Universidad de Valladolid

E.T.S Ingeniería Informática

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática
Mención Ingeniería de Software

Evaluación del sensor Kinect v2 para rehabilitación funcional

Autor:
An Chen
Tutores:
Miguel A. Laguna Serrano
Irene Lavín Perrino

ÍNDICE

LISTA DE FIGURAS

LISTA DE TABLAS

1. INTRODUCCIÓN

Este proyecto forma parte de los trabajos desarrollados en el grupo GIRO de la Universidad de Valladolid con el objetivo de encontrar una solución para facilitar y mejorar la vida de los pacientes hemipléjicos mediante aplicaciones informáticas.

El proyecto consiste en desarrollar una aplicación que se ofrecerá a los pacientes como alternativa para que puedan realizar una terapia de rehabilitación funcional de forma remota. Además, los terapeutas dispondrán de un seguimiento riguroso de la evolución de los pacientes. Los datos recogidos permitirán a los terapeutas realizar estudios y análisis detallados del progreso de cada paciente.

La aplicación se dividirá en dos partes. Por una parte, actividades de la vida diaria que servirán principalmente para recolectar informaciones de los pacientes. Por otra parte, los mini-juegos que servirán como ejercicios de rehabilitación.

Todo el desarrollo de la aplicación está basado en la tecnología Kinect v2 para Windows y Xbox One. El sensor Kinect v2 permite medir de forma objetiva y con mucha exactitud las posiciones de las articulaciones principales de un cuerpo, lo que le convierte en un candidato perfecto para el desarrollo de una aplicación de rehabilitación funcional. Sin embargo, debido a las características técnicas de Kinect es obligatorio imponer unos requisitos básicos para el correcto funcionamiento de la aplicación.

1.1. Descripción general de la aplicación

La aplicación desarrollada tiene dos objetivos principales: por un lado, ofrece múltiples ejercicios que simulan las actividades diarias en un entorno virtual 3d mediante un avatar de forma humanoide que monitoriza los movimientos de los pacientes y guarda los registros de los ejercicios realizados por los pacientes. Por otro lado, los mini-juegos que sirven como ejercicios de rehabilitación.

Para la monitorización de los movimientos de los pacientes se aprovechan los datos pre-calculados de Kinect, concretamente las '*jointorientations*'. Estos datos proporcionados por el sensor Kinect nos indican las rotaciones globales de las articulaciones de los pacientes. Sin embargo los datos proporcionados por el sensor Kinect no son tan exactos como cabría esperar y, por lo tanto, es necesario aplicar un filtro para obtener datos más fiables y exactos.

Para el registro de los datos obtenidos, el uso de SQLite como base de datos para almacenar las informaciones de los pacientes, ofrece numerosas ventajas:

- No requiere una configuración
- Sin necesitad de un servidor de base de datos
- Fácilmente portable (multiplataforma Windows, Linux, Mac, Android, etc.)
- Acceso rápido
- Soporte para casi cualquier lenguaje de programación
- Seguridad de datos (más de 2/3 del código está dedicado puramente a la prueba y verificación)
- Único archivo de Base de Datos KumbiaPHP soporta SQLite
- Registros de Longitud Variable

Con el motivo de conseguir la mayor independencia posible por parte de los pacientes para realizar los ejercicios, las interfaces se diseñarán de la manera más simple e intuitiva posible y siempre acompañadas de las instrucciones apropiadas para los pacientes.

1.2. Tecnologías empleadas

El desarrollo de la aplicación está basado en Kinect para Windows v2 y Kinect para Xbox One diseñados por Microsoft y la librería utilizada en la aplicación es la versión 2 de kit oficial de Microsoft para desarrollo de Kinect. Tecnologías complementarias empleadas son:

- Windows 8.1
- Unity 5.3.4f1 (Framework para la creación de entorno virtual 3d)
- Visual Studio 2015
- .Net 3.5 (Versiones posteriores no son posibles debido a la limitación de Unity)
- Sqlite

1.3. Objetivo y motivación

Uno de los objetivos de la aplicación es ofrecer a los pacientes una terapia de rehabilitación remota sin asistencias de los terapeutas. Al mismo tiempo que la realización de las actividades de la terapia, registrará las informaciones relevantes de los pacientes para su posterior análisis, que servirá como soporte para el control de progreso de los pacientes.

1.4. Contexto de la aplicación

La realización de una rehabilitación remota en un entorno familiar y cotidiano significa comodidad y accesibilidad para los pacientes. Los desplazamientos a los centros médicos normalmente limitan los horarios de los pacientes, además de los costes económicos que suponen.

Otra ventaja es el denominado apoderamiento, que convierte la terapia en un juego con metas medibles y que estimula a los pacientes a alcanzar dichos objetivos. Además, las animaciones y felicitaciones de los objetivos conseguidos son una buena fórmula para motivar a los pacientes.

2. HERRAMIENTAS DE DESARROLLO

2.1. Característica de sensor Kinect

Son muchas las diferencias y ventajas que tiene la versión 2 de Kinect con respecto a la versión anterior, las características principales de Kinect V2, que experimentaron un cambio drástico y que influyen directamente sobre el desarrollo de la aplicación son:

- Mayor campo de visión: 70º en horizontal y 60º en vertical (57º x 43º Antes). Gracias a esta mejora ahora es posible detectar simultáneamente a 6 personas. Y cabe destacar que esta versión Kinect no tiene motor de inclinación.

- Mayor Resolución: 1920 x 1080 Full HD (640 x 480 antes). Debido a ello es posible detectar el entorno con mucho más detalle incluso es posible llegar a diferenciar los dedos de las manos, mejorar el reconocimiento facial y por su puesto ofrecer imágenes de mayor calidad.

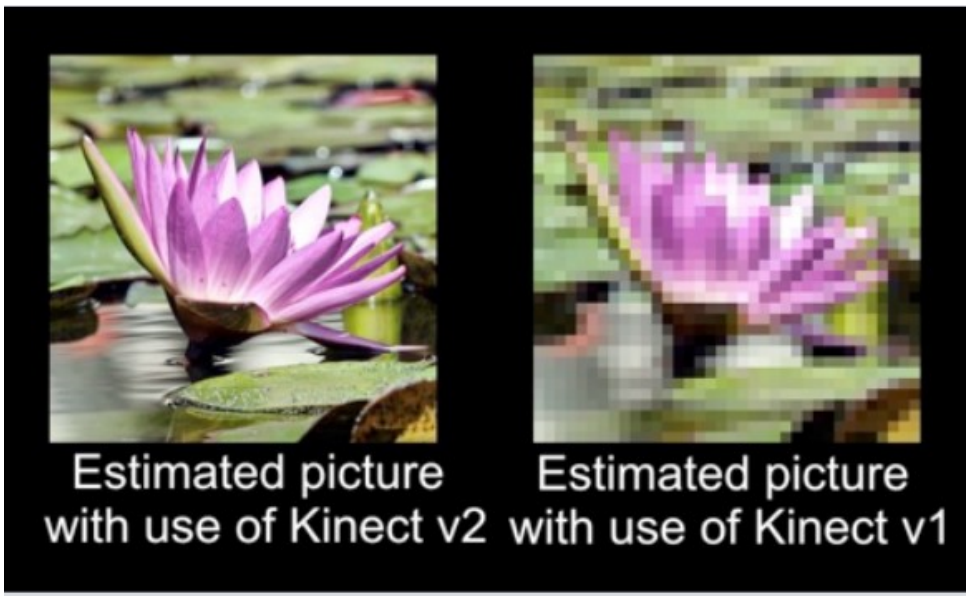

Figura 1: *Imagenes capturadas por Kinect v2(Derecha) y Kinect v1 (izaquierda)*

- Mejora del rango de profundidad del sensor: El rango pasa de 0.5 metros de antes a 4.5 metros.

- USB 3.0: Aumento considerable de la velocidad de transferencia de datos entre el sensor y el pc pasa de 90 ms a 60 ms, aunque esto significa también que el USB 3.0 es uno de los requisitos mínimos para las aplicaciones desarrolladas basadas en Kinect v2.

- Mejora de la captación de sonidos: Esta versión de Kinect incorpora una gran mejora en cuanto al reconocimiento de voz, eliminación más efectiva del ruido ambiental.

- Captación de movimientos a oscuras: El uso de cámara infrarroja posibilita detección de movimientos en un ambiente oscuro.

- Kinect 2 permite calcular/analizar la fuerza de nuestros músculos y medir el ritmo cardíaco, aunque se desconoce de su fiabilidad.

2.2. Características del SDK para Kinect

The Kinect for Windows Software Development Kit (SDK) 2.0 creado por Microsoft específicamente para el desarrollo de las aplicaciones con Kinect v2 proporciona soporte para la gestión de reconocimiento de voz e imágenes, mediante varios elementos:

2.2.1. Color stream (ColorFrame)

El sensor Kinect nos proporciona un array de valores de píxeles RGB de la imagen capturada con la cámara de color con resolución 1920 x 1080.

2.2.2. Depth stream (DepthFrame)

Depth stream contiene valor de profundidad de cada punto en la zona visible. La profundidad es la distancia entre sensor y el punto en cuestión, en general representamos los puntos distantes con color negro y puntos más cercanos con el color blanco, aunque tenemos la libertad de configurarlos con otros colores. Por lo tanto, para representar esta información es necesario dos arrays, un array de profundidad proporcionado por Kinect y otro que incluye los valores de los colores.

2.2.3. Infrared stream (InfraredFrame)

El sensor de infrarrojo de Kinect nos proporciona visión en un entorno oscuro y su gestión es similar al de profundidad en el que se requieren dos arrays.

2.2.4. Body tracking

Mediante tratamiento de informaciones proporcionadas por la cámara de profundidad el SDK Kinect nos ofrece informaciones relativas a los cuerpos.

En la primera versión de Kinect solo incluye 20 puntos del cuerpo humano que será mejorado en la segunda versión con 25 puntos del cuerpo. Aunque los datos que se recogen de algunos puntos del cuerpo tienden con frecuencia al error, como los dedos de la mano y de los pies.

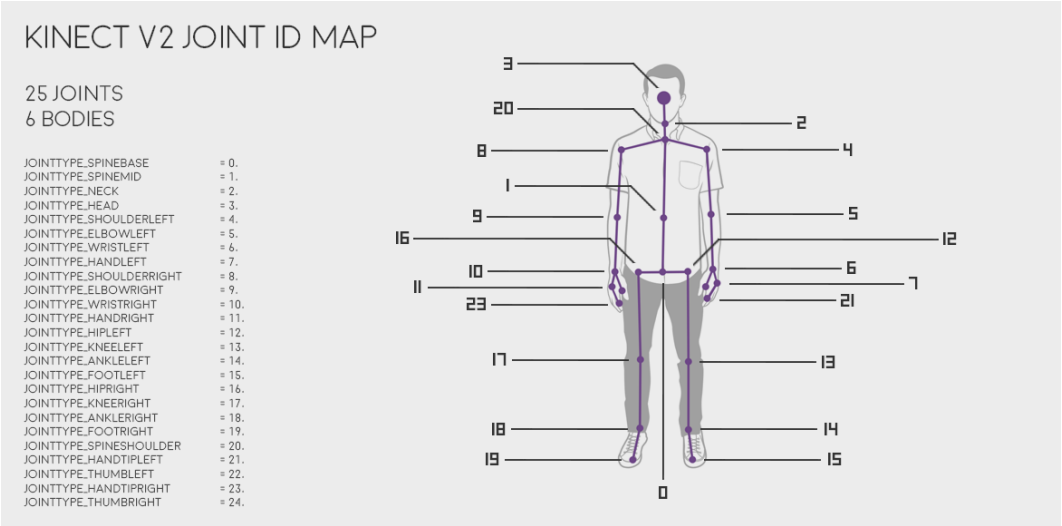

Figura 2: *Puntos del cuerpo capturados por el sensor Kinect*

Entre la información relativa a los puntos del cuerpo cabe destacar:

- **"position":** Representa la posición absoluta de cada punto del cuerpo mediante un vector 3d.

- **"orientation":** Cada punto del cuerpo guarda la información de rotación absoluta de su padre (Ver la figura 3), esto quiere decir que no hay información de rotación para las extremidades como cabeza y dedos. Sin embargo, la orientación de la cabeza es posible obtenerlo a través de reconocimiento facial.

Las rotaciones se representan mediante cuaterniones (x,y,z,w) concretamente cuarteniones unitarios respecto a la representación de ángulos de Euler, que son más simples de componer y evitan el problema del bloqueo del cardán.

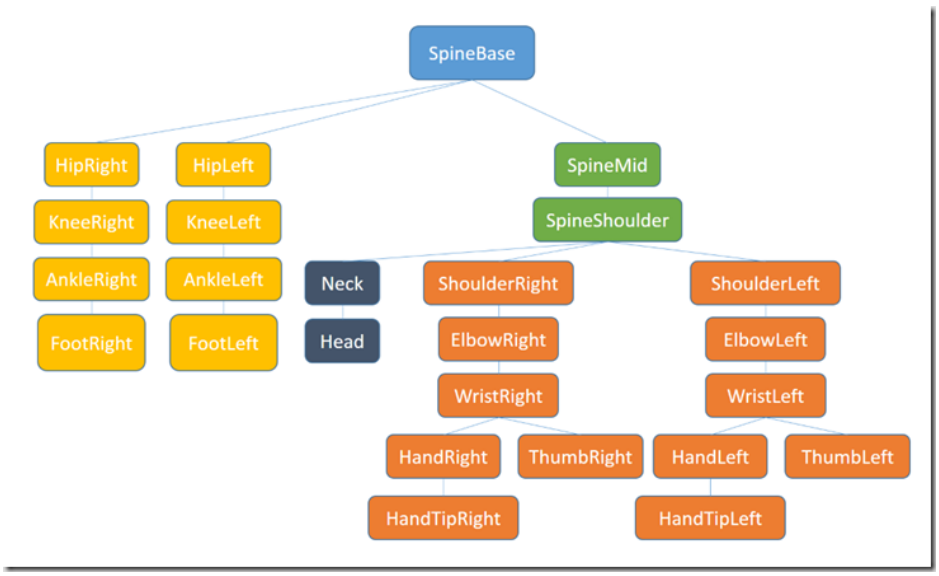

Figura 3: *Jerarquía de los puntos del cuerpo capturados por el sensor Kinect*

2.3. Cuaterniones

Los cuarteniones son utilizados ampliamente para las representaciones de rotaciones y orientaciones en un espacio 3D, respecto a las representaciones mediante ángulos de Euler, son más simples de componer y evitan el problema del bloqueo del cardán y en comparación con las matrices son más eficaces computacionalmente.

Las representaciones de rotaciones y orientaciones mediante cuarteniones están basadas en la teoría de rotación de Euler: **cualquier rotación o secuencia de rotaciones de un cuerpo rígido o sistema de coordenadas sobre un punto fijo es equivalente a una sola rotación en un ángulo dado θ alrededor de un eje fijo (llamado eje Euler) que pasa por el punto fijo.**

Los cuarteniones técnicamente tienen cuatro números de los cuales tres de ellos tienen componentes imaginarios. Por lo tanto, la definición de cuartenión seria q = w + xi + yj + zk, donde w, x, y, z son reales e i, j, son imaginarias. Debido a estas propiedades, las multiplicaciones de los cuarteniones no son conmutativas. Sin embargo, todo esto es irrelevante en caso de las representaciones de las rotaciones y orientaciones en 3D ya que la parte imaginaria nunca será tenida en cuenta en estas representaciones. Con esto quiere decir que la definición de cuartenión para rotaciones y orientaciones es: q (w, x, y, z), en esta definición de cuarteniones los componentes x, y, z representan el eje sobre el cual realiza la rotación y w representa la cantidad de rotación sobre este eje concurrente (ver figura 4).

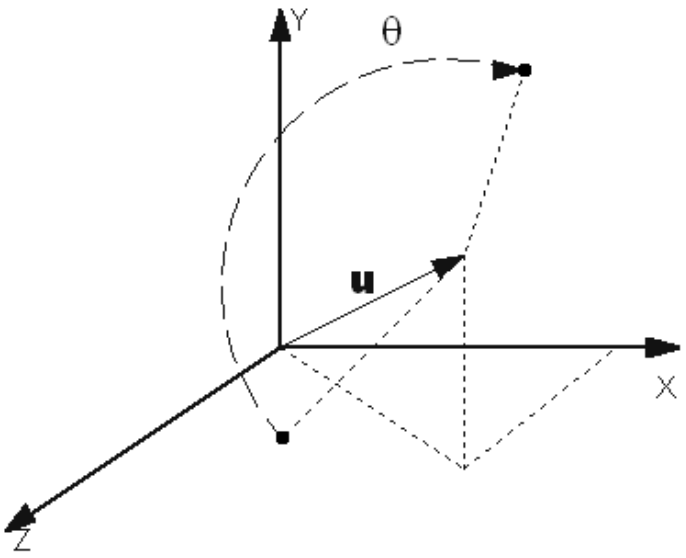

Figura 4: *Representación gráfica de Cuaternión*

Una de las operaciones más importantes de los cuaterniones es la multiplicación entre dos cuaterniones, que generalmente representan la rotación. Suponiendo $Q_3 = Q_1 * Q_2$.

$$Q_3.w = w_1w_2 - x_1x_2 - y_1y_2 - z_1z_2$$

$$Q_3.x = w_1x_2 + x_1w_2 + y_1z_2 - z_1y_2$$

$$Q_3.y = w_1y_2 - x_1z_2 + y_1w_2 + z_1x_2$$

$$Q_3.z = w_1z_2 + x_1y_2 - y_1x_2 + z_1w_2$$

2.4. Unity

2.4.1. Introducción

La primera versión de Unity fue creado en 2004 y empezó como una herramienta de producción exclusivamente para Mac. Aunque fue diseñado específicamente para la plataforma de Apple y cualquier desarrollo se debía llevar acabo en un entorno Apple, sin embargo, podía ser desplegado en otro entorno de sistema operativo. Esta herramienta se creó fundamentalmente para el desarrollo de videojuegos, no obstante, debido a que es una herramienta multiplataforma y a su potente conjunto de motores para el desarrollo de entornos 3D, es muy usado en otros ámbitos, tales como la animación, simulación, realidad aumentada, desarrollo de aplicaciones didácticas, etc.

En el 2008 aprovechando el lanzamiento de iPhone, que desató la fiebre de compatibilidad entre las plataformas, y la llegada de plataforma Android se produjo el auge de Unity. Sin embargo, para convertirse en uno de los motores de videojuegos más populares hizo falta una mejora drástica en la calidad gráfica, hecho realidad con la versión 4.0 de Unity. Además, a raíz de esta mejora grafica pudo cerrar acuerdos con Sony, Microsoft y Nintendo para que el motor fuera compatible con sus sistemas. Actualmente Unity es compatible con muchas plataformas: PC, Mac, Linux, iOS, Android, …).

Figura 5: *Plataformas soportadas por Unity*

2.4.2. Características:

- El núcleo del sistema de procesador se basa en OpenGL. Una de las mejoras en los últimos años que cabe destacar es su alto rendimiento.

- Una de las ventajas de Unity es que es gratuita para cualquier persona, además, el usuario dispone de casi todas las características del motor. A excepción de ciertos efectos de post-procesado, y algunos requisitos de reconocimiento por uso de Unity.

- Otro punto fuerte del Unity es su compatibilidad con la red, algo que carecen muchos otros motores de videojuegos.

- Para programar en Unity se puede utilizar el lenguaje de script Javascript o C# y con la libertad de elegir el editor.

- Unity cuenta con una documentación muy detallada y concisa.

- En cuanto a la física, el provedor PhysX es el motor estándar de físicas de la industria, el cual es utilizado por Unity.

- A partir de 2010, Unity dispone de Unity Asset Store con una colección de más de 15.000 paquetes Assets en una amplia gama de categorías, incluyendo modelos 3D, texturas y materiales, sistemas de partículas, música y efectos de sonido, tutoriales y proyectos, paquetes de scripts, extensiones para el editor y servicios en línea.

- Unity incorpora numerosas librerías que facilitan el desarrollo de los juegos. Como las operaciones relacionadas con el posicionamiento y rotación en el espacio 3D, que son muy útiles para el proyecto.

3. GESTIÓN DEL PROYECTO

3.1. Justificación de uso del proceso unificado de desarrollo de software

Debido al aspecto relativamente investigador de este proyecto, y a la necesidad de modificar los requisitos a medida que avanza el proyecto, visualizando las posibilidades con las que se cuenta el desarrollo, un modelo de desarrollo pesado resulta inadecuado.

En cuando al modelo puramente ágil se necesita un equipo de desarrollo con experiencia para poder completar el proyecto de forma satisfactoria. Por lo tanto, en este caso tampoco es adecuado.

Debido a los motivos anteriores, este proyecto requiere un modelo de desarrollo que proporciona un enfoque iterativo e incremental y a la vez no requiere personales experimentados: el proceso unificado de desarrollo de software propuesto por Ivar Jacobson, Grady Booch y James Rumbaugh.

3.2. Proceso unificado de desarrollo de software

El proceso unificado tiene las siguientes características:

- **Iterativo e incremental:** El Proceso Unificado es un marco de desarrollo iterativo e incremental compuesto de cuatro fases denominadas Inicio, Elaboración, Construcción y Transición. Cada una de estas fases es a su vez divididas en una serie de iteraciones. En cada nueva iteración se añade o mejora las funcionalidades del sistema en desarrollo.

- **Dirigido por los casos de uso:** para construir un sistema fructuoso es necesario conocer que es lo que los usuarios quieren y necesitan. Por eso en cada iteración del proceso unificado los desarrolladores identifican y especifican los casos de uso relevantes y los usan como la guía tanto para definir los objetivos de la iteración como para implementar el diseño de sistema que satisfacen estos casos de uso.

- **Centrado en la arquitectura:** partiendo de una visión global del sistema van refinando hacia niveles inferiores hasta definir cada uno de los componentes básicos del sistema.

- **Enfocado en los riesgos:** para disminuir la probabilidad de que ocurra un riesgo. Los riesgos deben ser identificados en una etapa temprana del ciclo de vida y tratados con la mayor prioridad en todas las iteraciones.

El proceso unificado está dividido en cuatro fases:

- **Inicio:** Teniendo en cuenta el enfoque relativamente ágil del proceso unificado que se ha aplicado para este proyecto. Es obligatorio reducir lo máximo posible la duración de esta fase. Las tareas de esta fase son:

    - Desarrollar una descripción del producto final y presentar el análisis de negocio.

    - Realizar una identificación inicial de los principales riesgos del sistema.

    - Establecer las funcionalidades primarias del sistema.

    La fase de inicio termina con el hito de los objetivos del desarrollo.

- **Elaboración:** En esta fase deberían capturar la mayoría de los requisitos funcionales del sistema, diseñar una arquitectura inicial y aborda los riesgos ya identificados con el fin de despejar los factores de incertidumbres del proyecto.

  La fase de elaboración termina al alcanzar el hito de la arquitectura del sistema.

- **Construcción:** Es la fase más larga del desarrollo. En esta fase toma la arquitectura obtenida de la fase anterior como la base y partiendo de ella en cada iteración incorpora nuevas funcionalidades de modo que a final de cada iteración se obtiene una nueva versión ejecutable del producto.

  Esta fase concluye con el hito de obtención de una aplicación con todas sus funcionalidades.

- **Transición:** Esta es la fase final del proyecto. Durante esta fase hay que lleva acabo la validación del sistema y la corrección de los fallos identificados.

  Esta fase concluye con el producto final.

Durante cada una de las iteraciones de cada fase se realizarán las actividades definidas en el ciclo de vida clásico: requisitos, análisis, diseño, implementación, prueba e implantación. Aunque todas las iteraciones suelen incluir toda la secuencia de ciclo de vida clásico, el grado de esfuerzo dentro de cada una de ellas varía a lo largo del proyecto.

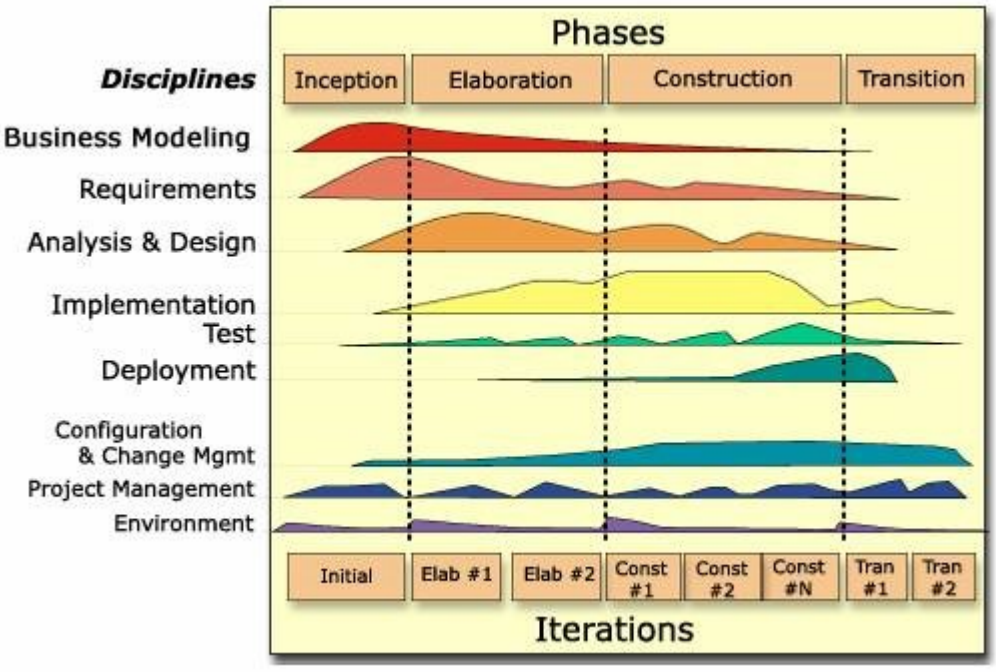

Figura 6: *Representación gráfica del proceso unificado de desarrollo de software*

3.3. Gestión de riesgos

Los riesgos son eventos cuyas ocurrencias son difíciles de predecir y sus apariciones tendrían efectos negativos o positivos sobre los objetivos del proyecto. Aunque, en cuando a la gestión de riesgos se centra en los riesgos que tengan efectos negativos. Muchas veces conseguir evitar los problemas potenciales que pueden aparecer en futuros suponer un gran ahorro de costes y tiempo para el proyecto. Por ellos, es importante identificar, analizar y preparar un plan de contingencia para los riesgos antes de que ocurran. Sin embargo, a pesar de todas las prevenciones es muy probable que ocurran algunos de los riesgos y es entonces cuando hay que llevar a cabo un plan de acción de forma sistemática y siguiendo a la documentación previamente planificada para minimizar el impacto. Además, es necesario documentar la evolución de los riesgos en todo el proyecto para mejorar la gestión de riesgos en los futuros proyectos.

Es muy conveniente cuantificar los riesgos para que de esa manera pueden ordenar los riesgos por impacto. Desafortunadamente, las probabilidades y las consecuencias de los riesgos suelen ser muy complicados de predeterminar, además las dos medidas separadas no sirven para determinar el impacto sobre el proyecto. Por ello hay que recurrir a la exposición al riesgo para determinar el impacto de los riesgos sobre el proyecto que se puede obtener de la siguiente manera:

Exposición al riesgo = Probabilidad x Consecuencia

Modificación de requisitos		
Número: 01	Fecha: 02/2016	Fase: Inicio
Probabilidad: Muy alta	Impacto: Bajo	Exposición: 5 (*)
Descripción del riesgo: Debido la mala captación de los requisitos iniciales o cambios de funcionalidad debido al requerimiento del cliente pueden sufrir cambios significativos en los requisitos de sistema.		
Consecuencia		
Modificaciones sobre los requisitos previamente capturados, que significará un aumento en coste y tiempo.		
Planificación del Riesgo		
Estrategia • Reducción del riesgo • Reservar el riesgo	**Plan de acción** Dedican más recursos para minimizar el cambio producido y reservar un amplio margen de holguras para no alterar la planificación inicial.	

Tabla 1: *Modificación de requisitos*

Tiempos de espera de feedback elevados		
Número: 02	Fecha: 02/2016	Fase: Construcción
Probabilidad: Muy baja	Impacto: Medio	Exposición: 2 (*)
Descripción del riesgo: Para el correcto avance del proyecto es necesario la obtención de las informaciones de retroalimentaciones del cliente lo antes posible. Para poder desarrollar la aplicación de acuerdo con el requerimiento del cliente.		
Consecuencia		
Los retrasos en la obtención de las informaciones de retroalimentaciones del cliente implican aumento de tiempo en desarrollo.		
Planificación del Riesgo		
Estrategia • Reservar el riesgo	**Plan de acción** Utilizar la holgura del presupuesto; realizar otras actividades mientras espera la respuesta del cliente.	

Tabla 2: *Tiempos de espera de feedback elevados*

Falta de experiencia con las herramientas		
Número: 03	Fecha: 02/2016	Fase: Construcción
Probabilidad: Muy alta	Impacto: Alto	Exposición: 7 (*)
Descripción del riesgo: Falta de experiencia del personal en la utilización de las herramientas del proyecto.		
Consecuencia		
Es necesario invertir mayor cantidad de recursos y tiempo en el proyecto.		
Planificación del Riesgo		
Estrategia • Evitación del riesgo	**Plan de acción** Realización de cursos y lecturas de tutoriales para aprender el uso y funcionalidad de las herramientas.	

Tabla 3: *Falta de experiencia con las herramientas*

Indisposición del personal		
Número: 04	Fecha: 02/2016	Fase: Durante todo el proyecto
Probabilidad: Muy baja	Impacto: Medio	Exposición: 2 (*)
Descripción del riesgo: Durante el proyecto a causa de ciertos motivos como enfermedad, accidente laboral… puede que el personal no está disponible para realizar las tareas asignadas.		
Consecuencia		
El aplazamiento del trabajo al realizar durante ese periodo del tiempo de indisposición, significa alargamiento de la duración del proyecto.		
Planificación del Riesgo		
Estrategia • Reservar el riesgo	**Plan de acción** Redistribuir esas tareas entre el tiempo restante del proyecto para disminuir el impacto.	

Tabla 4: *Idisposición del personal*

Fallos de los equipos del desarrollo		
Número: 05	Fecha: 02/2016	Fase: Durante todo el proyecto
Probabilidad: Media	Impacto: Bajo	Exposición: 3 (*)
Descripción del riesgo: A lo largo del proyecto es posible que estropee algún equipo de desarrollo y requiere reparación.		
Consecuencia		
La falta de recursos para el desarrollo conlleva retrasos en el proyecto.		
Planificación del Riesgo		
Estrategia • Protección del riesgo • Reducción del riesgo	**Plan de acción** Dedicar el debido cuidado y revisión periódica a los equipos de desarrollo y reservar algún equipo adicional para caso de que ocurran.	

Tabla 5: *Fallos de los equipos del desarrollo*

Estimación incorrecta del proyecto		
Número: 06	Fecha: 02/2016	Fase: Durante todo el proyecto
Probabilidad: Alta	Impacto: Medio	Exposición: 5 (*)
Descripción del riesgo: Debido la mala planificación del proyecto y las estimaciones de los recursos hace que tiempos requeridos no corresponder con que realmente se necesita.		
Consecuencia		
Una mala planificación puede suponer un aumento de costes y tiempo en lo planificado e incluso puede causar el fracaso del proyecto.		
Planificación del Riesgo		
Estrategia • Protección del riesgo	**Plan de acción** Monitorizar el progreso del proyecto detectar cualquier posible fallo de planificación lo antes posible y redistribuir de manera que reduzca el impacto del riesgo.	

Tabla 6: *Estimación incorrecta del proyecto*

*La exposición de todos los riesgos ha sido calculada utilizando el ISO/IEC 27005:2008 mostrada en la siguiente ilustración.

		Likelihood of Incident Scenario				
		Very Low	Low	Medium	High	Very High
Business Impact	Very Low	0	1	2	3	4
	Low	1	2	3	4	5
	Medium	2	3	4	5	6
	High	3	4	5	6	7
	Very High	4	5	6	7	8

Figura 7: *Matriz de evaluación de riesgos*

3.4. Planificación del proyecto

El objetivo de este capítulo será detallar la planificación del proyecto. Para llevar a cabo dicha planificación se ha preparado planes a dos niveles (planes que contienen los grandes rasgos de la planificación y planes que contienen detalles de una iteración).

Estructura de capitulo:

- Plan de fases
- Plan de cada iteración
- Plan de desarrollo real

3.3.1. Plan de fases

El plan de fases se genera al principio de la fase de inicio y se actualiza a lo largo del proyecto. En este plan se debe especificar de forma breve las fases del proyecto con sus correspondientes iteraciones.

Los objetivos que ha de cumplir en cada iteración de cada fase:

- **Inicio:** En esta primera y la única iteración de fase de inicio hay que completar los siguientes objetivos:
    - Capturar las funcionalidades primarias del sistema y comprender las patologías de la hemiplejia.
    - Realizar investigación sobre Kinect for Windows V2 y Unity, realizando cursos de tutoriales y simples ejemplos que proporciona un aprendizaje básico del uso y manejo de las dichas tecnologías.
    - Familiarizar con el lenguaje de programación C#.
    - Identificar los riesgos potenciales del proyecto.
- **Elaboración, iteración 1:** Capturar los primeros requisitos funcionales del sistema y con ellos comprende las especificaciones detalladas de los casos de uso más importantes. Hay que elaborar también en esta fase un plan de contingencia para los riesgos ya identificados.
- **Elaboración, iteración 2:** Posible modificación sobre los casos de uso y capturas de los nuevos requisitos funcionales, mayoría de los requisitos funcionales deben ser capturados en esta iteración y llevar a cabo un diseño inicial de la arquitectura del sistema.
- **Construcción, iteración 1:** Basando la arquitectura de sistema previamente construido se debe completar la implementación de la base de datos, comenzar con el desarrollo de los controladores de avatar y crear primera visión de interfaz de usuario.

- **Construcción, iteración 2:** En esta iteración se debe finalizar la implementación de los controladores de avatar y ordenar los casos de uso por prioridad y aplicar mejoras en la interfaz de usuario.

- **Construcción, iteración 3:** Últimos retoques en los controladores de avatar y completar restos de los casos de uso y aplicar modificaciones finales sobre la interfaz de usuario.

- **Transición:** Esta última fase de desarrollo consiste en realizar las pruebas e identificar los posibles fallos del sistema. En cuanto los fallos son identificados hay que actualizar el sistema hasta obtener un sistema consistente.

*** Las planificaciones siguientes se basan en una jornada laboral de 4 horas diarias**

	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1	Comienzo de proyecto	0 días	lun 22/02/16	lun 22/02/16	
2	**Proyecto**	**71 días**	**lun 22/02/16**	**vie 10/06/16**	
3	**Inicio**	**10 días**	**lun 22/02/16**	**vie 04/03/16**	
4	Iteración 1	10 días	lun 22/02/16	vie 04/03/16	1
5	**Elaboración**	**20 días**	**lun 07/03/16**	**mié 13/04/16**	
6	Iteración 1	10 días	lun 07/03/16	vie 18/03/16	4
7	Iteración 2	10 días	jue 31/03/16	mié 13/04/16	6
8	**Construcción**	**33 días**	**jue 14/04/16**	**mié 01/06/16**	
9	Iteración 1	12 días	jue 14/04/16	vie 29/04/16	7
10	Iteración 2	12 días	mar 03/05/16	mié 18/05/16	9
11	Iteración 3	10 días	jue 19/05/16	mié 01/06/16	10
12	**Transición**	**7 días**	**jue 02/06/16**	**vie 10/06/16**	
13	Iteración 1	7 días	jue 02/06/16	vie 10/06/16	11
14	Fin de proyecto	0 días	sáb 11/06/16	sáb 11/06/16	13

Tabla 7: *Plan de fases*

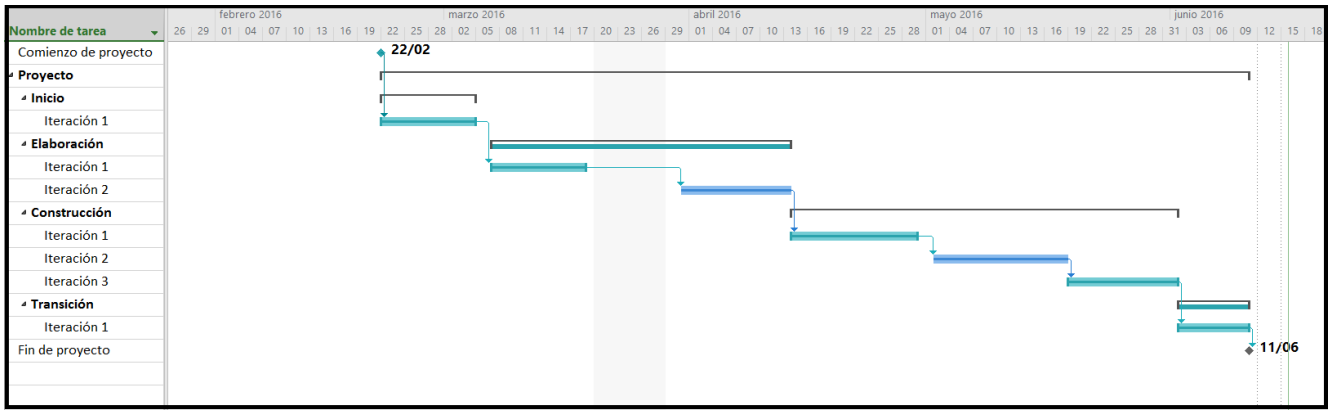

Figura 8: *Plan de fases*

3.3.2. Plan de iteraciones

En cada iteración están activos dos planes de iteración, el pan de iteración actual y el plan de iteración siguiente. En cada plan de iteración se detalla los objetivos y los artefactos que hay que conseguir en dicha iteración para poder saber si ha tenido éxito en su finalización. Además, hay que llevar una lista de actividades a completar de la iteración con sus correspondientes tiempos estimados.

