Universidad de Valladolid

Escuela de Ingeniería Informática

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

(Mención Ingeniería del Software)

Rehabilitación funcional de extremidades basada en Android

Autor:
Alberto Ceruelo Andrés

Tutor:
Miguel Ángel Laguna Serrano

Resumen

En este trabajo se ha desarrollado una aplicación que ayuda en la rehabilitación de pacientes mediante un tipo de terapia ocupacional que consiste en la realización de actividades cotidianas. Para ello, en este documento, se presenta el diseño y la implementación de dos aplicaciones. La primera, para ordenador personal, utiliza la plataforma Unity para mostrar y controlar las distintas actividades- La segunda, para teléfonos móviles inteligentes con sistema operativo Android, obtiene los valores proporcionados por los sensores incorporados en los mismos y los envía a la aplicación basada en Unity para que esta calcule y muestre el desplazamiento del dispositivo y por tanto de los movimientos del paciente.

Para la comunicación entre las dos aplicaciones anteriores se utiliza un servidor Node JS que las relaciona, pudiendo representar el movimiento obtenido en el móvil dentro del avatar que refleja las distintas actividades creadas en Unity.

Los distintos ejercicios que se pueden realizar están orientados a la rehabilitación del movimiento de hombro y codo, pudiendo llevar un control de los avances obtenidos gracias a la utilización de una base de datos en la que se almacenan los distintos resultados facilitando su seguimiento. La accesibilidad y facilidad de uso del sistema ayudan a tener un mayor control sobre los avances del paciente, así como una mayor independencia a la hora de realizar la rehabilitación.

Por tanto, podemos decir que el trabajo que a continuación se describe tiene estas tres características fundamentales que ayudan en la recuperación de los pacientes: apoyo en un sistema accesible, sencillez en su uso y almacenamiento de datos recogidos que informan sobre su evolución.

Tabla de contenido

Lista de figuras

Lista de tablas

Capítulo 1

1 Introducción

Esta aplicación se ofrecerá a los pacientes para que puedan realizar una terapia de rehabilitación funcional de forma presencial o remota, permitiendo a los terapeutas llevar un seguimiento de la evolución de los pacientes gracias al análisis de los datos recogidos.

La solución que se ha propuesto en este Trabajo de Fin de Grado es la sustitución del Kinect por un Smartphone con el sistema operativo Android, pudiendo realizar las mismas actividades, aunque de una manera ligeramente más acotada pero sensiblemente más independiente.

Constará de tres actividades, las dos primeras se encontrarán más orientadas a la rehabilitación mediante acciones cotidianas que se realizan en el día a día; por otra parte trataremos la rehabilitación mediante un mini-juego.

1.1 Descripción general de la aplicación

La aplicación tiene como objetivos principales simular, mediante una serie de ejercicios, actividades diarias en un entorno virtual 3d, mediante el uso de un avatar que representa y monitoriza los movimientos de los pacientes, almacenando un registro de los ejercicios realizados.

Por otro lado, se encuentra el minijuego como otro sistema de rehabilitación, diferenciándose en cierto modo de las anteriores actividades cotidianas.

El equipamiento para su desarrollo lo componen:

- El dispositivo móvil o Smartphone que, a través de una aplicación creada, se encarga de monitorizar los movimientos del paciente mediante los datos calculados por el sensor de rotación del móvil, para posteriormente enviarlos al servidor Node.

- El Ordenador que consta de:

    - La aplicación hecha en Unity, que representa los movimientos del paciente en un entorno virtual 3d usando un avatar humanoide, en forma de actividades, haciendo uso de los datos proporcionados por él móvil.

- El Servidor Node JS como punto de unión entre el móvil y el mencionado programa de Unity, encargándose de obtener los datos del primero y pasarlos al segundo.

Uno de los principales problemas encontrados fue la representación de los movimientos del paciente en la aplicación hecha en Unity, puesto que se trataba de plasmar lo que hace el paciente con el móvil en el avatar. Para solucionarlo se ha optado por el uso de los cuaterniones, elementos que se explicarán más en detalle en el capítulo 2 (punto 2.1 Cuaterniones). Tanto el sensor utilizado en el Smartphone como la plataforma Unity los usan para mostrar los movimientos, dado que estos se emplean habitualmente en la representación de orientaciones y rotaciones de objetos en tres dimensiones, permitiendo llevar los movimientos del usuario al avatar.

Para el almacenamiento de los datos obtenidos, se hace uso de SQLite como base de datos.

Todo se ha desarrollado tratando de obtener la mayor libertad por parte de los pacientes a la hora de realizar las distintas actividades. Para ello las interfaces se han realizado lo más intuitivas posibles y siempre estando acompañadas de las instrucciones apropiadas.

En el desarrollo de este trabajo se ha hecho uso del diseño de los avatares Unity ya realizados en el TFG de An Chen (“Evaluación del sensor Kinect v2 para rehabilitación funcional”).

1.2 Objetivos y motivación

El objetivo de esta aplicación es conseguir mediante la terapia ocupacional (Rehabilitación funcional), que abarca un conjunto de técnicas, métodos y actuaciones a través de actividades aplicadas con fines terapéuticos, prevenir y mantener la salud, favoreciendo la restauración de la función supliendo las deficiencias incapacitantes e intentando conseguir la mayor independencia del paciente en todos sus aspectos: mental, físico y social.

Todo esto se consigue mediante ejercicios con actividades básicas de la vida diaria, realizando tareas específicas de los miembros superiores: rango articular, fuerza, presa, pinza fina, pinza gruesa.

Además de ofrecer a los pacientes una terapia de rehabilitación lo más autónoma que sea posible, limitando a lo mínimo las asistencias de los terapeutas, se buscará una alternativa al desarrollo de actividades de rehabilitación que presentaba la cámara Kinect, tratando de lograr una mayor independencia de la que esta nos proporcionaba, sobre todo ante su discontinuidad por parte del fabricante impidiendo su uso en proyectos futuros.

Teniendo como motivación la oportunidad de poder utilizar como alternativa los dispositivos móviles, aparatos que nos ofrecen múltiples posibilidades en este campo gracias a la gran cantidad de sensores con los que cuentan y a la alta disponibilidad de estos por parte de los pacientes. También la ocasión de poder introducirme en el uso de herramientas orientadas a la creación de videojuegos y entornos 3D como es el caso de Unity, con el añadido de hacer posible la interacción entre ambas plataformas (Smartphone y Unity).

1.3 Metodología empleada

Para el desarrollo de este proyecto se ha decidido utilizar la metodología del Proceso Unificado.

El principal motivo de elección de esta metodología es que se caracteriza por un desarrollo iterativo e incremental, permitiendo la modificación de los requisitos a medida que avanza el proyecto, además no se requiere un equipo con experiencia para la conclusión de este.

A lo largo del capítulo 3 (Plan de Desarrollo del proyecto) se da una explicación más detallada de las características del Proceso Unificado.

1.4 Tecnologías empleadas

Para este proyecto se ha hecho uso de las siguientes tecnologías:

- Android 8 Oreo
- SQLite
- Unity 5.6.2f1
- C#

Como soporte se ha usado:

- Visual Studio Code
- Android Studio 3
- Windows 10

1.5 Estructura de la memoria

A continuación, se presenta la estructura que posee el presente documento:

- **Capítulo 1: Introducción.** En el aparecen el contexto, la motivación y los objetivos de este proyecto. Además, incluye la metodología empleada para su desarrollo, la estructura de toda la memoria y el contenido del CD entregado como parte de la documentación del Trabajo de Fin de Grado.

- **Capítulo 2: Herramientas empleadas para el desarrollo**. Breve descripción del software empleado para desarrollar la aplicación.

- **Capítulo 3: Planificación del proyecto.** Contiene toda la información relativa a cómo se ha desarrollado este proyecto, así como la gestión de los riesgos.

- **Capítulo 4: Análisis.** Incluye la elicitación de requisitos, diagrama y descripción de los casos de uso. Por último, se muestra el modelo de dominio.

- **Capítulo 5: Diseño.** Abarca todo lo relativo al diseño de las arquitecturas y de las interfaces. Los diagramas de secuencia, de clases de diseño, de vista física, de clases y diagrama entidad relación de la base de datos.

- **Capítulo 6: Implementación de la aplicación**. Implementación de las diferentes partes que forman el sistema que se ha desarrollado.

- **Capítulo 7: Pruebas.** Pruebas realizadas y el resultado obtenido en cada una de ellas.
- **Capítulo 8: Conclusiones.** Se detallan los objetivos alcanzados y los posibles trabajos futuros.

- **Bibliografía:** Todas las referencias consultadas para la realización del Trabajo de Fin de Grado.

- **Apéndice:** Manual de usuario de la aplicación.

1.6 Contenido del CD

El contenido del CD es:

- **EjeRehabilitación.zip:** ejecutables de que forman la aplicación:
  - **RehabilitacionUN:** archivos necesarios para que funcione el ejecutable (rehabilitación.exe) de la aplicación de Unity.
  - **MaRehabilita.apk:** ejecutable de la aplicación Android.
  - **NodeRest:** carpeta con el servidor Node JS.
  - **ScriptsWifi:** tres scripts para la creación, arranque y parada de un punto wifi en Windows.
- **Código:** contiene el código fuente de todas las partes que forman la aplicación (Unity, Android y servidor Node JS).
- **MemoriaTFG.pdf:** versión en pdf de la memoria del Trabajo de Fin de Grado.

Capítulo 2

2 Herramientas empleadas para el desarrollo

2.1 Introducción

En este capítulo se van a explicar las diferentes herramientas utilizadas en el desarrollo del proyecto, encontrándonos con SQLite, Android, Unity, Cuaterniones y Node JS.

2.2 Base de datos SQLite

SQLite es un sistema de gestión de bases de datos relacional, contenida en una relativamente pequeña biblioteca escrita en C.

A diferencia de los sistemas de gestión de bases de datos cliente-servidor, el motor de SQLite no es un proceso independiente con el que el programa principal se comunica. En lugar de eso, la biblioteca SQLite se enlaza con el programa pasando a ser parte integral del mismo. El programa utiliza la funcionalidad de SQLite a través de llamadas simples a subrutinas y funciones. Esto reduce la latencia en el acceso a la base de datos, debido a que las llamadas a funciones son más eficientes que la comunicación entre procesos. El conjunto de la base de datos (definiciones, tablas, índices, y los propios datos), son guardados como un sólo fichero estándar en la máquina host. Este diseño simple se logra bloqueando todo el fichero de base de datos al principio de cada transacción.

