceso de registro de usuario estaremos en el estado de “Usuario no identificado”, al usuario en proceso de registrarse se le pedirá su nombre el cual procederá a ser guardado, entrando en el estado de “nombre guardado”. Luego procederemos a pedir un nick, en este momento el usuario entra la posibilidad de escribir un código secreto que dará acceso al registro de operador y no al de usuario, dependiendo de si dicho código es introducido por el usuario entraríamos en dos posibles estados “Jugador a registrar” o “Operador a registrar” donde ya ha sido identificado el tipo de usuario que va a ser registrado. El operador ya estará habilitado a introducir la contraseña de su gusto. En el caso del jugador, su Nick tendrá que ser validado, es decir, no puede estar duplicado. En caso de no introducir un Nick válido más de dos veces se entrará en el estado “operación no válida” que finalmente nos llevará al inicio. En caso de Nick válido será hora de pedir validar la contraseña. En caso de que la contraseña no sea válida más de dos veces se entrará en “operación no válida” y nos llevará al inicio. En caso de que la contraseña esté validada entramos en el estado “usuario registrado” para concluir el registro.

4.2. Diagramas de estados del inicio de sesión

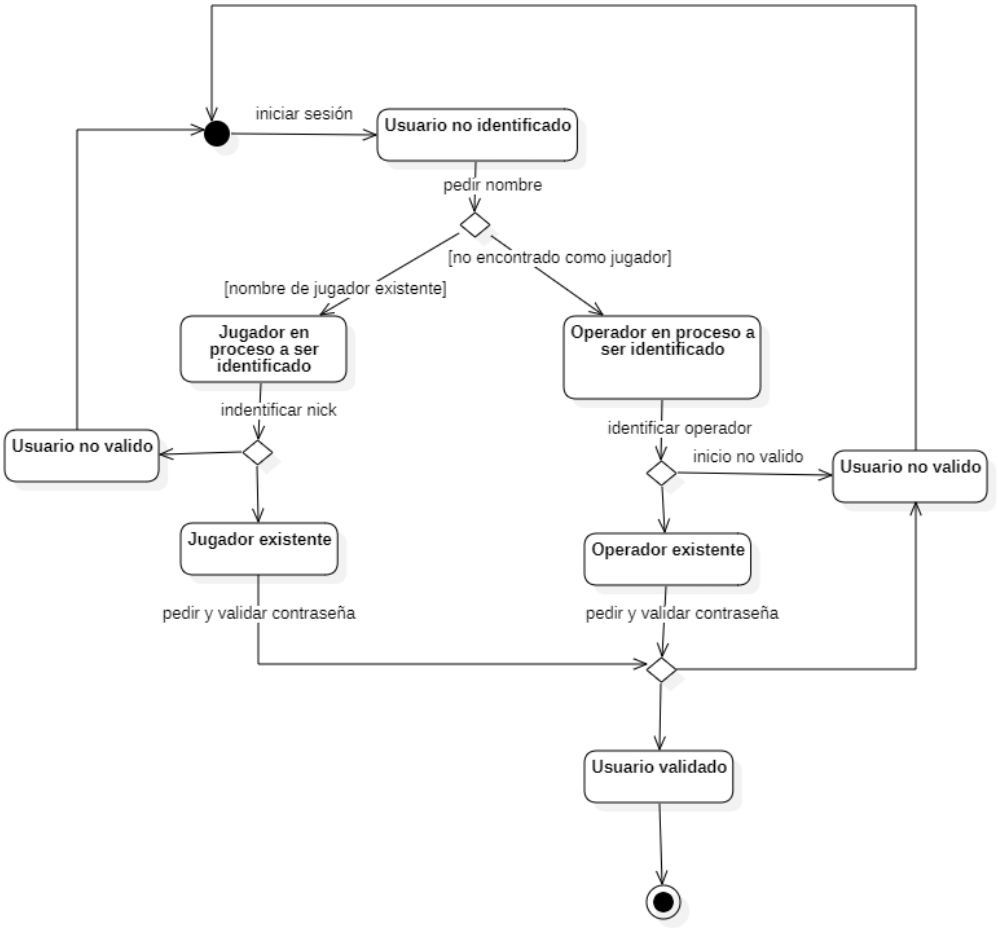

Imagen 15 Diagrama de estados de inicio de sesión

Al iniciar sesión estamos en el estado de “usuario no identificado”, posteriormente se le pedirá al usuario su nombre, a partir de aquí hay dos caminos. Primero se explicará qué sucede si existe un jugador con dicho nombre. En caso de encontrar a un jugador con el nombre dado se entrará en el estado de “Jugador en proceso a ser identificado”, en caso de haber más de un jugador con ese nombre se le pedirá al usuario su nick, en caso de no existir se entra en el estado “usuario no valido” que nos lleva al inicio del inicio de sesión, en caso contrario entramos en el estado “jugador existente”, posteriormente se le pedirá al jugador que introduzca su contraseña para validarla, si no es correcta entraremos en “usuario no valido” y si lo es entramos en “usuario validado” donde el usuario habrá iniciado sesión. En el caso de que el nombre dado por el usuario no exista en la lista de jugadores entraremos en el estado “operador en proceso a ser identificado”, ahí se verificará si existe en la lista de operadores el nombre introducido, si no existe entramos en “usuario no válido”, en caso contrario entramos en “operador existente” y luego se procederá a pedir y validar contraseña como al jugador para terminar de ejecutar el inicio de sesión.