Fase de Inicio

	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1	Comienzo	0 días	lun 22/02/16	lun 22/02/16	
2	**Inicio**	**10 días**	**lun 22/02/16**	**vie 04/03/16**	
3	**Proceso**	**10 días**	**lun 22/02/16**	**vie 04/03/16**	
4	Visión general de proyecto	2 días	lun 22/02/16	mar 23/02/16	1
5	Búsqueda de información sobre la hemiplejia	1 día	mié 24/02/16	mié 24/02/16	4
6	Aprender sobre Kinect	2 días	jue 25/02/16	vie 26/02/16	5
7	Adquirir conocimientos básicos para desarrollar con Unity	2 días	lun 29/02/16	mar 01/03/16	6
8	Aprender a programar con C#	2 días	mié 02/03/16	jue 03/03/16	6
9	Crear primer modelo de casos de uso	1 día	vie 04/03/16	vie 04/03/16	7;8
10	**Gestión**	**10 días**	**lun 22/02/16**	**vie 04/03/16**	
11	Análisis de riesgos inicial	3 días	lun 22/02/16	mié 24/02/16	1
12	Plan de fases	1 día	jue 25/02/16	jue 25/02/16	11
13	Plan de Iteración inicial	1 día	vie 26/02/16	vie 26/02/16	12
14	Plan de iteración siguiente	5 días	lun 29/02/16	vie 04/03/16	13
15	Seguimiento de la iteración actual	10 días	lun 22/02/16	vie 04/03/16	1
16	Fin	0 días	sáb 05/03/16	sáb 05/03/16	9;14;15

Tabla 8: *Fase de inicio*

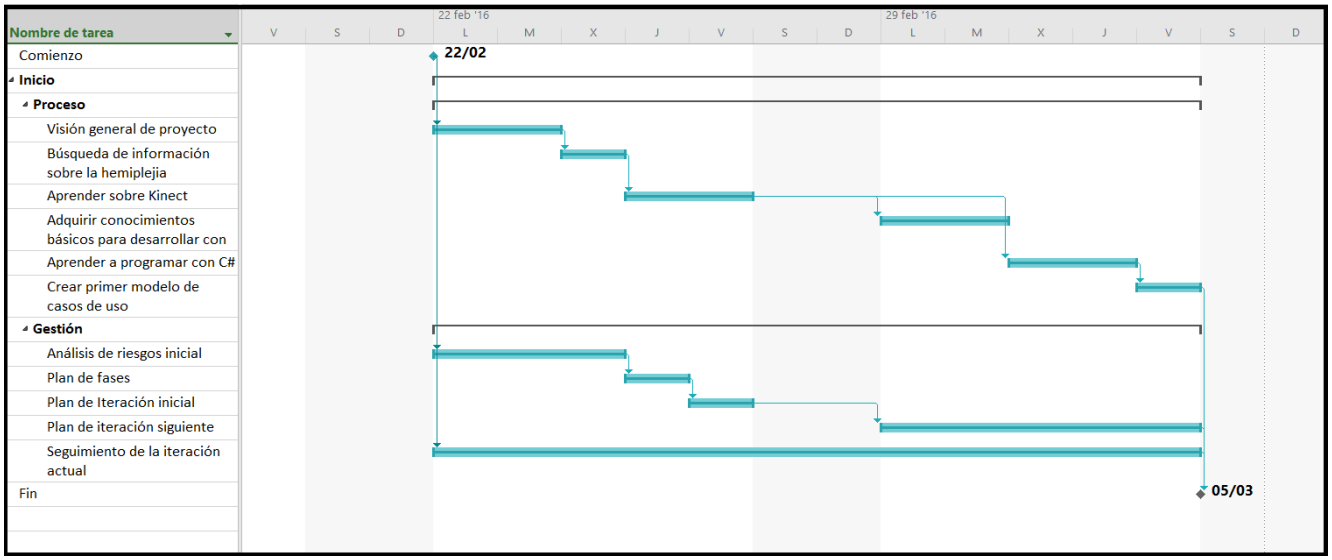

Figura 9: *Fase de inicio*

Fase de elaboración, Iteración 1

	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1	Inicio	0 días	lun 07/03/16	lun 07/03/16	
2	**Elaboración, iteracion 1**	**10 días**	**lun 07/03/16**	**vie 18/03/16**	
3	**Proceso**	**10 días**	**lun 07/03/16**	**vie 18/03/16**	
4	Modificar y detallar los casos de uso	6 días	lun 07/03/16	lun 14/03/16	1
5	Elicitación de requisitos funcionales	6 días	lun 07/03/16	lun 14/03/16	1
6	Prototipo de aplicación	4 días	mar 15/03/16	vie 18/03/16	4;5
7	**Gestión**	**10 días**	**lun 07/03/16**	**vie 18/03/16**	
8	Establecer plan de contingencia para los riesgos	5 días	lun 07/03/16	vie 11/03/16	1
9	Plan de iteración siguiente	5 días	lun 14/03/16	vie 18/03/16	8
10	Seguimiento de plan de iteración actual	10 días	lun 07/03/16	vie 18/03/16	1
11	Fin	0 días	sáb 19/03/16	sáb 19/03/16	6;10

Tabla 9: *Fase de elaboración, Iteración 1*

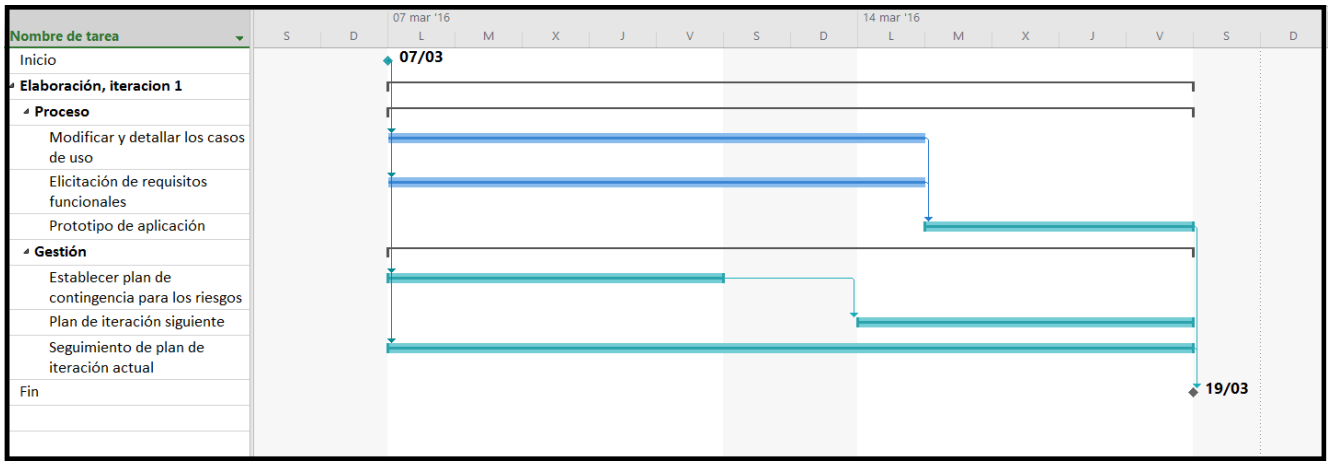

Figura 10: *Fase de elaboración, Iteración 1*

Fase de elaboración, Iteración 2

	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1	Inicio	0 días	jue 31/03/16	jue 31/03/16	
2	**Elaboración, iteración 2**	**10 días**	**jue 31/03/16**	**mié 13/04/16**	
3	**Proceso**	**10 días**	**jue 31/03/16**	**mié 13/04/16**	
4	Actualizar los casos de uso	3 días	jue 31/03/16	lun 04/04/16	1
5	Completar los requisitos funcionales	3 días	jue 31/03/16	lun 04/04/16	1
6	Diseño inicial de la arquitectura	5 días	mar 05/04/16	lun 11/04/16	4;5
7	Prototipo de interfaz usuario	2 días	mar 12/04/16	mié 13/04/16	6
8	**Gestión**	**10 días**	**jue 31/03/16**	**mié 13/04/16**	
9	Revisión de riesgos	5 días	jue 31/03/16	mié 06/04/16	1
10	Plan de iteración siguiente	5 días	jue 07/04/16	mié 13/04/16	9
11	Seguimiento de plan de iteración actual	10 días	jue 31/03/16	mié 13/04/16	1
12	Fin	0 días	jue 14/04/16	jue 14/04/16	11;7;10

Tabla 10: *Fase de elaboración, Iteración 2*

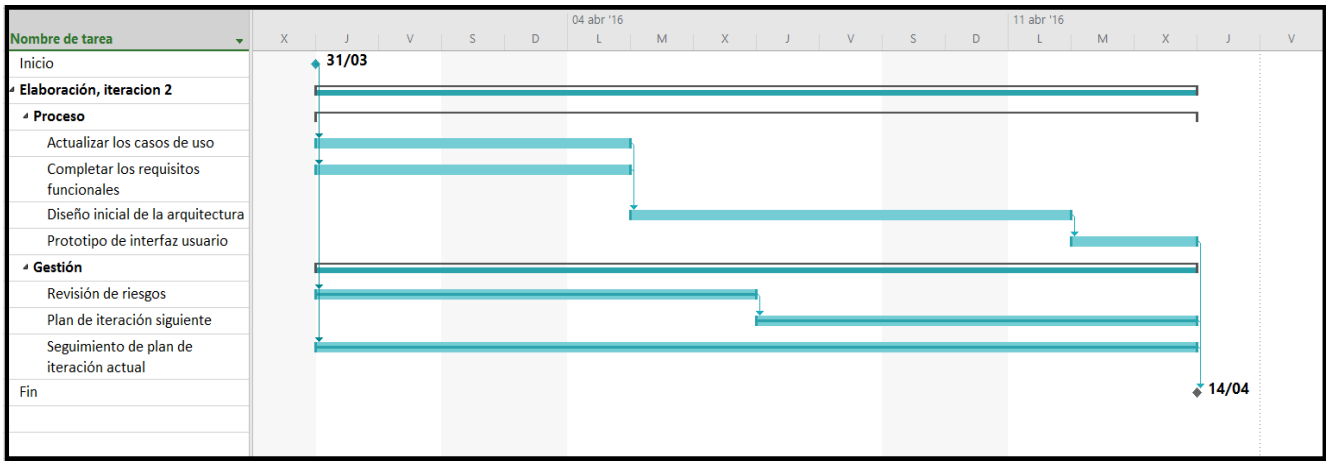

Figura 11: *Fase de elaboración, Iteración 2*

Fase de construcción, Iteración 1

	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1	Inicio	0 días	jue 14/04/16	jue 14/04/16	
2	**Construcción, iteración 1**	**12 días**	**jue 14/04/16**	**vie 29/04/16**	
3	**Proceso**	**12 días**	**jue 14/04/16**	**vie 29/04/16**	
4	Revisión de casos de uso, requisitos y diagramas	2 días	jue 14/04/16	vie 15/04/16	1
5	Diseño de base de datos	2 días	lun 18/04/16	mar 19/04/16	4
6	Implementación de base de datos	2 días	mié 20/04/16	jue 21/04/16	5
7	Implementación de los controladores de avatar con Kinect	3 días	vie 22/04/16	mar 26/04/16	6
8	Implementación de la primera versión de interfaz usuario	3 días	vie 22/04/16	mar 26/04/16	7
9	Implementación de los casos de uso	3 días	mié 27/04/16	vie 29/04/16	7
10	**Gestión**	**12 días**	**jue 14/04/16**	**vie 29/04/16**	
11	Revisión de riesgos	5 días	jue 14/04/16	mié 20/04/16	1
12	Plan de iteración siguiente	7 días	jue 21/04/16	vie 29/04/16	11
13	Seguimiento de plan de iteración actual	12 días	jue 14/04/16	vie 29/04/16	1
14	Fin	0 días	sáb 30/04/16	sáb 30/04/16	13;12;9;8

Tabla 11: *Fase de construcción, Iteración 1*

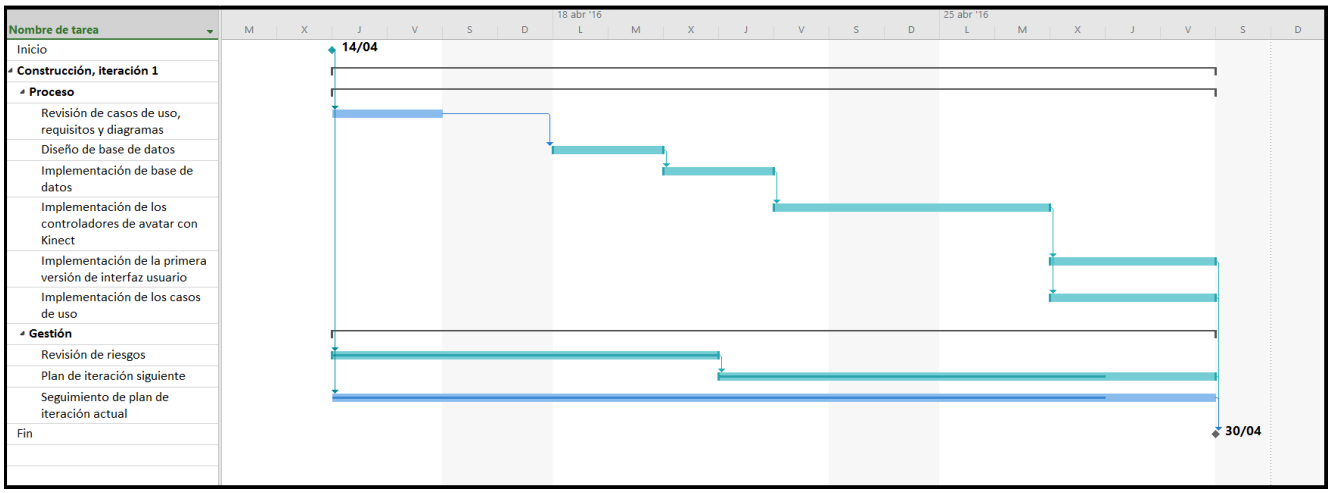

Figura 12: *Fase de construcción, Iteración 1*

Fase de construcción, Iteración 2

	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1	Inicio	0 días	mar 03/05/16	mar 03/05/16	
2	**Construcción, iteración 2**	**12 días**	**mar 03/05/16**	**mié 18/05/16**	
3	**Proceso**	**12 días**	**mar 03/05/16**	**mié 18/05/16**	
4	Actualizar los controladores de avatar	5 días	mar 03/05/16	lun 09/05/16	1
5	Implementación de los casos de uso	7 días	mar 10/05/16	mié 18/05/16	4
6	Mejora de la interfaz de usuario	7 días	mar 10/05/16	mié 18/05/16	4
7	**Gestión**	**12 días**	**mar 03/05/16**	**mié 18/05/16**	
8	Revisión de riesgos	5 días	mar 03/05/16	lun 09/05/16	1
9	Plan de iteración siguiente	7 días	mar 10/05/16	mié 18/05/16	8
10	Seguimiento de plan de iteración actual	12 días	mar 03/05/16	mié 18/05/16	1
11	Fin	0 días	jue 19/05/16	jue 19/05/16	10;9;6;5

Tabla 12: *Fase de construcción, Iteración 2*

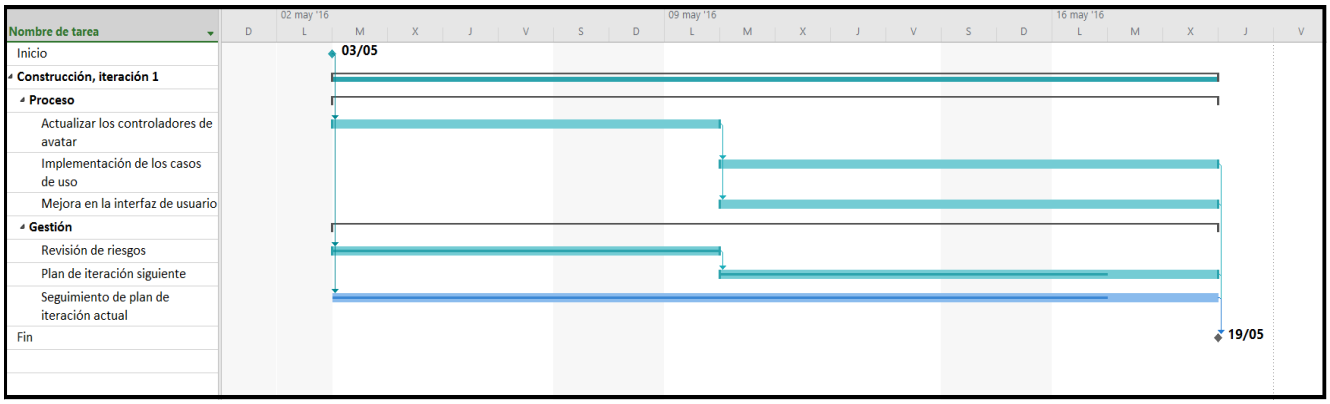

Figura 13: *Fase de construcción, Iteración 2*

Fase de construcción, Iteración 3

	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1	Inicio	0 días	jue 19/05/16	jue 19/05/16	
2	**Construcción, iteración 3**	**10 días**	**jue 19/05/16**	**mié 01/06/16**	
3	**Proceso**	**10 días**	**jue 19/05/16**	**mié 01/06/16**	
4	Modificación final de los controladores de avatar	5 días	jue 19/05/16	mié 25/05/16	1
5	Completar de los casos de uso restantes	5 días	jue 26/05/16	mié 01/06/16	4
6	Modificación final de la interfaz de usuario	5 días	jue 26/05/16	mié 01/06/16	4
7	**Gestión**	**10 días**	**jue 19/05/16**	**mié 01/06/16**	
8	Revisión de riesgos	5 días	jue 19/05/16	mié 25/05/16	1
9	Plan de iteración siguiente	5 días	jue 26/05/16	mié 01/06/16	8
10	Seguimiento de plan de iteración actual	10 días	jue 19/05/16	mié 01/06/16	1
11	Fin	0 días	jue 02/06/16	jue 02/06/16	10;9;6;5

Tabla 13: *Fase de construcción, Iteracción 3*

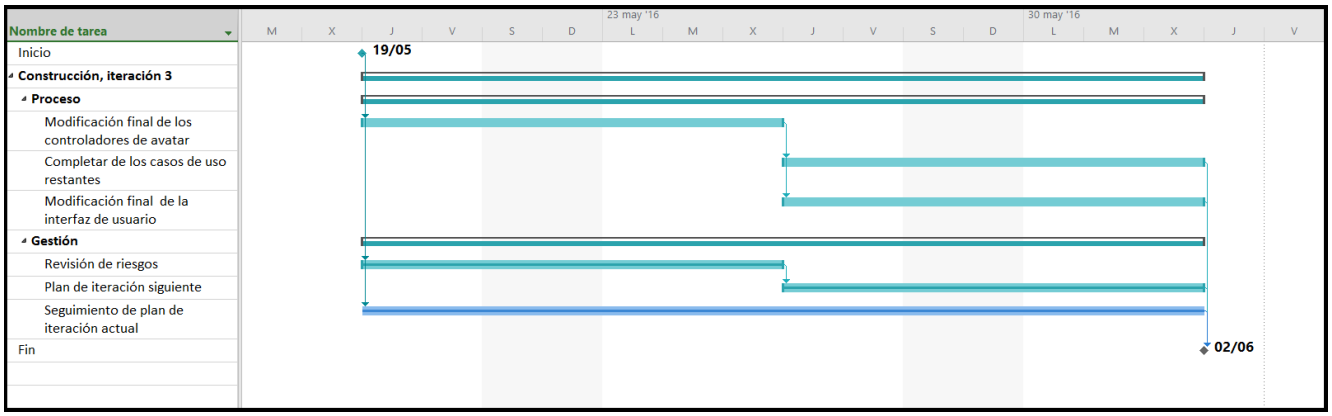

Figura 14: *Fase de construcción, Iteracción 3*

Fase de transición

	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1	Inicio	0 días	jue 02/06/16	jue 02/06/16	
2	**Transición**	**7 días**	**jue 02/06/16**	**vie 10/06/16**	
3	**Proceso**	**7 días**	**jue 02/06/16**	**vie 10/06/16**	
4	Realización de pruebas	2 días	jue 02/06/16	vie 03/06/16	1
5	Corrección de fallos detectados	4 días	lun 06/06/16	jue 09/06/16	4
6	Finalizar la aplicación	1 día	vie 10/06/16	vie 10/06/16	5
7	**Gestión**	**7 días**	**jue 02/06/16**	**vie 10/06/16**	
8	Revisión de riesgos	1 día	jue 02/06/16	jue 02/06/16	1
9	Seguimiento de plan de iteración actual	7 días	jue 02/06/16	vie 10/06/16	1
10	Fin	0 días	sáb 11/06/16	sáb 11/06/16	9;6

Tabla 14: *Fase de transición*

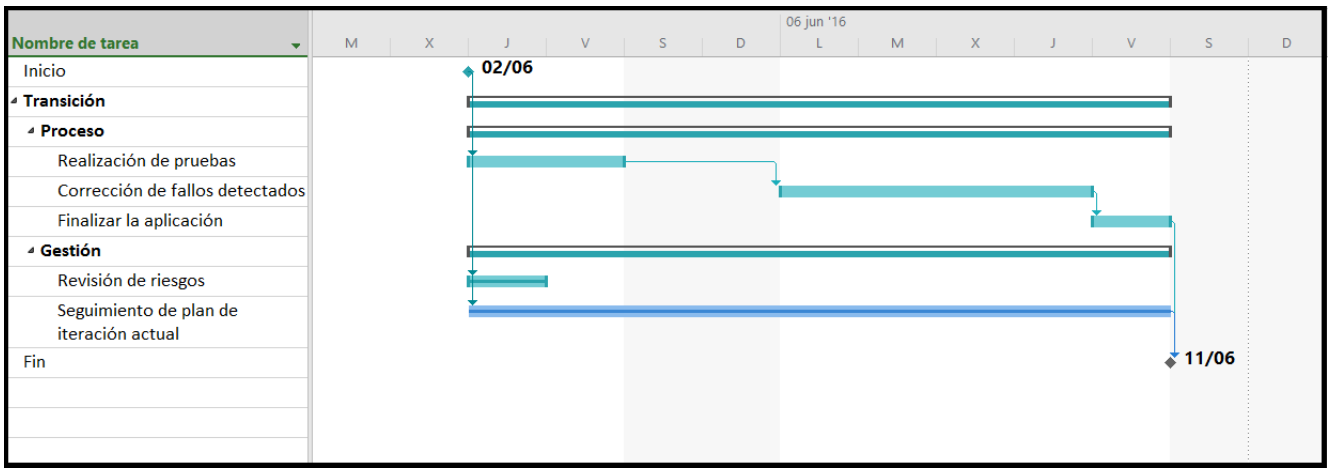

Figura 15: *Fase de transición*

3.3.3. Seguimiento de la planificación

A continuación, se muestra los retrasos y cambios sufridos en cada iteración de cada fase durante el desarrollo del proyecto.

	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1	construcción	0 días	lun 22/02/16	lun 22/02/16	
2	**Proyecto**	**71 días**	**lun 22/02/16**	**vie 10/06/16**	
3	**Inicio**	**12 días**	**lun 22/02/16**	**mar 08/03/16**	
4	Iteración 1	12 días	lun 22/02/16	mar 08/03/16	1
5	**Elaboración**	**20 días**	**lun 07/03/16**	**mié 13/04/16**	
6	Iteración 1	10 días	lun 07/03/16	vie 18/03/16	4
7	Iteración 2	10 días	jue 31/03/16	mié 13/04/16	6
8	**Construcción**	**36 días**	**jue 14/04/16**	**vie 03/06/16**	
9	Iteración 1	12 días	jue 14/04/16	vie 29/04/16	7
10	Iteración 2	16 días	lun 02/05/16	mar 24/05/16	9
11	Iteración 3	12 días	jue 19/05/16	vie 03/06/16	10
12	**Transición**	**7 días**	**jue 02/06/16**	**vie 10/06/16**	
13	Iteración 1	7 días	jue 02/06/16	vie 10/06/16	11
14	Fin de proyecto	0 días	sáb 11/06/16	sáb 11/06/16	13

Tabla 15: *Plan de desarrollo real*

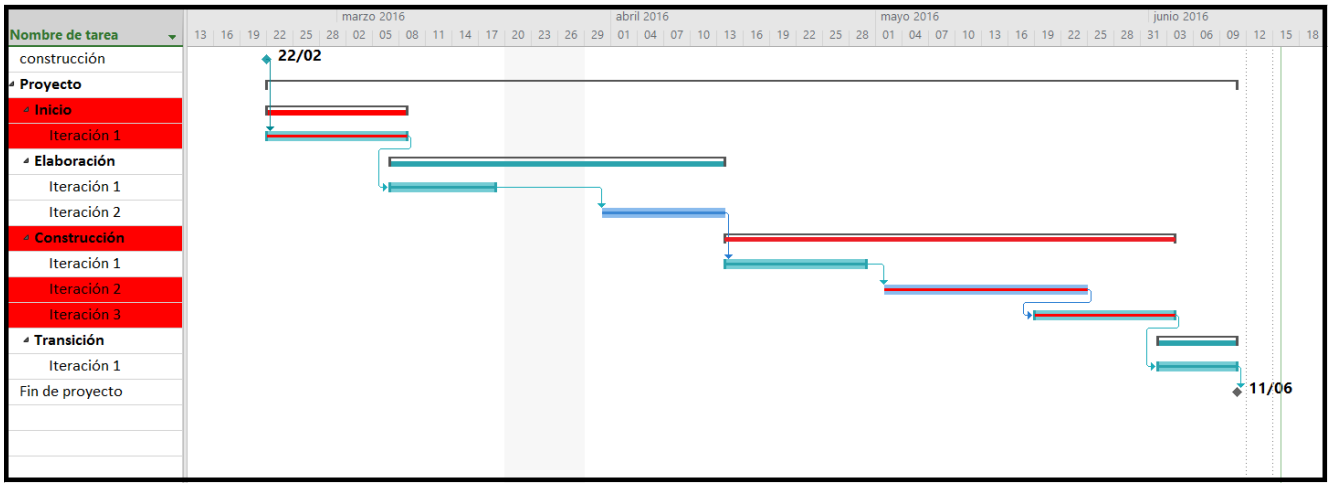

Figura 16: *Plan de desarrollo real*

Fase de inicio: Debido a la inesperada dificultad para encontrar informaciones y ejemplos adecuados de Kinect for Windows v2 para Unity, la duración de la iteración inicial se vio afectada y con el fin de respectar las fechas preestablecidas en la planificación tuvo que invertir horas extras.

Fase de construcción, iteración 2: La inestabilidad de los datos del esqueleto humano proporcionados por Kinect supuso un aumento significativamente para la duración de esta iteración. Para no alterar la planificación inicial fue necesario invertir horas extras.

Fase de construcción, iteración 3: Los fallos en el despliegue de la aplicación y problemas con la conexión del sensor Kinect a pc afectó negativamente la duración de esta iteración. A igual que los dos casos anteriores para poder seguir la planificación tuvo que trabajar horas extras.

3.5. Costes del proyecto

A continuación, se especificar los costes de proyectos en dos términos: costes de hardware y de personal.

3.5.1. Costes de hardware

En cuando a costes de hardware, la adquisición de hardware tuvo que respectar los requisitos mínimos exigidos para el funcionamiento de Kinect for Windows v2:

- Procesador Intel Core 5 o 7.
- 8 GB de RAM o más.
- Tarjeta Gráfica de más 64 MB de memoria RAM.
- Puertos USB 3.0.
- Espacio libre en el disco duro de más de 10 GB

Especificación de hardware adquirido:

- Portátil
  - Modelo: Asus R510JX-DM301D
  - Dimensions: 25.1x38x3.2cm
  - Weight: 2.4kg
  - Screen: 15.6in, 1920x1080
  - CPU: Intel Core i7 4720HQ, 2.6 GHz
  - RAM: 8GB DDR3L
  - Storage: 1TB HDD
  - GPU: NVIDIA GeForce 950M, 2GB RAM
  - Ports: 1 USB 2.0, 2 USB 3.0, VGA, HDMI, Ethernet, Headphones

- Optical drive: DVD-RW
- SO: ninguna
- Coste: 740,99 €

- El sensor Kinect v2.0 & adaptador para pc
  - Coste: 200 €

Coste de hardware imputable al proyecto: (740.99+200) * 6/48 = 117,62 €

3.5.2 costes de personal

El salario promedio de un programador por hora es 12 €, por lo tanto, el coste personal ha sido:

Coste ordinario: 71 x 4 x 12 = 3408 €
Coste ordinario más extra: 3408 + (5 x 4 x 12) = 3648 €

y el coste total de proyecto ha sido:

3648 + 117,62= 3765,62 €

4. ANÁLISIS Y DISEÑO DE LA APLICACIÓN

4.1. Requisitos del sistema

4.1.1. Actores

Breve descripción de roles de usuarios que interactúan con sistema.

ACT-001	Usuario
Descripción	Este actor representa a un usuario de la aplicación, puede ser tanto paciente como terapeuta.

ACT-002	Paciente
Descripción	Este actor representa al usuario que realizará los ejercicios de rehabilitación.

4.1.2. Requisitos funcionales

Definición de los servicios que el sistema debe proporcionar, cómo debe reaccionar a una entrada particular y cómo se debe comportar ante situaciones particulares.

Requisitos funcionales de la aplicación

FRQ-001	Registro
Descripción	El sistema deberá permitir al usuario registrar a un paciente.

FRQ-002	Identificar
Descripción	El sistema deberá permitir a todos los pacientes registrados identificarse con el Nick ante el sistema.

FRQ-003	Acceso
Descripción	El sistema deberá permitir el uso de aplicación a todos los pacientes identificados.

FRQ-004	Mostrar ejercicios
Descripción	El sistema deberá exponer todos los ejercicios disponibles de la aplicación en menú principal con la debida estructura.

FRQ-005	Mostrar informaciones
Descripción	El sistema deberá permitir consultar datos personales relativos a todos los pacientes.

FRQ-006	Iniciar ejercicios
Descripción	El sistema deberá iniciar la realización del ejercicio cuando los usuarios consideren oportuno.

FRQ-007	Pausar el ejercicio
Descripción	El sistema deberá pausar los ejercicios cuando los usuarios lo desean.

FRQ-007	Restaurar el ejercicio

Descripción	El sistema deberá permitir a los usuarios restaurar los ejercicios pausados.

FRQ-009	**Visualizar datos de ejercicios**
Descripción	El sistema deberá permitir visualizar los datos relevantes de los ejercicios realizados.

FRQ-010	**Lista de ejercicios realizados**
Descripción	El sistema deberá mostrar todos los ejercicios realizados del paciente identificado en forma de lista.

FRQ-016	**Control por voz**
Descripción	El sistema deberá permitir al usuario usa la aplicación mediante comandos por voz.

FRQ-018	**Instrucciones**
Descripción	El sistema deberá mostrar las instrucciones necesarias para que los usuarios pueden llevar acabo los ejercicios.

FRQ-019	**Cerrar sesión**
Descripción	El sistema deberá permitir al usuario cerrar la sesión siempre y cuando lo desean.

FRQ-020	**Cambios de informaciones de paciente**
Descripción	El sistema deberá permitir al usuario cambiar las informaciones personales y médicas de paciente.

FRQ-021	**Eliminación de paciente**
Descripción	El sistema deberá permitir al usuario eliminar las informaciones del usuario.

FRQ-022	**Eliminación de datos de los ejercicios**
Descripción	El sistema deberá permitir al usuario eliminar datos de los ejercicios realizados.

FRQ-023	**Fin de los ejercicios**
Descripción	El sistema deberá indicar adecuadamente al usuario la finalización de cada ejercicio.

FRQ-024	**Cambiar de ejercicios**
Descripción	El sistema deberá permitir al paciente cambiar de ejercicios en cualquier momento.

FRQ-025	**Salir de ejercicios**
Descripción	El sistema deberá permitir al paciente salir de ejercicio en cualquier momento.

Requisitos funcionales de actividad de salón (actividad de vida diaria)

FRQ-026	Ambientación de actividad de salón
Descripción	El sistema deberá simular el entorno apropiado de un salón.

FRQ-027	Avatar para actividad de salón
Descripción	El sistema deberá simular el paciente mediante un avatar humanoide.

FRQ-028	Control del avatar para actividad de salón
Descripción	El sistema deberá permitir al paciente controlar el avatar mediante las rotaciones de hombros, codos y muñecas.

FRQ-029	Animación de coger una taza
Descripción	El sistema deberá simular la animación de sostener una taza de café.

FRQ-030	Contacto entre taza y boca
Descripción	El sistema deberá detectar el contacto entre la boca y la taza del avatar controlado por el paciente.

Requisitos funcionales de actividad de lavabo (actividad de vida diaria)

FRQ-031	Configurar el número de repeticiones
Descripción	El sistema deberá permitir al usuario configurar el número de repeticiones de cepillado de los dientes.

FRQ-032	Ambientación de actividad de lavabo
Descripción	El sistema deberá simular el entorno apropiado de un baño con lavabo.

FRQ-033	Avatar para actividad de lavabo
Descripción	El sistema deberá simular al paciente mediante un avatar humanoide.

FRQ-034	Control del avatar para actividad de lavabo
Descripción	El sistema deberá permitir al paciente controlar el avatar mediante las rotaciones de hombros y flexiones de codos.

FRQ-035	Animación de coger cepillo de dientes
Descripción	El sistema deberá simular la animación de agarrar un cepillo de dientes cuando la mano derecha del avatar entra en contacto con el cepillo.

FRQ-036	Animación de coger pasta de dientes
Descripción	El sistema deberá simular la animación de agarrar una pasta de dientes cuando la mano izquierda del avatar entra en contacto con la pasta.

FRQ-037	Coger cepillo de dientes
Descripción	El sistema deberá permitir al paciente coger el cepillo de dientes con la mano derecha del avatar controlado.

FRQ-038	Coger pasta de dientes
Descripción	El sistema deberá permitir al paciente coger la pasta de dientes con la mano izquierda del avatar controlado.

FRQ-039	Aplicar pasta de dientes
Descripción	El sistema deberá permitir al paciente aplicar la pasta sobre cepillo.

FRQ-040	Mojar el cepillo de dientes
Descripción	El sistema deberá permitir al paciente mojar el cepillo.

FRQ-041	Cepillar los dientes
Descripción	El sistema deberá detectar los movimientos de cepillado los dientes.

Requisitos funcionales de actividad de emparejar cartas (mini-juego)

FRQ-042	Ambientación de actividad de emparejar cartas
Descripción	El sistema deberá mostrar múltiples parejas de cartas de forma aleatoria que cubre toda la pantalla.

FRQ-043	Avatar para actividad de emparejar cartas
Descripción	El sistema deberá simular la mano derecha del paciente mediante un avatar de una mano derecha.

FRQ-044	Animación de cierre de la mano
Descripción	El sistema deberá simular la animación de cierre de la mano para el avatar.

FRQ-045	Seleccionar las cartas
Descripción	El sistema deberá permitir al paciente seleccionar las cartas mediante el cierre de la mano.

FRQ-046	Eliminar las cartas
Descripción	El sistema deberá permitir al paciente eliminar las cartas cuando haya seleccionado su pareja.

FRQ-047	Puntuación de actividad de emparejar cartas

Descripción	El sistema deberá permitir al paciente obtener puntuaciones mediante eliminación de las cartas iguales.

Requisitos funcionales de actividad de despejar (mini-juego)

FRQ-048	**Ambientación de actividad de despejar**
Descripción	El sistema deberá mostrar un entorno lleno de cubos que tienen que ser despejados por el paciente.

FRQ-049	**Avatar para actividad de despejar**
Descripción	El sistema deberá simular la mano derecha del paciente mediante un avatar.

FRQ-050	**Despejar puntos**
Descripción	El sistema deberá despejar el área donde la mano derecha del paciente la mantiene cerrada.

FRQ-051	**Despejar áreas**
Descripción	El sistema deberá despejar el área encerrada por la trayectoria del paso de la mano del paciente.

FRQ-052	**Puntuaciones de actividad de despejar**
Descripción	El sistema deberá puntuar al paciente dependiendo del área despejada.

4.1.3. Requisitos de información

FRQ-053	**Datos de actividad de salón**
Descripción	El sistema deberá almacenar el nombre del ejercicio, el id del paciente que realizó el ejercicio, la fecha de la realización, el tiempo empleado en la actividad y ángulos de máximo y mínimo apertura de hombros y codos.

FRQ-054	**Datos de actividad de lavabo**
Descripción	El sistema deberá almacenar el nombre del ejercicio, el id del paciente que realizó el ejercicio, la fecha de la realización, repeticiones realizadas, tiempo que tardó en acabar la actividad y los movimientos de las extremidades.

FRQ-055	**Datos de actividad de mini juegos**
Descripción	El sistema deberá almacenar el nombre de la actividad, el id del paciente que realizó el ejercicio, la fecha de la realización y la puntuación obtenido.

FRQ-056	**Información personal**

Descripción	El sistema deberá almacenar los datos personales de los pacientes registrados (Nombre, Apellidos, Dirección, Teléfono, Código postal, Provincia, Localidad).

FRQ-057	**Información Médica**
Descripción	El sistema deberá almacenar información médica del paciente relevante para la aplicación, miembros de cuerpo afectado y fecha de la lesión.

4.1.4. Requisitos no funcionales, limitaciones que afectan a los servicios o funciones del sistema, tales como limitaciones de tiempo, sobre el proceso de desarrollo, estándares, etc.

NFR-001	**Usabilidad**
Descripción	El sistema deberá implementar la interfaz de las aplicaciones disponibles para Windows 8

NFR-002	**Aprendizaje de la interfaz**
Descripción	El interfaz de sistema deberá ser tan simple como sea posible que requerirá menos de un día de aprendizaje.

NFR-003	**Requisitos mínimos del sistema**
Descripción	●Procesador de 64-bit (x64) dual core de 3.1 GHz (2 núcleos lógicos por cada uno) o más rápido ● Puertos USB 3.0 ● 4 GB de RAM ● Tarjeta gráfica que soporte DirectX 11 ● Windows 8.1

NFR-004	**Fiabilidad de las medidas ángulos**
Descripción	El margen de error de los ángulos obtenidos por el sistema tiene que estar comprendida entre 0 y 5 grados.

NFR-005	**Implementación software**
Descripción	El sistema deberá desarrollarse bajo las siguientes herramientas de desarrollo: C# , Visual Studio 2015, Microsoft Kinect SDK Versión 2.4 y Unity 5.3.4f1

NFR-006	**Fecha y hora**
Descripción	El sistema deberá tomar la fecha y hora del reloj del sistema.

NFR-007	**Kinect**
Descripción	El sistema deberá ser compatible con Kinect 2.0 for Windows y Kinect for Xbox One.

NFR-008	**Resolución**
Descripción	El sistema deberá mostrar los contenidos correctamente en las pantallas de resolución 1280 x 720.

NFR-009	**Micrófono**

Descripción	El sistema deberá reconocer los comandos por voz mediante los micrófonos incorporados en el sensor Kinect.

4.2. Casos de uso

4.2.1. Diagrama de casos de uso

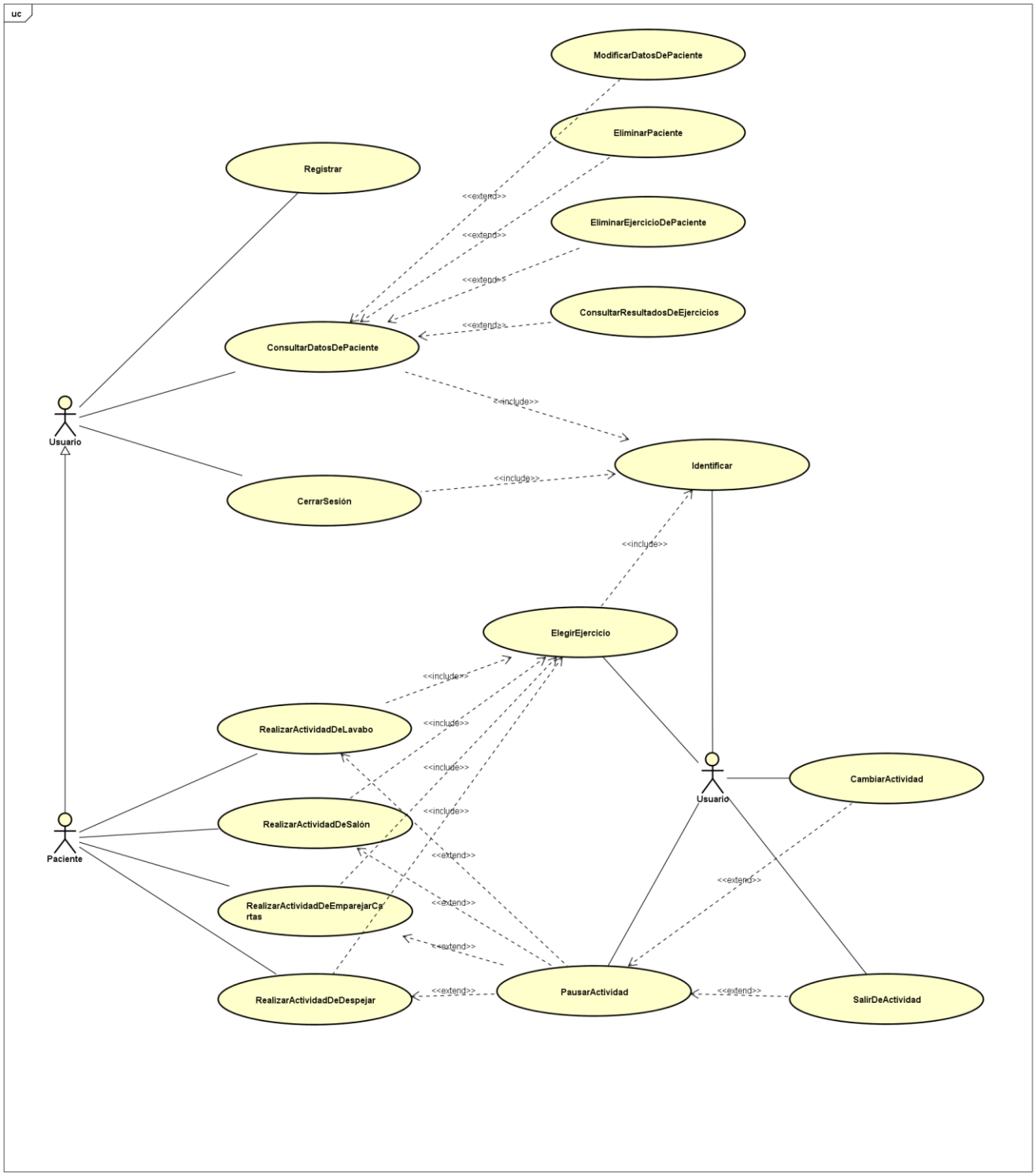

Figura 17: *Diagrama de casos de uso*

Descripción y diagramas de los casos de uso

Registrar

UC-001	**Registrar**	
Descripción	El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cuando el actor Usuario (ACT-001) desee registrar un paciente en el sistema.	
Precondición	El paciente no se encuentra en el sistema.	
Secuencia normal	**Paso**	**Acción**
	1	El actor Usuario (ACT-001) selecciona registrar.
	2	El sistema solicita la información del paciente a registrar.
	3	El actor Usuario (ACT-001) rellena el formulario con datos de paciente.
	4	El sistema comprueba los datos introducidos y registra al paciente dentro del sistema.
Postcondición	El paciente (ACT-002) queda registrado en el sistema.	
Excepción	**Paso**	**Acción**
	4	Si el paciente ya está registrado, el sistema informa al actor Usuario (ACT-001) del error y el caso de uso continuara en el paso 3.
	4	Si existen campos obligatorios sin rellenar, el sistema informa al actor Usuario (ACT-001) del error y el caso de uso continuara en el paso 3.

Tabla 16: *Casos de uso - Registrar*

Identificar

UC-002	**Identificar**	
Descripción	El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cuando el actor Usuario (ACT-001) desee identificar ante el sistema.	
Precondición	El paciente debe estar previamente registrado en el sistema.	
Secuencia normal	**Paso**	**Acción**
	1	El actor Usuario (ACT-001) introduce el Nick de paciente y procede a iniciar la sesión.
	2	El sistema comprueba el Nick de paciente, extraer los datos de paciente e iniciar la sesión del paciente.
Postcondición	El paciente queda identificado ante el sistema.	
Excepción	**Paso**	**Acción**
	2	Si el Nick introducido por actor Usuario (ACT-001) no es válido, el sistema informa del error. El caso de uso continuara en el paso 1.

Tabla 17: *Casos de uso - Identificar*

Consultar datos de paciente

UC-003	**Consultar datos de paciente**	
Descripción	El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cuando el actor Usuario (ACT-001) desee consultar los datos del paciente identificado.	
Precondición	Debe haber identificado ante el sistema.	
Secuencia normal	**Paso**	**Acción**
	1	El actor Usuario (ACT-001) selecciona consultar datos de paciente.
	2	El sistema muestra los datos de paciente, incluyendo la lista de las actividades realizadas.
	3	El actor usuario (ACT-001) selecciona visualizar detalles de una actividad. A continuación, este caso de uso continuara en el caso de uso “consultar resultados de ejercicio”.
Postcondición	El sistema muestra los datos del actor paciente (ACT-002).	
Excepción	**Paso**	**Acción**
	3	Si el actor Usuario (ACT-001) selecciona eliminar los datos del paciente, A continuación, este caso de uso continuara en el caso de uso de “Eliminar paciente”.
	3	Si el actor Usuario (ACT-001) selecciona modificar los datos del paciente, A continuación, este caso de uso continuara en el caso de uso de “Modificar datos de paciente”.
	3	Si el actor Usuario (ACT-001) selecciona eliminar los datos de una actividad, A continuación, este caso de uso continuara en el caso de uso de “Eliminar ejercicio de paciente”.

Tabla 18: *Casos de uso – Consultar datos de paciente*

Consultar resultado de ejercicio

UC-004	**Consultar resultado de ejercicio**	
Descripción	El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cuando el actor Usuario (ACT-001) desee visualizar los datos detallados de un ejercicio.	
Precondición	El actor Usuario (ACT-001) debería haber realizado correctamente el caso de uso “Consultar datos de paciente”.	
Secuencia normal	**Paso**	**Acción**
	1	El actor Usuario (ACT-001) selecciona un ejercicio entre los ejercicios de la lista para visualizar sus detalles.
	2	El sistema muestra los datos detallados de ejercicio seleccionado al actor Usuario (ACT-001).
Postcondición	Los datos detallados de ejercicio deben ser mostrado al actor Usuario (ACT-001).	
Excepción	**Paso**	**Acción**
	-	-

Tabla 19: *Casos de uso – Consultar resultado de ejercicio*

Eliminar Paciente

UC-005	**Eliminar Paciente**	
Descripción	El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cuando el actor Usuario (ACT-001) desee eliminar el paciente identificado del sistema.	
Precondición	El actor Usuario (ACT-001) debe haber realizado correctamente el caso de uso de “Consultar los datos de paciente”.	
Secuencia normal	**Paso**	**Acción**
	1	El actor Usuario (ACT-001) selecciona eliminar el paciente identificado.
	2	El sistema pide la confirmación al actor Usuario para proceder la eliminación.
	3	El actor Usuario (ACT-001) confirma la acción.
	4	El sistema elimina los datos de paciente.
Postcondición	Todos los datos relacionados al paciente deben ser eliminados	
Excepción	**Paso**	**Acción**
	3	Si el actor Usuario (ACT-001) cancela la acción, a continuación, este caso de uso queda sin efecto.

Tabla 20: *Casos de uso – Eliminar paciente*

Modificar datos de paciente

UC-006	**Modificar datos de paciente**	
Descripción	El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cuando el actor Usuario (ACT-001) desee modificar los datos del paciente.	
Precondición	El actor Usuario (ACT-001) debe haber realizado correctamente el caso de uso de “Consultar los datos de paciente”.	
Secuencia normal	**Paso**	**Acción**
	1	El actor Usuario (ACT-001) introduce los datos que requiere realizar el cambio y selecciona modificar los datos de paciente identificado.
	2	El sistema comprueba que no quede campos obligatorios sin rellenar y procede a modifica los datos del paciente. Una vez finalizado la operación el sistema informa al usuario del éxito de la operación.
Postcondición	Los datos del paciente quedan modificados en el base de datos.	
Excepción	**Paso**	**Acción**
	2	Si existen campos obligatorios sin rellenar, el sistema informa al actor Usuario (ACT-001) del error y el caso de uso continuara en el paso 1.