Esta base de datos ofrece numerosas ventajas:

- SQLite es una librería compacta.
- No necesita configuración.
- Es de código abierto.
- No necesita un servidor.
- Transaccional.
- Compatible con múltiples lenguajes de programación.
- Fácilmente portable.
- Permite registros de longitud variable.
- Seguridad de datos.

2.3 Android

Android es un sistema operativo de código abierto basado en el núcleo Linux. Fue diseñado principalmente para dispositivos móviles con pantalla táctil: teléfonos inteligentes, tabletas, relojes inteligentes, televisores y automóviles.

Posee las siguientes características:

- Framework de aplicación que posibilita la reutilización y reemplazo de componentes.
- Máquina virtual Dalvik optimizada para móviles. A partir de la versión 5 se sustituyó por ART (Android Runtime).
- Navegador integrado basado en WebKit.
- Gráficos optimizados por una librería gráfica 2D propia; gráficos 3D basados en la especificación OpenGL ES 1.0.
- SQLite para almacenamiento de datos estructurados.
- Soporte para gran variedad de archivos multimedia (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF, etc…).
- Telefonía GSM.
- Bluetooth, GSM, GPRS, EDGE, UMTS, LTE, WiFi, etc.…
- Cámara, GPS, brújula, acelerómetro, ...
- Tienda de aplicaciones Google Play.
- Entorno de desarrollo completo incluyendo emulador, herramientas de depuración, profiling de memoria y rendimiento y plugin para el IDE Eclipse (sustituido por Android Studio como IDE principal).
- Tethering.

2.3.1 Sensores

La mayoría de los dispositivos con Android tienen sensores incorporados que miden el movimiento, la orientación y varias condiciones ambientales. Estos sensores son capaces de proporcionar datos en bruto con alta precisión y exactitud. Son útiles si se desea monitorear el movimiento o posicionamiento tridimensional del dispositivo, o si desea monitorear cambios en el entorno ambiental. Por ejemplo, un juego puede rastrear las lecturas del sensor de gravedad de un dispositivo para inferir gestos y movimientos complejos del usuario, como inclinación, movimiento, rotación o balanceo. Del mismo modo, una aplicación meteorológica podría usar el sensor de temperatura y el sensor de humedad de un dispositivo para calcular e informar del punto de rocío, o una aplicación de viaje podría usar el sensor de campo geomagnético y el acelerómetro para informar del rumbo de una brújula.

La plataforma Android admite tres amplias categorías de sensores:

- Sensores de movimiento: miden fuerzas de aceleración y fuerzas de rotación a lo largo de tres ejes. Esta categoría incluye acelerómetros, sensores de gravedad, giroscopios y sensores de vector giratorio.

- Sensores ambientales: calculan varios parámetros ambientales, como la temperatura y presión del aire ambiente, la iluminación y la humedad. Esta categoría incluye barómetros, fotómetros y termómetros.

- Sensores de posición: calibra la posición física de un dispositivo. Incluye sensores de orientación y magnetómetros.

2.4 Unity

Es un motor de videojuegos orientado principalmente al desarrollo de videojuegos 3d, permitiendo también el desarrollo de videojuegos 2d, pudiendo combinar ambos.

Originalmente fue lanzado en exclusiva para Mac (2005), dado que su diseño fue específicamente desarrollado para esta plataforma. Desde 2008 empezó a firmar acuerdos con diferentes compañías haciendo que pasara a ser compatible con un gran número de plataformas diferentes: Mac, Windows, Linux, Android …

Las ventajas que ofrece son varias:

- Soporta OpenGL ES 3.0 (plataforma Android).
- Es multiplataforma.
- Permite la programación utilizando una gran variedad de lenguajes de scripts.
- Permite llamar a funciones personalizadas escritas en C/C++.
- Proporciona una API para acceder a diversos datos de entrada y de ajustes de Android.
- Documentación muy completa en la página del fabricante.
- Orientado a componentes que permitan aumentar los módulos de un videojuego.
- Dispone de una modalidad de licencia gratuita que nos permite desarrollar juegos desde el primer momento sin coste alguno, con acceso a recursos para facilitar esta tarea.

Sus características principales son:

- Editor de Unity: permite agrupar rápidamente todas las escenas en un espacio de trabajo mediante el uso de un editor intuitivo y fiable (es posible organizar y controlar diferentes escenas desde un solo editor).
- Desarrollo de videojuegos de gran calidad en pocos pasos, que se adaptan a todo tipo de resoluciones, proporcionando un control absoluto de las escenas creadas.
- Posibilita la publicación en numerosas plataformas sin realizar ninguna tarea de implementación extra.
- Herramientas dedicadas para la creación de contenido 2D y 3D.
- Importación de modelos y animaciones realizadas con otras aplicaciones 3D, como pueden ser Blender, Maya, 3ds Max, Modo, Cinema 4D, etc. En ellas Unity realizará y actualizará los cambios en todo el proyecto.
- Construcción rápida de escenas (niveles de juego) para añadir nuestros objetos 2D y 3D.
- Control exhaustivo de los recursos consumidos con una ventana.
- Integración con los motores de físicas de NVIDIA(r) PhysX(r) y Box2D.
- Iluminación de sombras en tiempo real además de proporcionar una herramienta llamada “Particle System”, encargada de simular líquidos, llamas o nubes mediante el uso de pequeñas imágenes 2D en la escena.

2.5 Cuaterniones

Los cuaterniones son números complejos ampliamente usados para la representación de la orientación y las rotaciones de objetos en 3 dimensiones.

De acuerdo con el teorema de rotación de Euler, toda rotación o secuencia de rotaciones de un cuerpo rígido o sistema de coordenadas con respecto a un punto fijo es equivalente a una única rotación de ángulo α alrededor de un eje fijo (eje de Euler) que pasa por el punto fijo. Es decir, cualquier rotación en 3 dimensiones puede ser representada como una combinación de un vector u (que representa un eje) y un escalar α (que representa un ángulo). Los cuaterniones proveen una manera simple para codificar esta representación ángulo-eje con 4 números <w, x, y, z> donde w es la rotación alrededor del eje <x, y, z>.

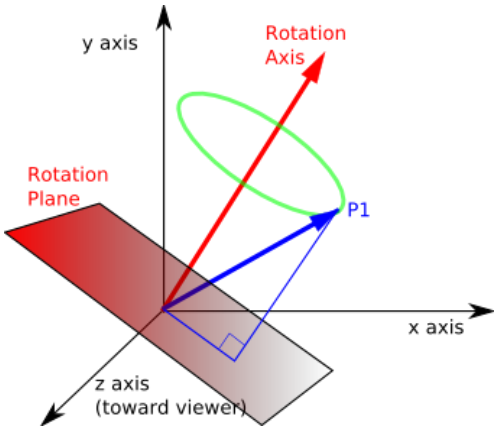

Ilustración 1: Cuaterniones-Eje

En comparación con los ángulos de Euler, estos son más simples de componer y evitan el problema del bloqueo del cardán. Este consiste en la pérdida de un grado de libertad en una suspensión cardán de tres rotores (ocurre cuando los ejes de dos de los tres rotores se colocan en paralelo bloqueando el sistema en una rotación de un espacio bidimensional degenerado) y, en comparación con las matrices de rotación, son más compactos y pueden ser más eficientes.

Un cuaternión es un número de la forma w + xi + yj + zk, donde x, y, z, y w son números reales unívocamente determinados por cada cuaternión.

Una de las operaciones más importantes de los cuaterniones es la multiplicación entre dos de ellos, que generalmente representan la rotación. El producto no es conmutativo: ij = k, ji = -k, jk = i, kj = -i, ki = j, ik = -j. Suponiendo Q3 = Q1*Q2.

Q3.w = w1w2 - x1x2 - y1y2 - z1z2
Q3.x = w1x2 + x1w2 + y1z2 - z1y2
Q3.y = w1y2 - x1z2 + y1w2 + z1x2
Q3.z = w1z2 + x1y2 - y1x2 + z1w2

2.5.1 Cálculo del ángulo

Para el cálculo del ángulo se ha hecho uso del producto escalar $\vec{x} \cdot \vec{y} = |\vec{x}||\vec{y}|$, necesitando para ello la obtención de tres puntos diferentes (posición de tres articulaciones). Así obtenemos los dos vectores con los cuales realizar la operación.

2.6 Node JS

Node JS es un entorno en tiempo de ejecución para javascript multiplataforma de código abierto, construido mediante el motor de javascript de Chrome V8.

Usa un modelo de operaciones de E/S sin bloqueo y está orientado a eventos, haciendo que sea más liviano y eficiente. Está dirigido hacia la capa de servidor, pero no solo se limita a esta.

Node JS funciona con un modelo de evaluación de un único hilo de ejecución, usando entradas y salidas asíncronas que pueden ejecutarse concurrentemente en un número de hasta cientos de miles sin incurrir en costos asociados al cambio de contexto.

2.6.1 Módulos

Node JS sigue un sistema vasado en módulos que permite ampliar sus usos, haciendo que este sea una gran herramienta para múltiples tareas, y aunque traiga por defecto algunos módulos instalados también es posible la descarga de módulos de terceros que ayudan y mejoran el trabajo, haciendo por lo general más simple y fácil su desarrollo.

Adicionalmente Node presenta el Node Package Module (NPM), que es una forma integrada de instalar y administrar los módulos que esté usando. Esta herramienta sirve para manejar automáticamente dependencias, de manera que cualquier módulo que se quiera instalar lo hará correctamente con todas sus partes necesarias. También se utiliza como una forma de publicar nuestros propios módulos en la comunidad Node.