4.3. Diagramas de estados del proceso de retar

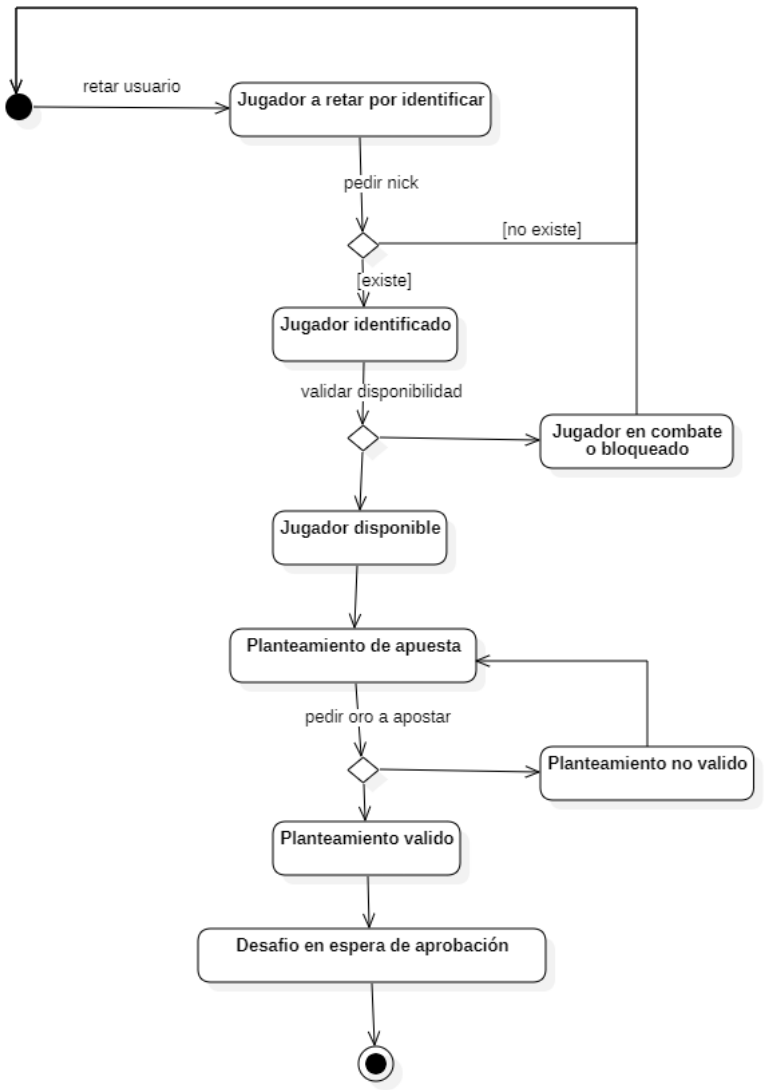

Imagen 16 Diagrama de estados del proceso de retar

Cuando un jugador reta a otro, el sistema pasará por múltiples estados. Al iniciar el proceso estaremos en el estado "Jugador por identificar" donde el sistema pedirá al usuario el nick del jugador a retar, si este jugador existe entraremos en el estado de "jugador identificado". Luego el sistema verificará en qué estado está dicho jugador, si no está disponible el programa estará en el estado de "Jugador en combate o bloqueado" que nos llevará al inicio de retar a un jugador, en caso contrario el sistema entra en estado "jugador disponible". Posteriormente entramos en el estado de "planteamiento de apuesta", donde el jugador propondrá la cantidad de oro a apostar y el sistema verificará si ambos jugadores serán capaces de pagar en caso de perder el combate, dependiendo de esto se entrará en el estado de "planteamiento válido" o "planteamiento válido". Finalmente, el sistema entra en estado de "desafío en espera de aprobación" donde dicho desafío ha sido creado y quedará en espera a ser aprobado por un operador.