Tabla 21: *Casos de uso – Modificar datos de paciente*

Eliminar ejercicio de paciente

UC-007	**Eliminar ejercicio de paciente**	
Descripción	El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cuando el actor Usuario (ACT-001) desee eliminar uno de los ejercicios la lista.	
Precondición	El actor Usuario (ACT-001) debe haber realizado correctamente el caso de uso de “Consultar los datos de paciente”.	
Secuencia normal	**Paso**	**Acción**
	1	El actor Usuario (ACT-001) selecciona eliminar uno de los ejercicios de la lista.
	2	El sistema eliminar los datos del ejercicio del base de datos.
Postcondición	Los datos de ejercicio seleccionado son borrados del base de datos.	
Excepción	**Paso**	**Acción**
	-	-

Tabla 22: *Casos de uso – Eliminar ejercicio de paciente*

Cerrar sesión

UC-008	**Cerrar sesión**	
Descripción	El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cuando el actor Usuario (ACT-001) desee cerrar la sesión del paciente actual.	
Precondición	Debe haber inicializado una sesión de paciente.	
Secuencia normal	**Paso**	**Acción**
	1	El actor Usuario (ACT-001) selecciona cerrar la sesión.
	2	El sistema procede a cerrar la sesión actual y volver a la pantalla de Login.
Postcondición	Se cierra la sesión de paciente actual y volver a la pantalla de Login.	
Excepción	**Paso**	**Acción**
	-	-

Tabla 23: *Casos de uso – Cerrar sesión*

Elegir ejercicio

UC-009	**Elegir ejercicio**	
Descripción	El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cuando el actor Usuario (ACT-001) desee iniciar uno de los ejercicios de la lista.	
Precondición	Debe haber inicializado una sesión de paciente.	
Secuencia normal	**Paso**	**Acción**
	1	El actor Usuario (ACT-001) selecciona iniciar ejercicio.
	2	El sistema muestra dos categorías de ejercicios disponibles.
	3	El actor Usuario (ACT-001) selecciona una categoría de las mostradas.
	4	El sistema muestra la lista de los ejercicios disponibles de la categoría seleccionada.
	5	El actor Usuario (ACT-001) selecciona la actividad de lavabo. A continuación, este caso de uso continuara en el caso de uso “Realizar actividad de Lavabo”.
Postcondición	Inicializara la actividad seleccionada.	
Excepción	**Paso**	**Acción**
	5	Si el actor Usuario (ACT-001) selecciona la actividad de Emparejar cartas. A continuación, este caso de uso continuara en el caso de uso “Realizar actividad de Emparejar cartas”.
	5	Si el actor Usuario (ACT-001) selecciona la actividad de despejar. A continuación, este caso de uso continuara en el caso de uso “Realizar actividad de Despejar”.
	5	Si el actor Usuario (ACT-001) selecciona la actividad de salón. A continuación, este caso de uso continuara en el caso de uso “Realizar actividad de Salón”.

Tabla 24: *Casos de uso – Elegir ejercicio*

Realizar actividad de lavabo

UC-010	**Realizar actividad de lavabo**	
Descripción	El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cuando el actor Paciente (ACT-002) desee iniciar la actividad de lavabo.	
Precondición	Debe haber realizado el caso de uso “Elegir ejercicio” correctamente.	
Secuencia normal	**Paso**	**Acción**
	1	El actor Usuario (ACT-001) selecciona iniciar la actividad de lavabo e introduce el número de repeticiones, “n”.
	2	El sistema cargar la escena de la actividad de lavabo y pide al actor Paciente (ACT-002) situarse en el centro del campo de detección del sensor Kinect.
	3	El actor Paciente (ACT-002) sitúa en el centro.
	4	El sistema inicia la cuenta atrás del inicio de la actividad. Y una vez finalizada pide al actor Paciente (ACT-002) alcanzar la pasta de dientes y el cepillo de dientes.
	5	El actor Paciente (ACT-002) alcanza la pasta de dientes y el cepillo de dientes.
	6	El sistema pide al actor Paciente (ACT-002) aplicar la pasta sobre el cepillo.
	7	El actor Paciente (ACT-002) aplica la pasta sobre el cepillo.
	8	El sistema pide al actor Paciente (ACT-002) mojar el cepillo en el agua.
	9	El actor Paciente (ACT-002) moja el cepillo en el agua.
	10	El sistema pide al actor Paciente (ACT-002) cepillar “n” veces los dientes.
	11	El actor Paciente (ACT-002) cepilla “n” veces los dientes.
	12	El sistema indica al actor Paciente (ACT-002) fin de la actividad y guarda los datos del ejercicio.
Postcondición	Los datos del ejercicio realizado son guardados en el base de datos.	
Excepción	**Paso**	**Acción**
	3, 5, 7, 9, 11	Si actor Usuario pausa la actividad, A continuación, este caso de uso continuara en el caso de uso “Pausar actividad”.

Tabla 25: *Casos de uso – Realizar actividad de lavabo*

Realizar actividad de salón

UC-011	**Realizar actividad de salón**	
Descripción	El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cuando el actor Paciente (ACT-002) desee realizar la actividad de salón.	
Precondición	Debe haber realizado el caso de uso “Elegir ejercicio” correctamente.	
Secuencia normal	**Paso**	**Acción**
	1	El actor Usuario (ACT-001) selecciona iniciar la actividad de salón.
	2	El sistema cargar la escena de la actividad de salón y pide al actor Paciente (ACT-002) situarse en el centro del campo de detección del sensor Kinect.
	3	El actor Paciente (ACT-002) sitúa en el centro.
	4	El sistema pide al actor Paciente (ACT-002) coger la taza de café.
	5	El actor Paciente (ACT-002) coge la taza de café.
	6	El sistema pide al actor Paciente (ACT-002) beber con la taza de café.
	7	El actor Paciente (ACT-002) lleva la taza hasta la boca.
	8	El sistema indica al actor Paciente (ACT-002) fin de la actividad y guarda los datos del ejercicio.
Postcondición	Los datos del ejercicio realizado son guardados en el base de datos.	
Excepción	**Paso**	**Acción**
	3, 5 ,7	Si actor Usuario pausa la actividad, A continuación, este caso de uso continuara en el caso de uso “Pausar actividad”.

Tabla 26: *Casos de uso – Realizar actividad de salón*

Realizar actividad emparejar cartas

UC-012	**Realizar actividad emparejar cartas**	
Descripción	El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cuando el actor Paciente (ACT-002) desee realizar la actividad emparejar cartas.	
Precondición	Debe haber realizado el caso de uso “Elegir ejercicio” correctamente.	
Secuencia normal	**Paso**	**Acción**
	1	El actor Usuario (ACT-001) selecciona iniciar la actividad de emparejar cartas.
	2	El sistema cargar la escena de la actividad de emparejar cartas y pide al actor Paciente (ACT-002) situarse en el centro del campo de detección del sensor Kinect.
	3	El actor Paciente (ACT-002) sitúa en el centro.
	4	El sistema iniciar la cuenta atrás y una vez finalizada pide al actor Paciente (ACT-002) eliminar todas las parejas de las cartas.
	5	El actor Paciente (ACT-002) selecciona todas las parejas de cartas y elimina todas ellas.
	6	El sistema indica al actor Paciente (ACT-002) fin de la actividad y guarda los datos del ejercicio.
Postcondición	Los datos del ejercicio realizado son guardados en el base de datos.	
Excepción	**Paso**	**Acción**
	2	Si es la primera vez que realizar la actividad el sistema muestra las instrucciones básicas al actor Paciente (ACT-002). A continuación, este caso de uso continuara en el paso 2.
	3, 5	Si actor Usuario pausa la actividad, A continuación, este caso de uso continuara en el caso de uso “Pausar actividad”.

Tabla 27: *Casos de uso – Realizar actividad emparejar cartas*

Realizar actividad despejar

UC-013	**Realizar actividad despejar**	
Descripción	El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cuando el actor Paciente (ACT-002) desee realizar la actividad despejar.	
Precondición	Debe haber realizado el caso de uso “Elegir ejercicio” correctamente.	
Secuencia normal	**Paso**	**Acción**
	1	El actor Paciente (ACT-002) selecciona realizar la actividad despejar.
	2	El sistema cargar la escena de la actividad de emparejar cartas y pide al actor Paciente (ACT-002) situarse en el centro del campo de detección del sensor Kinect.
	3	El actor Paciente (ACT-002) sitúa en el centro.
	4	El sistema iniciar la cuenta atrás y una vez finalizada pide al actor Paciente (ACT-002) despejar el máximo área posible, a su vez inicia la cuenta atrás para finalizar la actividad.
	5	El actor Paciente (ACT-002) despeja el área.
	6	El sistema finaliza la cuenta atrás y guarda los datos de ejercicio.
Postcondición	Los datos del ejercicio realizado son guardados en el base de datos.	
Excepción	**Paso**	**Acción**
	2	Si es la primera vez que realizar la actividad el sistema muestra las instrucciones básicas al actor Paciente (ACT-002). A continuación, este caso de uso continuara en el paso 2.
	3, 5	Si actor Usuario pausa la actividad, A continuación, este caso de uso continuara en el caso de uso “Pausar actividad”.

Tabla 28: *Casos de uso – Realizar actividad despejar*

Pausar actividad

UC-014	**Pausar actividad**	
Descripción	El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cuando el actor Usuario (ACT-001) desee pausa la actividad.	
Precondición	Está en el curso de la realización de una actividad.	
Secuencia normal	**Paso**	**Acción**
	1	El actor Usuario (ACT-001) ejecutar el comando de pausar la actividad.
	2	El sistema pausar la actividad y muestra el menú de la emergencia.
	3	El actor Usuario (ACT-001) ejecutar el comando de reanudar la actividad.
	4	El sistema recuperar la actividad en pause.
Postcondición	Se reanudará la actividad en pause.	
Excepción	**Paso**	**Acción**
	3	Si el actor Usuario (ACT-001) selecciona Cambiar de actividad. A continuación, este caso de uso continuara en el caso de uso “Cambiar actividad”.
	3	Si el actor Usuario (ACT-001) selecciona Salir de actividad. A continuación, este caso de uso continuara en el caso de uso “Salir de actividad”.

Tabla 29: *Casos de uso – Pausar actividad*

Cambiar actividad

UC-015	**Cambiar actividad**	
Descripción	El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cuando el actor Usuario (ACT-001) desee cambiar la actividad actual por otra.	
Precondición	Debe haber realizado el caso de uso “Pausar actividad” correctamente.	
Secuencia normal	**Paso**	**Acción**
	1	El actor Usuario (ACT-001) selecciona una de las opciones de cargar una nueva actividad.
	2	El sistema reiniciar la escena de la actividad actual.
Postcondición	Debe cargar correctamente la escena de la actividad seleccionada.	
Excepción	**Paso**	**Acción**
	2	Si el actor Usuario (ACT-001) ha seleccionado cargar la actividad siguiente, el sistema cargara la escena de la actividad siguiente.
	2	Si el actor Usuario (ACT-001) ha seleccionado cargar la actividad anterior, el sistema cargara la escena de la actividad anterior.

Tabla 30: *Casos de uso – Cambiar actividad*

Salir de actividad

UC-016	**Salir de actividad**	
Descripción	El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cuando el actor Usuario (ACT-001) desee salir de la actividad actual a menú principal.	
Precondición	Debe haber realizado el caso de uso “Pausar actividad” correctamente.	
Secuencia normal	**Paso**	**Acción**
	1	El actor Usuario (ACT-001) selecciona la opción de salir de la actividad.
	2	El sistema finalizar la actividad en pause sin guarda los datos en el base de datos y cargar la escena de menú principal.
Postcondición	Debe haber finalizado la actividad en pause y cargado la escena de menú principal.	
Excepción	**Paso**	**Acción**
	-	-

Tabla 31: *Casos de uso – Salir de actividad*

4.3. Modelo de dominio

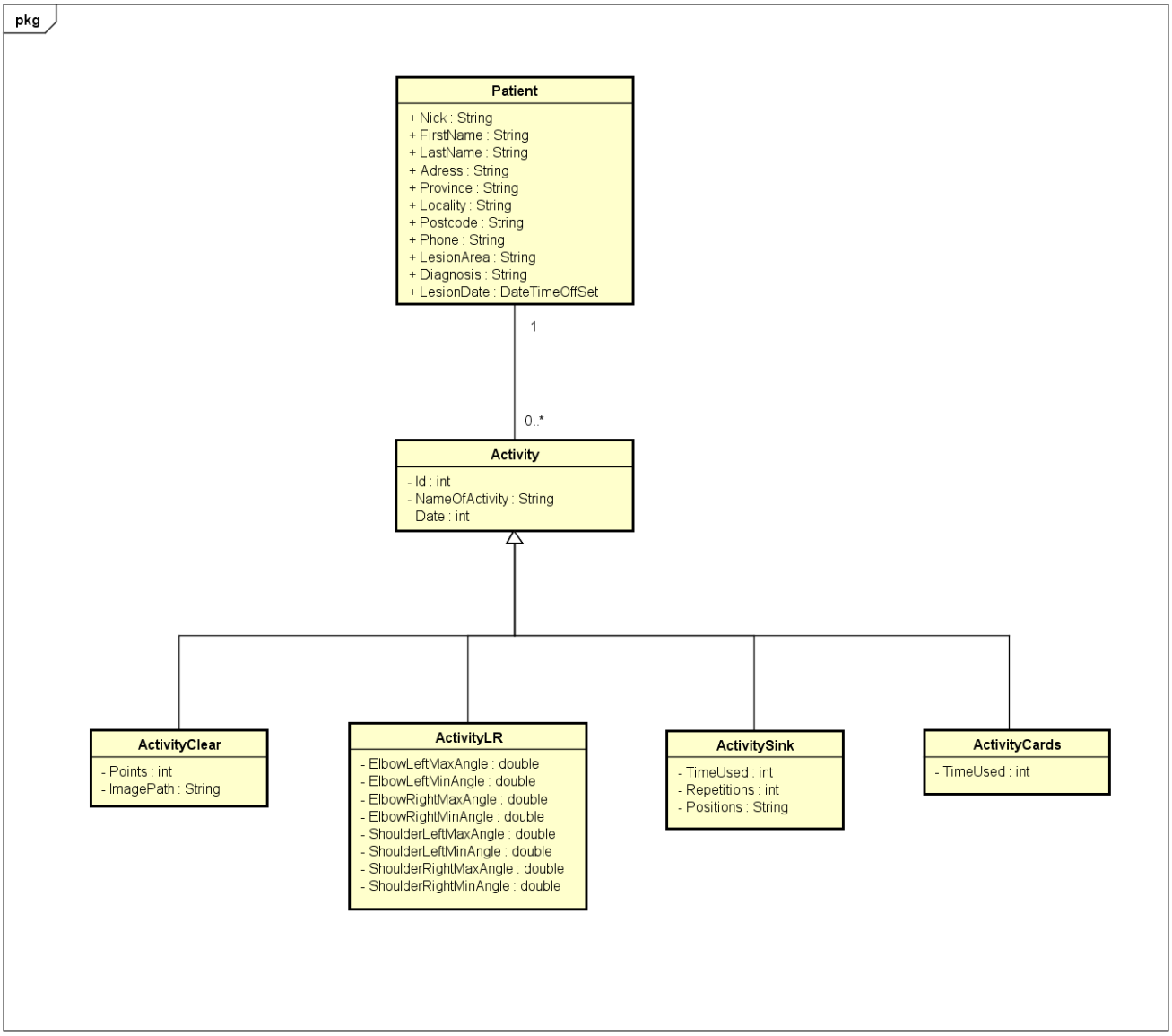

Figura 18: *Diagrama de modelo de dominio*

4.4. Diagrama de clases de diseño

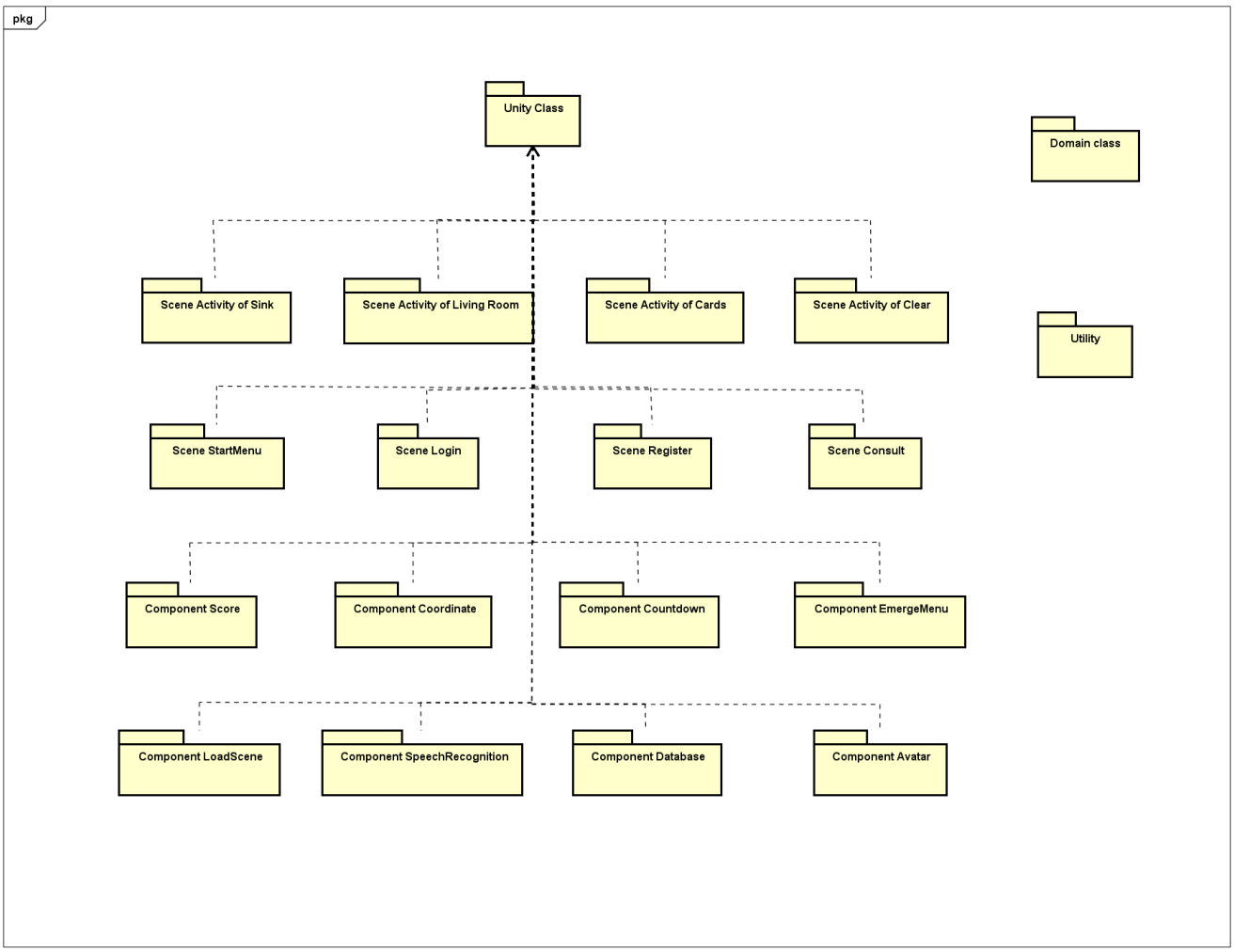

Figura 19: *Diagrama de clases de diseño*

4.4.1. Clases de motor de videojuegos Unity:

Clases pertenecientes al motor de videojuegos Unity. Las funcionalidades de estas clases son detalladas en el apéndice A.

4.4.2. Clases de la aplicación:

Clases propias de la aplicación, la mayoría de ellos son clases descendientes de la clase MonoBehavior. A continuación, una breve descripción de estas clases:

Controladores de escena de actividad de Salón:

- **ActivityLRController:** clase controlador de la escena de actividad de Salón. Se encarga de gestionar el progreso de la actividad de Salón. Por ejemplo, mostrar las instrucciones correspondientes en cada momento de la escena para que el paciente puede finalizar correctamente la actividad. Se encarga también de conectar con la base de datos para guardar los resultados del ejercicio.

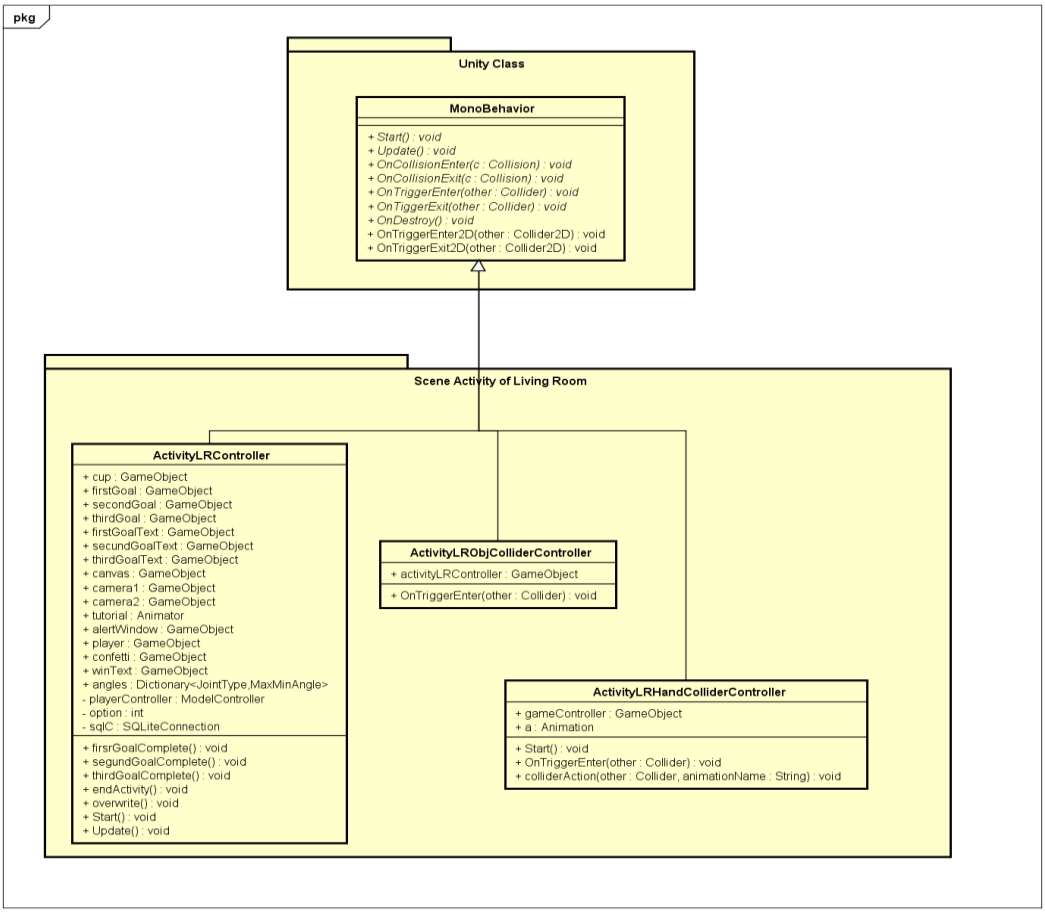

Figura 20: Clases de la escena de actividad de salón

- **ActivityLRObjColliderController:** esta clase se encarga de gestionar las colisiones entre los objetos de la escena.

- **ActivityLRHandColliderController:** la función principal de esta clase es gestionar las colisiones de las manos del avatar con los objetos de la escena para que puedan activar las animaciones apropiadas en cada situación.

Controladores de escena de actividad de Cartas:

- **ActivityCardsController:** esta clase muestra las indicaciones de cada paso al usuario y conecta con la base de datos para guarda informaciones relevantes de la actividad.

- **ActivityCardsHandColliderController:** clase controlador que lleva a cabo la gestión de las colisiones entre la mano y los objetos de la escena. Su función principal es seleccionar las cartas de la escena.

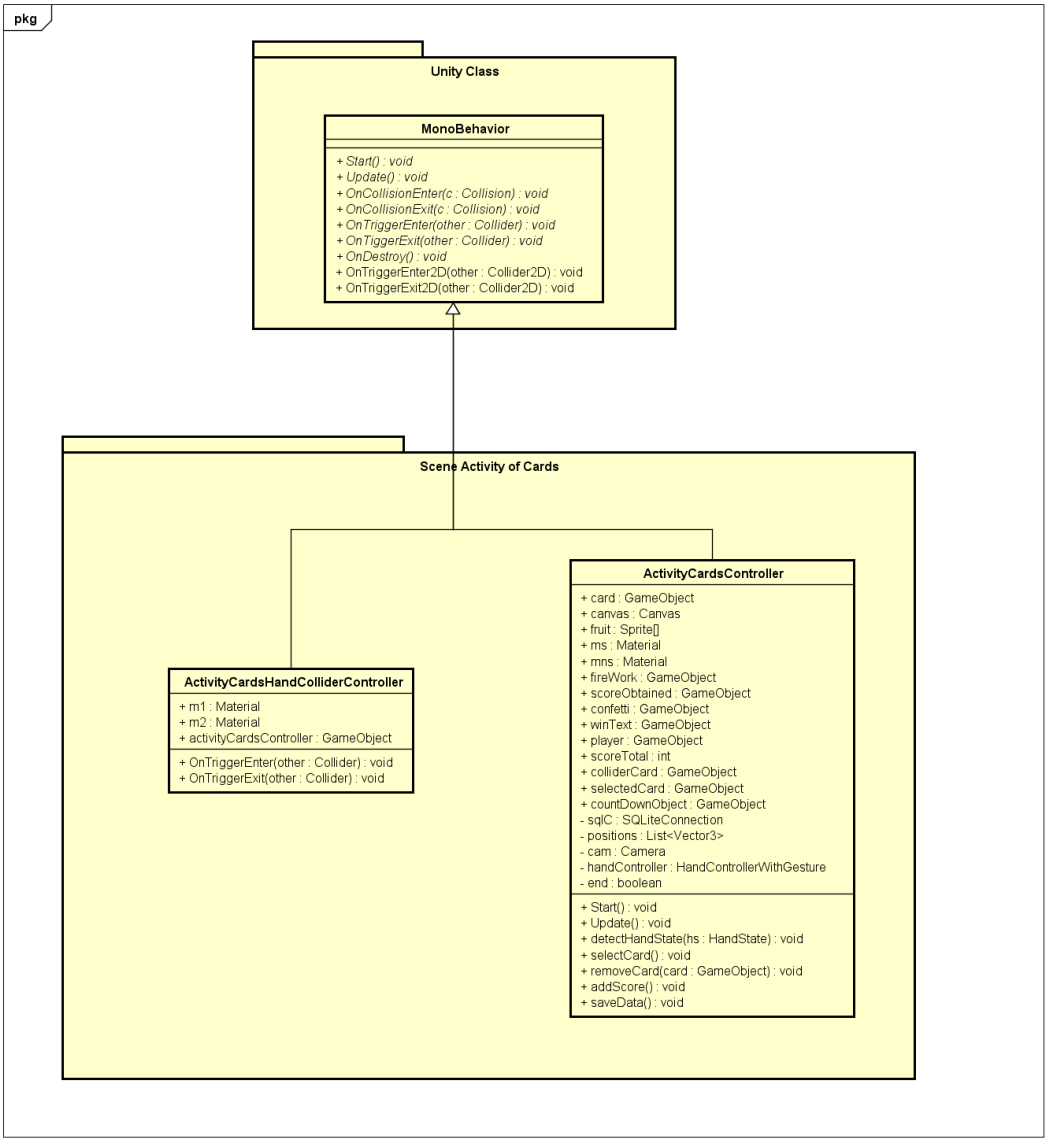

Figura 21: *Clases de la escena de actividad de Cartas*

Controladores de escena de actividad de Despejar:

- **ActivityClearController:** las tareas que desempeña esta clase son las siguientes: mostrar instrucciones al usuario, conectar con la base de datos, gestionar la matriz de cubos y mantener la trayectoria del jugador para poder guardar la trayectoria en base de datos como una imagen a finalizar la escena.

- **ActivityClearCubController:** esta clase se encarga principalmente de detectar las colisiones entre los cubos y el jugador.

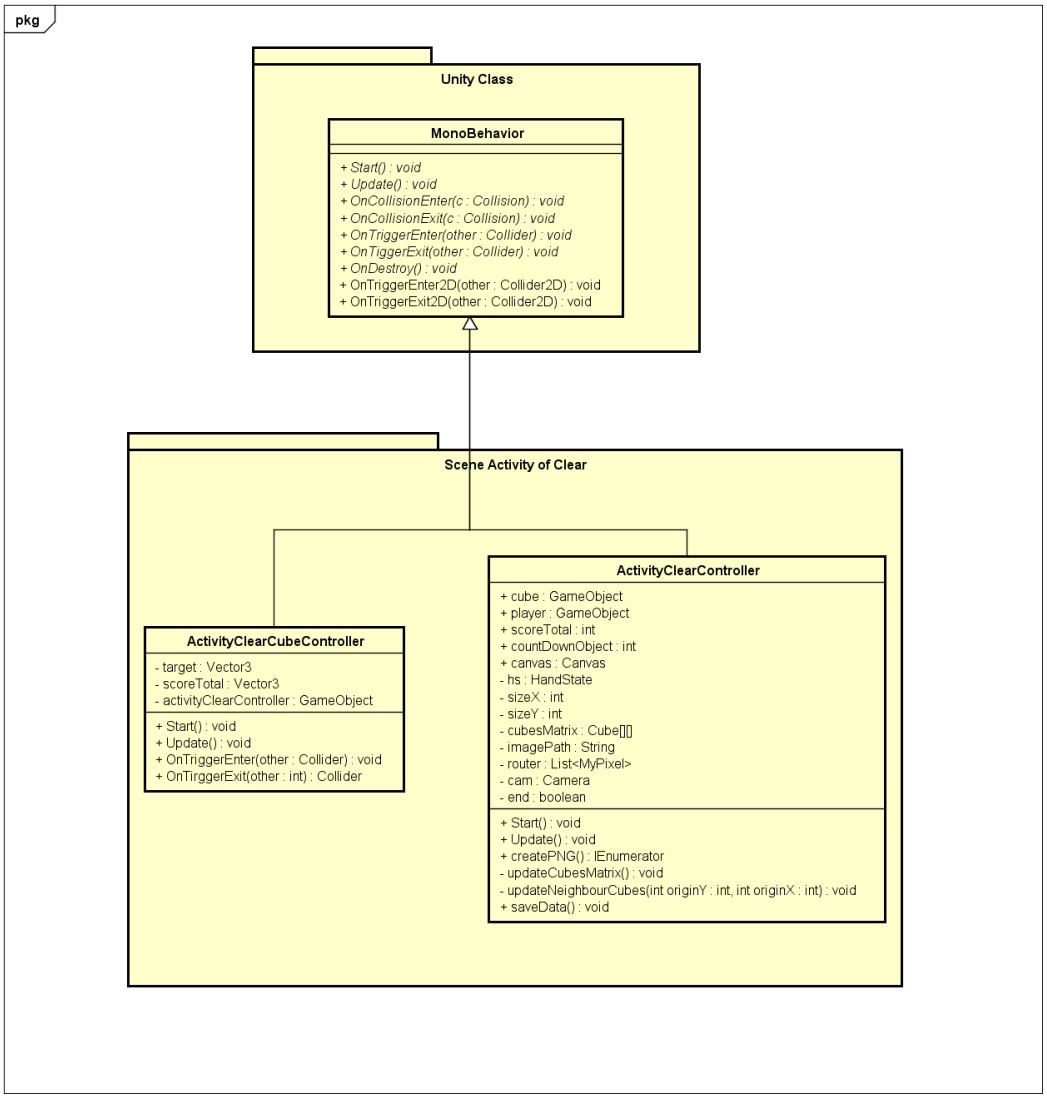

Figura 22: *Clases de la escena de actividad de Despejar*

Controladores de escena de actividad de Lavabo:

- **ActivitySinkController:** clase controlador de la escena de actividad de Lavabo. Se encarga de mostrar instrucciones al usuario, conectar con la base de datos y llevar la cuenta atrás de las repeticiones del ejercicio.

- **ActivitySinkHandColliderController:** esta clase desempeña la misma función que la clase “ActivityLRHandColliderController” de la escena de la actividad de Salón, es decir, detectar las colisiones entre las manos de avatar y otros objetos de la escena y activar las animaciones.

- **ActivitySinkObjColliderController:** esta clase sirve para detectar las colisiones entre los objetos de la escena, como la pasta y el cepillo.

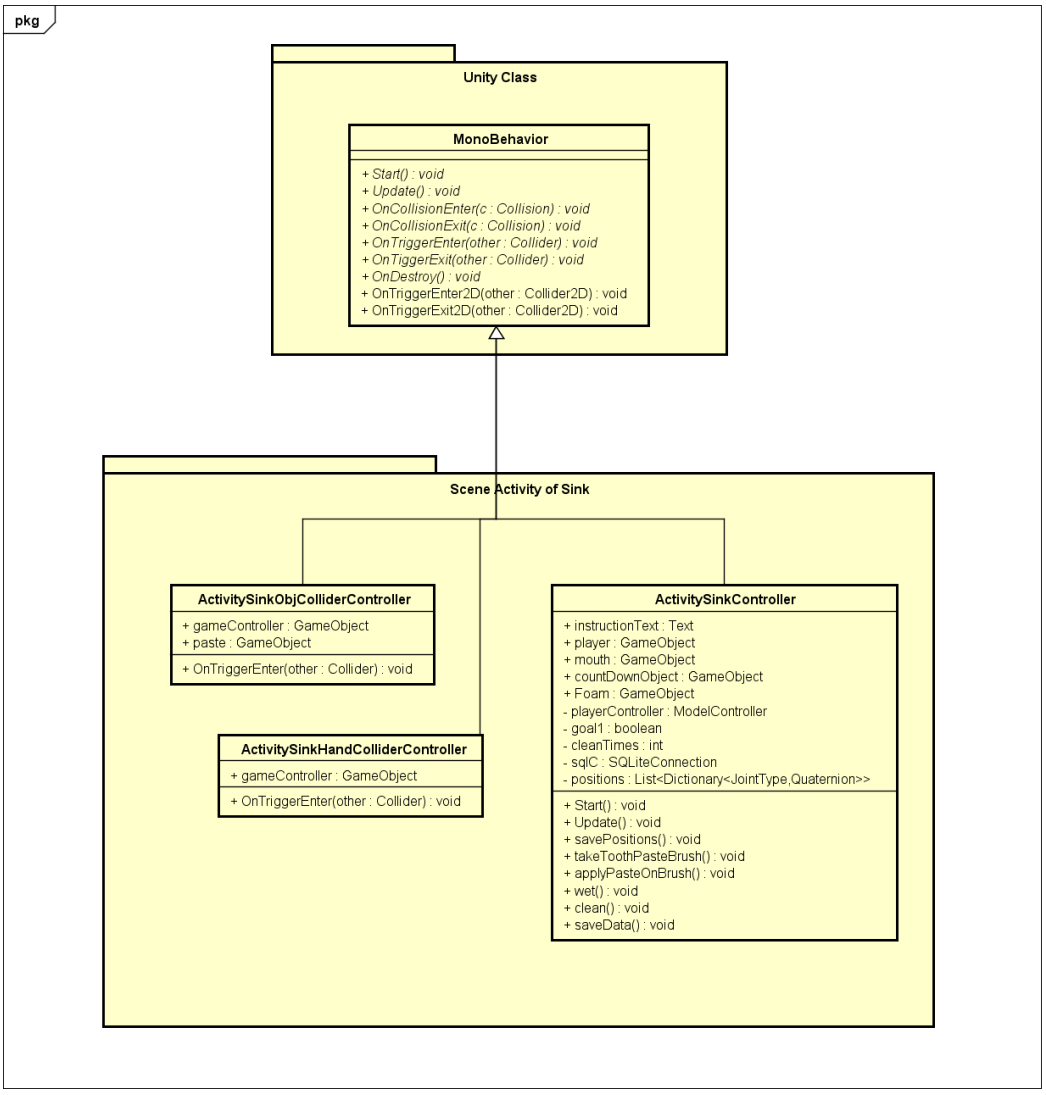

Figura 23: *Clases de la escena de actividad de Lavabo*

Controladores de la escena de consultar:

- **ActivitySelect:** clase controlador de la escena de consulta. Se encarga de detectar la actividad seleccionada por el usuario y mostrar los detalles de la actividad seleccionada.

- **ConsultController:** esta clase se encarga de conectar con la base de datos y mostrar los datos personales del paciente y la lista de las actividades realizadas.

- **ChangePatientInfo:** cuando el usuario haya modificado los datos personales del paciente en el formulario, esta clase se encarga de establecer conexión con la base de datos para aplicar estas modificaciones.

- **DeletePatient:** la única función de esta clase es eliminar todos los datos relacionados con el paciente de la base de datos.

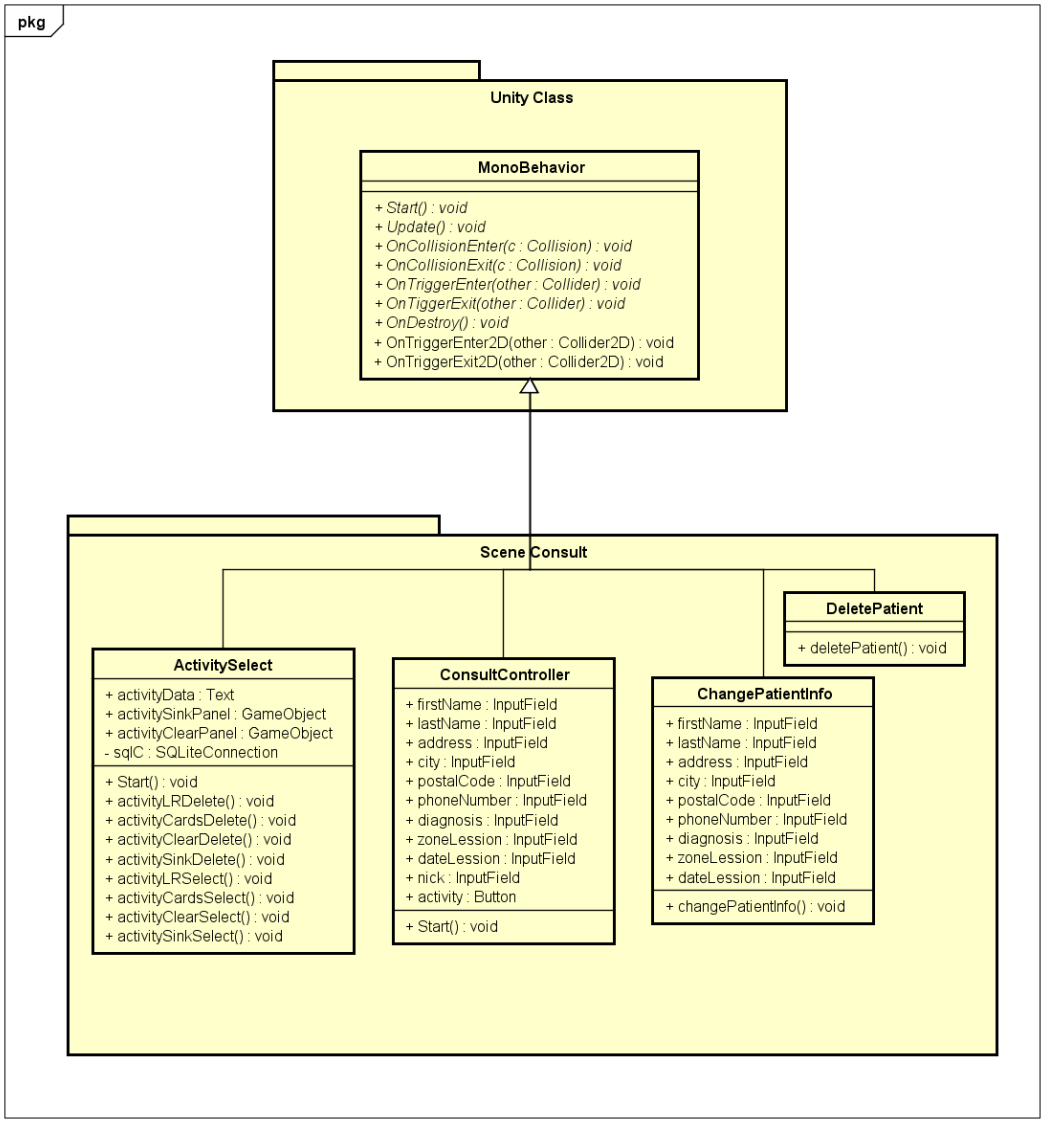

Figura 24: *Clases de la escena de Consulta*

Controlador de la escena de Login:

- **LoginController:** en el momento que el usuario intenta iniciar la sesión este controlador conecta con la base de datos para realiza la comprobación y dependiendo de resultado de la comprobación iniciará la sesión o lo rechazará.

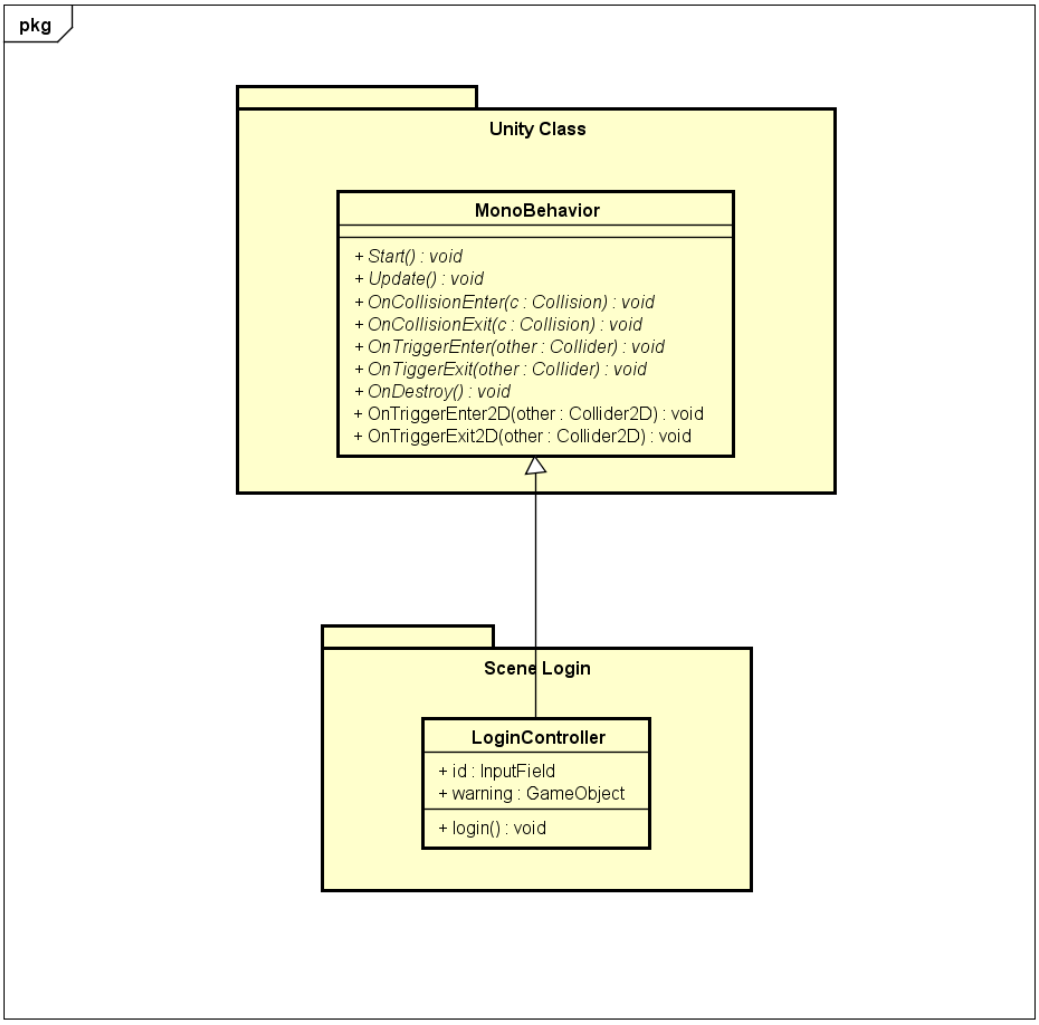

Figura 25: *Clases de la escena de Login*

Controlador de la escena de menú del inicio:

- **MenuController:** la funcionalidad primaria de esta clase es controlar el flujo de las listas de opciones que se presentan en la escena en cada momento.