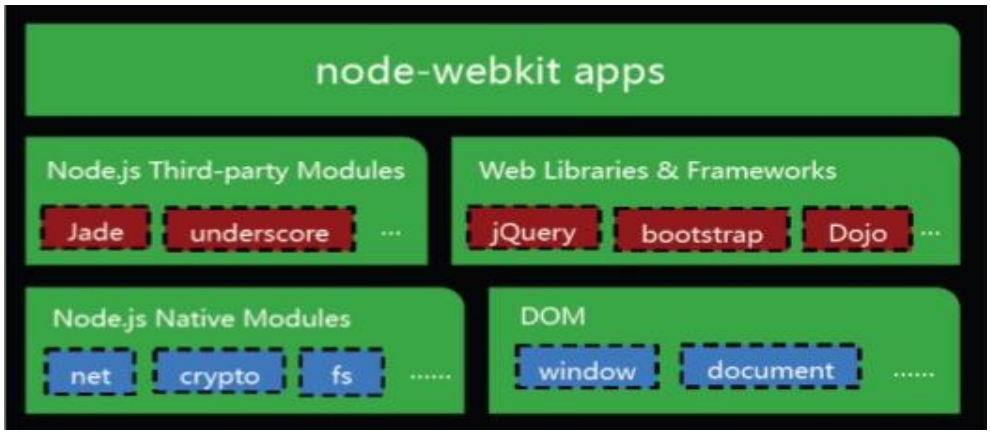

Ilustración 2: Ilustración 57: Node-Webkit

Capítulo 3

3 Planificación del proyecto

3.1 Introducción

En este capítulo se va a hablar de la elección del proceso unificado como modelo escogido para la planificación del desarrollo del proyecto, sus características y el plan de trabajo. Tratando también los distintos riesgos que pueden aparecer, así como la respuesta a los mismos.

3.2 Elección del Proceso Unificado

La elección del modelo de Proceso Unificado se ha debido principalmente a tres factores:

1. El uso de un modelo puramente ágil necesita un equipo de desarrollo con experiencia para la elaboración del proyecto.
2. En un modelo de desarrollo pesado resulta inadecuada la modificación de los requisitos a medida que avanza el proyecto.
3. El modelo unificado es un proceso iterativo e incremental que tiene en cuenta los riesgos.

Los principales elementos que resumen el Proceso Unificado son:

- Centrado en la arquitectura: nos permite gestionar los distintos puntos de vista que componen el sistema, ya que no hay un modelo único que abarque todos los aspectos que esta posee, permitiendo generar un patrón que dirija la construcción del sistema en las primeras fases.

- Enfocado en los riesgos: el Proceso Unificado requiere que se abarquen los riesgos desde etapas tempranas del ciclo de vida del proyecto. En cada iteración son ordenandos para que los principales se traten primero.

- Dirigido por casos de uso: estos son la guía que permite definir los diseños y la implementación de cada iteración y así poder satisfacer las demandas de los usuarios. Para ello, en cada iteración se identifican un conjunto de casos de uso o escenarios, llevándolos a cabo con las diferentes disciplinas.

- Iterativo e incremental: permite realizar pequeños pasos manejables con poca planificación, diseñando, especificando e implementando en cada iteración. Esto nos permitirá comprobar si en cada paso el resultado es favorable, avanzando en consecuencia, pudiendo adaptar los objetivos y permitiéndonos comprender en primeras etapas los riesgos, determinando la viabilidad.

3.3 Modelo de proceso

El Proceso Unificado está orientado a la elaboración del proyecto software, el cual se podría describir como un marco de trabajo genérico que se puede especializar para gran cantidad de sistemas software, tipos de organización, diferentes áreas de aplicación, etc.

Se pueden diferenciar 4 fases en el ciclo de vida del proyecto: inicio, elaboración, construcción y transición. Estas ayudan al desarrollo y a la resolución de problemas en momentos tempranos del mismo.

3.4 Plan de trabajo

Para la realización del plan de trabajo se hará uso de la herramienta MS Project de la que se obtendrán los diagramas y datos.

3.5 Actividades de trabajo

El Proceso Unificado divide en cuatro fases determinantes el proyecto:

Tabla 1: Fases del Proceso Unificado

Fases	Proceso
Fase de inicio	- Reunión de información. - Agrupar los conocimientos para usar la información. - Descubrir los objetivos faltantes.
Fase de elaboración	- Modelo de negocio completo. - Actualización de los Modelos: casos de uso, análisis, despliegue e implementación. - Determinar la arquitectura software propuesta. - Revisión del plan de gestión de riesgos.
Fase de construcción	- Generar plan de proyecto para la fase de transición. - Sistema software ejecutable en una fase beta. - Descripción de la arquitectura ya actualizada y modificada mínimamente. - Primera versión del manual de usuario para que sirva en la etapa beta. - Análisis de negocio que muestre la situación final de la fase.
Fase de transición	- Sistema software ya ejecutable junto con el instalador. - Versión final del manual de usuario. - Descripción completa y actualizada de la arquitectura.

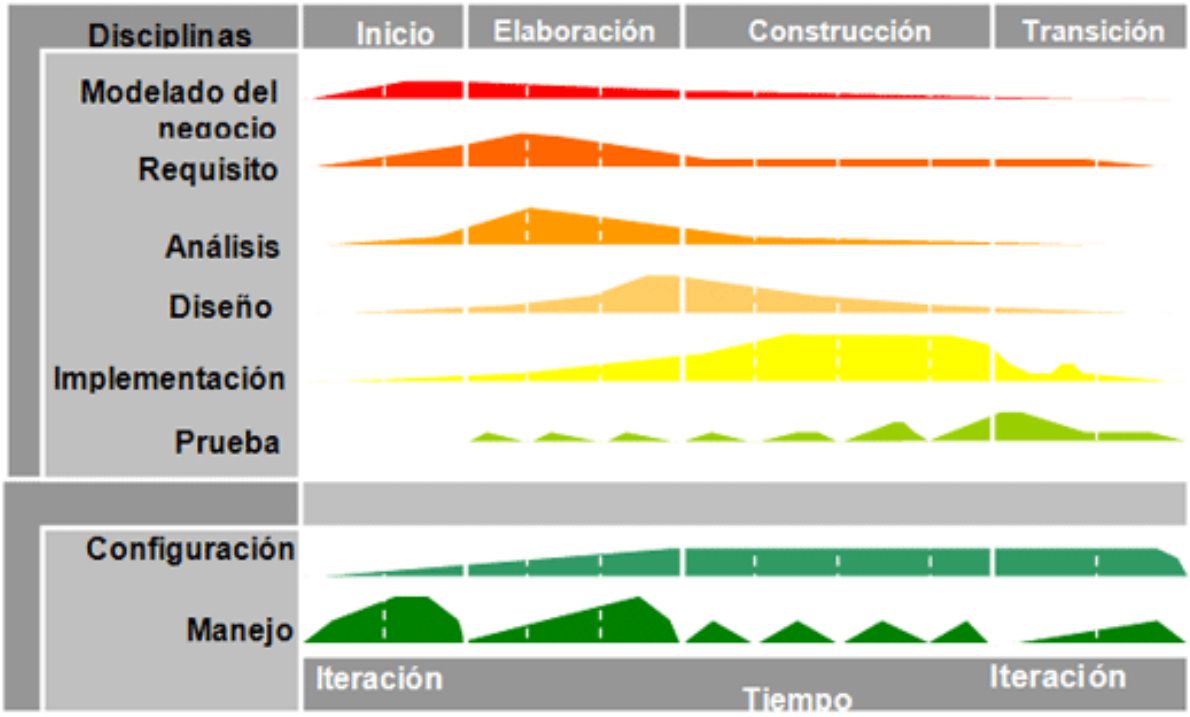

Ilustración 3: Plan de fases

3.6 Asignación de recursos y calendario

Para llevar a cabo dicha planificación se han preparado planes a dos niveles:

- Plan de fases.
- Plan de Iteración.

3.6.1 Plan de fases

El plan de fases se genera al principio de la fase de inicio y se actualiza tantas veces como sea necesario.

En este plan se deben especificar las fases del proyecto con sus correspondientes iteraciones.

Los objetivos que ha de cumplir en cada iteración de las diferentes fases son:

- **Inicio:** es la primera y en ella hay que completar los siguientes objetivos:
    - Capturar las funcionalidades primarias del sistema.
    - Realizar investigación sobre Unity, realizando tutoriales y simples ejemplos que proporcionen un aprendizaje básico del uso y manejo de dicha tecnología.

    - 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 ello las especificaciones detalladas de los casos de uso más importantes. Hay que elaborar también 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. La mayoría deben ser capturados en esta iteración llevando a cabo un diseño inicial de la arquitectura del sistema.

- **Construcción, iteración 1:** basándonos en 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 una primera visión de interfaz de usuario.

- **Construcción, iteración 2:** en esta iteración se debe finalizar la implementación de los controladores de avatar, ordenar los casos de uso por prioridad y aplicar mejoras en la interfaz de usuario, desarrollo de la aplicación móvil y del servidor Node JS.

- **Construcción, iteración 3**: últimos retoques en los controladores de avatar, 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 uno consistente.