4.4. Diagramas de estados del combate

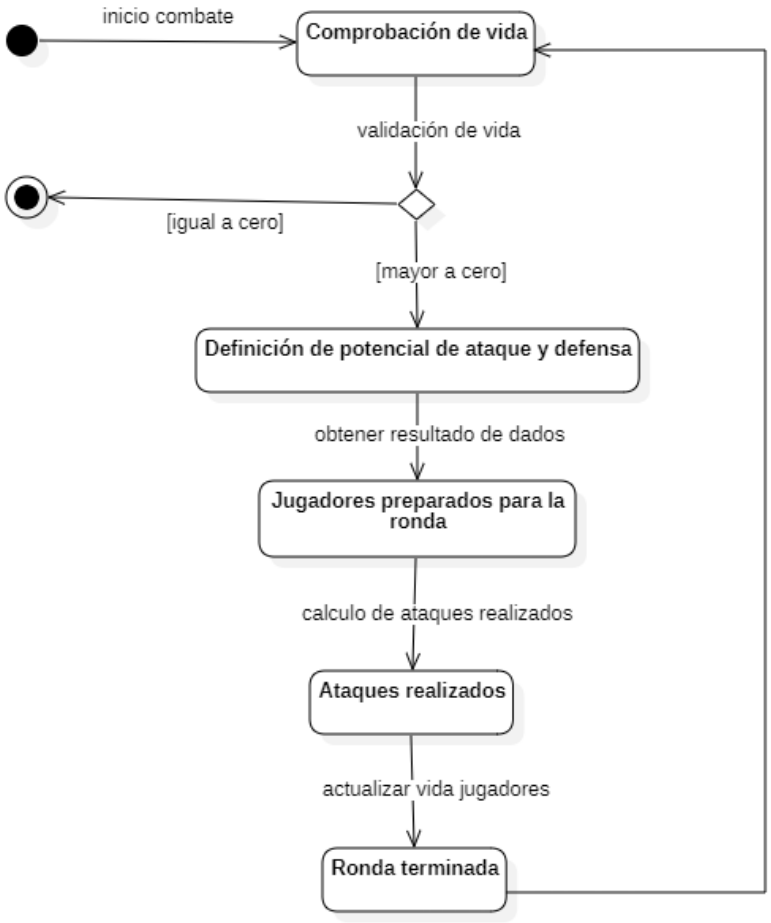

Imagen 17 Diagrama de estados del combate

El combate pasará por varios estados durante su ejecución, estos estados están atados a un bucle ya que el combate acabará cuando uno de los jugadores quede con su vida en cero. Al iniciar el bucle siempre entraremos en una fase de comprobación del estado de vida de ambos jugadores, si una de estas llega a cero se finalizará el combate. Posteriormente, el combate entrará en la fase de definición de potenciales, tanto de ataque como de defensa que posteriormente se utilizaran para verificar si los jugadores perdieron puntos de vida, esto último ocurrirá en la fase de ataques realizados. Finalmente entramos en la fase de ronda terminada, donde no queda nada más que actualizar el número de ronda y volver a el primer estado de la ronda, es decir, el estado de comprobación de vida.