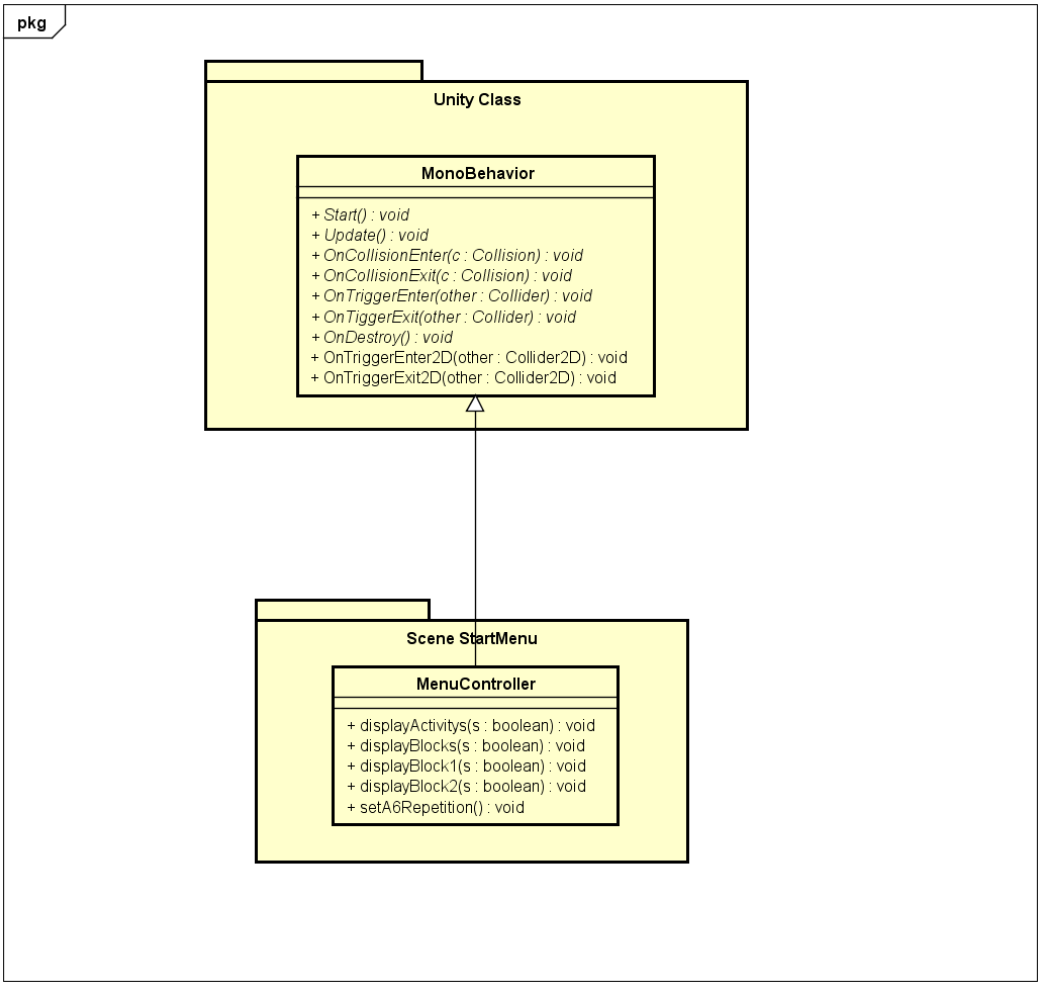

Figura 26: *Clases de la escena de menú del inicio*

Controlador de la escena de registrar:

- **RegisterController:** esta clase se encarga de obtener todos los datos introducidos por el usuario en el formulario, comprobar sus valideces y guardarlos en la base de datos.

- **DayAvailableController & YearAvailableController:** clases controladores de la escena de registrar, cuya funcionalidad principal es mostrar las fechas validas al usuario.

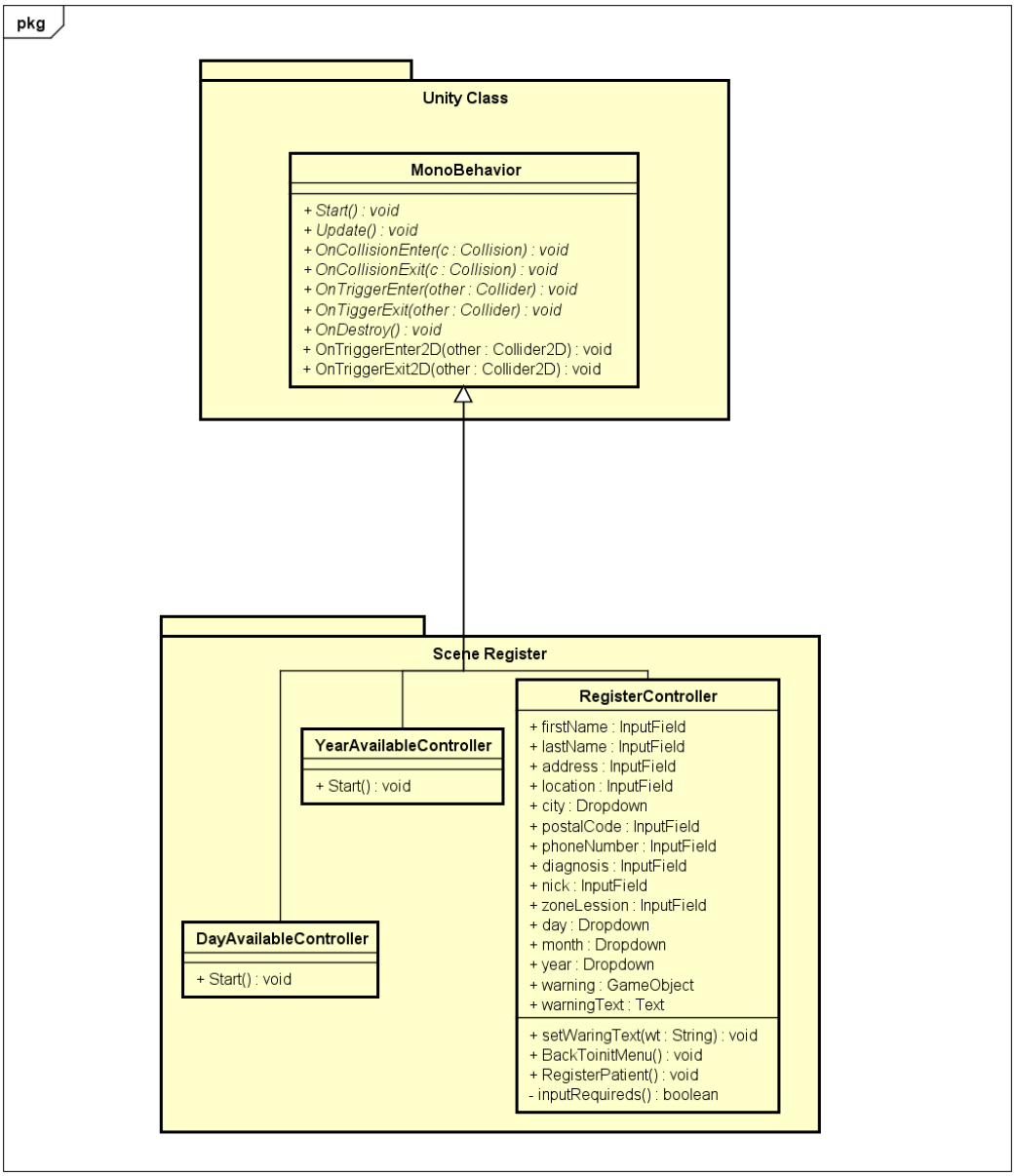

Figura 27: *Clases de la escena de registrar*

Controlador del componente creador de base de datos:

- **DBCreater:** el propósito de esta clase es comprobar la existencia de la base de datos en el inicio de la aplicación y, en caso de que no exista, se encarga de crear una.

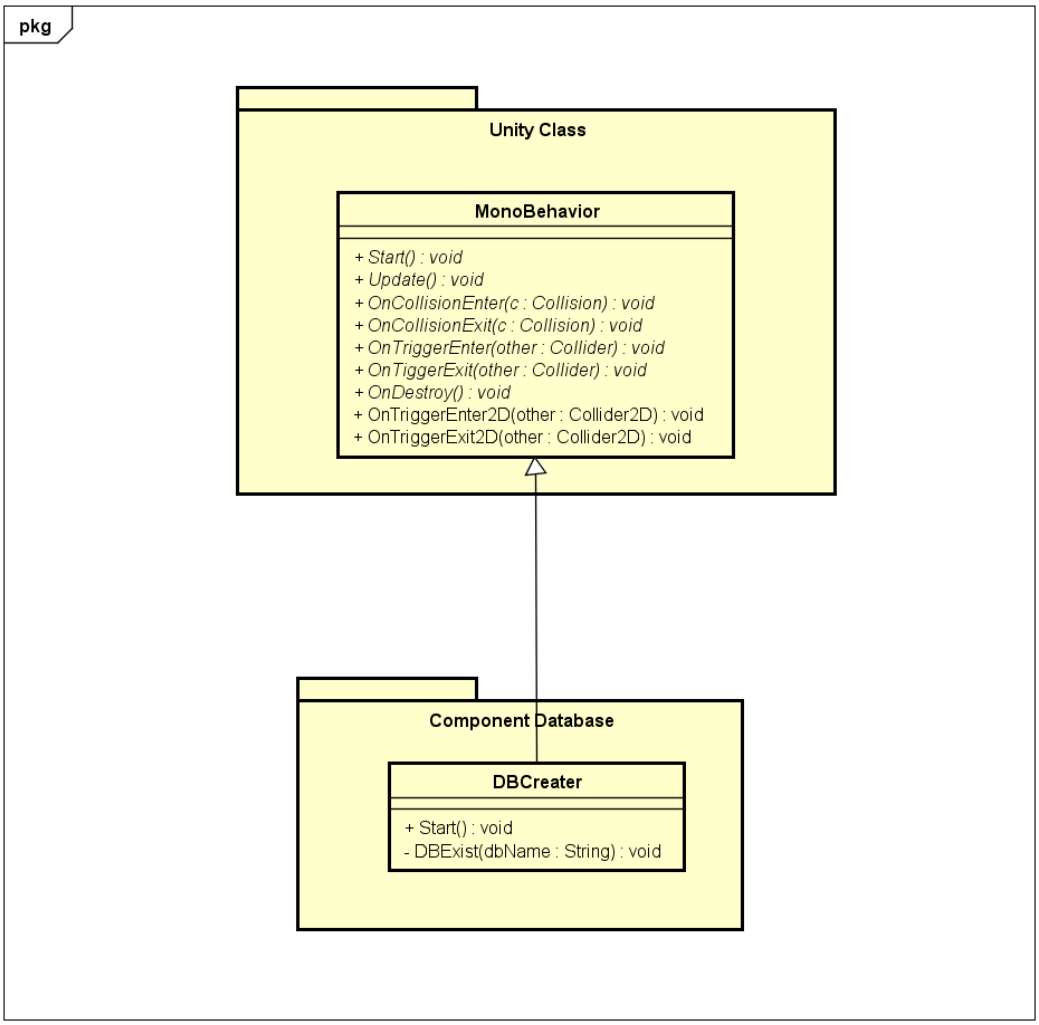

Figura 28: *Clases del componente creador de base de datos*

Controlador del componente reconocimiento por voz:

- **SpeedchRecognition:** esta clase sirve para capturar los comandos de voz del usuario.

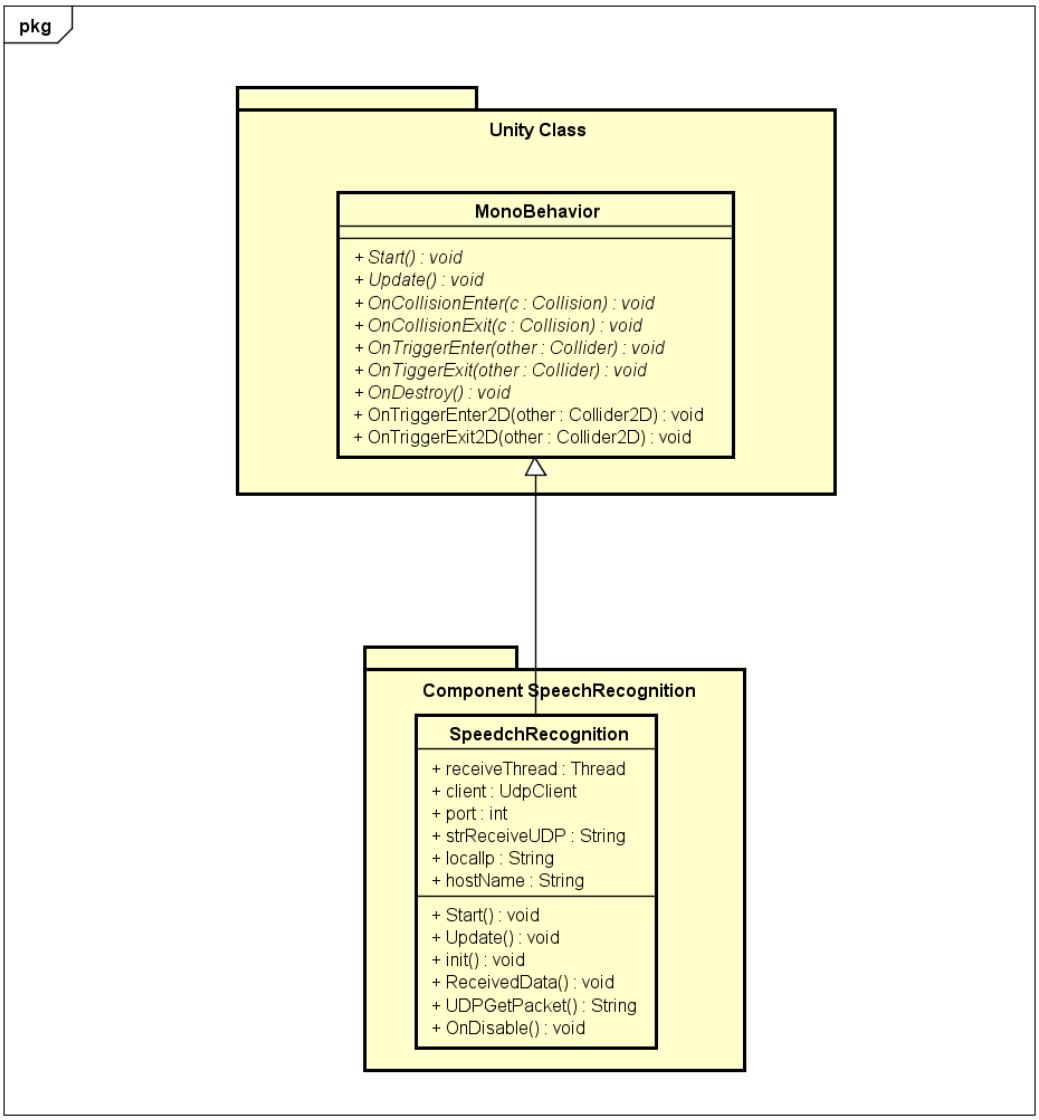

Figura 29: *Clases del componente reconocimiento por voz*

Controlador del componente cuenta atrás:

- **CountDownController:** esta clase se encarga de llevar las cuentas de tiempo empleado en las escenas y mostrarlas al usuario.

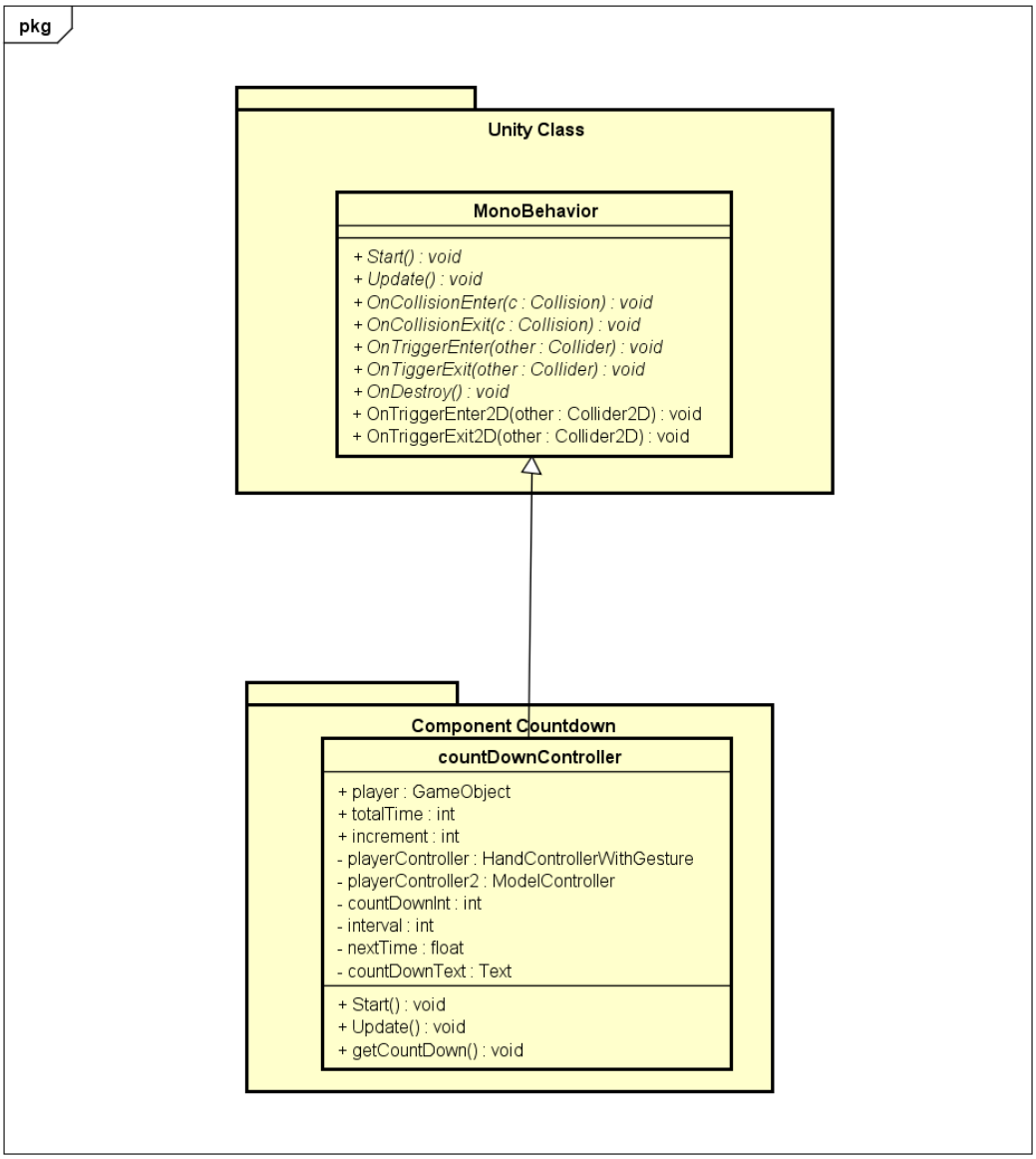

Figura 30: *Clases del componente cuenta atrás*

Controlador del componente puntuación:

- **ScoreTotalController & ScoreObtainedController:** estas clases se encargan de detectar la obtención de la puntuación y controlar las animaciones del objeto “puntuación total” de la escena.

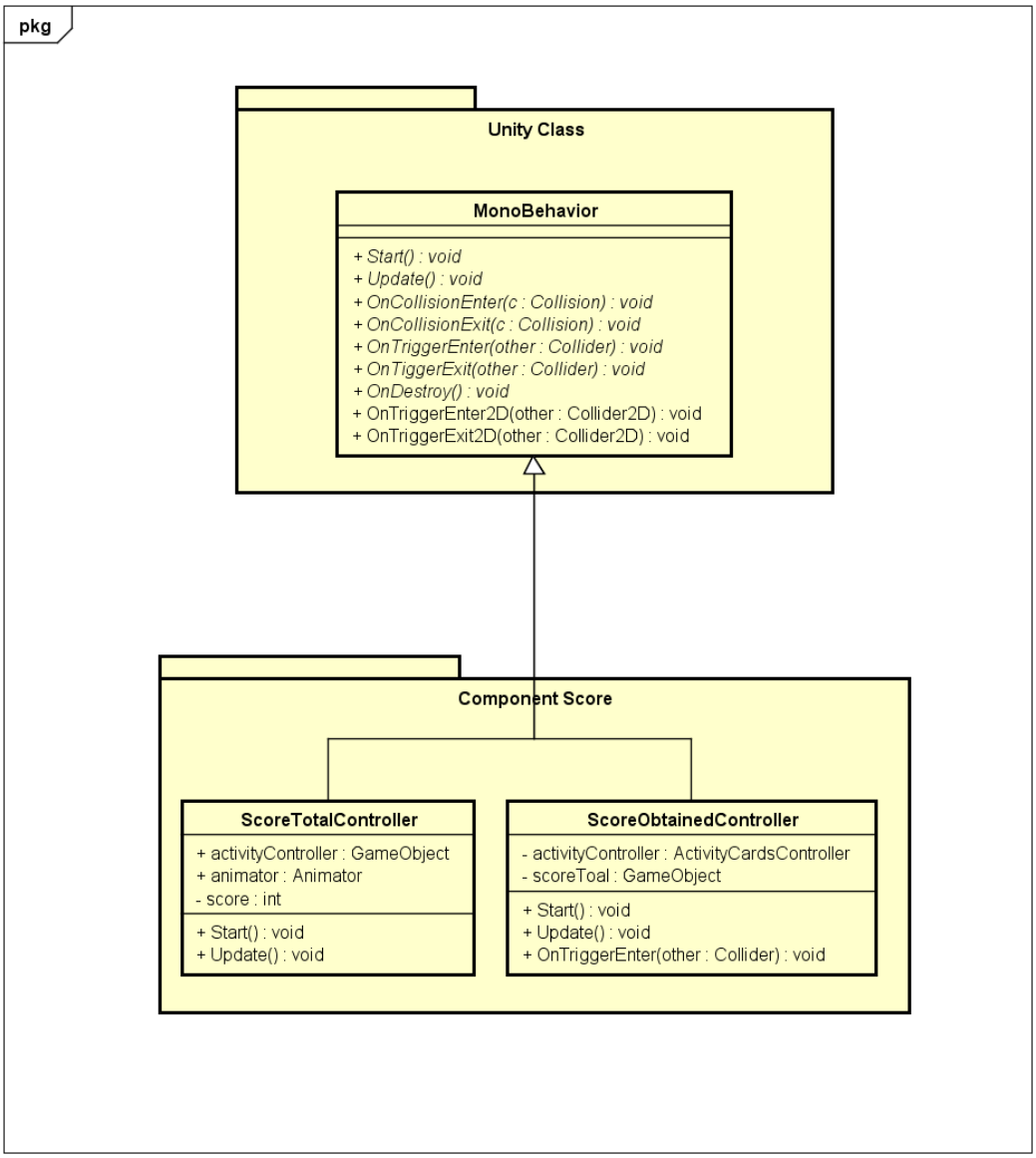

Figura 31: *Clases del componente puntuación*

Controlador del componente coordinación:

- **HumanIconController:** la función de esta clase es comunicar con el sensor Kinect y obtener datos de ella para controlar el avatar, en este caso solo la coordenada del "SpineBase".

- **Coordinate & HumanIconColliderController:** la malla controlada por "HumanIconColliderController" al colisionar con la malla de posición objetiva invoca la función de "Coordinate" que se encarga de iniciar el proceso de la actividad principal.

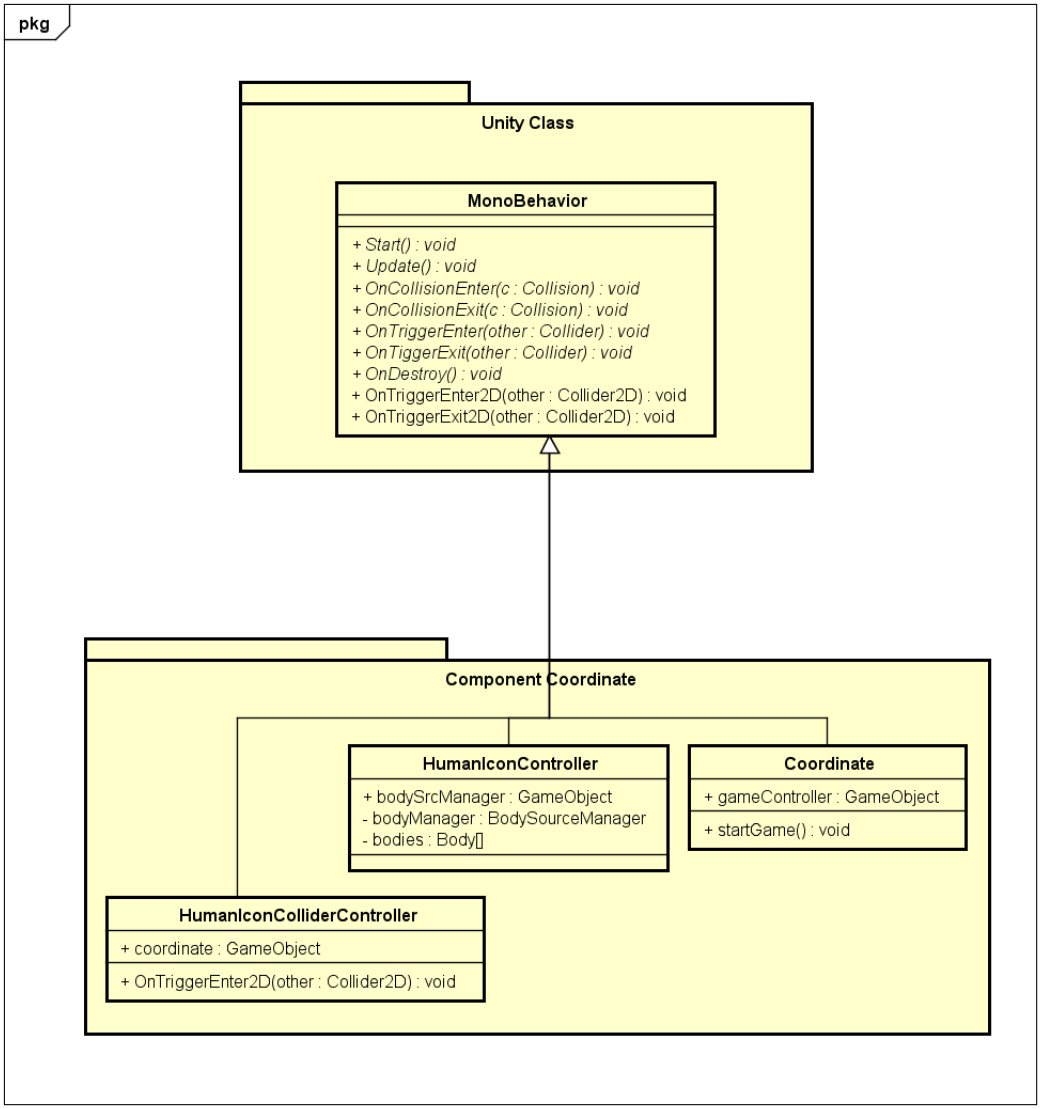

Figura 32: *Clases del componente coordinación*

Controlador del componente cargar escena:

- **LoadSceneController:** como su nombre indica la funcionalidad primaria de esta clase es cargar escenas.

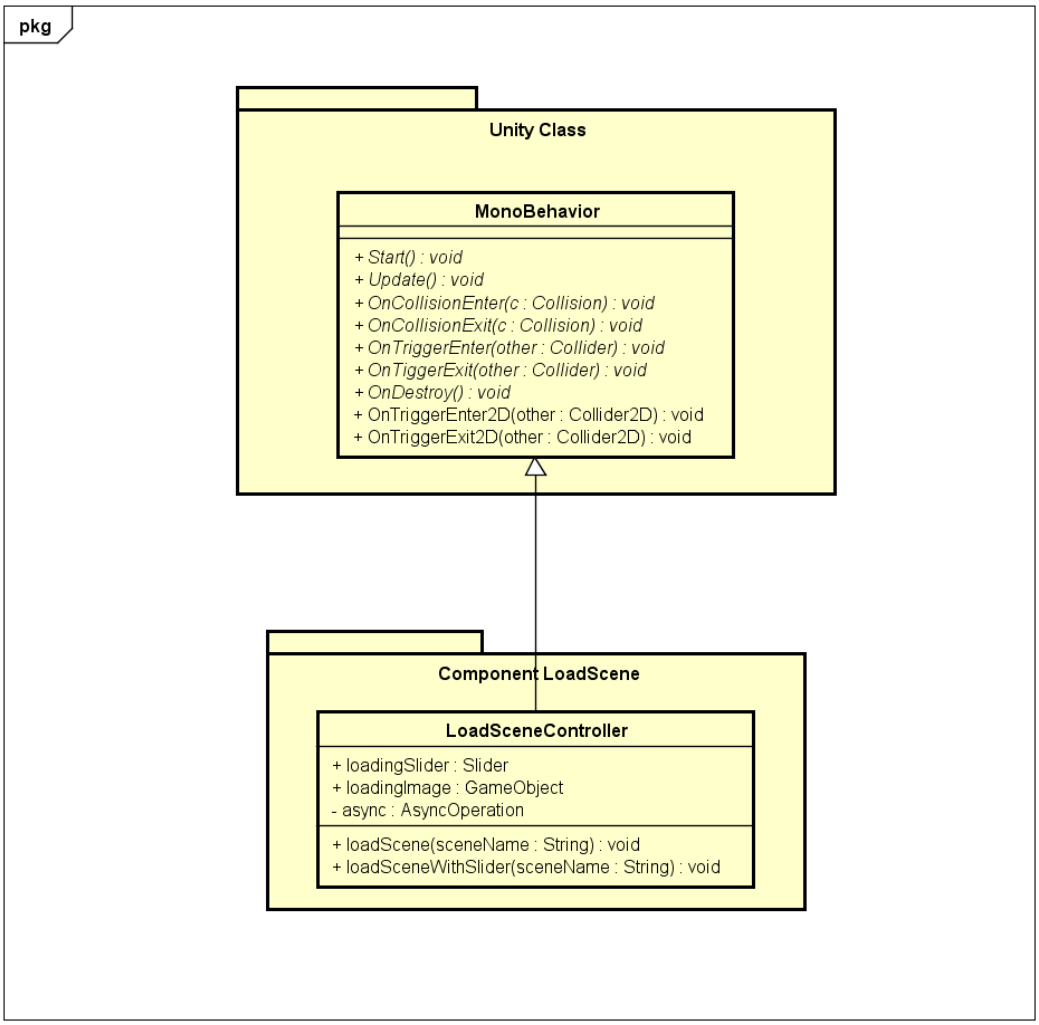

Figura 33: *Clases del componente cargar escena*

Controlador del componente avatar:

- **ModelController:** esta clase se encarga de controlar el avatar con forma humanoide. Para ello establece conexión con el sensor Kinect para obtener los datos de las rotaciones de los huesos que necesitan ser controlados. Pero debido a la inestabilidad y la falta de fiabilidad en algunos de los huesos como los dedos de las manos se precisa aplicar un filtro a estos datos obtenidos.

- **HandControllerWithGesture:** gracias a que solo hace falta obtener las posiciones de las manos y no las de los dedos para este controlador no es necesario aplicar ningún tipo de filtro, ya que estos datos son suficientemente estables. En cuando a la función de esta clase al igual que la clase “ModelController” sirve para controlar el avatar del jugador, aunque esta vez un avatar con forma de la mano.

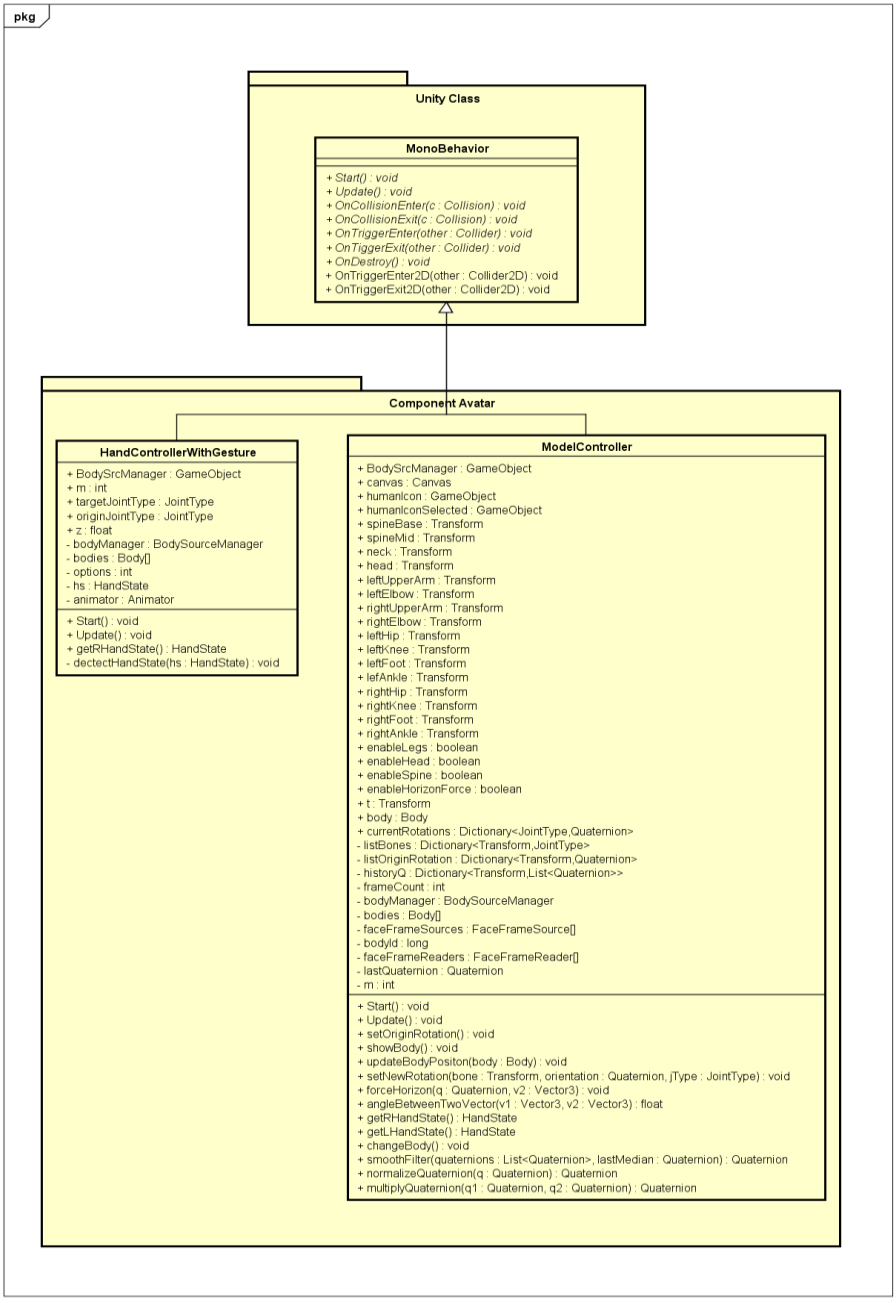

Figura 34: *Clases del componente avatar*

Controlador del componente menú de emergencia:

- **EmergeMenuController:** El proposito de esta clase es mostrar, ocultar y gestionar el menú de emergencia.

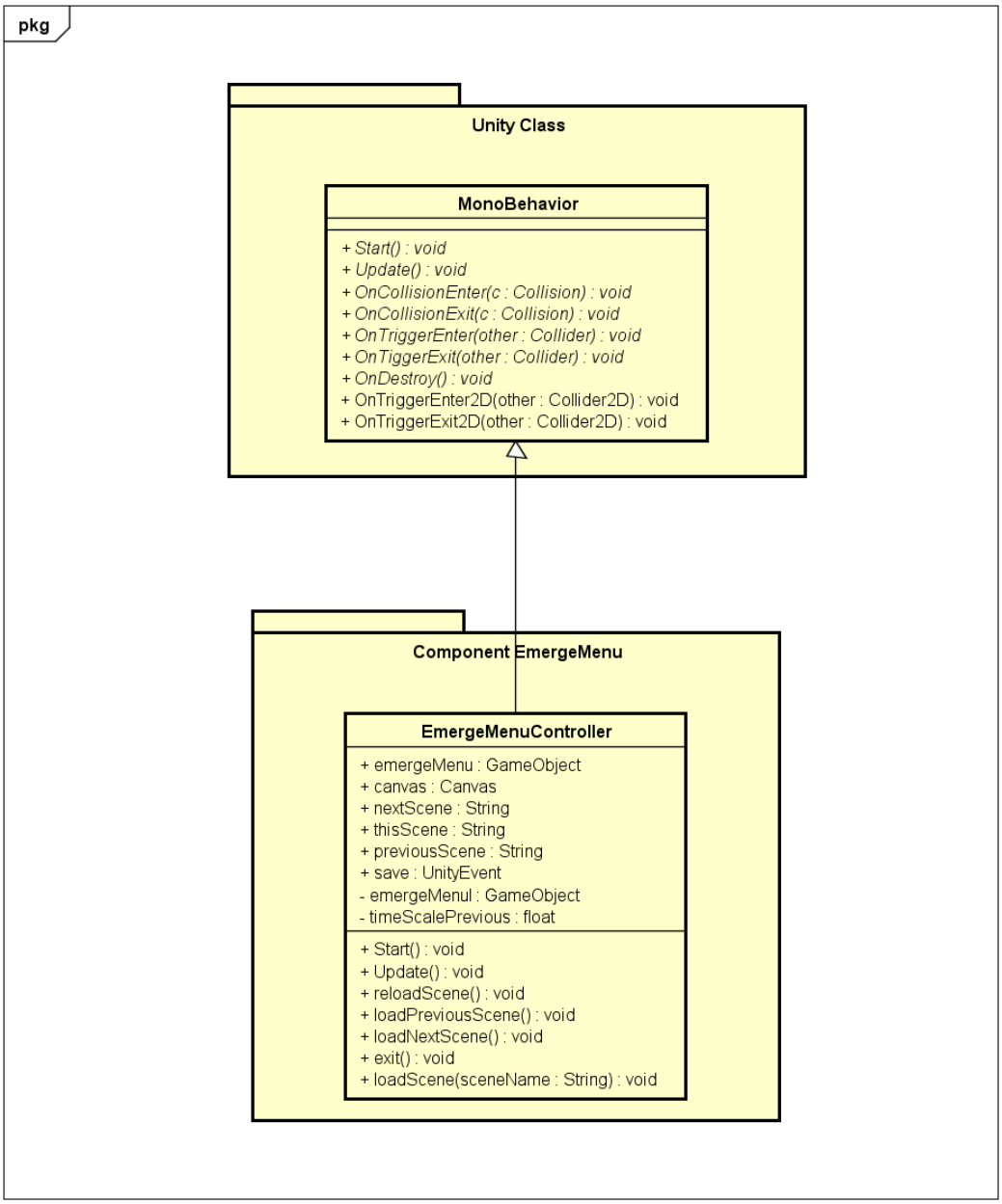

Figura 35: *Clases del componente menú de emergencia*

Clases auxiliares:

- **MaxMinAngle:** esta clase se encarga de almacenar el ángulo máximo y mínimo obtenido de una articulación del cuerpo.

- **Cube:** esta clase solo aparece en la escena de actividad de Despejar, que sirve para almacenar las informaciones relativas a un cubo.

- **GameState:** esta clase sirve como conector entre las diferentes escenas para poder pasar las informaciones entre ellas.