Calendario del plan de fases con jornadas de 5 horas:

	Nombre	Duración	Comienzo	Fin	Predecesoras
1	Comienzo de proyecto	0 días	enero 29 2018 9:00 AM	enero 29 2018 9:00 AM	
2	Proyecto	90 días	enero 29 2018 9:00 AM	junio 1 2018 7:00 PM	
3	Inicio	4 días	enero 29 2018 9:00 AM	febrero 1 2018 7:00 PM	
4	Iteración1	4 días	enero 29 2018 9:00 AM	febrero 1 2018 7:00 PM	1
5	Elaboración	20 días	febrero 2 2018 9:00 AM	marzo 1 2018 7:00 PM	
6	Iteración1	10 días	febrero 2 2018 9:00 AM	febrero 15 2018 7:00 PM	4
7	Iteración2	10 días	febrero 16 2018 9:00 AM	marzo 1 2018 7:00 PM	6
8	Construcción	60 días	marzo 2 2018 9:00 AM	mayo 24 2018 7:00 PM	
9	Iteración1	20 días	marzo 2 2018 9:00 AM	marzo 29 2018 7:00 PM	7
10	Iteración2	20 días	marzo 30 2018 9:00 AM	abril 26 2018 7:00 PM	9
11	Iteración3	20 días	abril 27 2018 9:00 AM	mayo 24 2018 7:00 PM	10
12	Transición	5 días	mayo 25 2018 9:00 AM	mayo 31 2018 7:00 PM	
13	Iteración1	5 días	mayo 25 2018 9:00 AM	mayo 31 2018 7:00 PM	11
14	Fin del proyecto	0 días	junio 1 2018 9:00 AM	junio 1 2018 9:00 AM	13

Ilustración 4: Calendario plan de fases

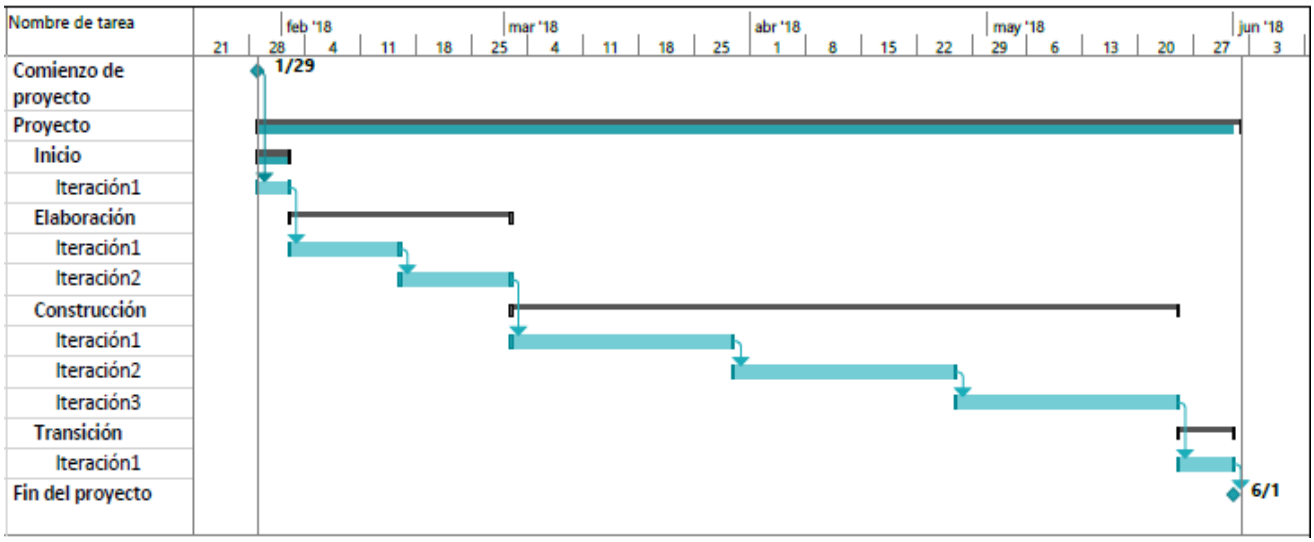

Ilustración 5 : Calendario plan de fases

3.6.2 Plan de iteración

Es un plan detallado, solo hay uno por iteración pudiendo tener dos planes a la vez, siendo estos el de la iteración actual y el de la iteración siguiente.

En cada uno se detallan los objetivos y los artefactos que hay que conseguir en dicha iteración.

Fase de Inicio

Nombre de tarea	Duración	Comienzo	Fin	Predecesoras	Nombres de los recursos
Comienzo	0 días	lun 1/29/18	lun 1/29/18		
Inicio	**4 días**	**lun 1/29/18**	**jue 2/1/18**		
Proceso	**4 días**	**lun 1/29/18**	**jue 2/1/18**		
Vision general de proyecto	1 día	lun 1/29/18	lun 1/29/18	1	
Adquirir conocimientos basicos sobre Unity	2 días	mar 1/30/18	mié 1/31/18	4	
Adaptarse a programar en C#	2 días	lun 1/29/18	mar 1/30/18	1	
Crear primer modelo de casos de uso	2 días	mié 1/31/18	jue 2/1/18	6	
Gestión	**4 días**	**lun 1/29/18**	**jue 2/1/18**		
Analisis de riesgos inicial	2 días	lun 1/29/18	mar 1/30/18	1	
Plan de fases	1 día	mié 1/31/18	mié 1/31/18	1,9	
Plan de iteración inicial	1 día	jue 2/1/18	jue 2/1/18	9,1	
Plan de iteración siguiente	2 días	mié 1/31/18	jue 2/1/18	9	

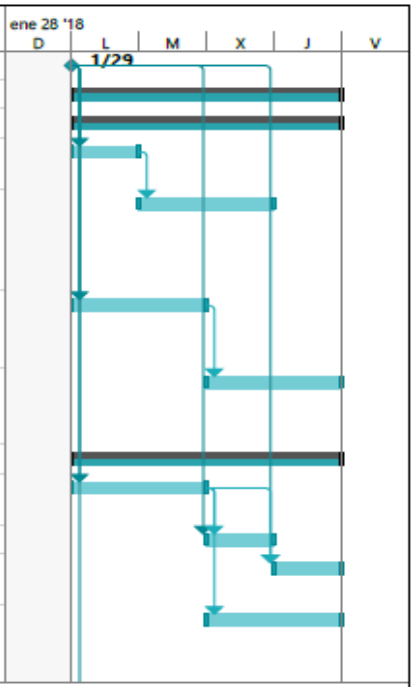

Ilustración 6: Calendario plan de fases-Fase de inicio

Fase de Elaboración, Iteración 1

Id	Modo de tarea	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras	Nombres de los recursos
1		Comienzo	0 días	vie 2/2/18	vie 2/2/18		
2		Elaboración, Iteración 1	10 días	vie 2/2/18	jue 2/15/18		
3		Proceso	10 días	vie 2/2/18	jue 2/15/18		
4		Elicitación de requisitos funcionales	2 días	vie 2/2/18	lun 2/5/18	1	
5		Actualizar y explicar casos de uso	3 días	mar 2/6/18	jue 2/8/18	4	
6		Prototipo	5 días	vie 2/9/18	jue 2/15/18	4,5	
7		Gestión	10 días	vie 2/2/18	jue 2/15/18		
8		Plan de contingencia frente a los riesgos	5 días	vie 2/2/18	jue 2/8/18	1	
9		Plan de iteración siguiente	5 días	vie 2/9/18	jue 2/15/18	8	
10		Seguimiento del plan de iteración actual	10 días	vie 2/2/18	jue 2/15/18	1	
11		Fin	0 días	vie 2/16/18	vie 2/16/18	6,8	

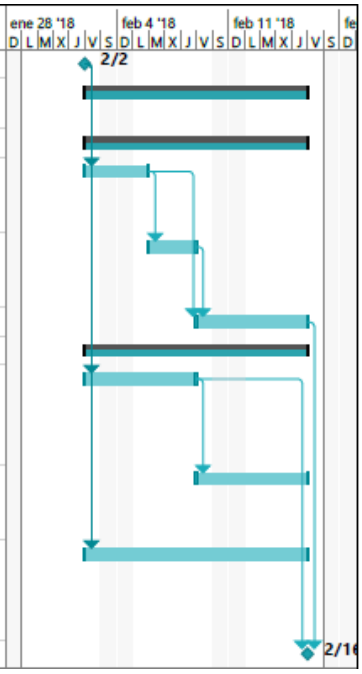

Ilustración 7: Calendario plan de fases-Fase de Elaboración, Iteración 1

Fase de Elaboración, Iteración 2

Id	Modo de tarea	Nombre de tarea	Duración	Comienzo	Fin
1		Comienzo	0 días	vie 2/16/18	vie 2/16/18
2		Elaboración, Iteración 2	10 días	vie 2/16/18	jue 3/1/18
3		Proceso	10 días	vie 2/16/18	jue 3/1/18
4		Completar requisitos funcionales	2 días	vie 2/16/18	lun 2/19/18
5		Diseño inicial de la arquitectura	5 días	mar 2/20/18	lun 2/26/18
6		Prototipo inicial de la interfaz de usuario	3 días	mar 2/27/18	jue 3/1/18
7		Gestión	10 días	vie 2/16/18	jue 3/1/18
8		Revisión de riesgos	5 días	vie 2/16/18	jue 2/22/18
9		Plan de la iteración siguiente	5 días	vie 2/23/18	jue 3/1/18
10		Seguimiento del plan de iteración actual	10 días	vie 2/16/18	jue 3/1/18
11		Fin	0 días	vie 3/2/18	vie 3/2/18

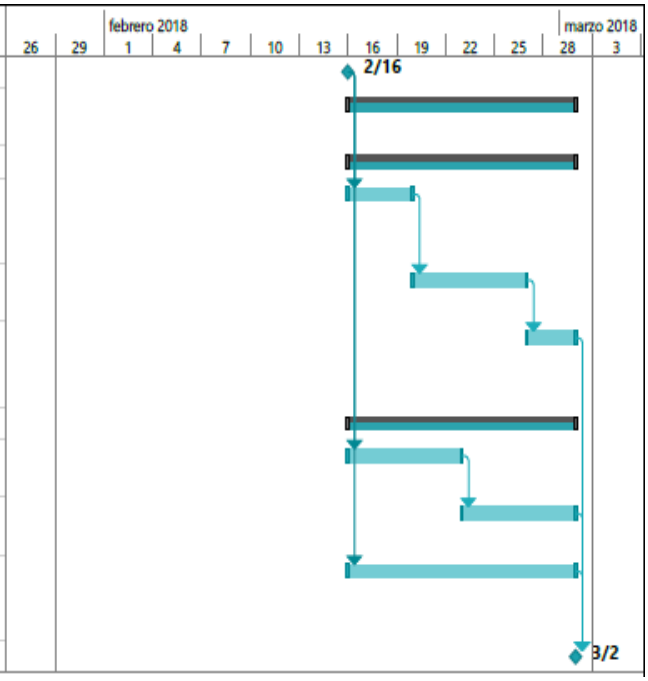

Ilustración 8: Calendario plan de fases-Fase de Elaboración, Iteración 2

Fase de Construcción, Iteración 1

Id	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1	**Comienzo**	**0 días**	**vie 3/2/18**	**vie 3/2/18**	
2	**Construcción, Iteración 1**	**20 días**	**vie 3/2/18**	**jue 3/29/18**	
3	**Proceso**	**20 días**	**vie 3/2/18**	**jue 3/29/18**	
4	Revisión de los modelos UML	2 días	vie 3/2/18	lun 3/5/18	1
5	Diseño de la base de datos	4 días	mar 3/6/18	vie 3/9/18	4
6	Implementación de la base de datos SQLite	2 días	lun 3/12/18	mar 3/13/18	5
7	Implementación de la primera versión de la interfaz de usuario	5 días	mié 3/14/18	mar 3/20/18	6
8	Implementación de algunos casos de uso	7 días	mié 3/14/18	jue 3/22/18	6
9	**Gestión**	**20 días**	**vie 3/2/18**	**jue 3/29/18**	
10	Revisión de riesgos	10 días	vie 3/2/18	jue 3/15/18	1
11	Plan de iteración siguiente	10 días	vie 3/16/18	jue 3/29/18	10
12	Seguimiento del plan de iteración	20 días	vie 3/2/18	jue 3/29/18	1
13	**Fin**	**0 días**	**vie 3/30/18**	**vie 3/30/18**	**7,8,12,11**