5. Diagramas de casos de uso

El propósito de esta sección consistirá en establecer las acciones disponibles para los diversos usuarios dentro del sistema. Esto se detalla principalmente mediante la representación en diagramas de casos de uso.

5.1. Definición de actores

5.1.1. Jugador

El jugador, como tipo de usuario registrado que participa activamente en las actividades relacionadas con el juego, desempeñará varias acciones importantes. Primero, el jugador podrá desafiar a otros jugadores registrados mediante el uso del Nick del contrincante. Mientras que, el jugador que ha sido desafiado podrá aceptar o negar dicho desafío

También, el jugador podrá gestionar a sus personajes. Esta gestión implica la selección y configuración del equipamiento de sus personajes, así como la creación de nuevos personajes y la eliminación de los existentes según sea necesario.

Por último, el jugador cuenta con acceso a los resultados de los combates. A través de un historial de combates, puede visualizar los detalles de estos enfrentamientos, lo que le permite analizar su rendimiento y evolución en el juego.

5.1.2. Operador

El operador, es el tipo de usuario registrado que como ocupación se encargará de gestionar el sistema. En esencia el operador es un administrador del sistema, este estará encargado de validar múltiples aspectos del juego durante la duración de este. El operador se encargará de:

- Validación de desafíos: El operador se encargará de verificar la validez de los desafíos propuestos por los distintos jugadores, es decir, verifican que cumplan los requisitos necesarios.
- Bloqueo de usuarios: En caso de que algún jugador viole las normativas del juego, el operador puede bloquear su cuenta. Además, podrá desbloquear usuarios previamente restringidos, según sea necesario.
- Edición de atributos de personajes: Los operadores podrán editar cualquier característica de los personajes, como poder añadir esbirros, añadir armas y armaduras, y también configurar tanto fortalezas como debilidades de estos personajes.

5.2. Diagrama de gestión de cuentas

En el siguiente diagrama, se representa la gestión de cuentas de los diferentes usuarios en el sistema. Los usuarios tienen la opción de registrarse en el sistema como jugadores o como operadores. Al llevar a cabo el proceso de registro, a los jugadores se les asignará automáticamente un número de identificación único generado por el sistema, mientras que a los operadores no se les asignará dicho número.

Además, los usuarios tendrán la capacidad de realizar las siguientes acciones:

- Iniciar sesión en el sistema.
- Cerrar sesión para finalizar su sesión activa.
- Darse de baja en el sistema en caso de que así lo deseen, lo que implica la eliminación de su cuenta y datos asociados.