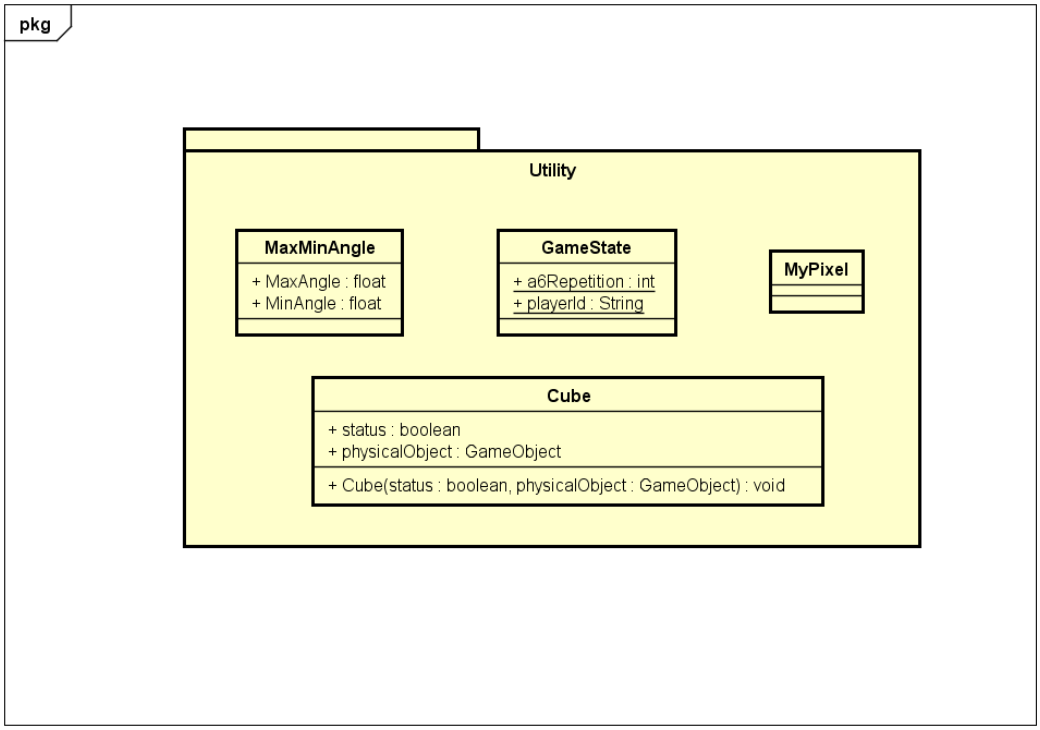

Figura 36: *Clases auxiliares*

4.5. Diagramas de secuencia de diseño

Registrar

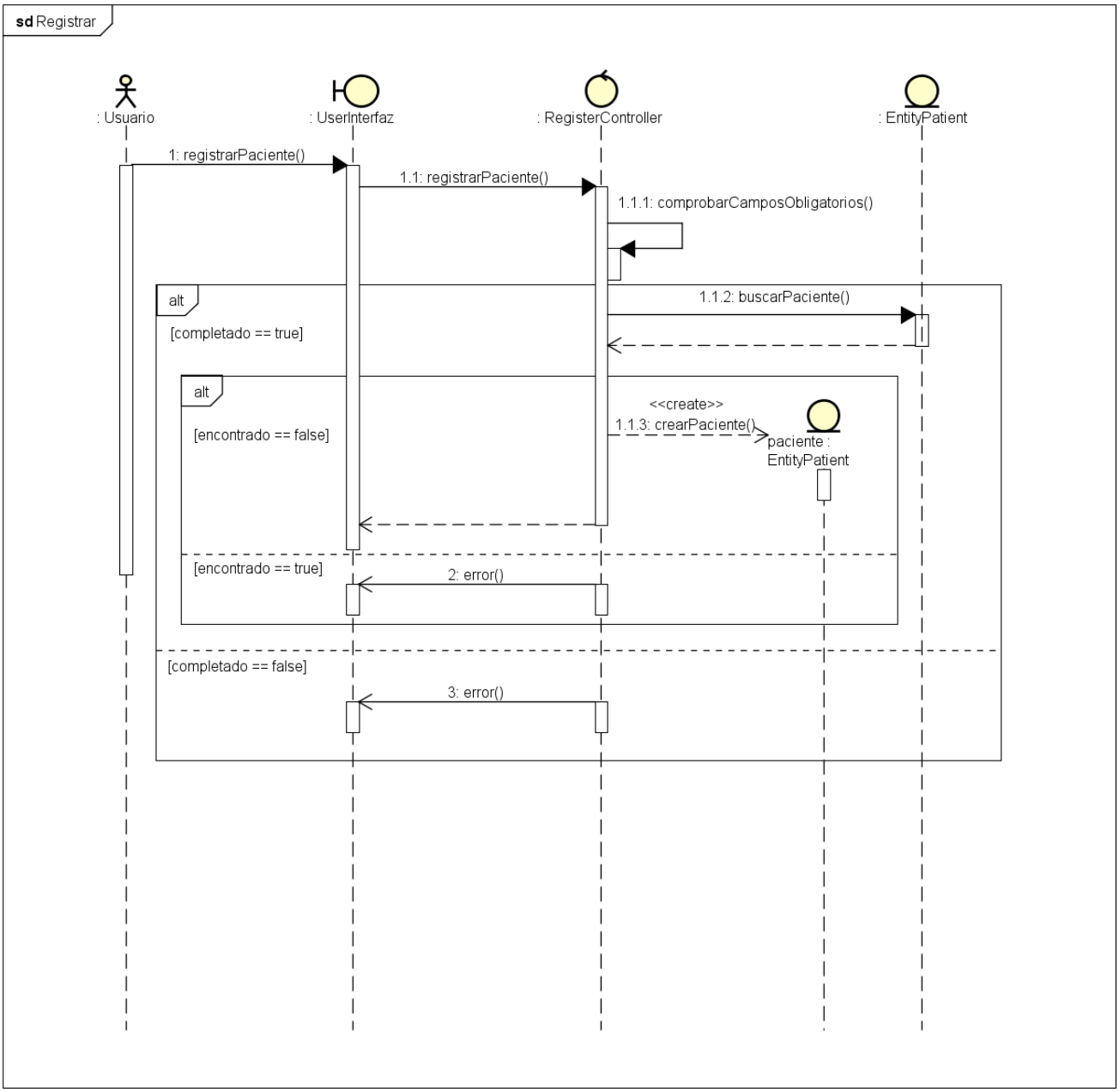

Figura 37: *DS – Registrar*

Identificar

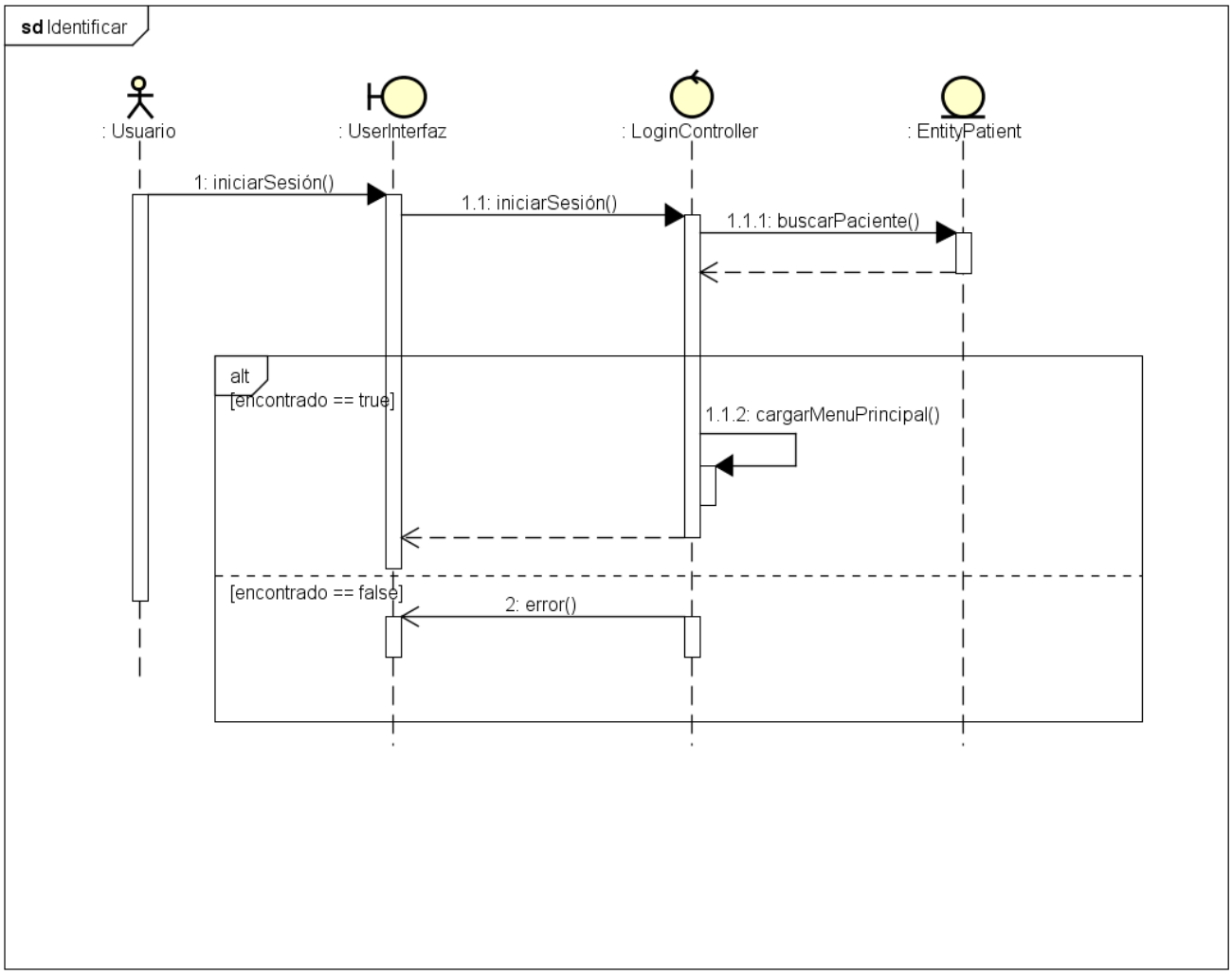

Figura 38: *DS – Identificar*

Consultar datos de paciente

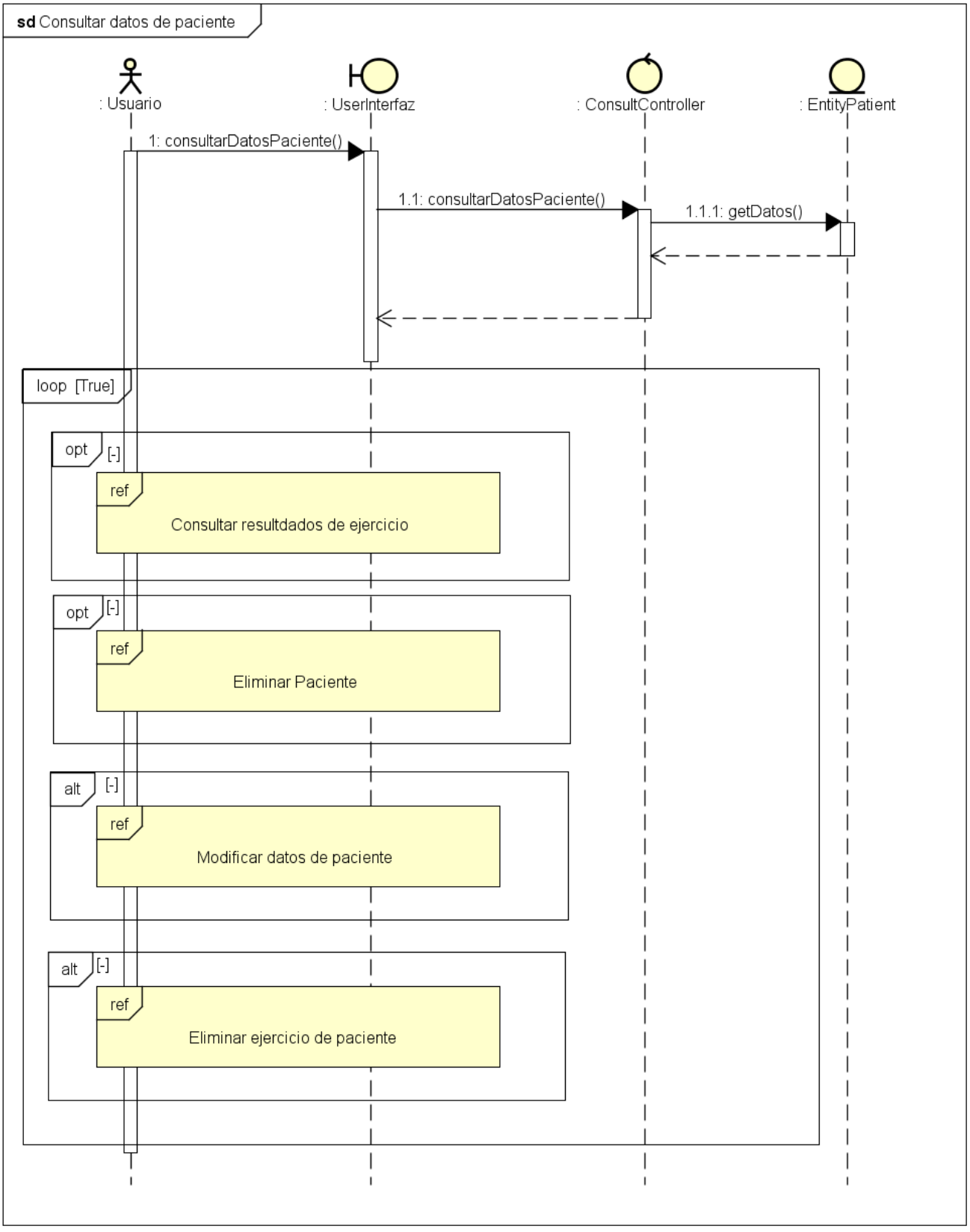

Figura 39: *DS – Consultar datos de paciente*

Consultar resultado de ejercicio

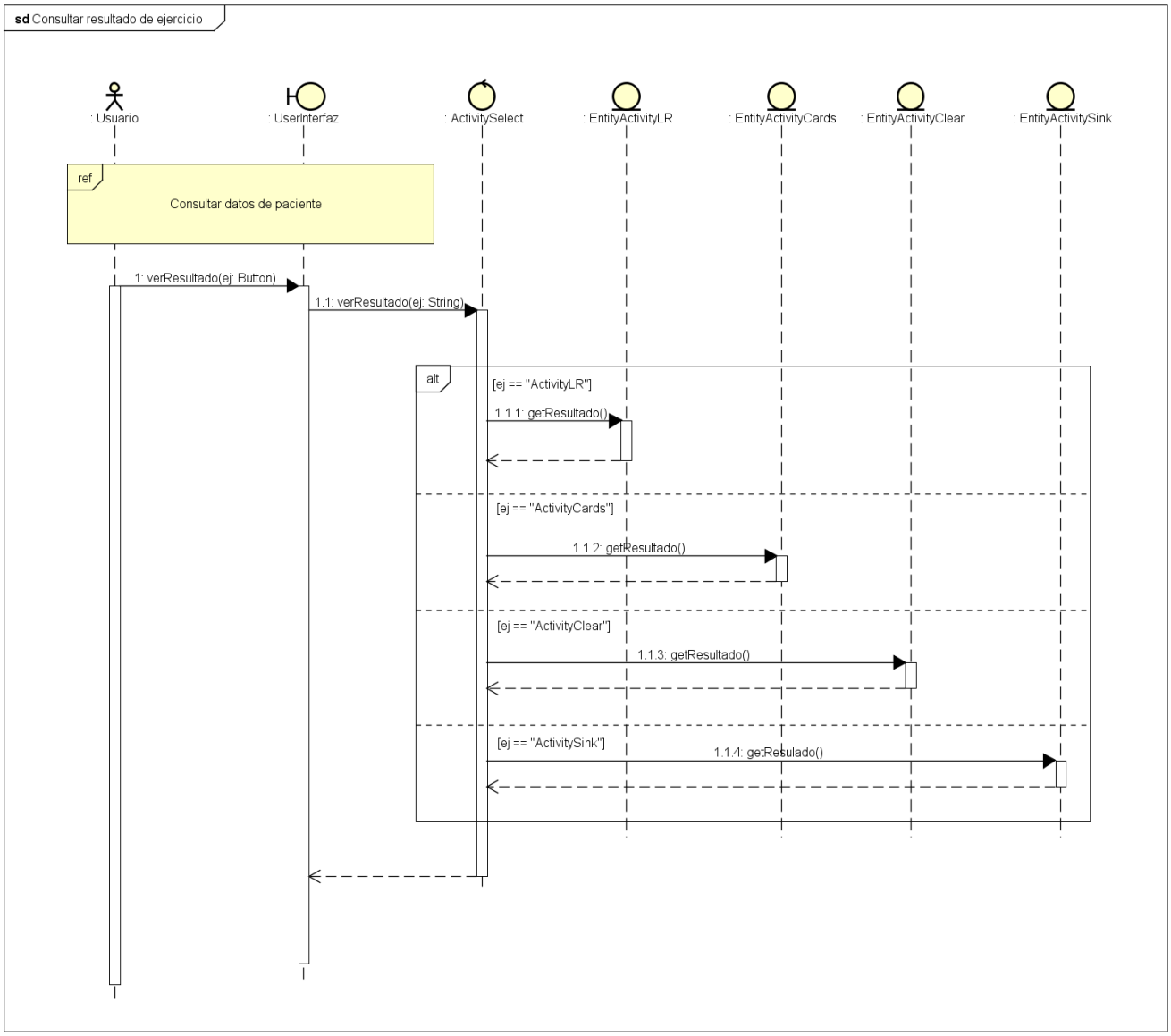

Figura 40: *DS – Consultar resultado de ejercicio*

Eliminar paciente

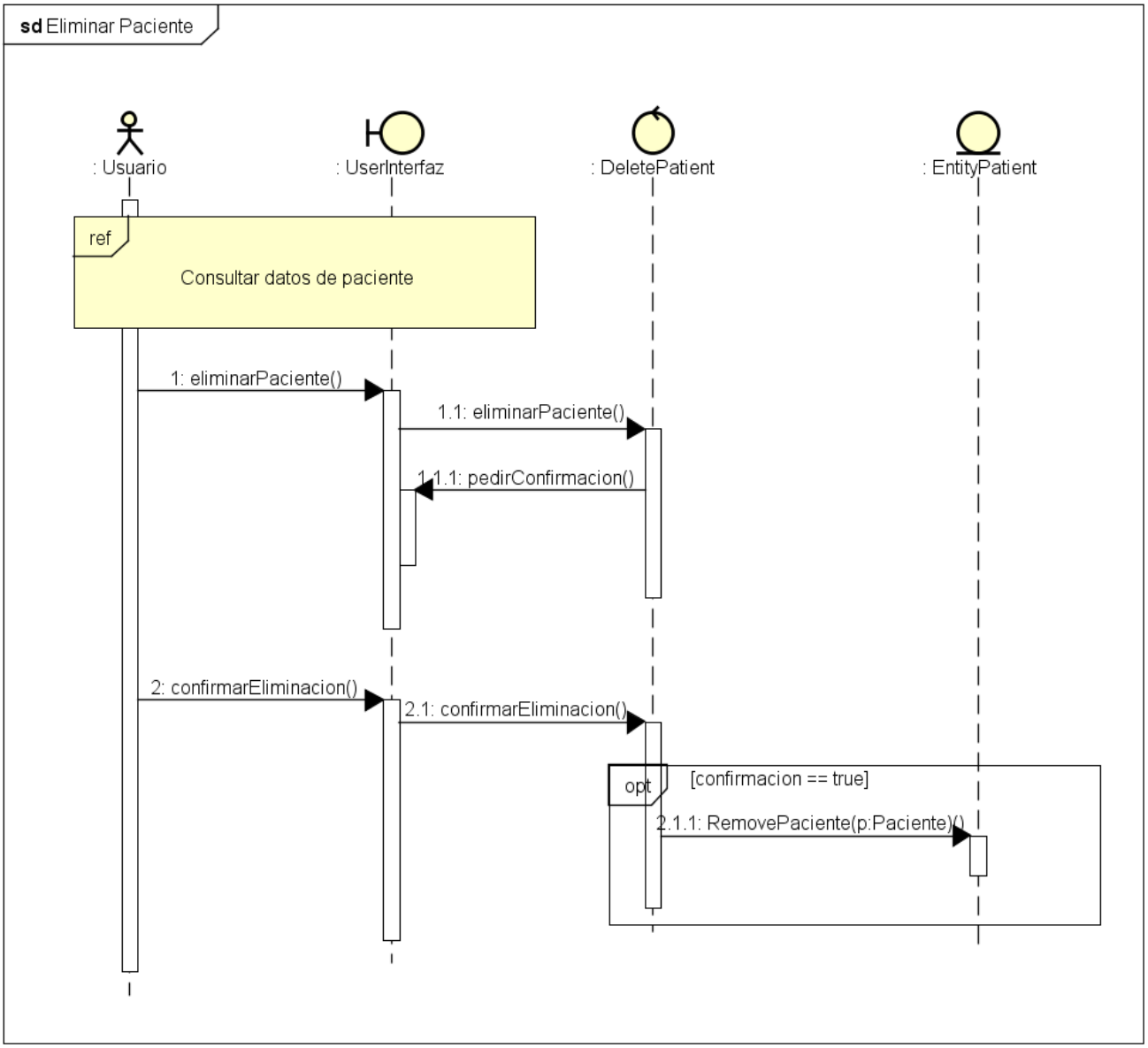

Figura 41: *DS – Eliminar paciente*

Modificar datos de paciente

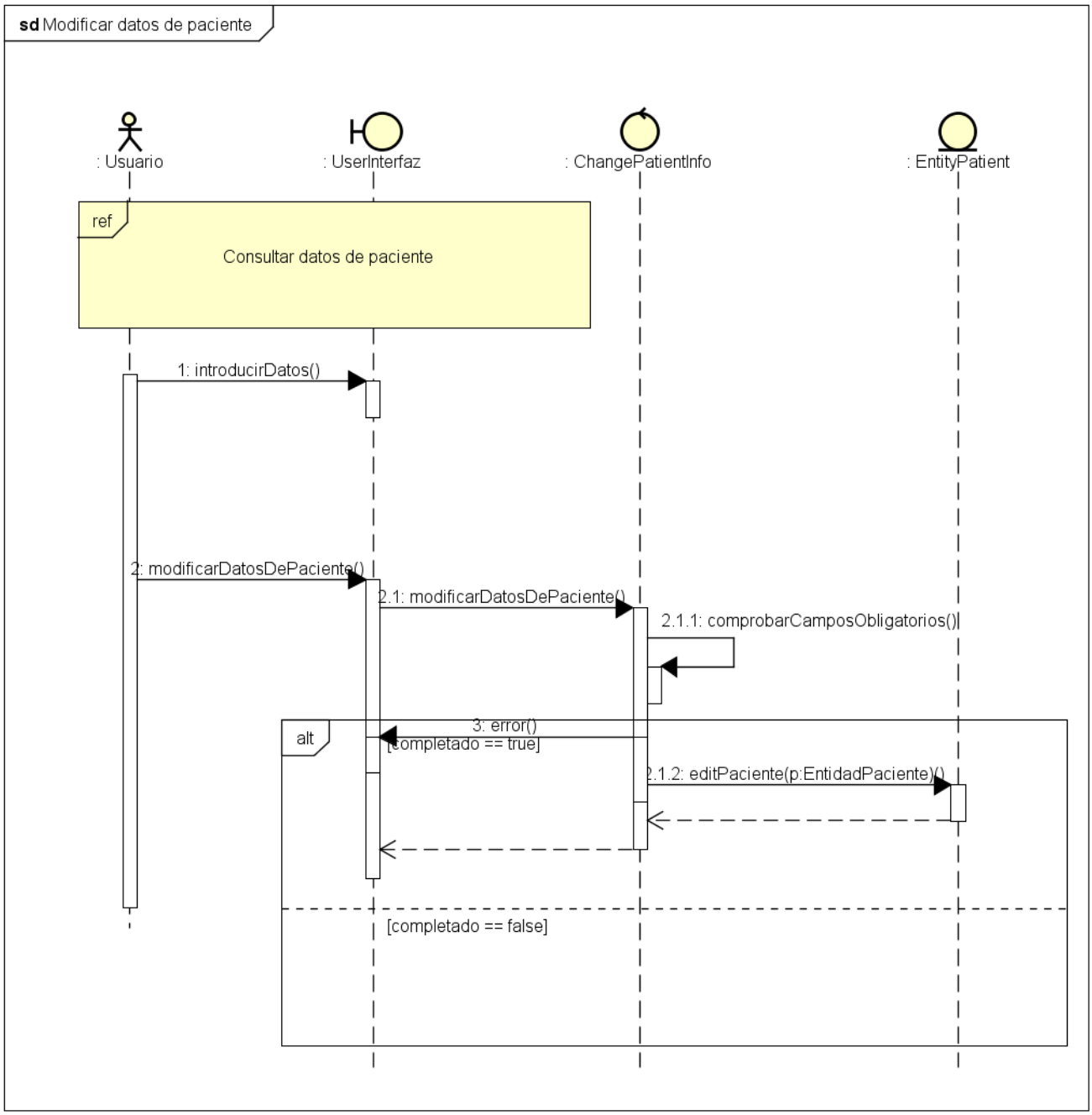

Figura 42: *DS – Modificar datos de paciente*

Eliminar ejercicio de paciente

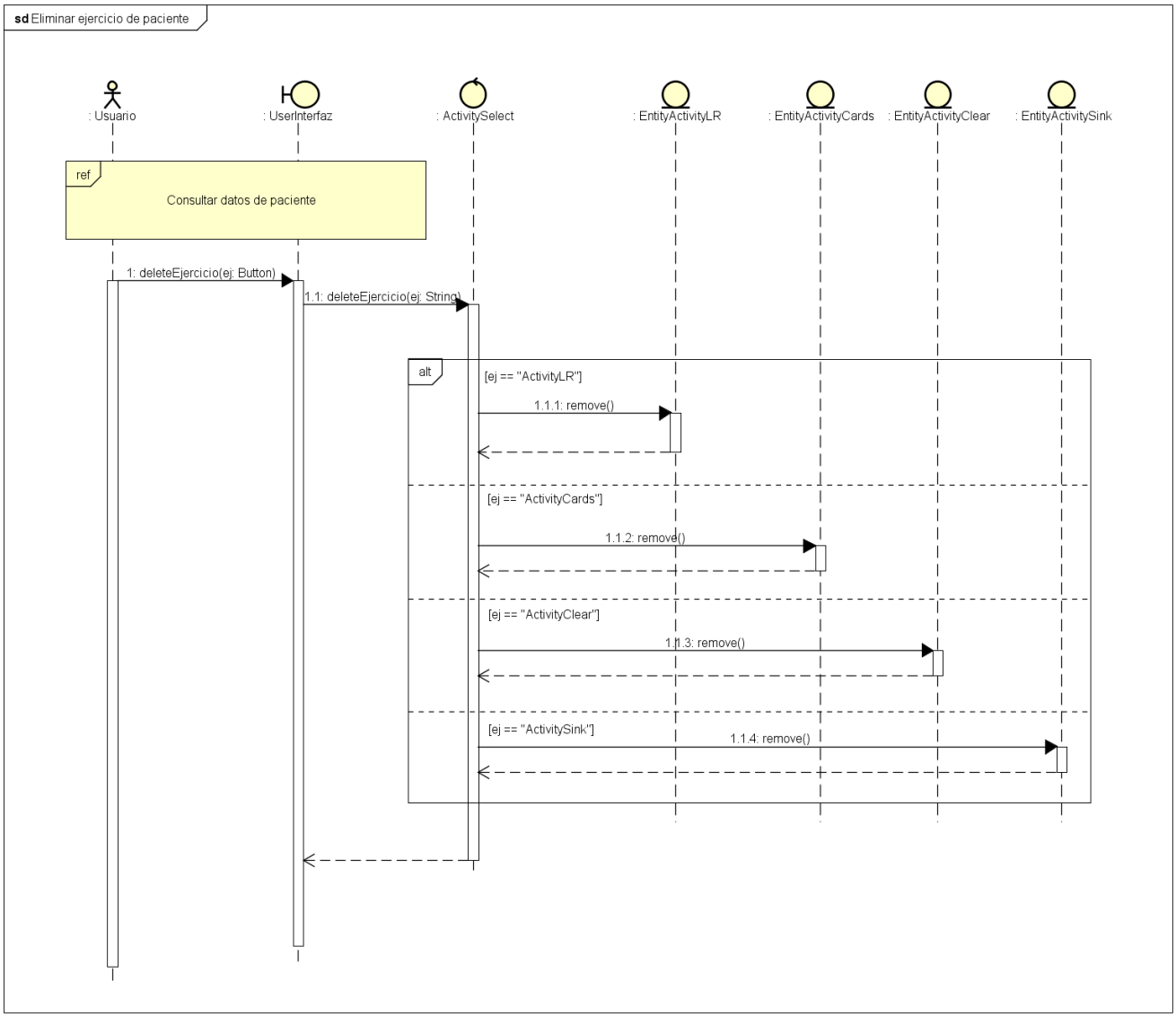

Figura 43: *DS – Eliminar ejercicio de paciente*

Cerrar sesión

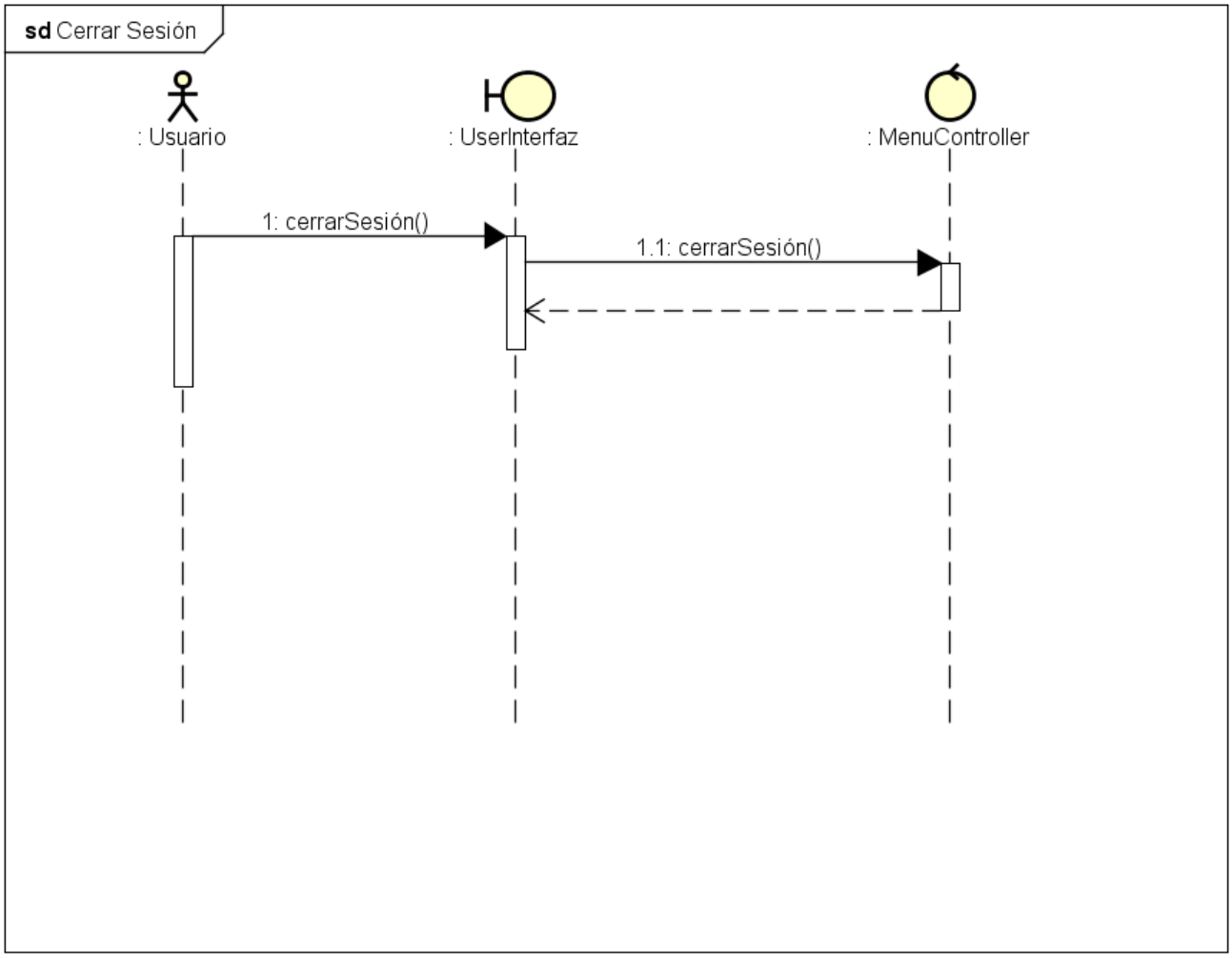

Figura 44: *DS – Cerrar sesión*

Elegir ejercicio

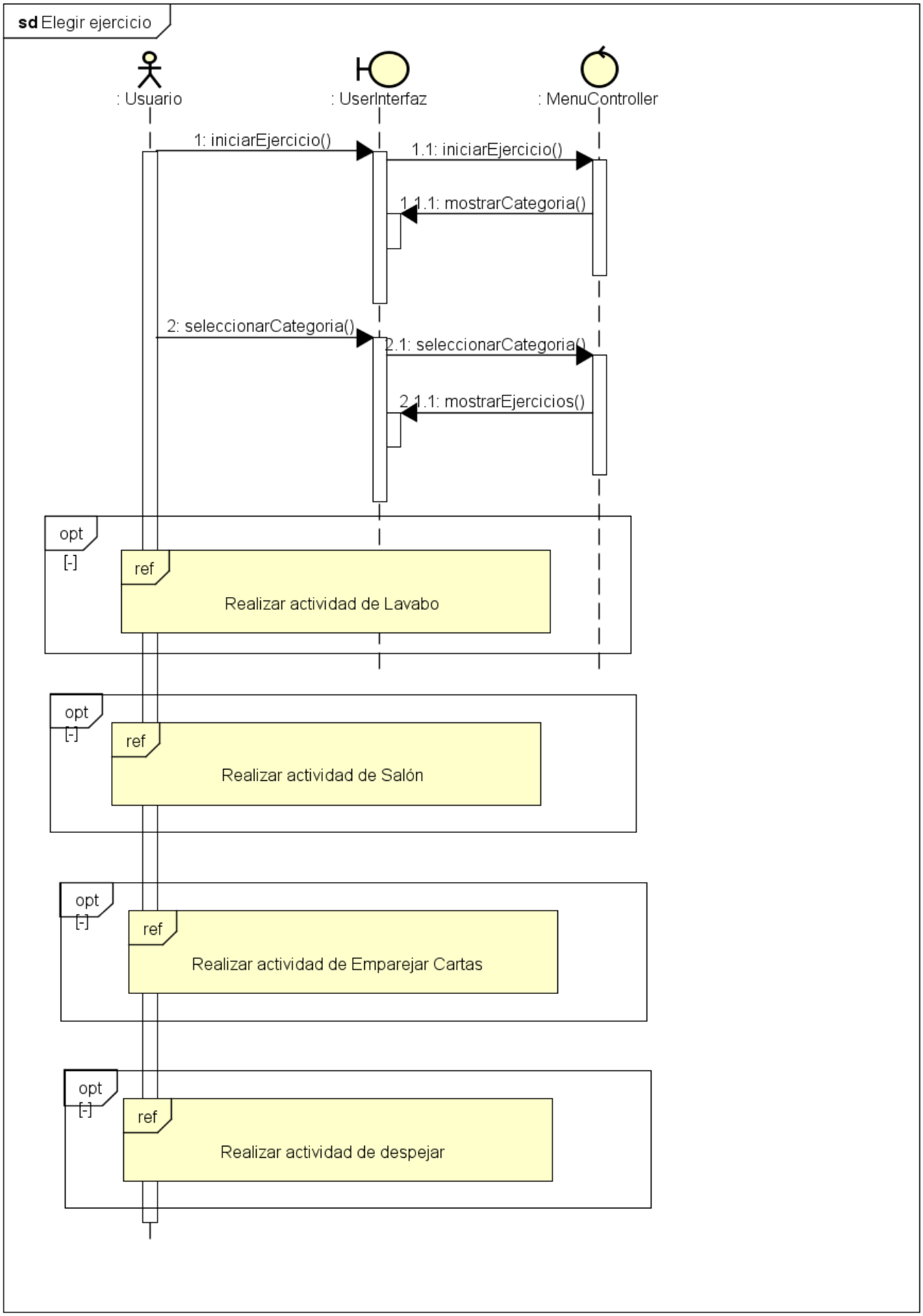

Figura 45: *DS – Elegir ejercicio*

Realizar actividad de lavabo

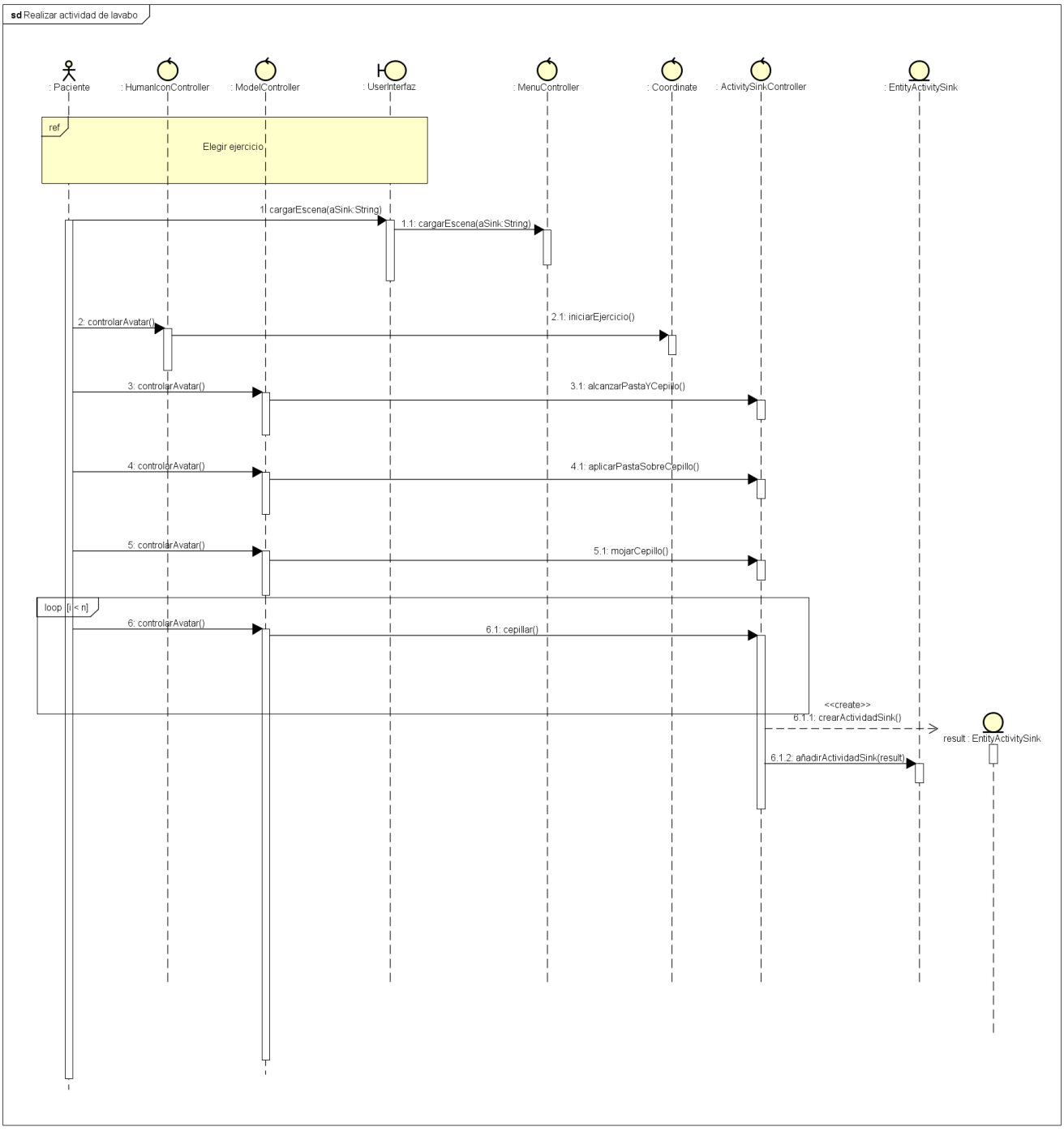

Figura 46: *DS – Realizar actividad de lavabo*

Realizar actividad de salón

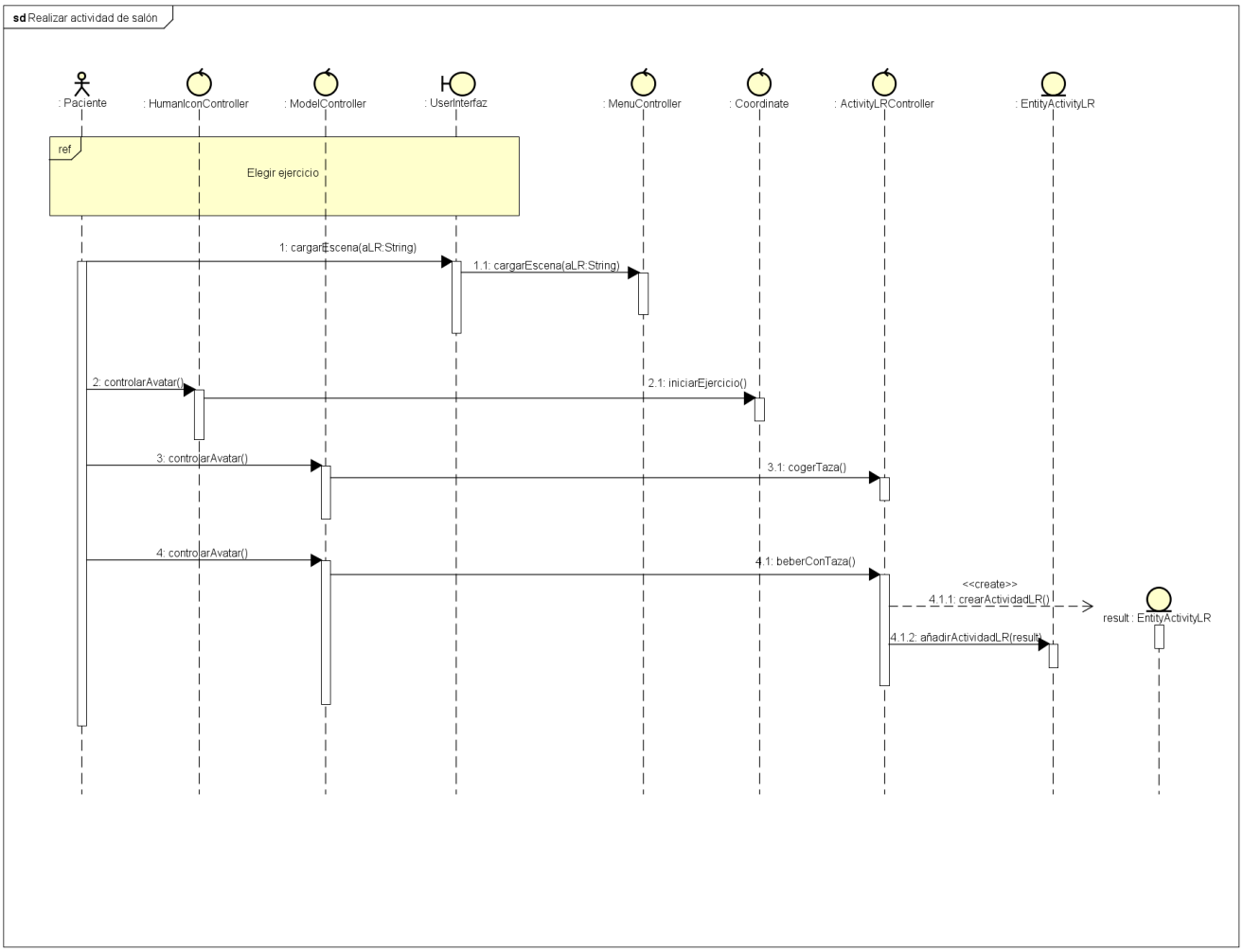

Figura 47: *DS – Realizar actividad de salón*

Realizar actividad emparejar cartas

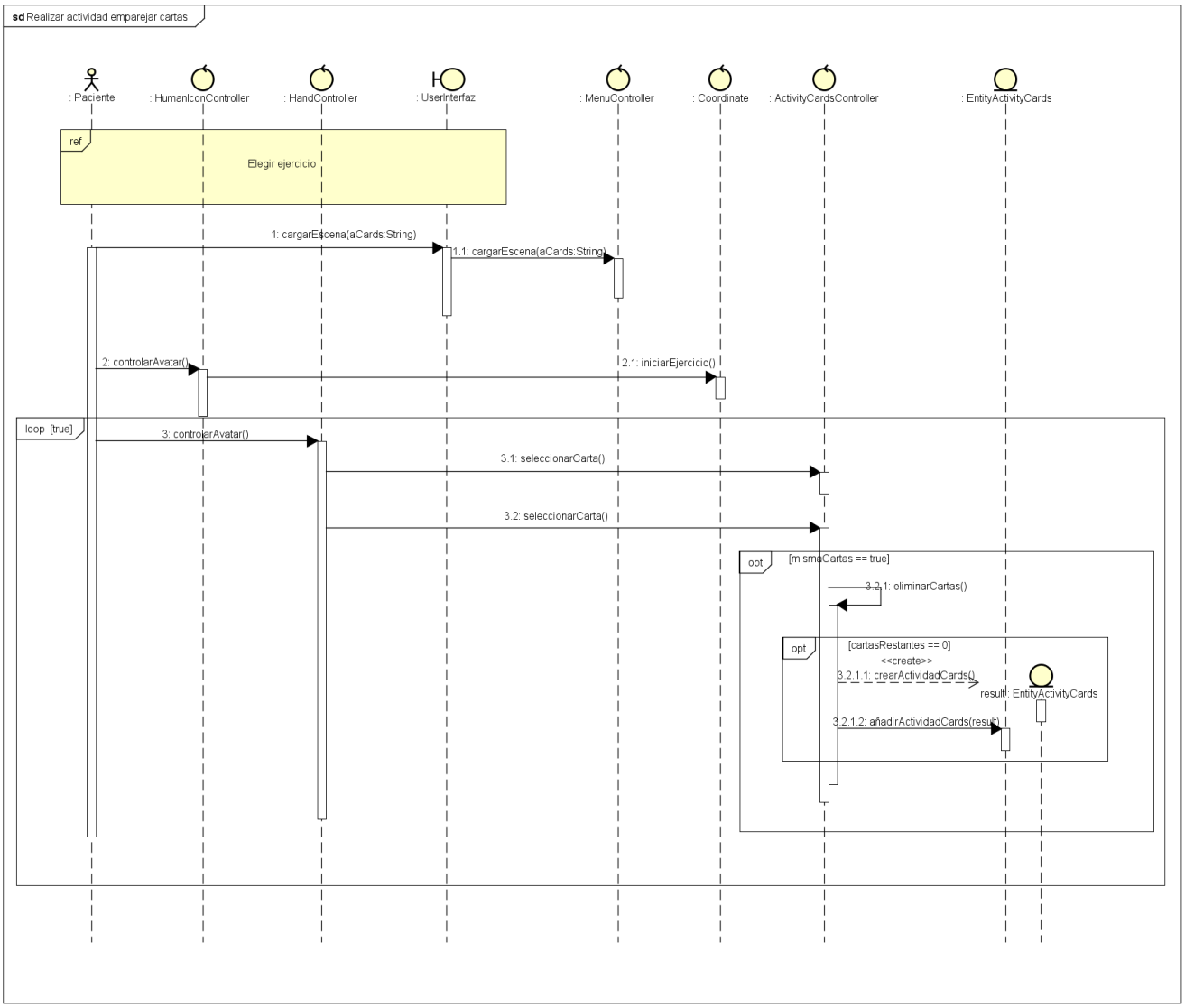

Figura 48: *DS – Realizar actividad emparejar cartas*

Realizar actividad de despejar

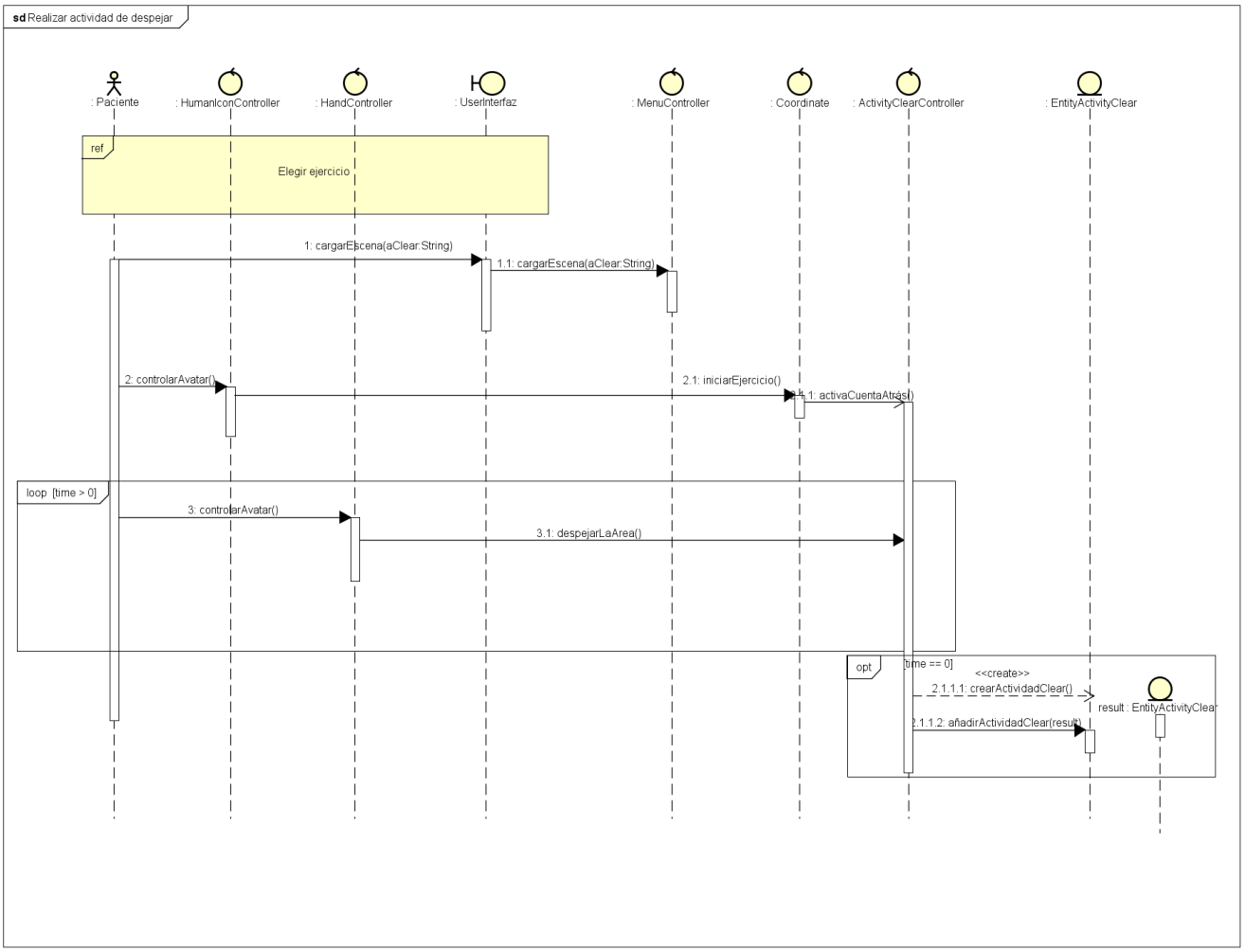

Figura 49: *DS – Realizar actividad de despejar*

Pausar actividad

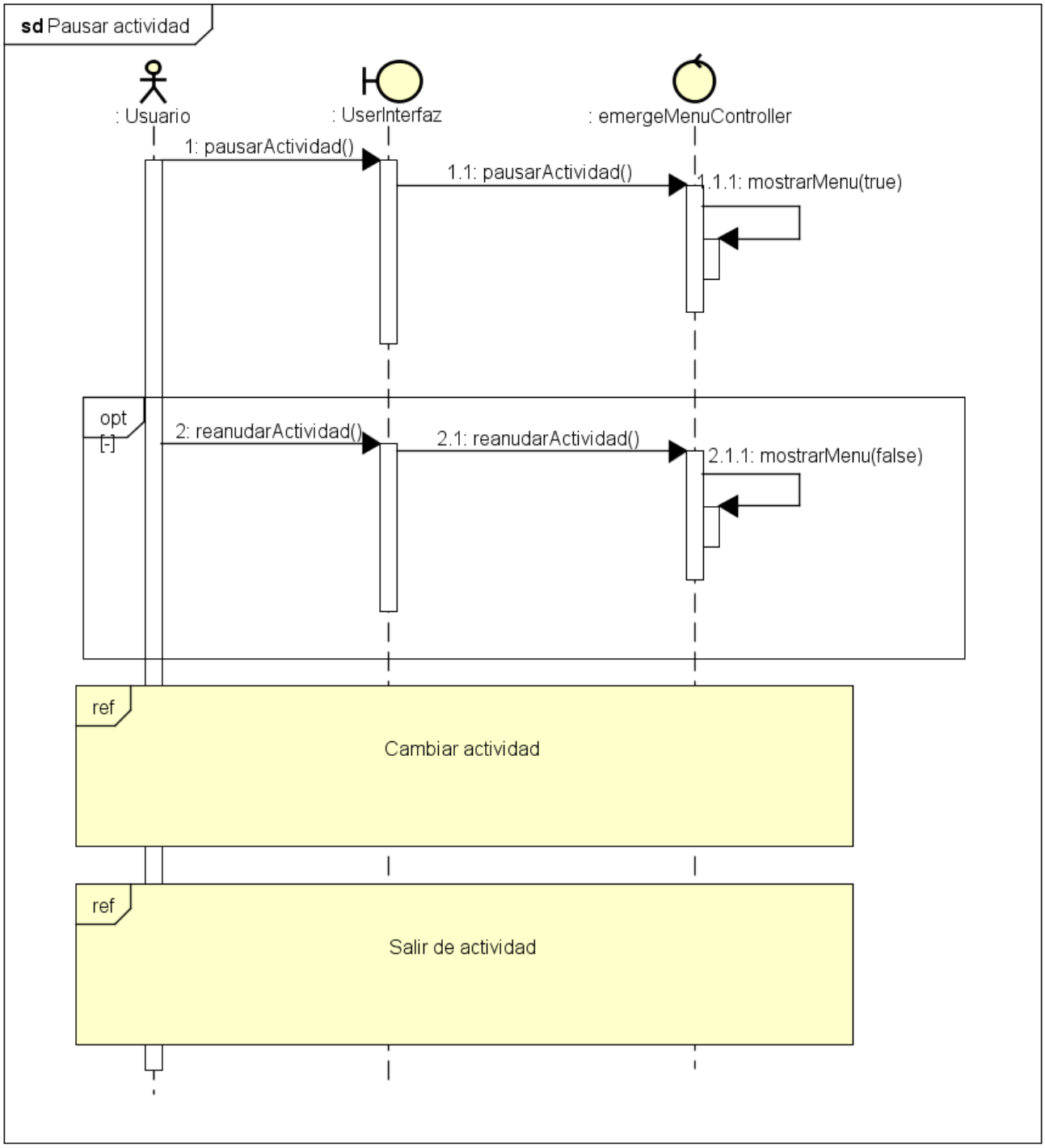

Figura 50: *DS – Pausar actividad*

Cambiar actividad

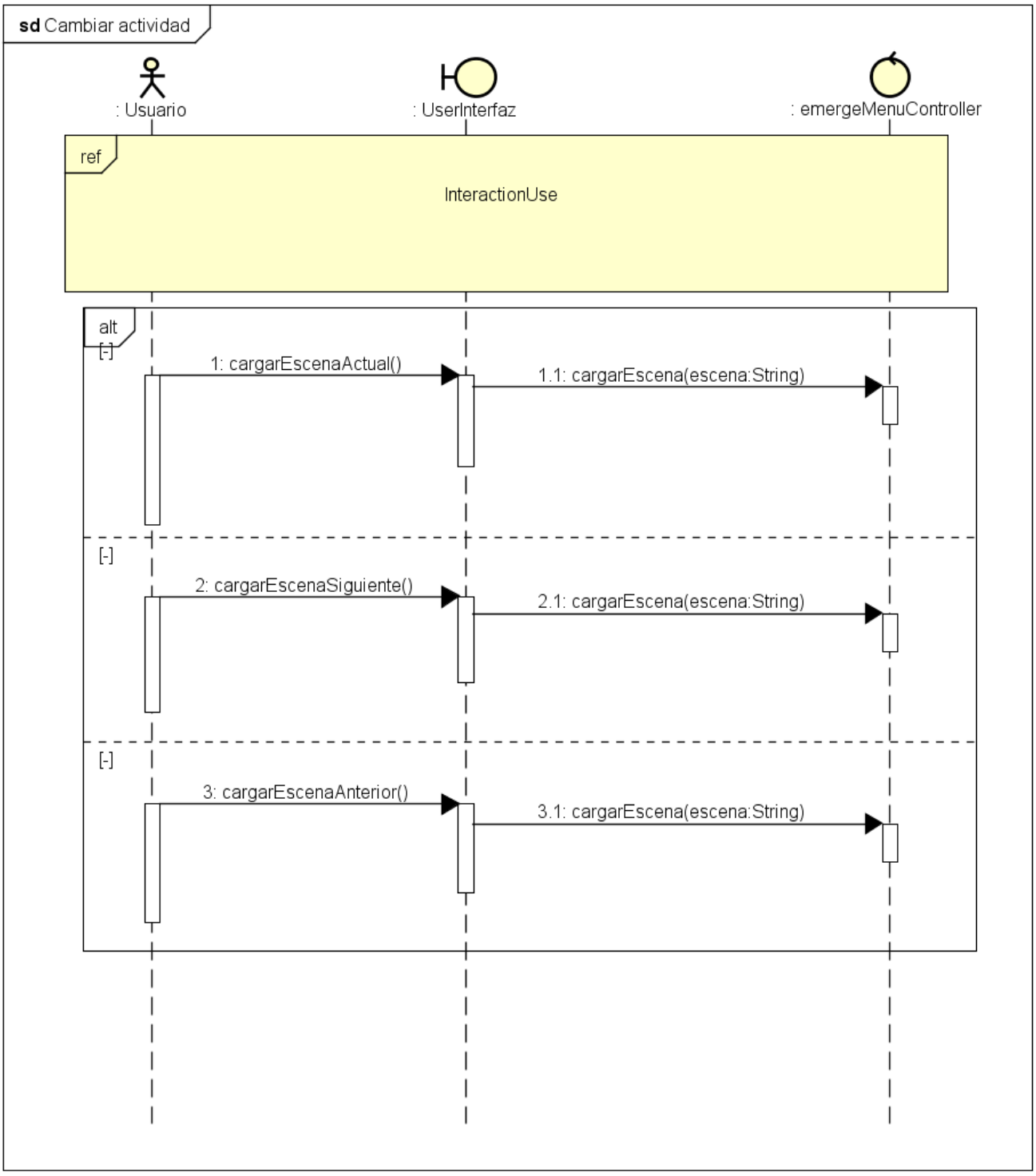

Figura 51: *DS – Cambiar actividad*

Salir de actividad

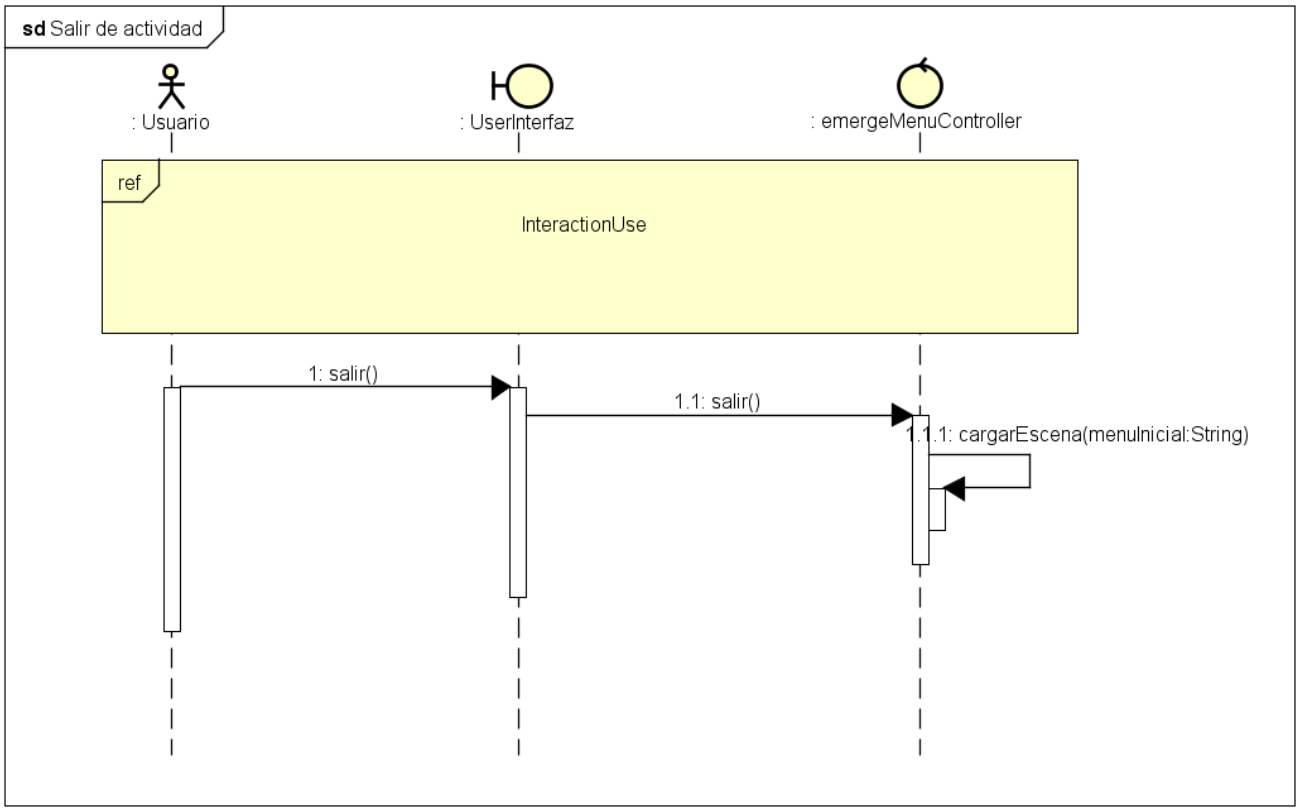

Figura 52: *DS – Salir de actividad*

4.6. Diseño de la interfaz

Para la interfaz se ha utilizado los recursos proporcionados por el Unity, lo que significa que la interfaz está dividida en diferentes escenas de Unity. A petición de cliente todos los diseños de las escenas están preparados para la pantalla con la resolución 1280x720. A continuación, se presentará el flujo de las escenas y capturas de las escenas.

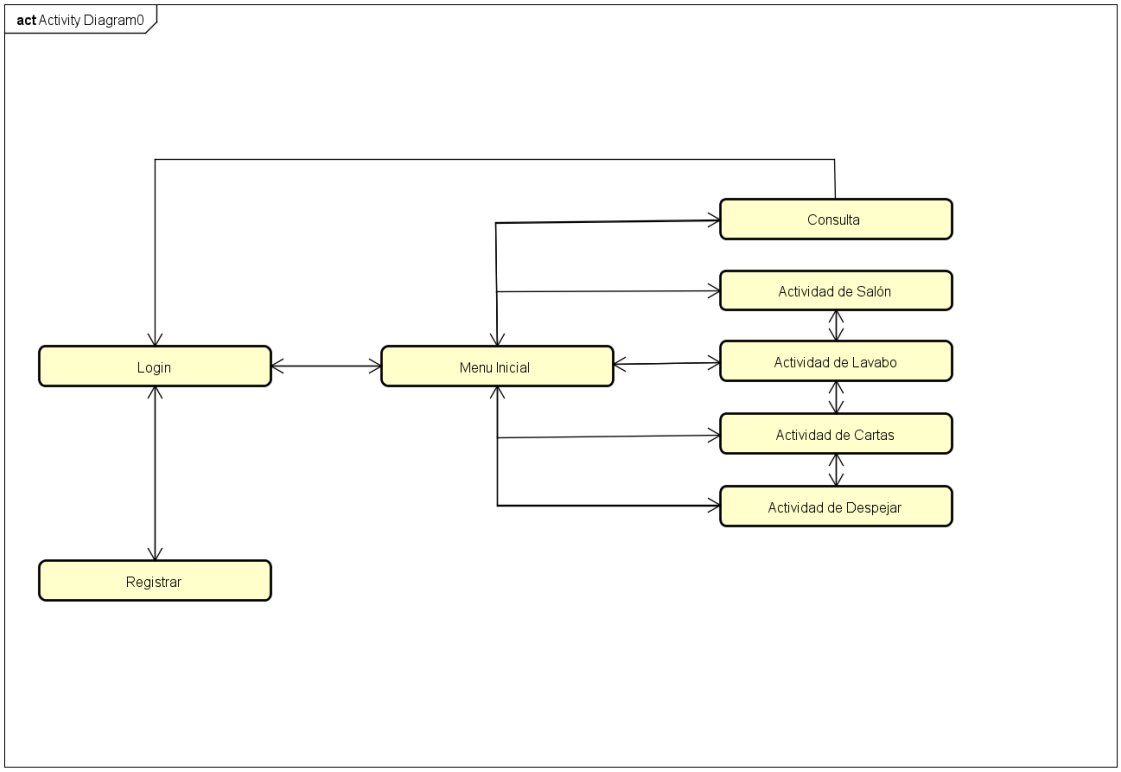

Figura 53: *Diagrama de navegación*

Escena Login:

Es la primera escena que se muestra al iniciar la aplicación, desde esta escena podemos navegar a la escena de registrar o identificarse para ir a la escena de menú inicial.

Figura 54: *Captura de escena Login*

Escena Registrar:

En esta escena la mayor parte de la pantalla se aprovecha para mostrar el formulario de registra y en la parte inferior derecha se sitúan los botones de “Registrar” y “Cancelar”.

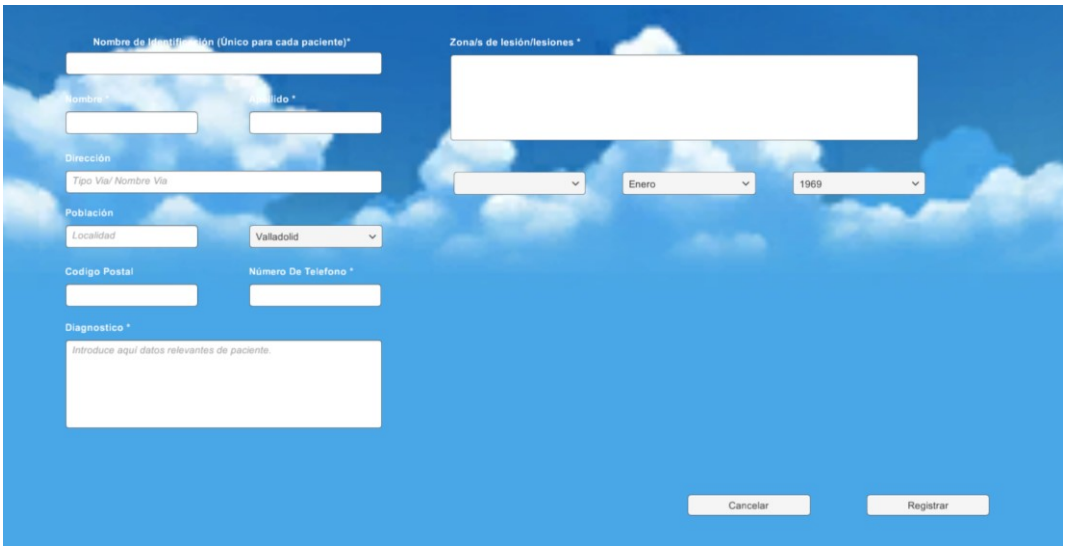

Figura 55: Captura de escena Registrar

Escena Menú Inicial:

La siguiente captura muestra la escena de menú inicial nada más inicia la sesión. Sin embargo, la lista de opción de la izquierda será sustituida para mostrar las categorías de ejercicios y la lista de ejercicios si el usuario selecciona la opción de “iniciar”.

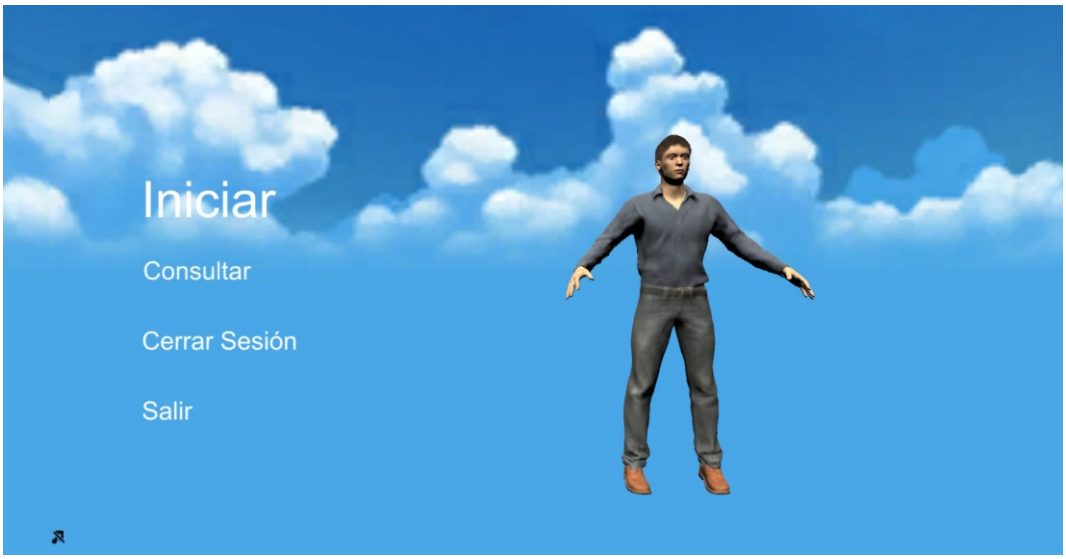

Figura 56: Captura de escena Menú inicial

Escena Consultar:

Esta escena está dividida en paneles, el panel izquierdo se muestra los datos personales y la lista de ejercicios realizados por el usuario. Y el panel derecho se muestra los detalles del ejercicio seleccionado.

Figura 57: *Captura de escena consultar*

Escena actividad de Salón:

Esta escena está decorada de forma que intenta imitar un salón normal para que el paciente se sienta familiarizado y cabe destacar tres objetos en ella: El avatar que simula los movimientos del paciente situado en la parte izquierda de la pantalla, la taza situada encima de la mesa y la pizarra que hay enfrente de paciente donde se muestran las instrucciones del ejercicio.

Figura 58: *Captura de escena actividad de salón*

Escena actividad de Lavabo:

Como puede observar en la **Figura 59** conforme con el avance de la actividad se modifica la perspectiva de la cámara de modo que el paciente disponen de las vistas que facilitan la realización de la actividad en todo el momento.

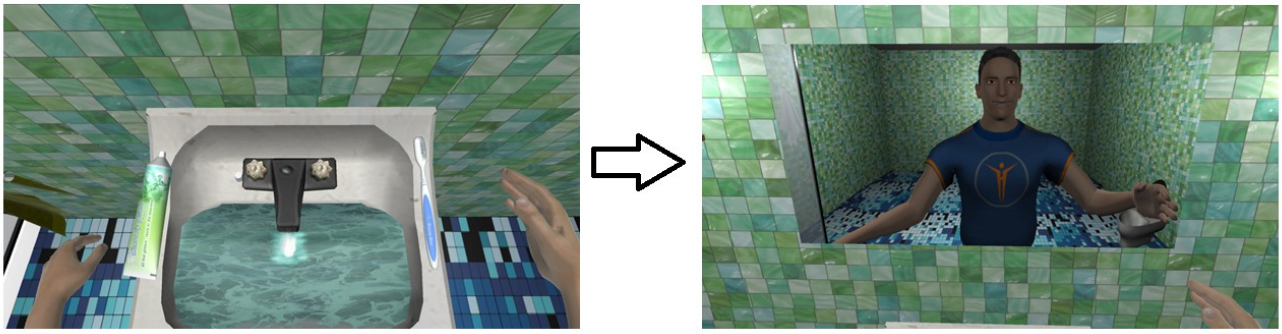

Figura 59: *Captura de escena actividad de Lavabo*

Escena actividad de Cartas:

Con el fin de ofrecer una interfaz intuitiva a los usuarios las parejas de cartas de esta escena son decoradas con imágenes fácilmente distinguibles.

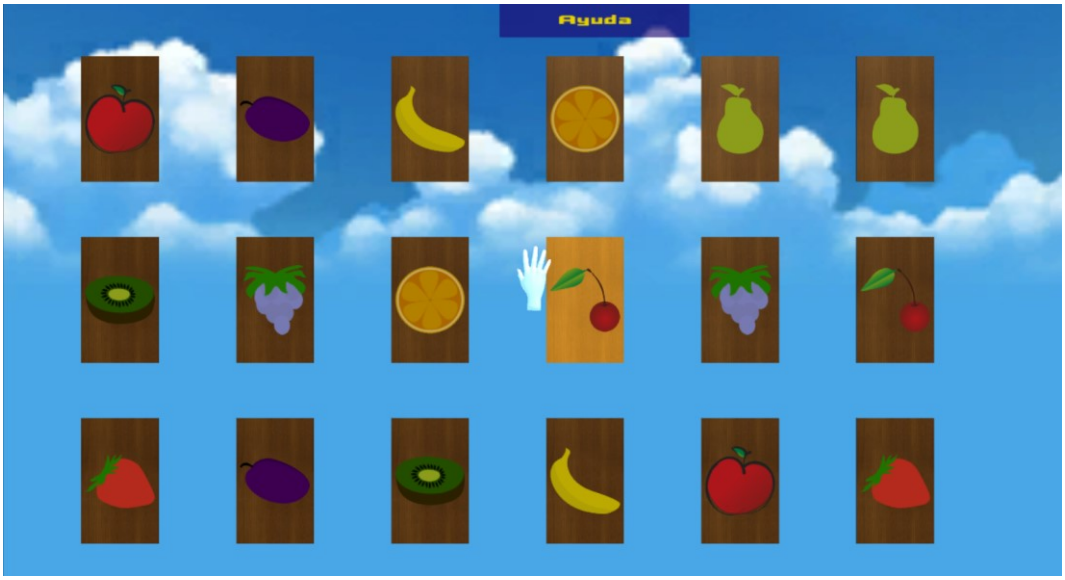

Figura 60: *Captura de escena actividad de Cartas*

Escena actividad de Despejar:

Para mantener un diseño sencillo de la interfaz, las áreas ocupadas son representadas simplemente por cubos semitransparentes y la ausencia de estos cubos indica que el área esta despejada.

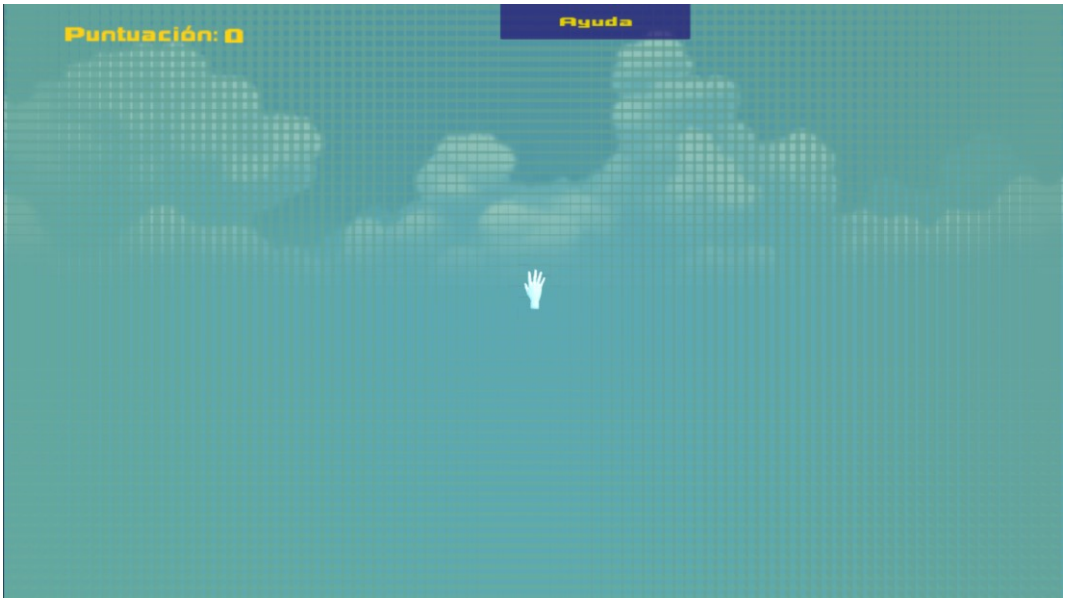

Figura 61: *Captura de escena actividad de Despejar*

5. IMPLEMENTACIÓN DE LA APLICACIÓN

5.1. Implementación de la base de datos

Para la base de datos se ha optad por SQLite y la biblioteca sqlite-net de código abierto escrito en C#. Gracias a las múltiples ventajas (mencionadas en la página 8) que proporciona SQLite es la idónea para la aplicación. En cuando la biblioteca sqlite-net el uso de esta biblioteca proporciona las siguientes ventajas:

- Es muy fácil de integrar con los proyectos existentes, solo hace falta copiar el fichero que contiene los códigos de sqlite-net a directorio del proyecto.
- No disminuye la eficiencia de SQLite.
- Métodos sencillos para realizar operaciones y consultas a base de datos.
- Permite realizar consultas y operaciones de forma síncrona y asíncrona.
- No crea dependencias adicionales.

A continuación, se describen los atributos que deben ser añadidos a la clase modelo para que sea reconocida por la SQLite.

Atributos de clase:

- **[Table (nombre de la tabla)]:** identifica la tabla de base de datos.

Atributos de las variables:

- **[PrimaryKey]:** identifica la clave primaria.
- **[ForeignKey]:** identifica la clave foránea.
- **[AutoIncrement]:** al insertar un nuevo elemento en la tabla, la variable con esta propiedad se generará e incrementará de forma automática. Esta propiedad solo se puede vincular a variable de tipo entero y a su vez es clave primaria.
- **[Column (nombre de la columna)]:** Identifica un nombre concreto para una columna, en lugar del nombre de la variable.
- **[Indexed] | [Indexed(string name, int order)]:** sirve para combinar dos tablas.
- **[Unique]:** los valores de la variable marcada con esta propiedad deben ser únicos.
- **[Ignore]:** identifica la variable que no queremos incluir en la tabla del base de datos.
- **[MaxLength(int length)]:** marca la longitud máxima de los valores de la variable.
- **[NotNull]:** El valor de la variable nunca puede ser vacía.

Los siguientes tipos de datos son suportados por la biblioteca:

- **Integers:** son almacenados usando *integer* o *bigint* de SQLite.

- **Boolean:** son almacenados como *integers*, con valor 1 para indicar True y el resto de los valores para indicar false.

- **Enums:** son almacenados como *integers* usando el valor de enumeración.

- **Singles & Doubles:** son almacenados como *float* de SQLite.

- **String:** son almacenados como *varchars* de SQLite con la longitud máxima marcada por el atributo *MaxLength*. Si el atributo no es especificado se tomará el valor por defecto 140.