Ilustración 9: Calendario plan de fases- Fase de Construcción, Iteración 1

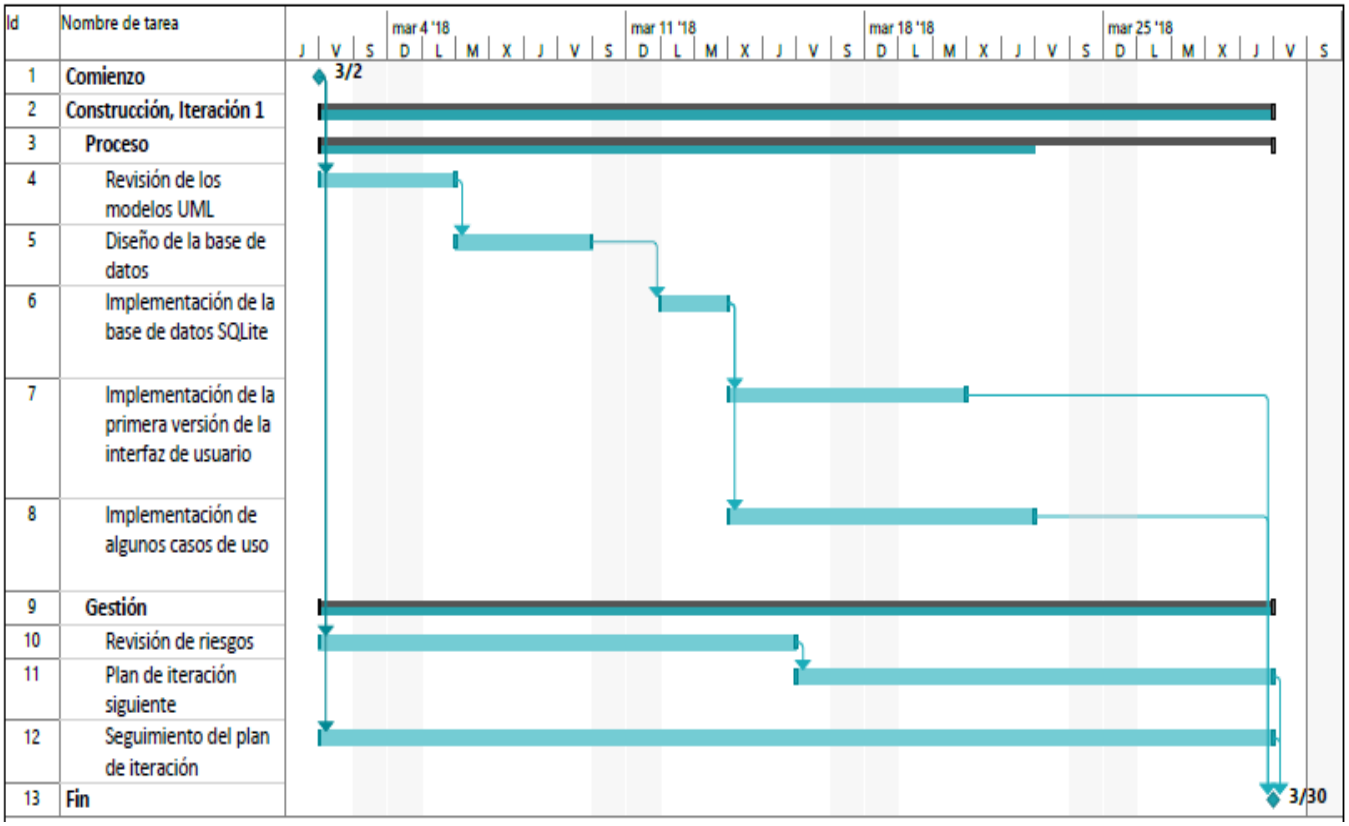

Ilustración 10: Calendario plan de fases- Fase de Construcción, Iteración 1 – Diagrama Gantt

Fase de Construcción, Iteración 2

Id	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1	**Comienzo**	**0 días**	**vie 3/30/18**	**vie 3/30/18**	
2	**Construcción, Iteración 2**	**20 días**	**vie 3/30/18**	**jue 4/26/18**	
3	**Proceso**	**20 días**	**vie 3/30/18**	**jue 4/26/18**	
4	Desarrollo de la app móvil y del servidor	10 días	vie 3/30/18	jue 4/12/18	1
5	Implementación de los casos de uso	4 días	vie 4/13/18	mié 4/18/18	4
6	Mejora de la interfaz de usuario	3 días	jue 4/19/18	lun 4/23/18	5
7	Controlador Avatar	3 días	mar 4/24/18	jue 4/26/18	5,6
8	**Gestión**	**20 días**	**vie 3/30/18**	**jue 4/26/18**	
9	Revisión de riesgos	10 días	vie 3/30/18	jue 4/12/18	1
10	Planificación de la iteración siguiente	10 días	vie 4/13/18	jue 4/26/18	9
11	Seguimiento de la iteración actual	20 días	vie 3/30/18	jue 4/26/18	1
12	**Fin**	**0 días**	**vie 4/27/18**	**vie 4/27/18**	7,10,11

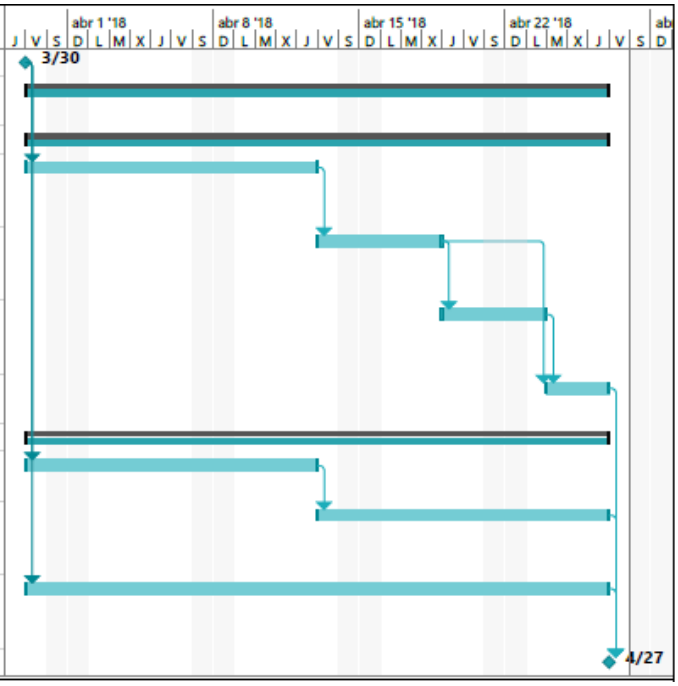

Ilustración 11: Calendario plan de fases- Fase de Construcción, Iteración 2 – Diagrama Gantt

Fase de Construcción, Iteración 3

Id	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1	**Comienzo**	**0 días**	**vie 4/27/18**	**vie 4/27/18**	
2	**Construcción, Iteración 3**	**20 días**	**vie 4/27/18**	**jue 5/24/18**	
3	**Proceso**	**20 días**	**vie 4/27/18**	**jue 5/24/18**	
4	Finalizar la implantación de los casos de uso	10 días	vie 4/27/18	jue 5/10/18	1
5	Versión final de la interfaz de usuario	5 días	vie 5/11/18	jue 5/17/18	4
6	Versión final de los controladores de avatar	5 días	vie 5/18/18	jue 5/24/18	4
7	**Gestión**	**20 días**	**vie 4/27/18**	**jue 5/24/18**	
8	Revisión de riesgos	10 días	vie 4/27/18	jue 5/10/18	1
9	Plan de la iteración siguiente	10 días	vie 5/11/18	jue 5/24/18	8
10	Seguimiento de la iteración actual	20 días	vie 4/27/18	jue 5/24/18	1
11	**Fin**	**0 días**	**vie 5/25/18**	**vie 5/25/18**	5,6,9,10

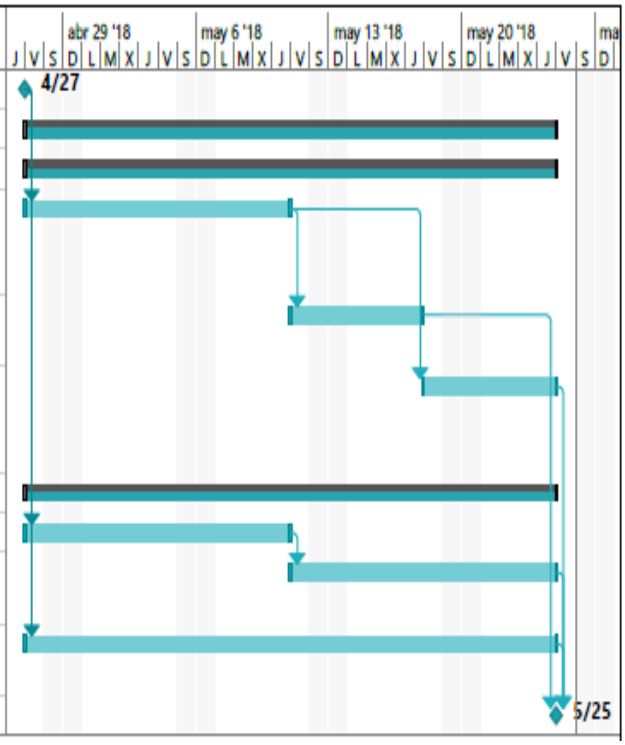

Ilustración 12: Calendario plan de fases- Fase de Construcción, Iteración 3 – Diagrama Gantt