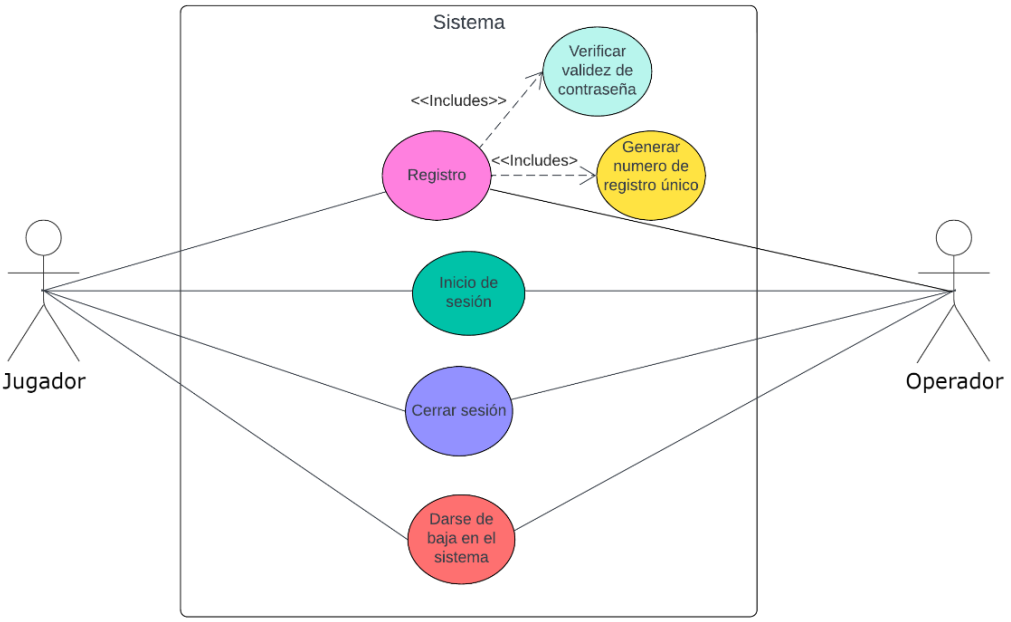

Imagen 18 Diagrama de gestión de cuentas

5.3. Diagrama de gestión de desafíos

El próximo diagrama detalla cómo es la gestión de los desafíos, tanto para los jugadores como operadores. Por un lado, los jugadores podrán desafiar distintos jugadores, a la vez que podrán aceptar o rechazar desafíos de otros jugadores. Los operadores, que tienen la función de ser administradores, tendrán la tarea de validar los desafíos establecidos por los jugadores antes de que estos puedan batallar. En caso de jugadores problemáticos, el operador podrá bloquear a estos jugadores y desbloquearlos posteriormente.

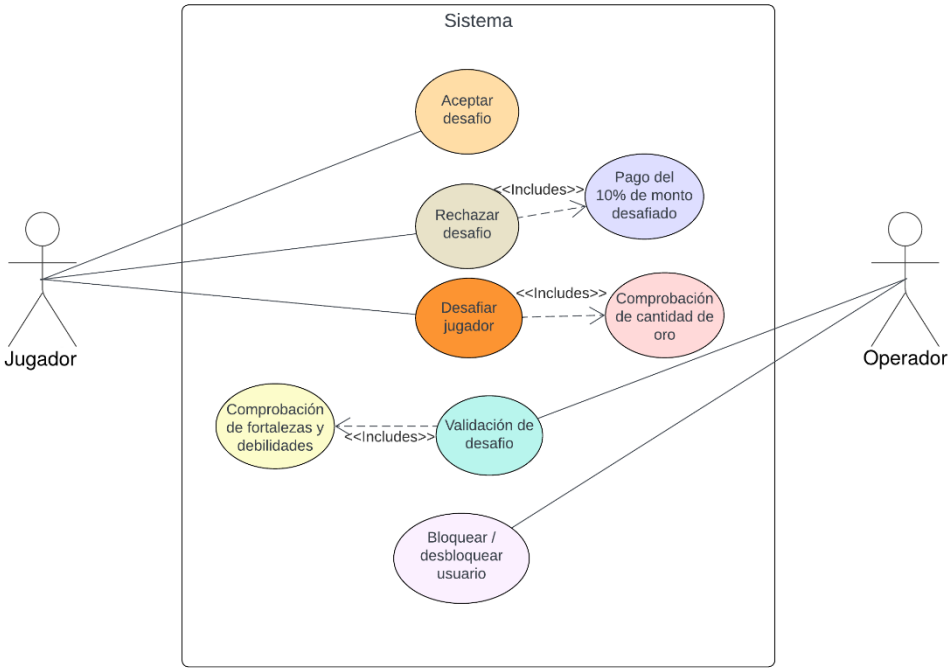

Imagen 19 Diagrama de gestión de desafíos

5.4. Diagrama de gestión de personajes

El diagrama presentado detalla cómo los personajes serán gestionados por los distintos usuarios en el programa. Los jugadores, como es ilustrado, tendrán la posibilidad de registrar personajes y eliminarlos a la vez que podrán equiparles a estos personajes distintas armas y armaduras. Por otro lado, podrá editar cualquier característica de los personajes, como poder añadir esbirros, añadir armas y armaduras, y también configurar tanto fortalezas como debilidades de estos personajes.

Imagen 20 Diagrama de gestión de personajes

5.5. Diagrama de consultas y reportes

Cómo presentado en el diagrama a continuación, siempre que el jugador esté en una sesión activa, este podrá acceder a información relacionada a él. Esto implica acceder al ranking global de jugadores y también al historial de sus combates, donde el jugador podrá ver información de dichos combates.