- **DateTimes:** son almacenados como *datetime* de SQLite y su precisión depende de SQLite.

- **Byte []:** son almacenados como *blob* de SQLite.

- **Guid:** son almacenados como *varchars (36)* de SQLite.

En cuando a los métodos para realiza las operaciones y consultas a base de datos disponemos de las siguientes:

- **CreateTable:** para generar una tabla en el base de datos.

```
var db = new SQLiteConnection("foofoo");
db.CreateTable<Stock>();
db.CreateTable<Valuation>();
```

- **Insert:** insertar una fila en una tabla.

```
public static void AddStock(SQLiteConnection db, string symbol) {
    var s = db.Insert(new Stock() {
        Symbol = symbol
    });
Console.WriteLine("{0} == {1}", s.Symbol, s.Id);
}
```

- **Update & Delete:** como sus nombres indican uno para actualizar y otro para borrar valores. Y su uso es similar a *Insert*.

- **Table:** una forma fácil de realizar consultas a base de datos con la restricción de que solo se puede realizar consultas de tipo *WHERE* y *ORDER BY.*

```
    var conn = new SQLiteConnection("foofoo");
    var query = conn.Table<Stock>().Where(v =>
v.Symbol.StartsWith("A"));
```

```
    foreach (var stock in query)
        Debug.WriteLine("Stock: " + stock.Symbol);
```

- **Query:** sirve para realizar consultas a bajo nivel.

```
public class Val {
    public decimal Money { get; set; }
    public DateTime Date { get; set; }
}
public static IEnumerable<Val> QueryVals (SQLiteConnection db, Stock
stock)
{
    return db.Query<Val> ("select 'Price' as 'Money', 'Time' as 'Date'
from Valuation where StockId = ?", stock.Id);
}
```

5.2. El patrón Entity-Component-Systems (Entidad-Componente-Sistemas)

La forma tradicional de implementar las entidades de juegos es la programación orientada a objetos. En POO para compartir los mismos comportamientos y funciones entre las entidades se hace a través del polimorfismo. Esto a menudo da lugar a jerarquías de clases grandes y rígidas que se hace cada vez más difícil encaja nueva entidad en la jerarquía. Además, puede crear ambigüedades debido a la herencia múltiple como caso de problema diamante.

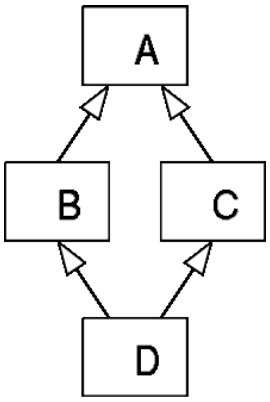

Figura 62: *Diagrama de herencia en diamante*

El patrón Entidad-Componente-Sistema (ECS) ofrece la solución a estos problemas que se presentan en la programación orientada a objetos. La construcción de entidades se hace mediante composición en lugar de herencia, una entidad es una simpe agregación de componentes. Esto supone que la creación de nuevas entidades es muy sencilla y eficiente.

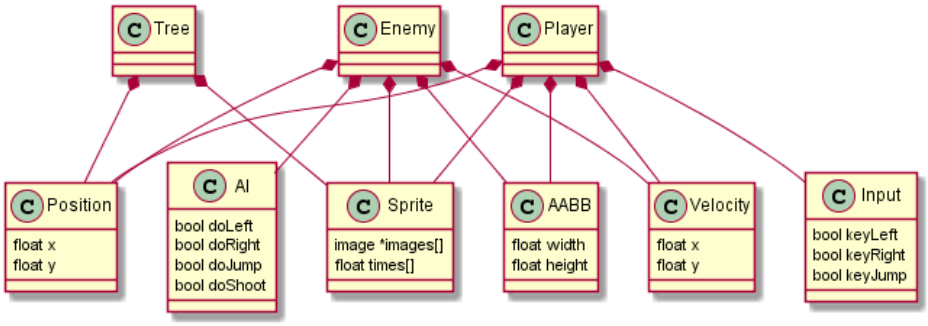

Figura 63: *Ejemplo de una diagrama de ECS*

Componente

Los componentes no tienen métodos y solo sirven para almacenar datos. Generalmente todos los componentes derivan de una misma clase abstracta, de esta forma facilita la obtención del tipo de componente y los componentes asignados a la entidad en tiempo de ejecución. Los componentes por si solos casi no tienen ningún sentido, pero junto con entidades y sistemas pueden resultar muy potentes.

Entidad

Una entidad es algo que está presente en el mundo de juego. Una entidad tiene un ID único para identificarse, todos los componentes asignados a una entidad compartirán ese ID. Si fuera necesario se puede eliminar o añadir componentes a la entidad durante la ejecución.

Sistema

Cada sistema funciona de forma continua es como si cada sistema tiene su propio hilo privado y lleva a cabo acciones globales en cada entidad que posee un componente del mismo aspecto que el sistema. Por ejemplo, un sistema para detectar las colisiones actuaría en todas las entidades que tienen un componente físico. Y la forma de comunicar entre los diferentes sistemas es almacenar los datos en componentes y actualizarlos constantemente. Sin embargo, esta solución puede ser muy poco eficiente ante una gran cantidad de eventos poco frecuentes. Otra solución sería usar el patrón observador, de forma que la acción asociada al evento solo se ejecutará una vez, cuando es invocado.

Todas estas características del patrón ECS lo hacen idóneo para el desarrollo con Unity que está basado en componentes. Todas las entidades son instancias de GameObject y se les puede asignar diferentes tipos de Component. Las implementaciones de las acciones asociadas a los eventos se implementan mediante scripts descendientes de MonoBehavior.

5.3. Control de avatar

En este apartado, se explica la gestión de los datos proporcionado por el sensor Kinect para controlar el avatar. La obtención y asignación de los datos brutos de Kinect al avatar es muy sencillo una vez importados los paquetes de Unity Kinect:

```
public Transform RightUpperArm;
private BodySourceManager bodyManager;

void Start()
{
     if(GameObject.FindGameObjectWithTag("KinectSrcManager") == null)
        {
            var b = Instantiate(BodySrcManager);
            bodyManager = b.GetComponent<BodySourceManager>();
        }
        else
        {
            bodyManager =
GameObject.FindGameObjectWithTag("KinectSrcManager").GetComponent<BodySourceManager>();
        }
}

void Update()
 {
     if (bodyManager != null)
        {
            bodies = bodyManager.GetData();

            if (bodies != null)
            {
                Body body = null;
                if (bodies.Length != 0)
                {
                   body = bodies[0];
                }
                Var orientation = body.JointOrientation[bone.Value].Orientation;
                Var q = new Quartenion(orientation.x, orientation.y, orientation.z,
orientation.w);
                RightUpperArm.transform.rotation =q;
             }
          }
 }
```

Sin embargo, el control mediante los datos brutos del sensor Kinect es muy poco satisfactorio ya que los datos son muy inestables y poco fiables. Por lo tanto, es imprescindible aplicar un filtro a los datos obtenidos del sensor Kinect:

```
    private Quaternion SmoothFilter(List<Quaternion> quaternions, Quaternion lastMedian)
    {
        Quaternion median = new Quaternion(0, 0, 0, 0);

        foreach (Quaternion quaternion in quaternions)
        {
            float weight = 1 - (Quaternion.Dot(lastMedian, quaternion) / (Mathf.PI / 2));
// 0 degrees of difference => weight 1. 180 degrees of difference => weight 0.
            Quaternion weightedQuaternion = Quaternion.Lerp(lastMedian, quaternion,
weight);

            median.x += weightedQuaternion.x;
            median.y += weightedQuaternion.y;
            median.z += weightedQuaternion.z;
            median.w += weightedQuaternion.w;
```

```
        }

        median.x /= quaternions.Count;
        median.y /= quaternions.Count;
        median.z /= quaternions.Count;
        median.w /= quaternions.Count;

        return NormalizeQuaternion(median);
    }

    public Quaternion NormalizeQuaternion(Quaternion quaternion)
    {
        float x = quaternion.x, y = quaternion.y, z = quaternion.z, w = quaternion.w;
        float length = 1.0f / (w * w + x * x + y * y + z * z);
        return new Quaternion(x * length, y * length, z * length, w * length);
    }
```

5.4. Algoritmo de evaluación de matriz

Durante la implementación de la actividad de Despejar para poder conocer y eliminar los cubos aislados se necesitan evaluaciones constantes de la matriz de cubos y para que esto sea viable se precisa un algoritmo eficiente.