Fase de Transición

Id	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1	**Comienzo**	**0 días**	**vie 5/25/18**	**vie 5/25/18**	
2	**Transición**	**5 días**	**vie 5/25/18**	**jue 5/31/18**	
3	**Proceso**	**5 días**	**vie 5/25/18**	**jue 5/31/18**	
4	Realización de pruebas	2 días	vie 5/25/18	lun 5/28/18	1
5	Corrección de fallos detectados	2 días	mar 5/29/18	mié 5/30/18	4
6	Finalización de la aplicación móvil y pc	1 día	jue 5/31/18	jue 5/31/18	5
7	**Gestión**	**5 días**	**vie 5/25/18**	**jue 5/31/18**	
8	Revisión de riesgos	1 día	vie 5/25/18	vie 5/25/18	1
9	Seguimiento del plan de iteración actual	5 días	vie 5/25/18	jue 5/31/18	1
10	**Fin**	**0 días**	**vie 6/1/18**	**vie 6/1/18**	6,8,9

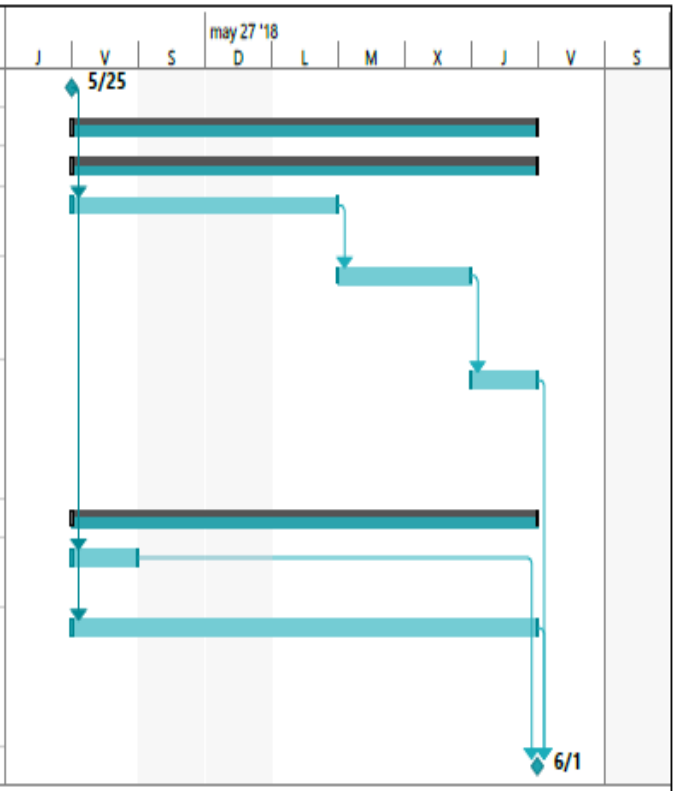

Ilustración 13: Calendario plan de fases- Fase de Transición

3.7 Métodos, herramientas y técnicas

En la siguiente lista se muestran las herramientas software necesarias en cada fase.

Fase de Inicio:

- Herramientas ofimáticas: se hará uso de estos productos para elaborar la documentación pertinente, así como los gráficos, tablas o el soporte para la planificación del proyecto. Se utilizará la familia de productos Office de Microsoft, concretamente Project y Word.

Fase de elaboración:

- Herramientas de control de versiones: se usará la herramienta Git junto a la web GitHub para mantener copias de seguridad del proyecto y de los documentos. En caso de fallo se podría recuperar una versión funcional.

- Herramientas CASE: en la parte de ingeniería del software se usarán herramientas que ayuden a mejorar la productividad del desarrollo. Se usará Astah para la construcción de los diferentes diagramas de casos de uso, secuencia, actividad, etc.

Fase de construcción, en esta se utilizarán las siguientes herramientas:

- Android estudio 3.0.1.
- Microsoft visual studio code.
- Unity 5.6.2f1.
- Lenguajes de programación, gestor base de datos, servidor web:(Se explicará más adelante)
- Dispositivo móvil con sistema Android.

3.8 Costes del proyecto

A continuación, se especifican los costes del proyecto incluyendo los referentes al hardware y los de personal.

3.8.1 Costes hardware

Para el desarrollo del proyecto se ha usado un teléfono móvil Xiaomi MI A1, que cuenta con las siguientes características:

- CPU: Snapdragon 625 de ocho núcleos a 2,2GHz
- GPU: Adreno 506
- RAM: 4 GB
- Almacenamiento: 64 GB
- Precio: 180 €

El ordenador utilizado ha sido un Lenovo Z50 cuyas características son:

- CPU: AMD FX-7500, 2.10 GHz
- GPU: Radeon R7 APU
- Pantalla: 15.6 pulgadas, 1366 x 768
- RAM: 8GB
- Almacenamiento: 1 Tb
- S.O: Windows 10, 64 bits
- Puertos: 1 USB 3.0, 2 USB 2.0, VGA, HDMI, Ethernet, Headphones, Sd reader
- Precio: 349 €

Coste hardware total imputable al proyecto: (180 +349) / 48 meses * 4 meses = **44,083€**

3.8.2 Coste de personal

El sueldo medio de un analista-programador se encuentra en torno a 12€ la hora.
Siendo así, el coste de personal es: 12€/hora * 5 horas/día * 90 días: **5400€**

3.8.3 Coste total del proyecto

Con los datos anteriores el coste es la suma del coste del hardware y del personal resultando en un total de: 44,083 + 5400 = **5444,083 €**

3.9 Gestión de riesgos

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

Exposición al riesgo = Probabilidad x Consecuencia

Matriz Impacto/Probabilidad

Impacto/ Probabilidad	Muy alto	Alto	Medio	Bajo	Muy bajo
Catastrófico	Alto	Alto	Moderado	Moderado	Bajo
Crítico	Alto	Alto	Moderado	Bajo	Ninguno
Marginal	Moderado	Moderado	Bajo	Ninguno	Ninguno
Despreciable	Moderado	Bajo	Bajo	Ninguno	Ninguno

Ilustración 14:Matriz Impacto/Probabilidad

Tabla 2: Pérdida de artefactos

Código	R01
Nombre	Pérdida de artefactos
Descripción	Pérdida de los artefactos en cualquiera de las fases del proyecto, tanto realizados como en proceso de realización.
Probabilidad	Bajo
Consecuencia	Pérdida parcial o completa de los artefactos (documentación o código), y todo el tiempo invertido en ello, con su incremento de tiempo que conllevaría volver a realizarlo.
Impacto	Catastrófico
Estrategia	• Evitar el riesgo.
Plan de contingencia	Realización de copias de seguridad de todos los artefactos cada vez que se realice una modificación, mediante un sistema de control de versiones.

Tabla 3: Modificación de los requisitos

Código	R02
Nombre	Modificación de los requisitos
Descripción	Mala captación de los requisitos iniciales o incorporación de otros nuevos debido al requerimiento del cliente, haciendo que los requisitos del sistema puedan sufrir cambios significativos.
Probabilidad	Medio
Consecuencia	Aumento del coste y del tiempo del desarrollo del proyecto.
Impacto	Medio
Estrategia	• Reducción del riesgo.
Plan de contingencia	Aumentar el margen de holgura para intentar modificar lo menos posible la planificación inicial y aplicar más recursos minimizando el cambio producido.

Tabla 4: Problemas conexión Unity y Android

Código	R03
Nombre	Problemas conexión Unity y Android
Descripción	Problemas en el intercambio de datos entre la plataforma móvil Android y la plataforma de desarrollo de juegos Unity.
Probabilidad	Bajo
Consecuencia	Buscar alternativa a Unity para poder desarrollar la aplicación.
Impacto	Alto
Estrategia	• Protección del riesgo.
Plan de contingencia	Informarse de antemano de la posibilidad de comunicar Unity con una aplicación Android.

Tabla 5: Ausencia de personal

Código	R04
Nombre	Ausencia de personal
Descripción	Falta de personal para desarrollar el proyecto por problemas de salud o por accidente laboral, impidiendo que pueda llevarse a cabo la tarea requerida.
Probabilidad	Bajo
Consecuencia	Aumento del tiempo de desarrollo que, dependiendo de la causa, será mayor o menor, pudiendo llegar incluso a su cancelación.
Impacto	Alto
Estrategia	• Reservar el riesgo.
Plan de contingencia	Redistribuir las tareas en el tiempo restante para tratar de reducir el impacto.

Tabla 6: Falta de conocimiento en el uso de las herramientas

Código	R05
Nombre	Falta de conocimiento en el uso de las herramientas
Descripción	Falta de experiencia del personal a la hora de usar las herramientas necesarias para la realización del proyecto.
Probabilidad	Alta
Consecuencia	Mayor inversión de recursos y tiempo en el desarrollo del proyecto.
Impacto	Bajo
Estrategia	• Evitar el riesgo.
Plan de contingencia	Aprender el uso y funcionalidad de las herramientas, mediante tutoriales, cursos y lecturas.

Tabla 7: Fallos en las herramientas de desarrollo

Código	R06
Nombre	Fallos en las herramientas de desarrollo
Descripción	Posible fallo al usar una herramienta ya sea por problemas de uso o por que deje de funcionar.
Probabilidad	Medio
Consecuencia	Aumentaría el tiempo de desarrollo y posiblemente el coste de este por la adquisición de una nueva herramienta.
Impacto	Alto
Estrategia	• Protección del riesgo. • Reducción del riesgo.
Plan de contingencia	Usar de forma cuidadosa las herramientas y realizar un mantenimiento periódico de estas.

Tabla 8: Falta de recursos del ordenador

Código	R07
Nombre	Falta de recursos del ordenador
Descripción	El ordenador no dispone de las características necesarias para mover todos los programas que se requieren a la hora del desarrollo.
Probabilidad	Media
Consecuencia	Aumento del coste y tiempo, por la adquisición de un software adecuado.
Impacto	Medio
Estrategia	• Protección del riesgo.
Plan de contingencia	Comprobar que todas las herramientas sean compatibles con los programas que se van a usar.