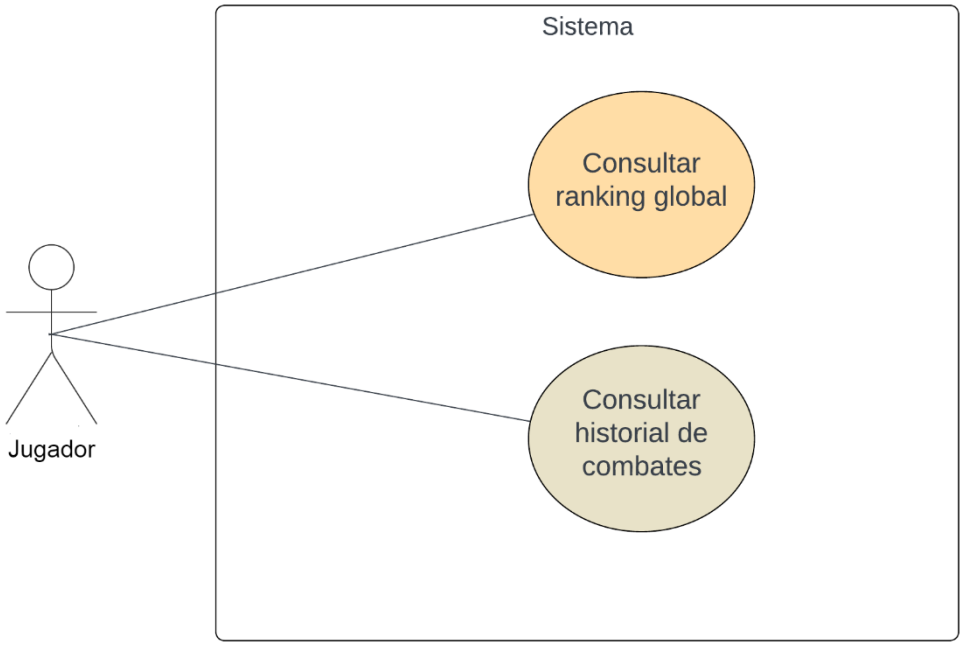

Imagen 21 Diagrama de consultas y reportes

ANEXO

Datos de ejemplo

I. Partida (Fichero binario)

- Mapa de jugadores
  - Jugador
    - Nombre: Andrés
    - Nick: ElBombas
    - Contraseña: Cuenta1]
    - Nº Registro: T15LP
    - Combates: null
    - Bloqueado: False
    - Personaje activo (el ejemplo se ve en lista de personajes)
    - Historial combate: null
    - Ultima derrota: null
- Lista de desafíos sin validar
  - Desafío
    - Jugador retado: Andrés
    - Jugador retador: Federico
    - Oro apostado: 20.5
    - Rondas: 1
    - Fecha: 20-01-2024
    - Ganador: null
    - Jugador con esbirros sin derrotar:
      - Andrés
      - Federico
    - Modificadores: null
    - Valido: True
- Lista de personajes
  - Personaje
    - Nombre: Vampiro
    - Habilidades especiales:
      - Nombre: Disciplina
      - Valor ATC: 1
      - Valor DFS: 2
      - Coste: 1.5
    - Armas: null
    - Armas activas:
      - Nombre: Garras
      - Modificador: 2
      - Mod DFS: 1
      - Tipo: True
    - Armaduras: null
    - Armadura activa:
      - Nombre: Caparazón
      - Modificador: 3
      - Mod ATQ: 1.5
    - Esbirros:

        - Nombre: Ghoul
        - Salud: 20
        - Dependencia: 1
    - Oro: 120
    - Salud: 10
    - Poder: 5
    - Debilidades:
        - Nombre: Roce con el agua
        - Valor: 0.1
    - Fortalezas:
        - Nombre: Ego
        - Valor: 0.5
    - Versión: 1
    - Edad: 150
    - Sangre: 10

II. Fichero con operadores (Fichero de texto)

- Código secreto de operadores
    - JuEgOmPGiS2024]
- Operador
    - Nombre: Alfredo
    - Nick: ElOperador1
    - Contraseña: PrimerOp

Prototipos de pantallas

Es de vital importancia que la interfaz del sistema sea fácil de entender y usar, proporcionando una experiencia de usuario intuitiva. Esto implica que la disposición de los elementos en la interfaz debe usar elementos visuales claros y descriptivos para guiar a los usuarios. La navegación y el acceso a las distintas funciones debe ser sencillo, y la interfaz debe ofrecer retroalimentación visual inmediata sobre las acciones del usuario.