Se ha optado la implementación de algoritmo mediante algoritmo de Backtracking, a continuación, se muestra el árbol de búsqueda:

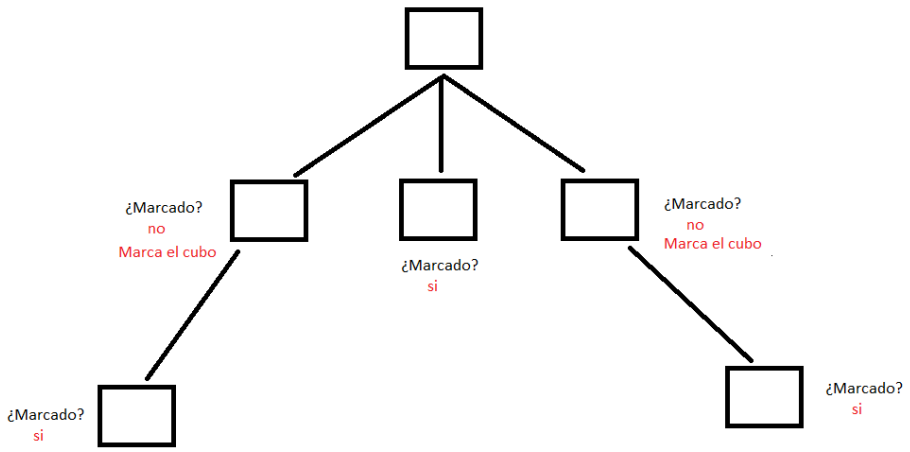

Figura 64: *Árbol de búsqueda*

Tras la evaluación de la matriz de cubos, los cubos marcados significan que no están aislados y los que no están marcados significa que están aislados y son eliminados de la matriz.

```
private void updateCubeM()
    {
```

```
        foreach (var c in cubeM)
        {
            if (c != null)
            {
                c.setStatus(false);
            }
        }
        updateNeighbour(0, 0);
        foreach (var c in cubeM)
        {
            if(c.getPhysicalObject()!= null && c.getStatus() == false)
            {
                c.getPhysicalObject().tag = "Off";
            }
        }
        var hS = playerController.getRHandState();
        if (!hS.Equals(handState))
        {
            if (!hS.Equals(HandState.Lasso) && !hS.Equals(HandState.Unknown) &&
!hS.Equals(HandState.NotTracked))
            {
                handState = hS;
                player.GetComponent<Collider>().enabled =
!player.GetComponent<Collider>().enabled;
            }
        }
    }

    private void updateNeighbour(int originY, int originX)
    {
        cubeM[originY, originX].setStatus(true);
        for (int i = -1; i <= 1; i++)
        {
            for (int j = -1; j <= 1; j++)
            {
                int x = originX + i;
                int y = originY + j;
                if (x >= 0 && x <= sizeX && y >= 0 && y <= sizeY)
                {
                    if(cubeM[y, x].getPhysicalObject() != null) {
                        if (cubeM[y, x].getStatus() == false && cubeM[y,
x].getPhysicalObject().tag.Equals("On"))
                        {
                            updateNeighbour(y, x);
                        }
                    }
                }
            }
        }
    }
```

6. PRUEBAS

6.1. Introducción

Este capítulo recoge los casos de pruebas que se han realizado sobre el sistema para detectar los posibles errores.

Todas las pruebas que se presenta a continuación son de caja negra, ya que las pruebas de caja blanca se han realizado durante la implementación del programa. Por lo tanto, el objetivo principal de estas pruebas es comprobar el cumplimiento de los requisitos especificados y que funciona de manera correcta.

6.2. Casos de prueba

Los siguientes casos de pruebas son organizados en función de los casos de usos:

UC-001 Registrar

Prueba	Resultado	Valoración
Registrar el paciente rellenando todos los campos del formulario.	El sistema guardó correctamente los datos de paciente.	OK
Registrar el paciente rellenando solo los campos obligatorios del formulario.	El sistema guardó correctamente los datos de paciente.	OK
Registrar el paciente sin rellena algunos campos obligatorios del formulario.	El sistema denegó registrar el paciente e informó del error.	OK
Registrar el paciente con un Nick ya existente.	El sistema denegó registrar el paciente e informó del error.	OK

Tabla 32: *Caso de prueba – Registrar*

UC-002 Identificar

Prueba	Resultado	Valoración
Iniciar la sesión con el identificador de un paciente registrado.	El usuario inició correctamente la sesión de paciente.	Ok
Iniciar la sesión con un identificador falso.	El sistema no pudo encontrar el paciente e informó del error.	Ok

Tabla 33: *Caso de prueba – Identificar*

UC-003 Consultar datos de paciente

Prueba	Resultado	Valoración
Seleccionar la opción de “Consultar” de menú inicial.	El sistema mostró correctamente los datos personales de paciente y su lista de ejercicios realizados.	Ok

Tabla 34: *Caso de prueba – Consultar datos de paciente*

UC-004 Consultar resultado de ejercicio

Prueba	Resultado	Valoración
Desde la ventana de datos personales, seleccionar un resultado de actividad de Salón.	El sistema mostró los resultados de actividad de Salón en el panel de detalle de actividades.	Ok
Desde la ventana de datos personales, seleccionar un resultado de actividad de Cartas	El sistema mostró los resultados de actividad de Cartas en el panel de detalle de actividades.	Ok
Desde la ventana de datos personales, seleccionar un resultado de actividad de Despejar.	El sistema mostró los resultados de actividad de Despejar en el panel de detalle de actividades.	Ok
Desde la ventana de datos personales, seleccionar un resultado de actividad de Lavabo.	El sistema mostró los resultados de actividad de lavabo en el panel de detalle de actividades.	Ok

Tabla 35: *Caso de prueba – Consultar resultado de ejercicio*

UC-005 Eliminar paciente

Prueba	Resultado	Valoración
Desde la ventana de datos personales, pulsar el botón “eliminar”.	El paciente y todos los datos relacionados con él es eliminado de la base de datos.	Ok

Tabla 36: *Caso de prueba – Eliminar paciente*

UC-006 Modificar datos de paciente

Prueba	Resultado	Valoración
Desde la ventana de datos personales, modificar todos los campos de datos con datos no vacíos.	Los datos modificados fueron almacenados correctamente en la base de datos.	Ok
Desde la ventana de datos personales, modificar algunos campos obligatorios con datos vacíos.	El sistema denegó la operación e informó del error.	Ok
Desde la ventana de datos personales, modificar algunos campos no obligatorios con datos vacíos.	Los datos modificados fueron almacenados correctamente en la base de datos.	Ok

Tabla 37: *Caso de prueba – Modificar datos de paciente*

UC-007 Eliminar ejercicio de paciente

Prueba	Resultado	Valoración
Desde la ventana de datos personales, eliminar resultado de actividad de Salón.	El resultado fue eliminado correctamente.	Ok
Desde la ventana de datos personales, eliminar resultado de actividad de Cartas.	El resultado fue eliminado correctamente.	Ok
Desde la ventana de datos personales, eliminar resultado de actividad de Despejar.	El resultado fue eliminado correctamente.	Ok
Desde la ventana de datos personales, eliminar resultado de actividad de Lavabo.	El resultado fue eliminado correctamente.	Ok

Tabla 38: *Caso de prueba – Eliminar ejercicio de paciente*

UC-008 Cerrar sesión

Prueba	Resultado	Valoración
Seleccionar la opción “Cerrar sesión” de menú inicial.	La sesión del paciente fue cerrada correctamente.	Ok

Tabla 39: *Caso de prueba – Cerrar sesión*

UC-009 Elegir ejercicio

Prueba	Resultado	Valoración
Desde menú inicial seleccionar la opción “Iniciar”.	El sistema mostró los dos bloques de actividades disponibles.	Ok
Después de haber seleccionado “Iniciar”, seleccionar “Actividad diaria”.	El sistema mostró la lista de actividades diarias disponibles.	Ok
Después de haber seleccionado “Iniciar”, seleccionar “Mini juegos”.	El sistema mostró la lista de mini juegos disponibles.	Ok
Después de haber seleccionado “Iniciar”, seleccionar el icono de la flecha.	El sistema volvió a mostrar las opciones de la lista inicial.	Ok
Después de haber seleccionado “Iniciar” y “Actividad diaria”, seleccionar “Salón”.	El sistema cargó la escena de actividad de Salón.	Ok
Después de haber seleccionado “Iniciar” y “Actividad diaria”, seleccionar “Lavabo”.	El sistema cargó la escena de actividad de Lavabo.	Ok
Después de haber seleccionado “Iniciar” y “Actividad diaria”, seleccionar el icono de la flecha.	El sistema volvió a mostrar los dos bloques de actividades.	Ok
Después de haber seleccionado “Iniciar” y “Mini juegos”, seleccionar “Cartas”.	El sistema cargó la escena de actividad de Cartas.	Ok
Después de haber seleccionado “Iniciar” y “Mini juegos”, seleccionar “Despejar”.	El sistema cargó la escena de actividad de Despejar.	Ok
Después de haber seleccionado “Iniciar” y “Mini juegos”, seleccionar el icono de la flecha.	El sistema volvió a mostrar los dos bloques de actividades.	Ok

Tabla 40: *Caso de prueba – Elegir ejercicio*

UC-010 Realizar actividad de lavabo

Prueba	Resultado	Valoración
Situar en el centro del campo de detección.	El sistema inició correctamente el ejercicio una vez el paciente situó en el centro.	Ok
Colisionar la mano izquierda de avatar con la pasta de dientes.	La pasta de dientes fue vinculada a la mano izquierda una vez son colisionados.	Ok
Colisionar la mano derecha de avatar con el cepillo de dientes.	El cepillo de dientes fue vinculado a la mano derecha una vez son colisionados.	Ok
Una vez cogido la pasta y el cepillo, colisionar la pasta con el cepillo.	La pasta fue aplicada en el cepillo una vez son colisionados.	Ok
Una vez aplicado la pasta, colisionar el cepillo con agua.	Tras la colisión el cepillo fue mojado.	Ok
Cuando esta mojado el cepillo, colisionar el cepillo con la boca.	Fueron capturados correctamente las repeticiones de cepillados.	Ok
Sin que haya aplicado la pasta sobre el cepillo, colisionar el cepillo con el agua.	No hubo ningún efecto.	Ok
Sin que haya mojado el cepillo, colisionar el cepillo con la boca.	No hubo ningún efecto.	Ok

Tabla 41: *Caso de prueba – Realizar actividad de lavabo*

UC-011 Realizar actividad de salón

Prueba	Resultado	Valoración
Situar en el centro del campo de detección.	El sistema inició correctamente el ejercicio una vez el paciente situó en el centro.	Ok
Colisionar la mano derecha con la taza de café.	La taza fue vinculada con la mano derecha.	Ok
Una vez cogido la taza de café, colisionarlo con la boca.	Fue capturada la colisión y finalizo correctamente la actividad.	Ok

Tabla 42: *Caso de prueba – Realizar actividad de salón*

UC-012 Realizar actividad emparejar cartas

Prueba	Resultado	Valoración
Situar en el centro del campo de detección.	El sistema inició correctamente el ejercicio una vez el paciente situó en el centro.	Ok
Seleccionar dos cartas idénticas.	Las dos cartas son eliminadas.	Ok
Seleccionar dos cartas distintas.	Las dos cartas seleccionadas son deseleccionadas.	Ok
Eliminar todas las cartas.	La actividad fue finalizada correctamente.	Ok

Tabla 43: *Caso de prueba – Realizar actividad emparejar cartas*

UC-013 Realizar actividad despejar

Prueba	Resultado	Valoración
Situar en el centro del campo de detección.	El sistema inició correctamente el ejercicio una vez el paciente situó en el centro.	Ok
Cerrar el puño en un punto.	Los cubos que se encontraban en la posición de la mano son eliminados.	Ok
Mantener cerrado el puño mientras dibuja una línea.	Los cubos que se encontraban en la trayectoria de la mano son eliminados.	Ok
Mantener cerrado el puño mientras dibuja una circunferencia.	Los cubos que se encontraban encerrados en el área trazada por la mano son eliminados.	Ok
Esperar hasta que el contador de cuenta atrás llegue a cero.	La actividad fue finalizada correctamente.	Ok

Tabla 44: *Caso de prueba – Realizar actividad despejar*

UC-014 Pausar actividad

Prueba	Resultado	Valoración
Pulsar botón “Esc” para pausar la actividad de Salón.	La actividad de Salón fue pausada y el menú emergente fue mostrado correctamente.	Ok
Pulsar botón “Esc” para pausar la actividad de Cartas.	La actividad de Cartas fue pausada y el menú emergente fue mostrado correctamente.	Ok
Pulsar botón “Esc” para pausar la actividad de Despejar.	La actividad de despejar fue pausada y el menú emergente fue mostrado correctamente.	Ok
Pulsar botón “Esc” para pausar la actividad de Lavabo.	La actividad de Lavabo fue pausada y el menú emergente fue mostrado correctamente.	Ok
Pulsar de nuevo el botón “Esc” para recuperar la actividad de Salón después de haber pausado.	La actividad de Salón recuperó correctamente de la pausa.	Ok
Pulsar de nuevo el botón “Esc” para recuperar la actividad de Cartas después de haber pausado.	La actividad de Cartas recuperó correctamente de la pausa.	Ok
Pulsar de nuevo el botón “Esc” para recuperar la actividad de Despejar después de haber pausado.	La actividad de Despejar recuperó correctamente de la pausa.	Ok
Pulsar de nuevo el botón “Esc” para recuperar la actividad de Lavabo después de haber pausado.	La actividad de Lavabo recuperó correctamente de la pausa.	Ok

Tabla 45: *Caso de prueba – Pausar actividad*

UC-015 Cambiar actividad

Prueba	Resultado	Valoración
Seleccionar cargar a la actividad siguiente desde menú emergente.	La escena de la actividad siguiente fue cargada correctamente.	Ok
Seleccionar cargar a la actividad anterior desde menú emergente.	La escena de la actividad anterior fue cargada correctamente.	Ok
Seleccionar recargar a la actividad actual desde menú emergente.	La escena de la actividad actual fue cargada correctamente.	Ok

Tabla 46: *Caso de prueba – Cambiar actividad*

UC-016 Salir de la actividad

Prueba	Resultado	Valoración
Seleccionar la opción de salir de la actividad desde menú emergente.	La escena del menú inicial fue cargada correctamente.	Ok

Tabla 47: *Caso de prueba – Salir de la actividad*

7. CONCLUSIONES

7.1. Objetivos alcanzados

El desarrollo del proyecto concluyó positivamente satisfaciendo los requisitos planteados por el cliente. Los resultados obtenidos son los siguientes:

- Se han implementado los controladores de los avatares usando Kinect que servirán en el futuro para implementar nuevas actividades.

- Se ha desarrollado un algoritmo de filtrado de los datos proporcionado por el sensor Kinect para ofrecer mayor estabilidad y fiabilidad a los controladores de los avatares.

- Se ha implementado una base de datos que almacena los datos personales del usuario y los resultados de las actividades realizadas del mismo.

- Se han desarrollado 2 escenas que simulan las actividades de la vida diaria y dos mini juegos que sirven como ejercicios para los usuarios.

- Se ha implementado una sencilla interfaz para que el usuario pueda usar la aplicación desde primer momento sin ayuda de profesionales.

7.2. Líneas de trabajo futuras

Teniendo en cuenta el tiempo disponible desde el inicio del proyecto se ha establecido los de límites para los cuales se dan por alcanzados los objetivos marcado. No obstante, es importante resaltar aquellas posibles mejoras que podrían ser incluidas en futuros desarrollos:

- Incluir más variedades de actividades de la vida diaria o mini juegos en la aplicación. Por ejemplo, la actividad de ducha.

- Implementar un servidor que centralice los datos de los usuarios, de esta manera los terapeutas pueden acceder a los datos de estos y llevar acabo sus evaluaciones.

- Incluir análisis estadísticos sobre los resultados de los ejercicios facilitando el trabajo de los terapeutas.

BIBLIOGRAFÍA

Confuted, “Using Quaternion to Perform 3D rotations”, Disponible en: http://www.cprogramming.com/tutorial/3d/quaternions.html (Última consulta: 16 de mayo de 2016)

De la Fuente, Pablo. “Planificación y Control de Proyecto Software” (Tema 2) & “Modelo de proceso de desarrollo” (Tema 3), material de estudio para la asignatura *Planificación y Gestión de Proyectos de Software*, 2015-2016, Universidad de Valladolid.

ETC Internal Unity3D Wiki, “Microsoft Kinect - Microsoft SDK” Disponible en: http://wiki.etc.cmu.edu/unity3d/index.php/Microsoft_Kinect_-_Microsoft_SDK (Última consulta: 12 de abril de 2016)

Gil Romero, Eva. “UN ESTUDIO ACERCA DEL DESARROLLO DE VIDEOJUEGOS MEDIANTE EL MOTOR GRÁFICO UNITY 3D” Disponible en: https://uvadoc.uva.es/handle/10324/5853 (Última consulta: 30 de junio de 2016)

Jacobson, Ivar & Booch, Grady & Rumbaugh, James. “The Unified Software Development Process” (1999)

Kapandji, Adalbert (2006). Fisiología Articular Tomo 1 6ed. Madrid: Panamericana

Koch, Jeff & MicrosoftLearningExperiences, “Programming Kinect for Windows v2: (04) Using Kinect with Other Frameworks or Libraries” Disponible en: https://channel9.msdn.com/Series/Programming-Kinect-for-Windows-v2/04 (Última consulta: 8 de junio de 2016)

Laguna, Miguel Ángel. “Modelado de requisitos” (Tema 2) & “Casos de uso” (Tema 3) & “Modelado del Dominio” (Tema 5), material de estudio para la asignatura *Modelado de Sistemas Software*, 2014-2015, Universidad de Valladolid.

López Gómez, Miguel Ángel.“La gestión de la exposición al riesgo y riesgos cambiarios” Disponible en: http://ocw.uc3m.es/economia-financiera-y-contabilidad/financiacion-internacional/material-de-clase-1/Tema12.-Gestion-Exposicion.pdf/view (Última consulta: 22 de mayo de 2016)

Marcal, Montserrat, “Características Kinect 2” Disponible en: http://www.kinectfordevelopers.com/es/2014/01/28/caracteristicas-kinect-2/ (Última consulta: 7 de marzo de 2016)

MakeHuman, “MakeHuman” Disponible en: http://www.makehuman.org/ (Última consulta: 13 de junio de 2016)

Microsoft MSDN, “Documentación Kinect SDK 2.0” Disponible en: https://msdn.microsoft.com/en-us/library/dn799271.aspx (Última consulta: 16 de junio de 2016)

Nystrom, Robert. “Game Programming Patterns” Disponible en: http://gameprogrammingpatterns.com/ (Última consulta: 27 de junio de 2016)

Pete D, “Avateering with Kinect V2 – Joint Orientations” Disponible en: http://peted.azurewebsites.net/avateering-with-kinect-v2-joint-orientations/ (Última consulta: 25 de abril de 2016)

Pixabay, “Imagen digital” Disponible en:
https://pixabay.com/es/ (Última consulta: 26 de mayo de 2016)

Rumen.F, “Kinect v2 Tips, Tricks and examples” Disponible en: https://rfilkov.com/2015/01/25/kinect-v2-tips-tricks-examples/#t1 (Última consulta: 3 de julio de 2016)

Sánchez Cabeza, Ángel. “Reaprendizaje motor orientado a tareas” (Tema 15), material de estudio para la asignatura *Terapia ocupacional*, Universidad de Rey Juan Carlos.

sqlite-net, “sqlite-net” Disponible en:
https://github.com/praeclarum/sqlite-net#sqlite-net (Última consulta: 27 de junio de 2016)

TarHead Studio, “Entity Component System” Disponible en:
http://tarheadstudio.com/entity-component-system/ (Última consulta: 27 de junio de 2016)

Tejero de Pablos, Miguel Ángel “Estudio de las posibilidades del sensor Kinect para la rehabilitación motora de pacientes hemipléjicos”, proyecto de fin de master de ingeniería informática, 2015, Universidad de Valladolid.

Unity, “Manual de Unity” Disponible en:
http://docs.unity3d.com/es/current/Manual/ (Última consulta: 14 de junio de 2016)

Unity, “Unity Assert Store” Disponible en:
https://www.assetstore.unity3d.com/en/ (Última consulta: 8 de julio de 2016)

Unity, “Entity system architecture with Unity - Unite Europe 2015” Disponible en:
https://www.youtube.com/watch?v=1wvMXur19M4 (Última consulta: 27 de junio de 2016)

Vaca Rodríguez, César & González Ferreras, César. “Análisis de Algoritmo” (Tema 1) & “Diseño de Algoritmos” (Tema 2), material de estudio para la asignatura *Análisis y Diseño de algoritmo*, 2014-2015, Universidad de Valladolid.

Wikipedia, “Entity component system” Disponible en:
https://en.wikipedia.org/wiki/Entity_component_system (Última consulta: 28 de junio de 2016)

Wikipedia, “Quaternion” Disponible en:
https://en.wikipedia.org/wiki/Quaternion (Última consulta: 11 de julio de 2016)

Wikipedia, “Unity” Disponible en:
https://es.wikipedia.org/wiki/Unity_(software) (Última consulta: 3 de julio de 2016)

ZJP, “[Windows] UDP Voice Recognition Server” Disponible en: https://community.unity.com/t5/Life-of-a-Unity-Game/Windows-UDP-Voice-Recognition-Server/td-p/1367797 (Última consulta: 7 de marzo de 2016)

APÉNDICE A – UNITY

Este apéndice está dedicado a explicar detalladamente el motor de videojuego que sea utilizado en este proyecto, Unity.

1. Proyecto

Unity está preparado tanto para el desarrollo de un proyecto 2d como para el de un proyecto 3d, incluso una combinación de ellos. La elección entre un proyecto 2d o 3d solo afecta a los ajustes del editor de Unity, por ejemplo, el modo en el que son importados los recursos por defecto.

Un proyecto de Unity contiene recursos y carpetas que componen la aplicación y está estructurado de la siguiente manera en el disco:

- **Assets:** Contiene todos los recursos añadidos por el desarrollador para su aplicación (Recursos gráficos, sonidos, scripts, etc)

- **Librerías:** Contiene todos los datos relacionados con la configuración del proyecto, además de los datos auto-generados por Unity para la caché de imágenes, metadatos y otros ficheros utilizados por el editor.

2. Interfaz

La interfaz de Unity se compone de varias ventanas interaccionadas, cada una con su funcionalidad específica que se explicarán en detalle a continuación, para hacer que el desarrollo sea lo más visual e intuitivo posible para el desarrollador.

Al igual que muchas aplicaciones de hoy en día se puede modificar la interfaz de Unity para que resulta más familiar para el desarrollador. Todas las ventanas se pueden redimensionarse, cambiarse de posición y ocultarse. Existe también muchas ventanas que, por defecto, están ocultas como la ventana de animation, la ventana de profiler …

Se puede acceder a estas ventanas a través de la opción de menú Windows. Además, es posible guardar las configuraciones personalizadas para cada momento del desarrollo a la hora de animar, editar escena …

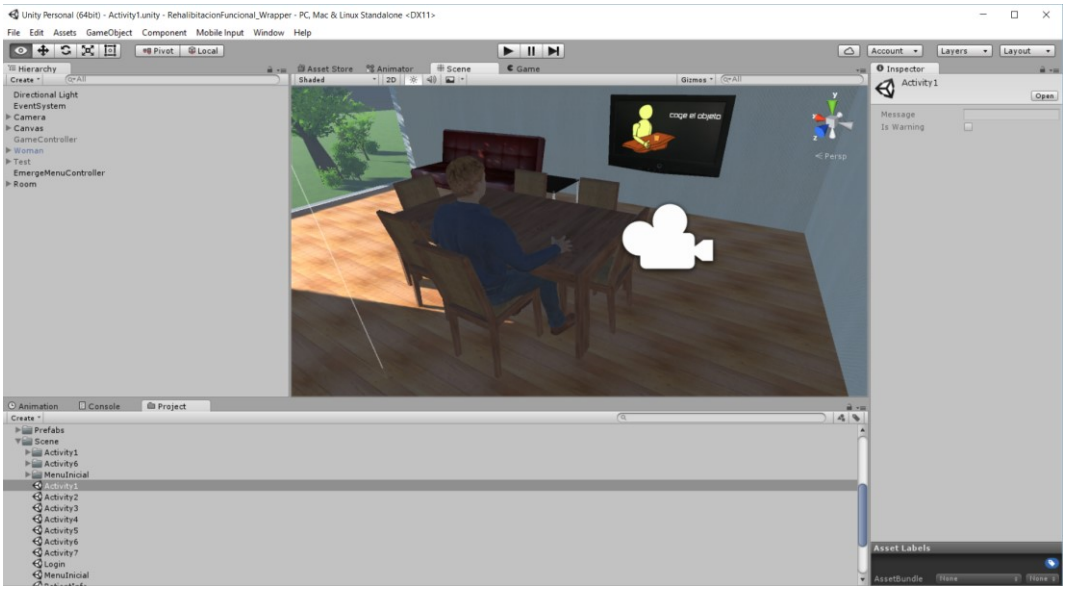

Figura 65: *Interfaz de Unity*

2.1. vista de proyecto

Desde la vista del proyecto se pueden visualizar los Assets de cada proyecto. Todos los componentes creados e importados, como modelos 3D, texturas, efectos de sonido, música, etc.

Debido a que una aplicación completa contiene muchos Assets, es imprescindible crear una buena estructura de carpetas para clasificar los diferentes recursos del proyecto.

Haciendo clic sobre el botón “Create” de la parte superior de la ventana se muestra una lista desplegable con varias opciones, entre ellos, crear carpetas, scripts, materiales, animaciones … y clic sobre uno de los ficheros o carpetas, se puede eliminar o copiar el elemento seleccionado.

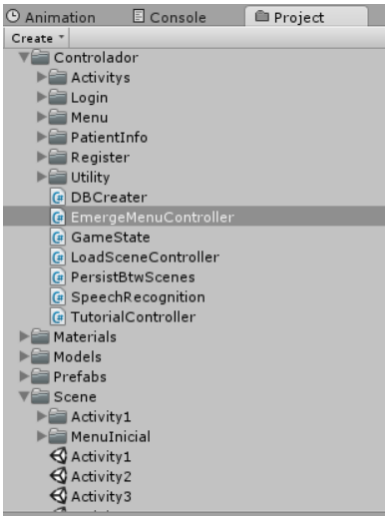

Figura 66: *Ventana de proyecto*

2.2. Editor de escena

Desde la ventana de editor de escena se pueden colocar los nuevos objetos arrastrando sobre ella, entonces se puede posicionarlos, rotarlos y escalarlos de forma gráfica y es posible recorrer el entorno 3d mediante la combinación de teclado “w”, “s”, “a” y “d” con la ayuda del ratón para rotar la cámara.

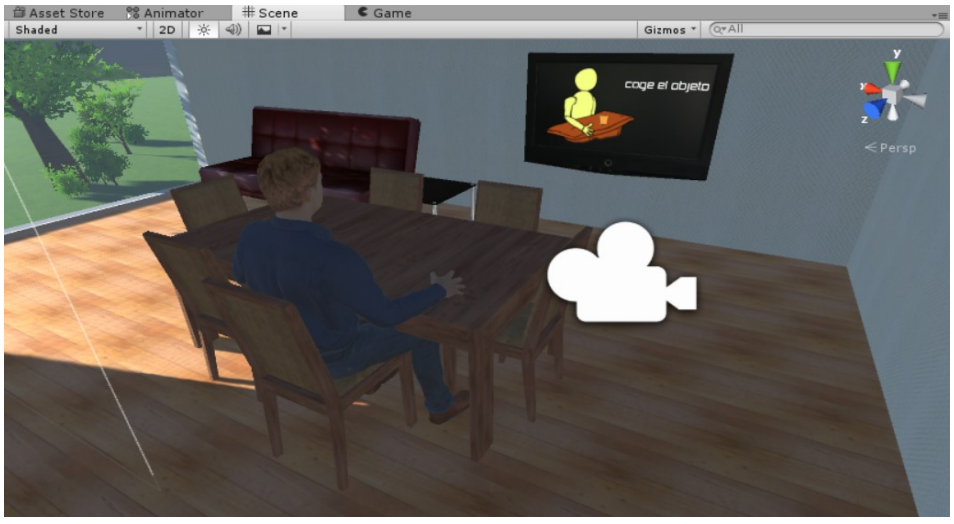

Figura 67: *Ventana de escena*

Por defecto, la vista de escena se muestra en una perspectiva 3D, pero se puede cambiar a una perspectiva 2D, que es muy útil para el desarrollo de una aplicación 2D o editar la interfaz de usuario.

Se puede aplicar varios filtros a la ventana de escena:

- **Shading Mode**: en el render mode se puede escoger entre Shaded (Representar las texturas), Wireframe (Solo muestra las mallas), Shaded Wireframe (Representar las texturas y mostrar las mallas sobre ellas).

- **Miscellaneous:** en esta opción se puede alternar entre Shadow Cascades (muestra la dirección de las luces y sombra), Render Paths (muestra cada objeto de la escena mediante un código de colores), Alpha channel (Representar con el color alpha),Overdraw (Representar los objetos de forma transparente con siluetas para que de esta manera se puedan observar los objetos ocultados detrás de otros) y MipMaps (muestra el tamaño de textura ideales mediante los colores.

- **Deferred:** Los modos de esta opción permiten visualizar cada elemento del G-Buffer (albedo, especular, suavidad y normal).

- **Global illumination:** Los modos de esta opción ayuda a los desarrolladores visualizar y gestionar la iluminación del sistema.

Además, en la ventana escena se dispone de herramientas visuales para modificar la posición, rotación y escalado de cada objeto en el momento que un objeto es seleccionado.

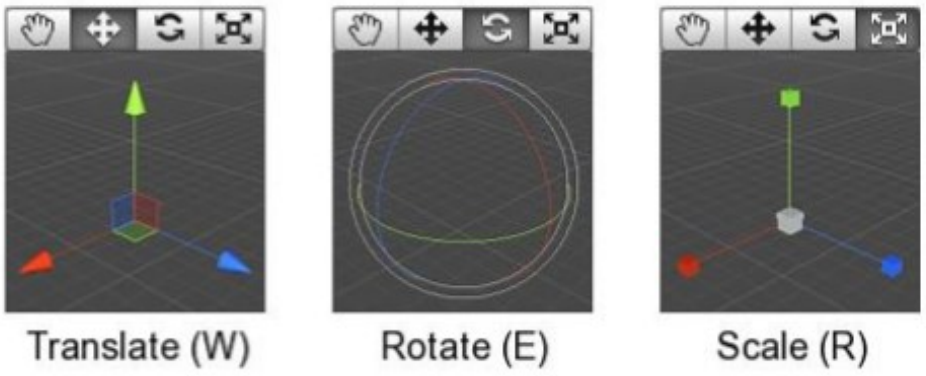

Figura 68: *Gizmo*

2.3. Ventana de Jerarquías

En la ventana de jerarquías se presenta la lista de los componentes incluidos en la escena. Esta lista se actualiza constantemente en cuanto se añade un nuevo objeto a la escena, el objeto se reflejará enseguida en la lista. Además, se puede duplicar, eliminar y crear objetos directamente desde esta ventana.

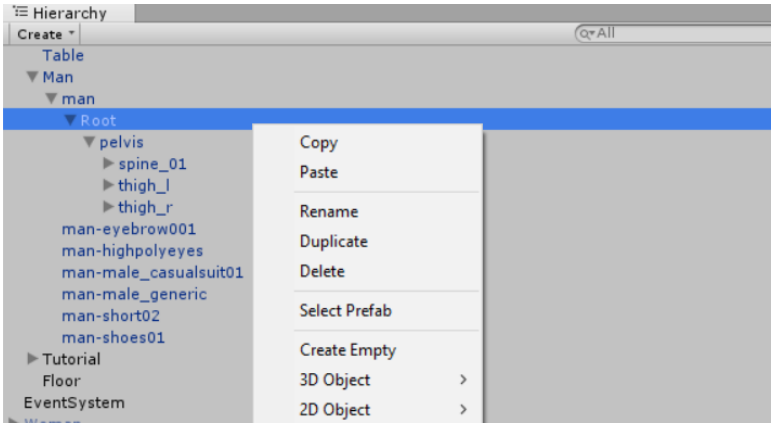

Figura 69: *Ventana de Jerarquías*

2.4. Inspector

En la pestaña de inspector se muestran los componentes y los parámetros del objeto seleccionado. Se pueden modificar los valores de los parámetros del objeto seleccionado directamente desde la pestaña de inspector.

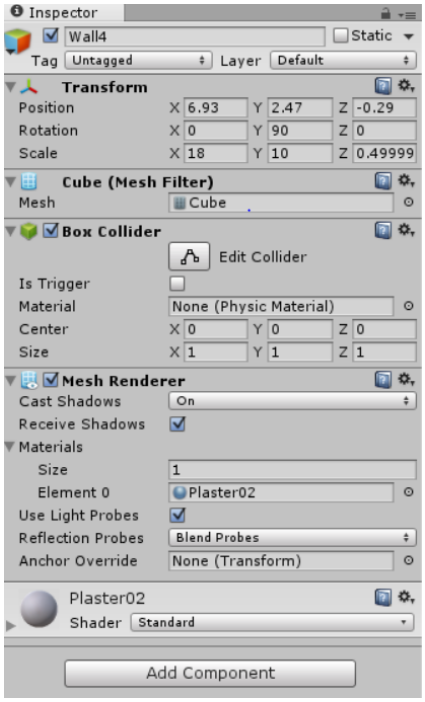

Figura 70: *Panel de inspector*

2.5. visualización del juego

En la ventana del juego se puede ver en directo la ejecución de la aplicación desarrollada hasta el momento, incluso gracias a la estructura basado en el componente de Unity es posible cambiar valores de parámetros de algún objeto en la ventana de inspector y observar su efecto inmediatamente en esta ventana durante la ejecución.

2.6. Consola

En la ventana de consola se registran todas las salidas del programa como los fallos o eventos que se produzcan durante la ejecución o errores críticos de los scripts fuera de ejecución. Lo que la hace extremadamente útil para localizar fallos en los scripts.

2.7. Animación

Desde la ventana de animación se pueden editar animaciones de los objetos.

3. Conceptos básicos de Unity

- **GameObjects:** El GameObject es la clase base de todas las entidades de la escena, cada escena está compuesto por un conjunto de GameObjects.

  Un GameObject básico o vacío contiene tres componentes: un “Transform” que sirve para posicionar, rotar y escalar el objeto en el espacio 3d mediante valores locales y globales; un “Layer” para indicar la capa física perteneciente del objeto con la finalidad de gestionar la colisión con otros objetos; Un “tag” cuya funcionalidad principal es diferenciarse de otros objetos, muy útil a nivel de programación para identificar determinados objetos específicos.

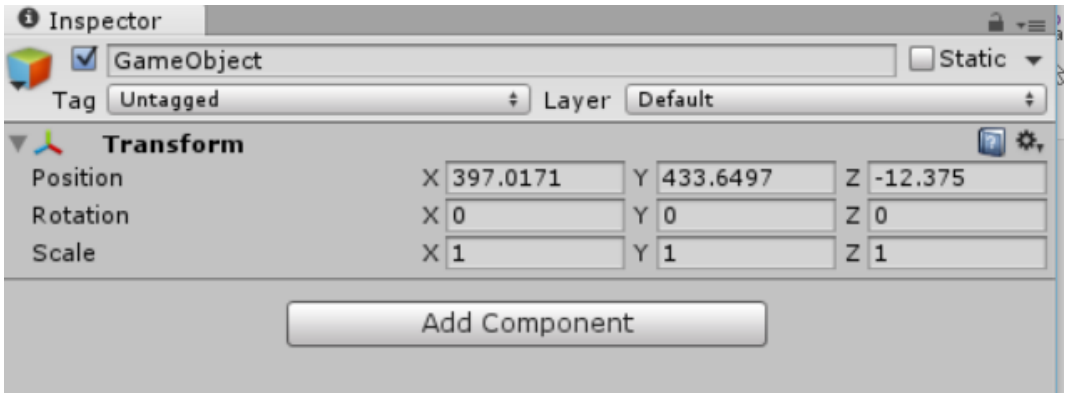

Figura 71: *GameObject vacío*

Una de las utilidades del GameObject es que se puede crear a partir de ella objetos complejos mediante la adición de los componentes adoptándoles propiedades físicas, gráficas e incluso comportamientos específicos con scripts propios.

- **Asset:** Un Asset es la representación de cualquier elemento que se puede utilizar en desarrollo del juego o proyecto. Un Asset puede ser un recurso creado mediante herramientas externas, tales como un modelo 3d, un archivo audio, una textura, un video, etc. También existen otros que son creados dentro del motor, como una animación de un objeto, material o texturas básicas. Formatos que permite Unity:

  - Gráficos 3D: Maya, Cinema 4D, 3ds Max, Cheetah3D, Modo, Lightwave, Blender, .FBX, .dae, .3DS, .dxf y .obj.

- Gráficos 2D: PSD, TIFF, JPG, TGA, PNG, GIF, BMP, IFF, PICT, incluso los PSD multicapa y TIFF multicapa.

- Audio: WAV, AIFF, MP3, OGG

- **Prefab:** Muchas veces entre diferentes escenas se comparten un mismo GameObject con una configuración similar o idéntica, configurar un GameObject desde uno básico puede ser una tarea farragosa y poco eficiente. Para solventar este problema Unity incorpora el concepto de Prefab, que es un GameObject con una configuración de componentes y valores de parámetros personalizados y reutilizables en diferentes escenas. Además, Prefab actua como si fuera una plantilla, a partir de ella, se pueden crear variaciones de instancias en la escena sin afectar a la configuración de Prefab.

- **Escena:** Cada escena contiene un conjunto de GameObjects configurados de la manera que desea el desarrollador. Las escenas junto con sus configuraciones de los GameObjects que las componen, son almacenadas en la carpeta del proyecto como un fichero de tipo “Unity Scene File” que se pueden cargar y editar posteriormente. Las escenas pueden ser usadas para crear un menú principal, niveles individuales y cualquiera otro tipo de escena.

  En el momento de despliegue hay que seleccionar y cargar todas las escenas que van a forma parte de la aplicación final, así como el orden de éstas. De acuerdo con esta selección Unity generará la aplicación final compilando solo los ficheros necesarios para la correcta ejecución de las escenas seleccionadas.

4. Importar y exportar Assets

Importar los recursos externos a Unity es extremadamente fácil y dispone de varias formas de hacerlo: Se puede arrastra simplemente el fichero hasta la ventana del proyecto y Unity se encargará automáticamente del resto;

También se puede guardar directamente en el directorio de la carpeta del proyecto y Unity será capaz de leerlo, se dará cuenta de nuevos cambios en el proyecto y volverá a importar los archivos si fuese necesario;

En cuando a la importación de una colección de assets se recomienda el uso de Unity Packages, los Unity Packages son colecciones de archivos y datos de proyectos comprimidos de forma similar a un archivo Zip, manteniendo su estructura original de los directorios y los meta datos de los assets(Ajustes de la importación y enlaces con otros assets).

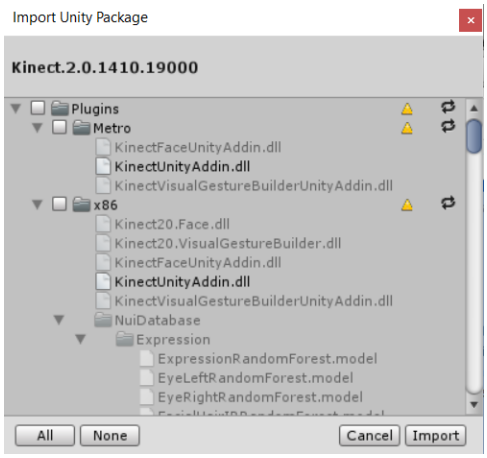

Figura 72: *Ventana de importación*

Generalmente para importa un assert hace falta una configuración de importación antes de su uso, a excepción de los assets importados mediante Unity Packages. Cada tipo de asset soportado por Unity tiene su propio conjunto de ajustes de importación (ver **Figura 73** y **Figura 74¡Error! No se encuentra el origen de la referencia.**) y la configuración del conjunto de ajuste de importación afectará directamente a la forma en que aparezca el asset en el proyecto.

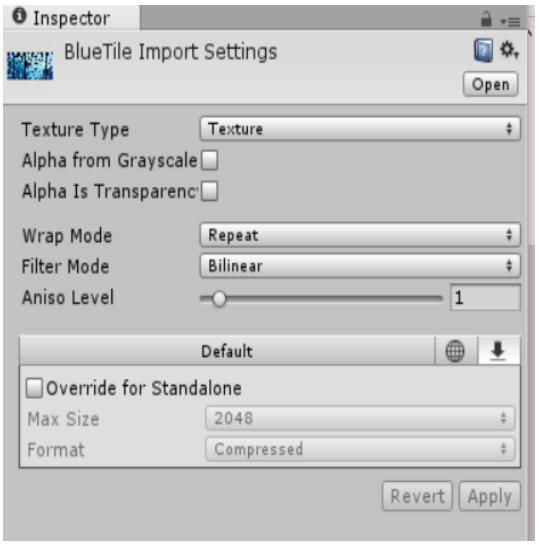

Figura 73: *Ventana de configuración de importación de una imagen*

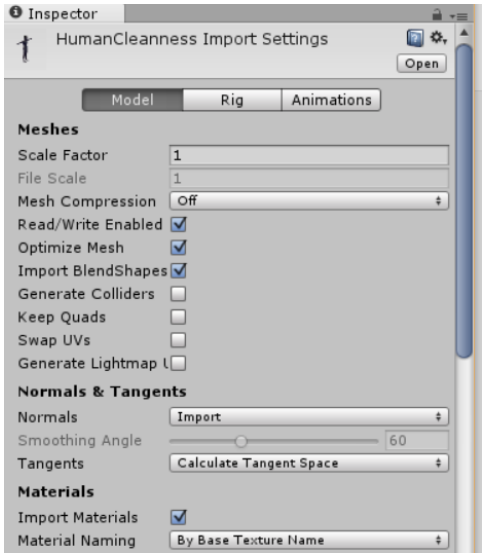

Figura 74: *Ventana de configuración de importación de textura 3d*

Al igual que podemos importar, también podemos exportar estos paquetes para poder ser importados después en otros proyectos de Unity.