Tabla 9: Actualizaciones en el software utilizado

Código	R08
Nombre	Actualizaciones en el software utilizado
Descripción	Posibles actualizaciones en el software usado, pudiendo este hacer incompatible los datos producidos con versiones anteriores del mismo.
Probabilidad	Medio
Consecuencia	Posible pérdida de parte del trabajo realizado haciendo que el tiempo de desarrollo aumente.
Impacto	Alto
Estrategia	• Reducir el riesgo.
Plan de contingencia	Buscar alternativas que permitan continuar con el trabajo.

Tabla 10: Mala estimación del proyecto

Código	R09
Nombre	Mala estimación del proyecto
Descripción	Mala planificación del proyecto haciendo que el tiempo requerido estimado no coincida con el tiempo que se necesita realmente para su desarrollo.
Probabilidad	Media
Consecuencia	La mala estimación supondrá un aumento de costes y del tiempo para su desarrollo, pudiendo llegar al fracaso del proyecto.
Impacto	Crítico
Estrategia	• Protección del riesgo. • Reducción del riesgo
Plan de contingencia	Controlar el progreso del proyecto, detectando posibles fallos de planificación y permitiendo redistribuirlo, de manera que se reduzca el impacto del riesgo.

Tabla 11: Problemas con la conexión

Código	R10
Nombre	Problemas con la conexión
Descripción	Mala transferencia de los datos del móvil a Unity provocando una mala sincronización entre ambos.
Probabilidad	Media
Consecuencia	La mala comunicación supondrá un aumento de costes y del tiempo de desarrollo.
Impacto	Alto
Estrategia	• Protección del riesgo. • Reducción del riesgo
Plan de contingencia	Controlar el progreso del proyecto, detectando posibles fallos de conexión y buscar posibles soluciones.

Tabla 12: Modificación de los requisitos de la interfaz de usuario

Código	R11
Nombre	Modificación de los requisitos de la interfaz de usuario
Descripción	Tratando de facilitar al máximo el entendimiento de la herramienta por el usuario, esta debe de ser sencilla y entendible.
Probabilidad	Media
Consecuencia	La mala estimación supondrá un aumento de tiempo en el aprendizaje y uso por parte del usuario.
Impacto	Bajo
Estrategia	• Reducción del riesgo
Plan de contingencia	Controlar el diseño de las diferentes interfaces con los usuarios. Rediseñándola si es necesario, para darle la máxima accesibilidad.

3.9.1 Seguimiento de la planificación

En este punto se muestran los retrasos sufridos en las iteraciones de cada fase durante el desarrollo del proyecto.

Tabla 13: Riesgos y solución

Riesgos	Descripción	Solución	Duración
R02- Modificación de los requisitos	Ampliación de los requisitos con la inclusión de una nueva actividad (Minijuego: emparejar cartas).	Modificar la documentación añadiendo la nueva información e incluir el minijuego en la aplicación de Unity	La iteración 3 de la fase de construcción se incrementó en 5 días.
R08- Actualizaciones en el software utilizado	La actualización incorrecta de la herramienta Android Studio, usada para el desarrollo de la aplicación móvil, hizo que esta no se pudiera compilar, parando su desarrollo.	Reinstalar Android Studio, teniendo que continuar con el desarrollo de la aplicación desde la última copia de seguridad realizada antes del fallo.	La iteración 2 de la fase de construcción se incrementó en un día.
R05- Falta de conocimiento en el uso de las herramientas	Debido a la falta de conocimientos en la herramienta de Unity, la representación de los movimientos del móvil en el avatar inicialmente requirió más tiempo del estimado.	Buscar más información y realizar pruebas.	La fase de inicio se incrementó en 2 días.
R11- Modificación de los requisitos de la interfaz de usuario	El diseño de la interfaz podría ser más simple, siendo en la mayoría de los casos el tamaño de la fuente el que habría que modificar.	Se ha modificado el tamaño de la letra y el menú inicial se ha simplificado.	La fase de transición se incrementó en 3 horas, teniendo un impacto casi nulo.

El calendario final del plan de fases con los cambios sufridos ha quedado como se muestra a continuación.

Tabla 14: Calendario plan de fases - final

Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
Comienzo de proyecto	0 días	lun 1/29/18	lun 1/29/18	
Proyecto	**90 días**	**lun 1/29/18**	**vie 6/1/18**	
Inicio	**6 días**	**lun 1/29/18**	**jue 2/1/18**	
Iteración1	6 días	lun 1/29/18	jue 2/1/18	1
Elaboración	**20 días**	**vie 2/2/18**	**jue 3/1/18**	
Iteración1	10 días	vie 2/2/18	jue 2/15/18	4
Iteración2	10 días	vie 2/16/18	jue 3/1/18	6
Construcción	**66 días**	**vie 3/2/18**	**jue 5/24/18**	
Iteración1	20 días	vie 3/2/18	jue 3/29/18	7
Iteración2	21 días	vie 3/30/18	jue 4/26/18	9
Iteración3	25 días	vie 4/27/18	jue 5/24/18	10
Transición	**5 días**	**vie 5/25/18**	**jue 5/31/18**	
Iteración1	5 días	vie 5/25/18	jue 5/31/18	11
Fin del proyecto	0 días	vie 6/1/18	vie 6/1/18	13

La duración del proyecto no se ha visto alterada gracias a la realización de horas extra para solucionar los riesgos que han ido ocurriendo.

Capítulo 4

4 Análisis

4.1 Introducción

Este capítulo desarrolla primero la especificación de los requisitos, diagrama y descripción de los casos de uso. Por último, se muestra el modelo de dominio.

4.2 Requisitos funcionales

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

FRQ-002	**Identificar**
Descripción	El sistema deberá permitir a los pacientes registrados identificarse mediante su ID.

FRQ-003	**Borrar**
Descripción	El sistema deberá permitir al paciente borrar su cuenta.

FRQ-004	**Acceso**
Descripción	El sistema permitirá a los pacientes registrados acceder a todas las aplicaciones.

FRQ-005	**Ver datos del paciente**
Descripción	El sistema deberá permitir visualizar los datos del paciente

FRQ-006	Mostrar ejercicios
Descripción	El sistema deberá exponer todos los ejercicios disponibles de la aplicación en el menú principal.

FRQ-007	Modificar datos del paciente
Descripción	El sistema deberá permitir la modificación de todos los datos de un paciente a excepción del Nombre de usuario y la fecha de lesión.

FRQ-008	Iniciar ejercicio
Descripción	El sistema deberá permitir iniciar el ejercicio al usuario cuando este quiera.

FRQ-009	Guardado de datos
Descripción	El sistema deberá guardar tanto datos personales como los resultados de los ejercicios de un paciente.

FRQ-010	Pausar ejercicio
Descripción	El sistema deberá permitir pausar un ejercicio cuando el usuario lo requiera.

FRQ-011	Restaurar ejercicio
Descripción	El sistema deberá permitir continuar con los ejercicios pausados.

FRQ-012	Visualizar datos de ejercicio
Descripción	El sistema deberá permitir visualizar los datos relevantes de los ejercicios realizados.

FRQ-013	Lista de ejercicios realizados
Descripción	El sistema deberá mostrar una lista de todos los ejercicios que ha realizado el paciente.

FRQ-014	Pausar ejercicio
Descripción	El sistema deberá permitir pausar un ejercicio cuando el usuario lo requiera.

FRQ-015	Instrucciones
Descripción	El sistema deberá mostrar las instrucciones necesarias para que el paciente pueda realizar el ejercicio.

FRQ-016	Seleccionar lado del ejercicio
Descripción	El sistema deberá permitir al usuario seleccionar que versión del ejercicio se va a realizar (Izquierda/Derecha).

FRQ-017	Cerrar sesión
Descripción	El sistema deberá permitir al usuario cerrar su sesión cuando este lo quiera.

FRQ-018	Eliminar datos de los ejercicios
Descripción	El sistema deberá permitir al usuario eliminar datos de los ejercicios realizados.

FRQ-019	Fin de ejercicio
Descripción	El sistema deberá indicar al usuario el final de los ejercicios.

FRQ-020	Salir del ejercicio
Descripción	El sistema deberá permitir al usuario salir del ejercicio en cualquier momento.

4.2.1 Requisitos funcionales de actividad de salón, baño y cartas

Requisitos funcionales Actividad Salón

FRQ-021	**Avatar para actividad de salón**
Descripción	El sistema deberá simular al paciente mediante un avatar.

FRQ-022	**Coger taza**
Descripción	El sistema deberá simular que el avatar del paciente agarra una taza mediante una animación.

FRQ-023	**Coger cepillo**
Descripción	El sistema deberá simular que el avatar del paciente agarra un cepillo mediante una animación.

FRQ-024	**Colisión entre boca y taza**
Descripción	El sistema deberá detectar el contacto entre la taza y la boca del avatar.

FRQ-025	**Colisión entre cepillo y pelo**
Descripción	El sistema deberá detectar el contacto entre el cepillo y el pelo del avatar.

FRQ-026	**Control del avatar para actividad de salón**
Descripción	El sistema deberá permitir al paciente controlar el avatar mediante las rotaciones del codo.

FRQ-027	**Ambiente actividad de salón**
Descripción	El sistema deberá simular el entorno de un salón.

Requisitos funcionales Actividad Baño

FRQ-028	**Ambiente actividad de baño**
Descripción	El sistema deberá simular el entorno de un baño.

FRQ-029	**Configurar repeticiones de cepillado**
Descripción	El sistema deberá permitir configurar el número de veces que se tiene que cepillar los dientes.

FRQ-030	**Control del avatar para actividad de baño**
Descripción	El sistema deberá permitir al paciente controlar el avatar mediante las rotaciones del codo.

FRQ-031	**Coger cepillo de dientes**
Descripción	El sistema deberá simular que el avatar del paciente agarra un cepillo de dientes mediante una animación.

FRQ-032	**Mojar el cepillo de dientes**
Descripción	El sistema deberá permitir mojar el cepillo de dientes.

FRQ-033	**Cepillar los dientes**
Descripción	El sistema deberá detectar que el cepillo de dientes está a la altura de la cabeza del avatar y puede realizar el gesto de cepillar los dientes.