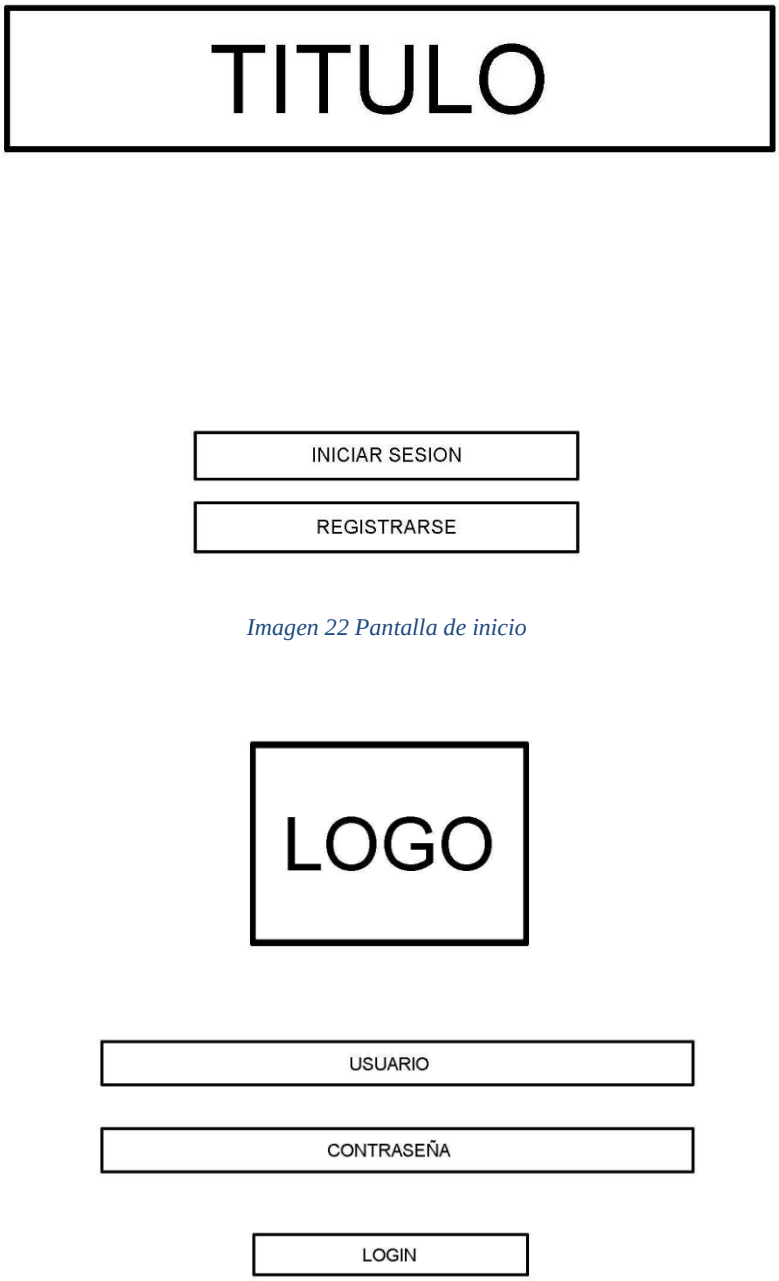

Imagen 22 Pantalla de inicio

Imagen 23 Pantalla de inicio de sesión

En las ilustraciones superiores podemos observar un esquema a seguir para la orientación de la interfaz de la aplicación.

-En ***ilustración 14*** se puede encontrar un espacio grande en la parte superior destinado al título y dos botones en la zona inferior que representan dos posibilidades comunes entre operadores y usuarios (iniciar sesión y registrarse).

-En ***ilustración 15*** se puede destacar en la parte superior un espacio reservado para el logo identificativo de la aplicación. Dos espacios centrales destinados a registrar los datos de los usuarios a la hora de iniciar sesión. Por último, un botón de envío del formulario en la parte inferior.