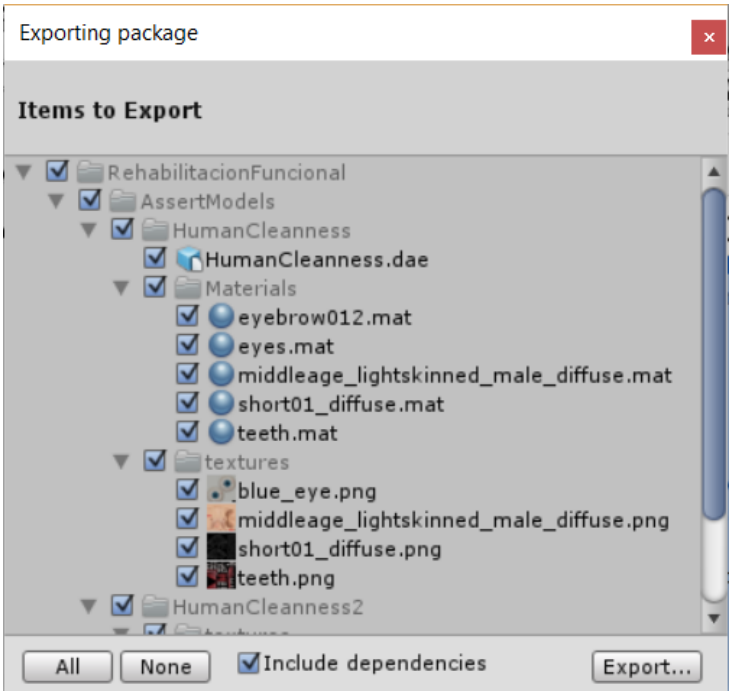

Figura 75: *Ventana de exportación*

5. Script

Unity fue concebido con la idea de componentes en la mente. Un proyecto de Unity está compuesto por un conjunto de objetos que a su vez están formados por un conjunto de componentes. A esta estructura se la puede llamar arquitectura basada en componentes. Esta arquitectura hace que Unity sea sumamente visual permitiendo a los desarrolladores observar todo en tiempo real, esto quiere decir, que el desarrollador puede editar los objetos y ver los cambios en directo sin la necesidad de recompilar la aplicación.

Esta arquitectura basada en componentes existe un componente base para todos los objetos de escena que es el componente GameObject. Además, se puede observar en la Figura 76 la adición de los componentes al GameObject que formará un objeto complejo con propiedades y comportamientos específicos.

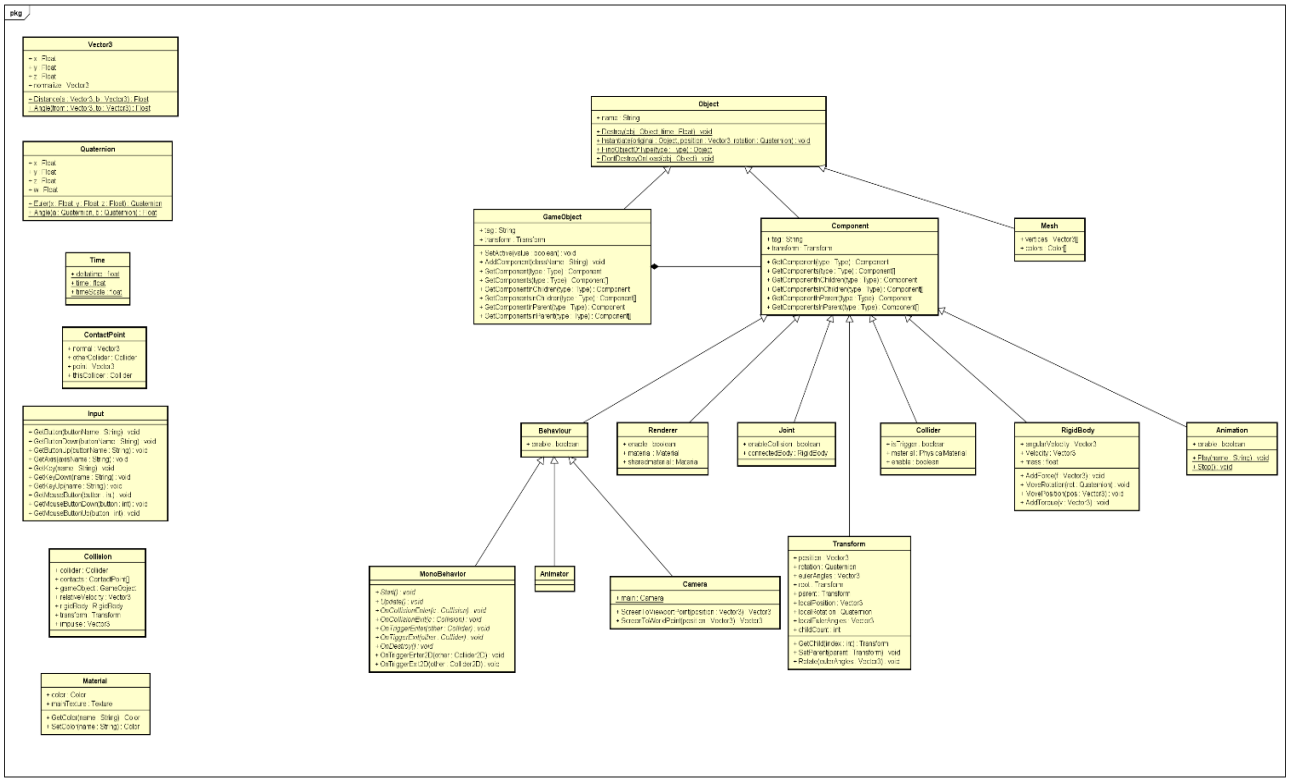

Figura 76: *Diagrama de clases de Unity*

Descripción breve de las clases de los componentes de la Figura 76:

- **Component:** La clase base de todos los componentes vinculados a los GameObjects.
- **Transform:** Cada objeto de la escena tiene un Transform que sirve para guardar y manipular la posición, rotación y escalar del objeto. Además, un Transform puede tener un padre que permite aplicar la posición, rotación y escalar de forma jerárquica.
- **RigidBody:** Agregando un RigidBody a un objeto, acondiciona sus movimientos bajo los efectos de motor de físicas de Unity (PhysX). Como la gravedad, la aceleración de forma realista, reacción ante las colisiones con otros objetos…
- **Animation:** Sirven para almacenar y reproducir una animación.
- **Collider:** Clase base de todos los Colliders, proporciona una estructura geométrica al objeto que interviene en las colisiones procesadas por el motor de físicas.
- **Joint:** Se puede juntar dos RigidBody de dos objetos mediante Joint e imponer ciertas restricciones de movimiento de la articulación formada.
- **Renderer:** Es el que se encarga de representar un objeto en la pantalla.
- **Camera:** Los usuarios pueden observar el espacio 3d a través de este objeto.
- **Animator:** Es la interfaz para controlar el sistema de las animaciones.
- **Behaviour:** Es un componente que se puede activar y desactivar.

- **MonoBehaviour:** Es la clase base de todos los scripts, se puede redefinir las funciones abstractas de la clase MonoBehaviour como Start() que es llamado después de la creación de objetos y antes del primer Update() o como Update() que es una función que es invocada en cada frame. Aparte de estas dos funciones que son las más usadas existen otras como OnTriggerEnter(), OnTriggerExit(), OnDestroy() …

6. Incorporación de Kinect a Unity

Para incorpora Kinect dentro de Unity solo se necesita importar un Unity Package oficial proporcionado por Microsoft que contiene los scripts básicos de control de Kinect. Además de los scripts dentro del paquete, también tiene dos escenas como ejemplos. Una vez importado estos scripts ya se pueden desarrollar aplicaciones de Kinect con Unity de forma similar a otros proyectos de Kinect normales.

APÉNDICE B - MANUAL DE USUARIO

1. Requisito software

- Sistema operativo Windows 8 o Windows 10
- Directx 11 o Directx 12

2. Requisito hardware

- Procesador: Intel Core i7-4750HQ 2.00 GHz
- Tarjeta gráfica: NVIDIA GeForce® GTX 950M
- Memoria: 2 GB RAM
- Disco duro: 1 GB de espacio libre

3. Descripción de la aplicación

Esta aplicación se compone de un conjunto de actividades de la vida diaria y mini juegos. Las actividades de la vida diaria tienen como objetivo evaluar sus capacidades funcionales, mientras los mini juegos le servirán como ejercicios.

Aparte de las actividades y juegos, este programa también le permitirá consultar sus datos personales y los resultados de los ejercicios realizados, así como las gestiones de estos datos.

4. Manual de usuario de aplicación

4.1. Inicio

La Figura 77 es la pantalla inicial de la aplicación y se muestra nada más abrir la aplicación. En esta pantalla podemos realizar dos acciones, si es primera vez que usamos la aplicación debemos pulsar el botón de “Registrar” para registrarnos (ver apartado 4.2). Si ya hemos registrado en el sistema podemos introducir directamente el Nick y pulsar el botón “Aceptar” para iniciar nuestra sesión (ver apartado 4.3).

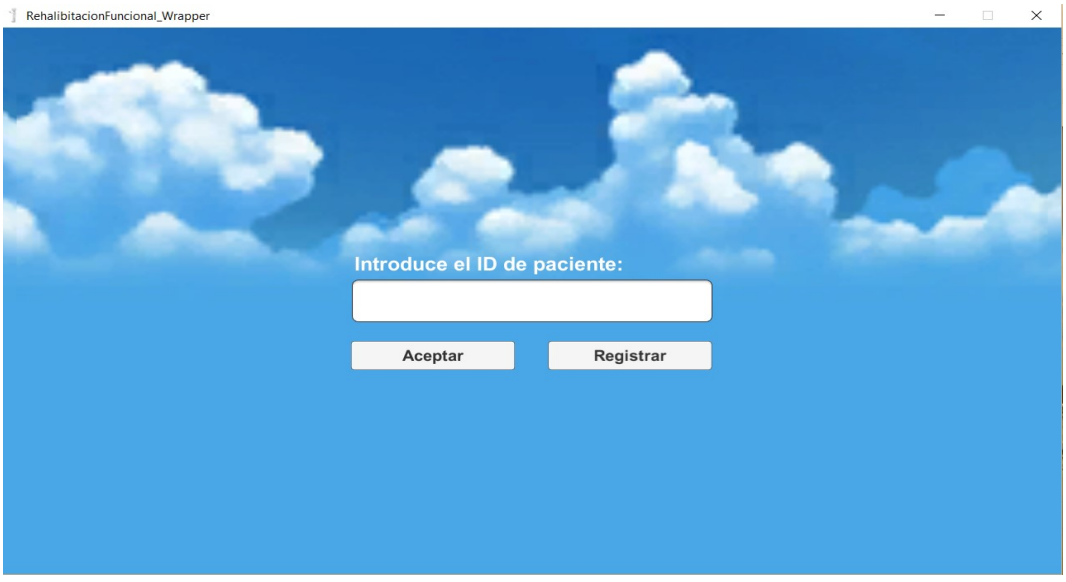

Figura 77: *Pantalla de Login*

4.2. Registrar

Si nos hemos elegido en la pantalla anterior la opción de registrar se nos mostrara una pantalla como la de Figura 78. En esta pantalla tenemos que rellenar todos los campos obligatorios (Los campos en los que aparece el símbolo * y los que no lo llevan son opcionales puede dejo vacío):

- Nombre de identificación o Nick * (Se recomienda usa el DNI)
- Nombre *
- Apellido*
- Dirección
- Población
- Código postal
- Número de teléfono*
- Diagnostico *
- Zona/s de lesión/lesiones*
- Fecha de lesión/lesiones*

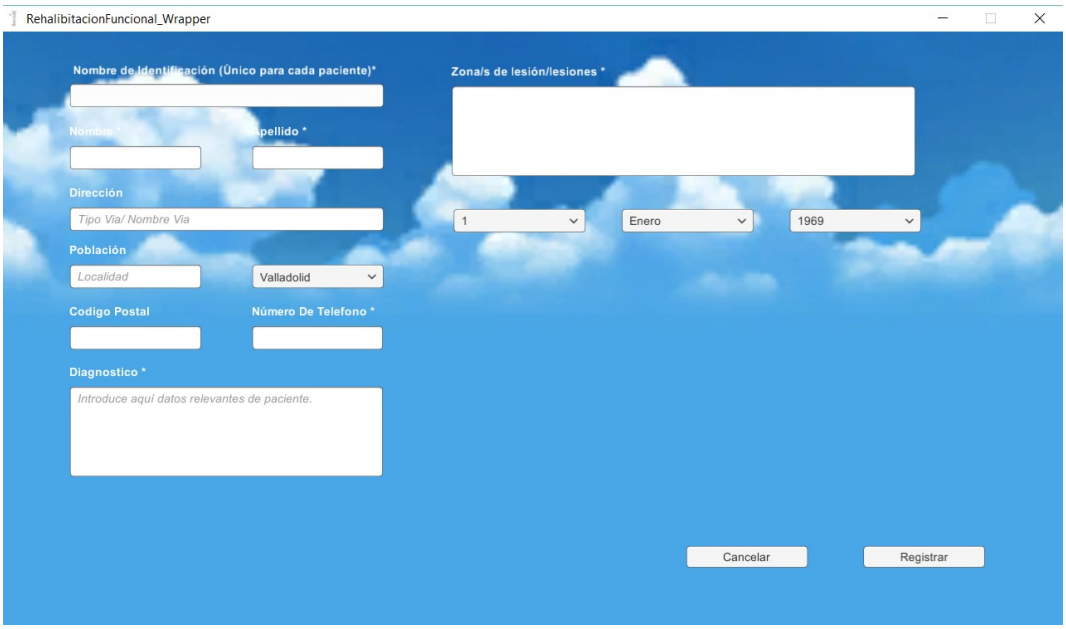

Figura 78: *Pantalla de registrar*

Una vez que hemos rellenado todos los campos obligatorios podemos pulsar el botón de “Registrar” para completar este proceso. Si hemos hecho todos correctamente nos devolverá a la pantalla de inicio, y si no nos mostrará las siguientes ventanas emergentes:

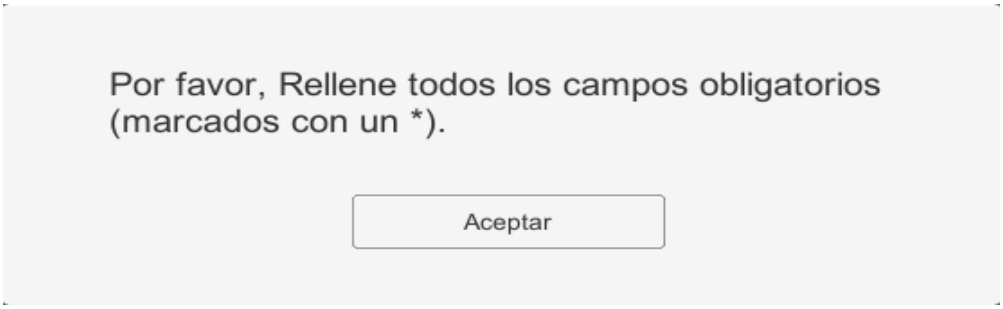

Figura 79: *Ventana emergente 1*

Si aparece la ventana de Figura 79, comprueba que no hemos dejado ningún campo/s obligatorio/s sin rellenar.

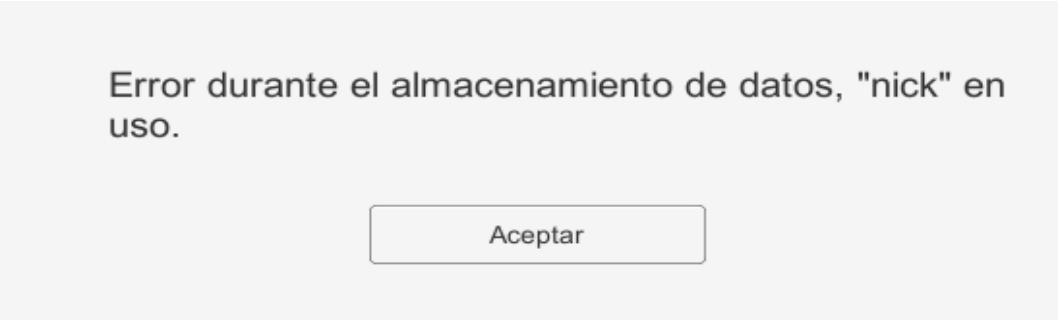

Figura 80: *Ventana emergente 2*

Si aparece la ventana de Figura 80, tenemos que registrar con otro Nick ya que el introducido está en uso.

4.3. Menú inicial

Una vez que hemos identificado ante el sistema nos llevará a la escena de Menú inicial, donde dispondremos de un menú con las siguientes opciones: “Iniciar”, “Consultar”, “Cerrar Sesión” y “Salir” (Figura 81). A continuación, detallaremos los servicios que nos proporciona cada una de estas opciones.

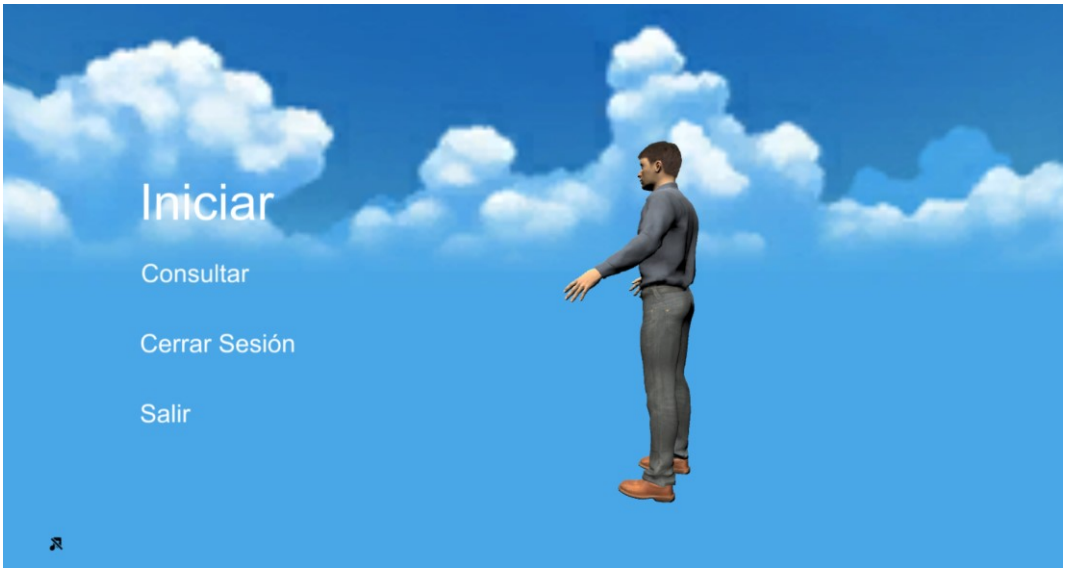

Figura 81: *Pantalla de menú inicial 1*

4.3.1. Iniciar

Si seleccionamos la opción “Iniciar” (Figura 81), nos mostrará el siguiente menú con los dos bloques de actividades: las actividades de la vida diaria y los mini juegos. Y la flecha que hay debajo nos da la opción de volver al Menú Inicial.

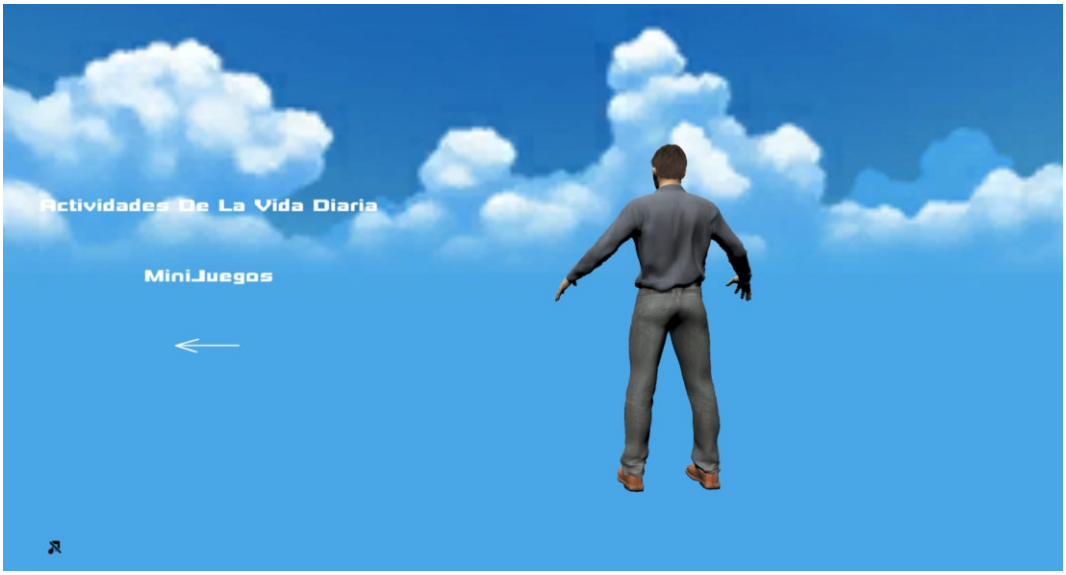

Figura 82: *Pantalla de menú inicial – Lista de Bloques*

Si seleccionamos el bloque “Actividad de la vida diaria” se nos mostrará las opciones de la **Figura 83**, y si seleccionamos el bloque “MiniJuegos” aparecerá las opciones de la **Figura 84**.En cuanto a las descripciones de estas opciones ver apartado 4.4.

Debemos tener cuidado con la opción “Lavabo” de la **Figura 83**, ya que para esta actividad necesitamos introducir las repeticiones de cepillados de dientes que deseamos realizar, si no nos pondrá por defecto 10 repeticiones.

Figura 83: *Pantalla de menú inicial – Bloque de actividad diaria*

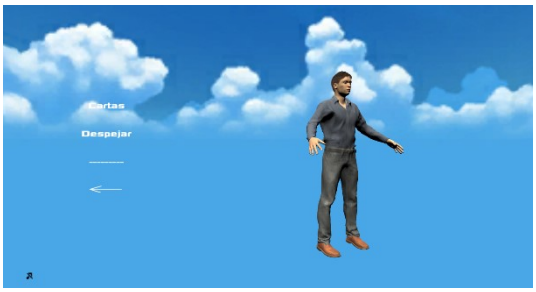

Figura 84: *Pantalla de menú inicial – Bloque de MiniJuegos*

4.3.2. Consultar

A seleccionar la opción “Consultar” nos mostrará los datos relativos al paciente identificado (Figura 85).

En la parte superior del panel izquierdo tenemos los datos personales del paciente, desde aquí podemos modificar directamente los datos del paciente a excepción de ID que es inmutable, una vez que hayamos realizado los cambios deseados podemos pulsar el botón “Guardar” situado justo debajo para guardar los cambios en el base de datos.

El botón “Eliminar” situado en el lado izquierdo de botón “Guardar” sirve para eliminar permanentemente todos los datos de pacientes.

En la parte inferior del panel izquierdo encontramos con la lista de las actividades realizadas. Si seleccionamos una de las actividades de la lista nos mostrará los detalles de la actividad en el panel derecho (Figura 86, Figura 87, Figura 88, Figura 89). Y si seleccionamos uno de los botones con el icono de papelera eliminaremos la actividad en cuestión.

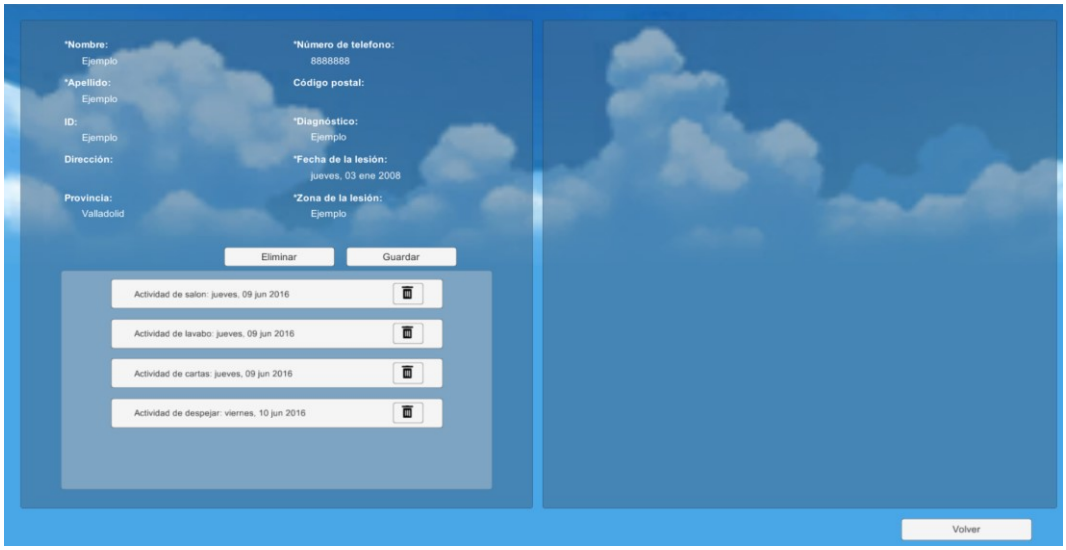

Figura 85: *Pantalla de Consultar*

Capturas de los resultados de las actividades:

- Resultado de la actividad de Salón

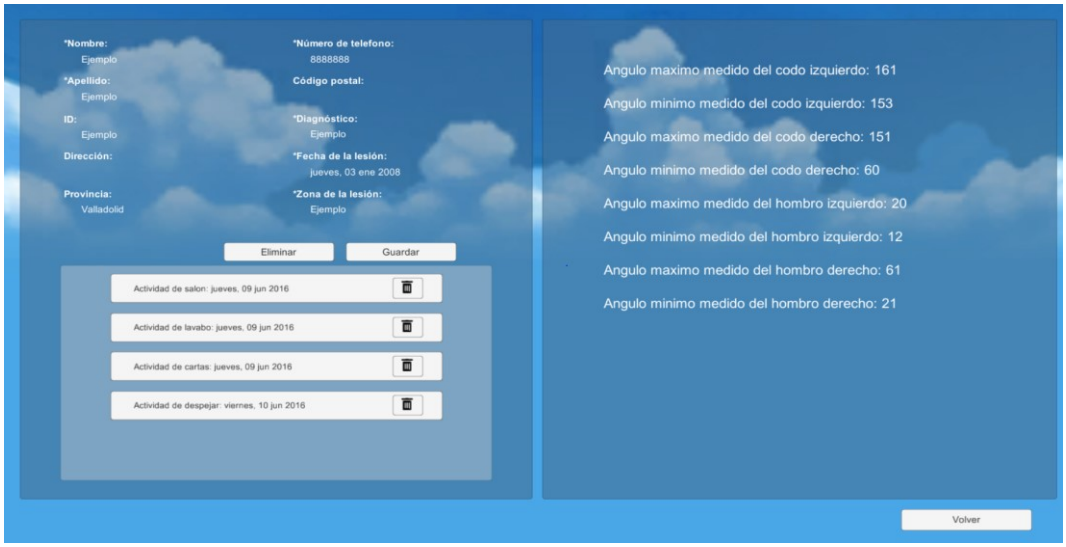

Figura 86: *Pantalla de Consultar con resultado de la actividad de salón*

- Resultado de la actividad de Lavabo

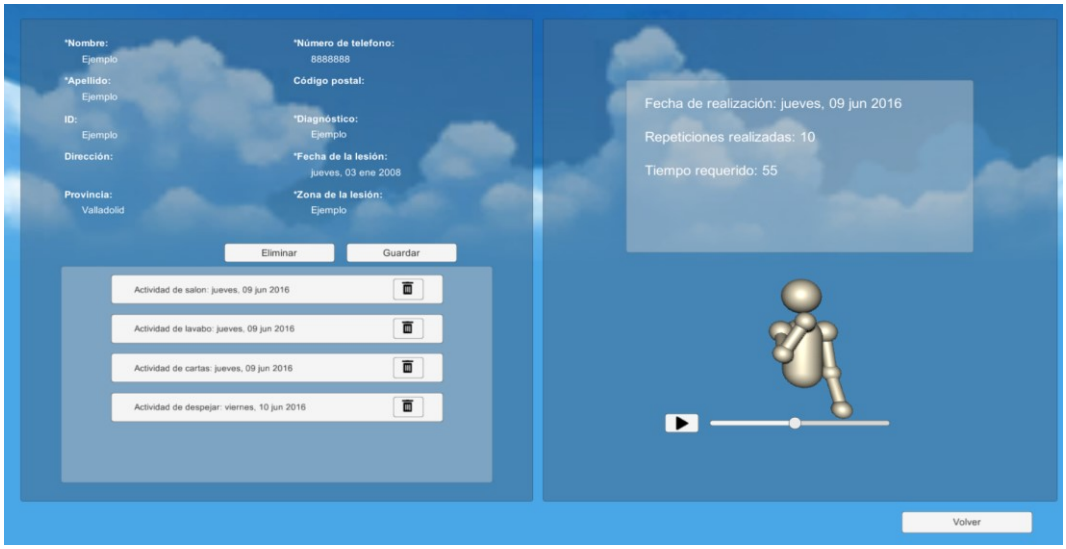

Figura 87: *Pantalla de Consultar con resultado de la actividad de Lavabo*

- Resultado de la actividad de Cartas

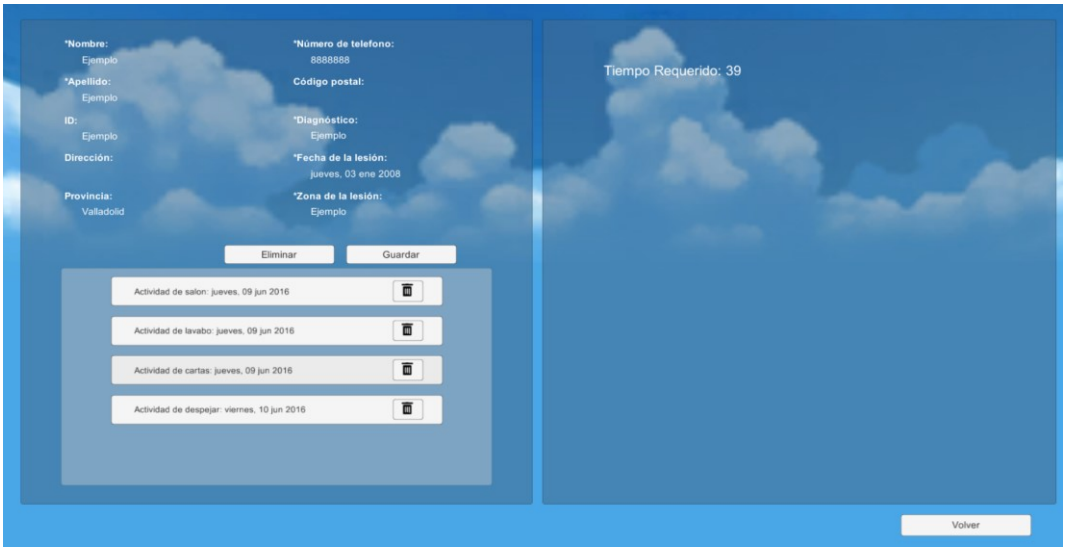

Figura 88: *Pantalla de Consultar con resultado de la actividad de Cartas*

- Resultado de la actividad de Despejar

Figura 89: *Pantalla de Consultar con resultado de la actividad de Despejar*

En la Figura 87 tenemos los detalles de una actividad de “Lavabo”. Los detalles de tipo de actividad a diferencia de otras, disponemos de una grabación de las posiciones del paciente durante la realización de la actividad. Podemos parar y reproducir los movimientos del paciente mediante el botón situado debajo de la figura, también podemos parar la reproducción y rota la figura mediante la combinación de teclados “Q” y “E”.

4.3.3. Cerrar Sesión

La opción “Cerrar Sesión” nos permite cerrar la sesión actual del paciente y abrir una nueva.

4.3.3. Salir

Si seleccionamos la opción “Salir” nos cerrará la aplicación.

4.4. Actividades

A continuación, detallamos aquellas funcionalidades relacionadas con las actividades. Para poder acceder a una actividad tenemos que haber realizado correctamente los pasos del apartado 4.3.1. Iniciar.

4.4.1. Coordinación

En el inicio de todas las actividades nos pedirá que centralicemos. Para ellos tenemos que movernos hasta en el centro del campo de detección del sensor Kinect.

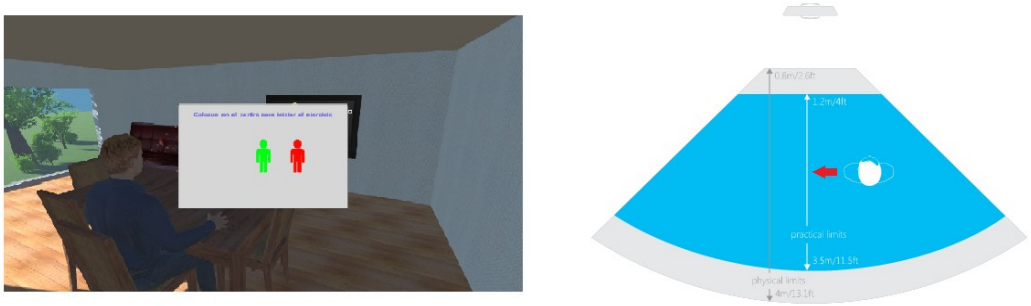

Figura 90: *Ventana de coordinación (izquierda), Rango de detectión del sensor Kinect (Derecho)*

4.4.2. Menú de emergencia

Disponemos de un menú de emergencia en todas las actividades, que aparecerá cuando pulsamos el teclado “ESC” para pausar la actividad.

Desde este menú podemos salir, cambiar y recargar la actividad, así como cambiar del jugador.

Si pulsamos de nuevo el teclado “ESC” el menú desaparecerá y reanudará la actividad que está en pausé.

Figura 91: *Ventana de menú de emergencia*

4.4.1. Actividad de Salón

El objetivo de esta actividad es beber con la taza que hay encima de la mesa. Para ello primero tenemos que coger la taza de café y luego llevarlo hasta la boca.

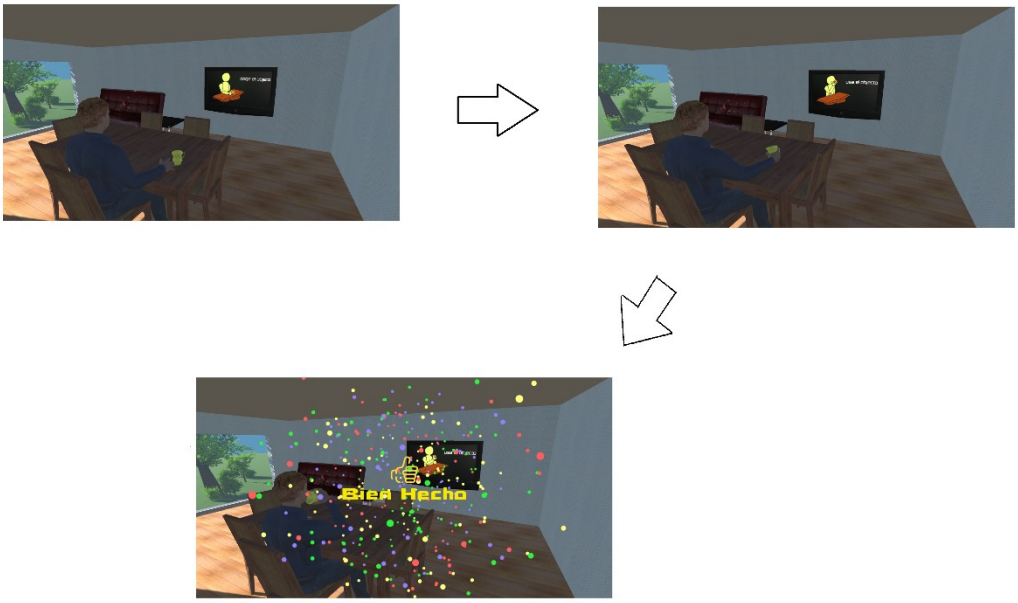

Figura 92: *Procedimiento de la actividad de Salón*

4.4.2. Actividad de Lavabo

En esta actividad tenemos que realizar el proceso de cepillado de los dientes, que consisten en cuatro pasos:

- Coger la pasta y el cepillo

- Aplicar la pasta sobre el cepillo
- Mojar el cepillo en el agua
- Cepillar los dientes

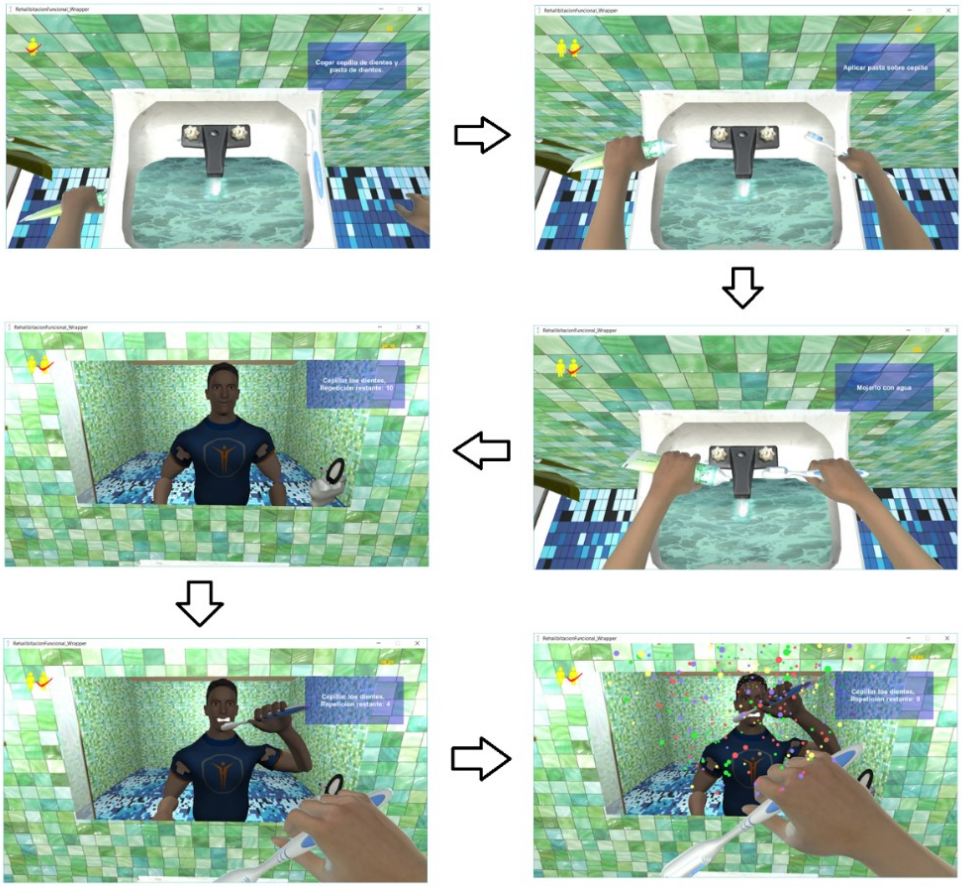

Figura 93: *Procedimiento de la actividad de Lavabo*

4.4.3. Actividad de Cartas

Para finalizar exitosamente esta actividad debemos eliminar todas las cartas de la pantalla, es decir, necesitamos seleccionar todas las parejas de las cartas de la pantalla. Para seleccionar una carta primero hay que situar nuestra mano encima de ella y una vez situado encima cerramos el puño para seleccionarla. Cuando hayamos seleccionado una carta, lo siguiente es buscar la pareja de ella y seleccionarla del mismo modo que la primera. Una vez seleccionado la pareja de cartas, las dos serán eliminadas.

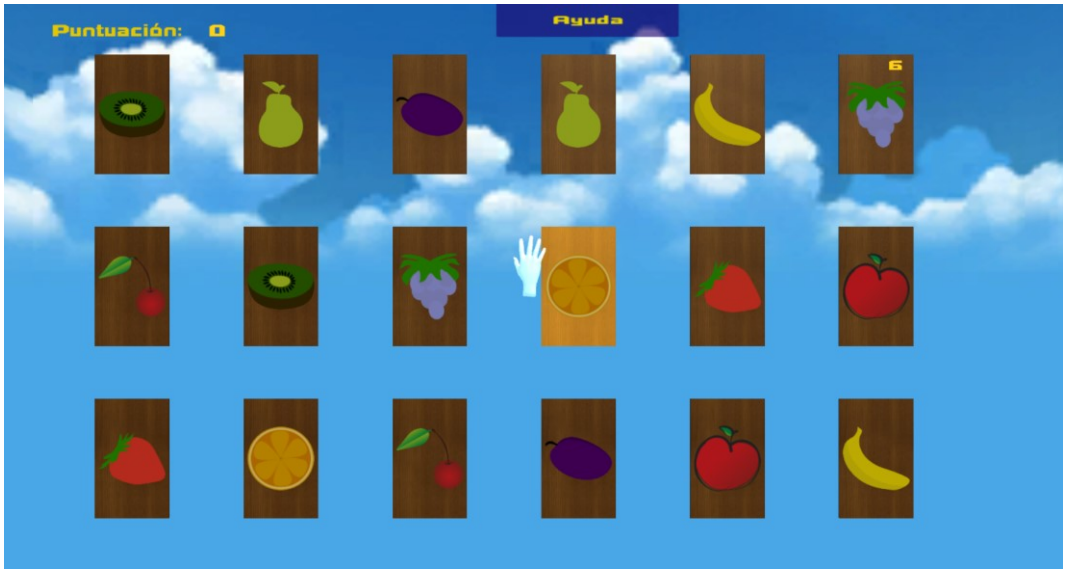

Figura 94: *Pantalla de la actividad de Cartas*

4.4.4. Actividad de Despejar

Antes de entrar en detalle con la realización de esta actividad, es conveniente que tengamos en cuenta que disponemos solo de 30 segundos para esta actividad.

Una vez que conocemos del tiempo disponible ya podemos comenzar con la explicación de la realización de la actividad. Si cerramos el puño en un punto determinado de la pantalla todos los cubos de este punto son eliminados y si mantenemos el puño cerrado mientras lo arrastramos, podemos observar que todos los cubos por los que hemos pasado también son eliminados. Además, si conseguimos encerrar las líneas de manera que aíslan los cubos de medio, todos los cubos encerrados son eliminados igualmente.

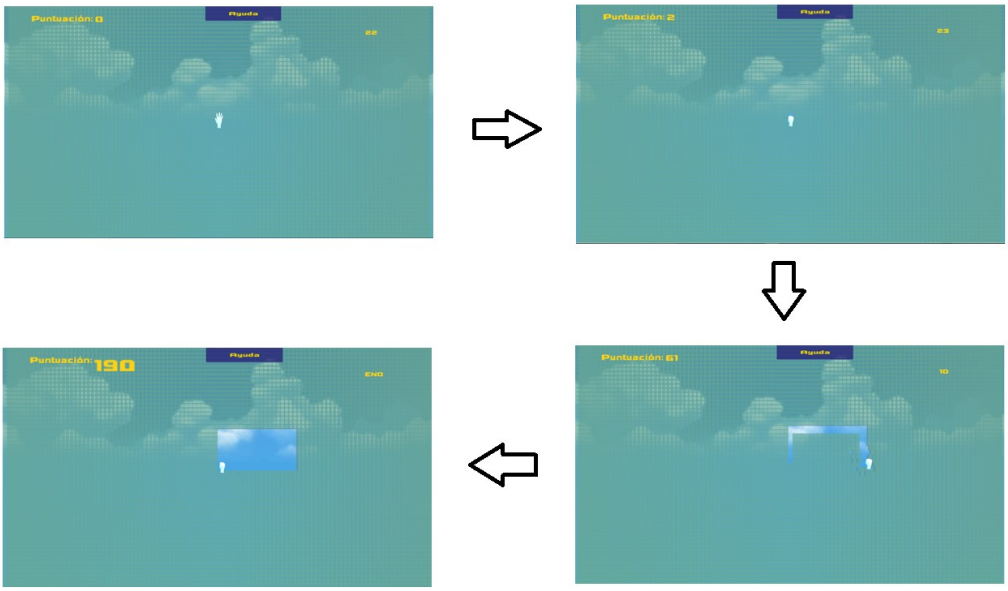

Figura 95: *Procedimiento de actividad de Despejar*