Requisitos funcionales Actividad Cartas

FRQ-034	**Ambiente actividad de emparejar cartas**
Descripción	El sistema deberá mostrar múltiples parejas de cartas de forma aleatoria que cubre toda la pantalla.

FRQ-035	**Avatar para actividad de emparejar cartas**
Descripción	El sistema deberá simular la mano del paciente mediante un avatar de una mano.

FRQ-036	**Inclinación de la mano de la actividad emparejar cartas**
Descripción	El sistema deberá simular que la mano del avatar se inclina tras el giro de la muñeca del paciente.

FRQ-037	**Seleccionar las cartas**
Descripción	El sistema deberá permitir al paciente seleccionar las cartas mediante el giro de la mano.

FRQ-038	**Eliminar las cartas**
Descripción	El sistema deberá permitir al paciente eliminar las cartas cuando haya seleccionado su pareja

FRQ-039	**Puntuación de actividad de emparejar cartas**
Descripción	El sistema deberá permitir al paciente obtener puntuaciones mediante eliminación de las cartas iguales.

.

4.3 Requisitos de información

FRQ-040	Información personal
Descripción	El sistema deberá almacenar los datos personales de los pacientes registrados (Nombre, Apellidos, Dirección, Teléfono, Código postal, Provincia, Localidad).

FRQ-041	Información medica
Descripción	El sistema deberá almacenar información médica del paciente relevante para la aplicación, miembros de cuerpo afectado y fecha de la lesión.

FRQ-042	Información Actividad de salón
Descripción	El sistema deberá almacenar el nombre del ejercicio, el id del paciente, el ángulo máximo y mínimo, si se ha finalizado la actividad, la fecha, el lado con el que se realizó, y el tiempo empleado.

FRQ-043	Información Actividad de baño
Descripción	El sistema deberá almacenar el nombre del ejercicio, el id del paciente, el número de repeticiones, si se ha finalizado la actividad, la fecha, el lado con el que se realizó, y el tiempo empleado.

FRQ-044	Información Actividad de cartas
Descripción	El sistema deberá almacenar el nombre del ejercicio, el id del paciente, el número de repeticiones, si se ha finalizado la actividad, la fecha, el lado con el que se realizó, y el tiempo empleado.

4.4 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	**Uso de la interfaz**
Descripción	La interfaz del sistema debe de ser tan simple que requiera menos de un día de aprendizaje.

NFR-002	**Requisitos mínimos del sistema**
Descripción	Procesador de 64-bit (x64) dual core de 2.1 GHz (2 núcleos lógicos por cada uno) o mayor. Tarjeta de red wifi. 4 GB de RAM o superior. Tarjeta gráfica que soporte DirectX 11 Windows 7. Android 5.0

NFR-003	**Fiabilidad de las medidas de los ángulos**
Descripción	El margen de error de los ángulos debe de ser de un 10%.

NFR-004	**Implementación software**
Descripción	El sistema deberá desarrollarse bajo las siguientes herramientas: C#, Unity 5.6.2f1, Android estudio 3, Node JS 6.11.4.

NFR-005	**Fecha y hora**
Descripción	Deberá tomar la fecha y hora del reloj del sistema.

NFR-006	**Disponibilidad de Acelerómetro y Giroscopio**
Descripción	El móvil deberá contar con un acelerómetro y un giroscopio.

NFR-007	**Facilidad de instalación de aplicación Android**
Descripción	La aplicación debe ser fácilmente instalable por cualquier usuario a partir de su apk.

4.5 Diagrama de Casos de uso

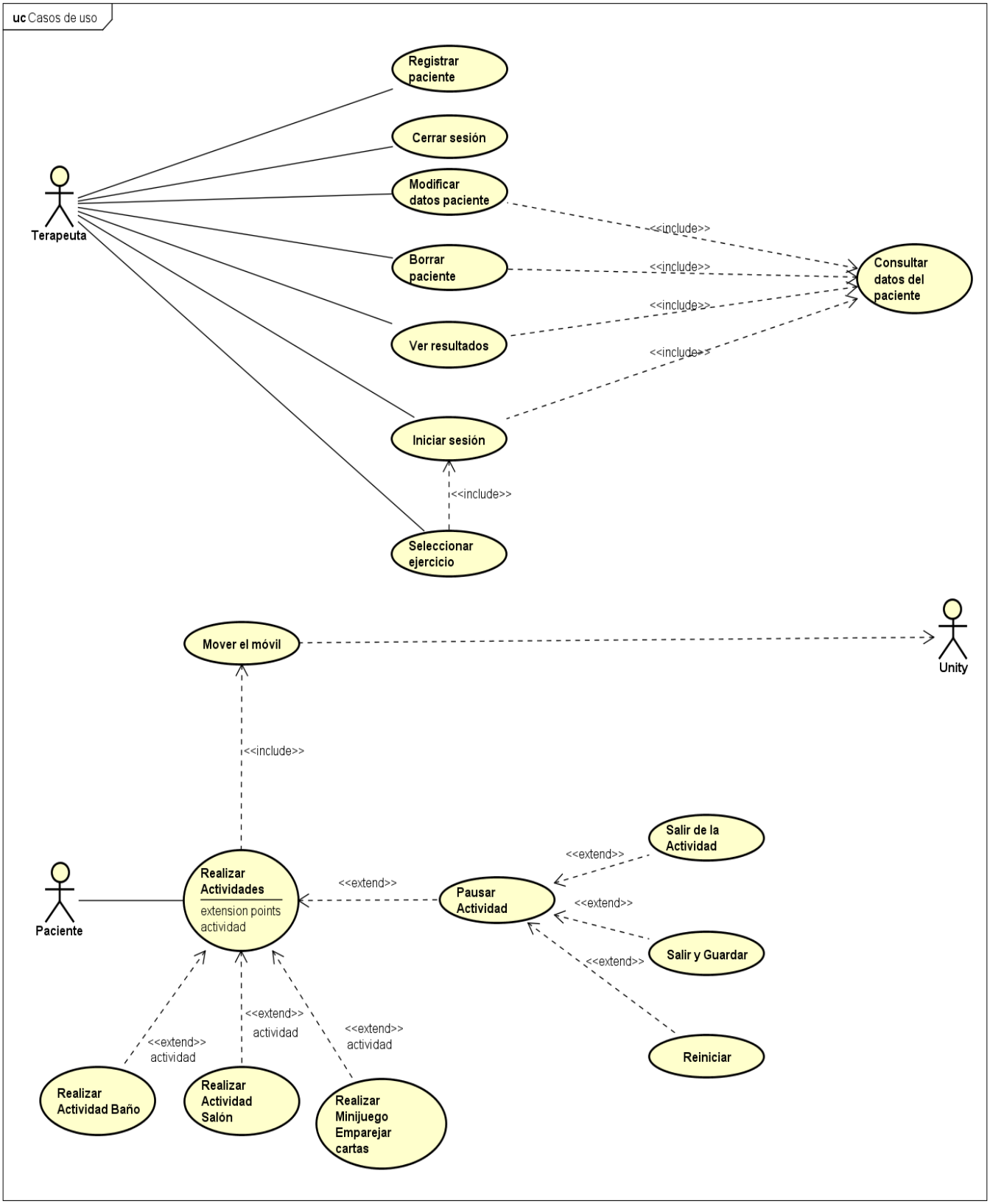

Ilustración 15: Diagrama casos de uso

4.6 Identificación de actores del sistema

- (ACT-001) Terapeuta = Es la persona encargada de manejar la aplicación del ordenador.
- (ACT-002) Paciente = Es la persona que realiza los ejercicios, mediante el uso del sistema móvil.

4.7 Descripción de los Casos de uso

Tabla 15: UC-001 Registrar Paciente

UC-001	**Registrar Paciente**
Descripción	El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cuando el actor Terapeuta (ACT-001), desee registrar un paciente en él sistema
Precondición	--
Secuencia normal	1. El actor Terapeuta (ACT-001) selecciona registrar un paciente. 2. El sistema solicita los datos necesarios para registrar el paciente 3. El actor Terapeuta (ACT-001) introduce los datos solicitados. 4. El sistema comprueba que se han introducido todos los datos necesarios y registra al paciente dentro del sistema.
Postcondición	El paciente se ha registrado en el sistema
Excepciones	4. El sistema detecta que no se han completado todos los campos marcados como obligatorios, o que el id de usuario ya existe. Informa al actor y vuelve al paso 2. 2. El actor Terapeuta (ACT-001) cancela la operación, y el caso de uso finaliza.

Tabla 16: UC-002 Iniciar sesión

UC-002	**Iniciar sesión**
Descripción	El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cuando el actor Terapeuta (ACT-001) desee identificarse ante el sistema.
Precondición	--
Secuencia normal	1. El actor Terapeuta (ACT-001) solicita iniciar sesión. 2. El sistema solicita el nombre de usuario. 3. El actor Terapeuta introduce los datos. 4. El sistema comprueba que el usuario exista e inicia sesión.
Postcondición	El terapeuta ha iniciado la sesión del paciente.
Excepciones	4. El sistema comprueba que el usuario introducido no es correcto, informa al actor y vuelve al paso 2.

Tabla 17: UC-003 Cerrar sesión

UC-003	**Cerrar sesión**
Descripción	El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cuando el actor Terapeuta (ACT-001) desee identificarse ante el sistema.
Precondición	La sesión está iniciada
Secuencia normal	1. El actor Terapeuta (ACT-001) solicita cerrar sesión. 2. El sistema cierra la sesión y vuelve al menú de login.
Postcondición	El terapeuta ha cerrado la sesión.
Excepciones	--

Tabla 18: UC-004 Consultar datos del paciente

UC-004	**Consultar datos del paciente**
Descripción	El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cuando el actor Terapeuta (ACT-001) desee consultar datos del paciente.
Precondición	La sesión está iniciada
Secuencia normal	1. El actor Terapeuta (ACT-001) solicita ver los datos del paciente. 2. El sistema muestra los datos del paciente.
Postcondición	El terapeuta ha consultado los datos, que el sistema le ha mostrado sobre el paciente.
Excepciones	1. Si el actor Terapeuta (ACT-001) selecciona eliminar los datos del paciente, este caso de uso continuará en el caso de uso de “Eliminar paciente”. 1. Si el actor Terapeuta (ACT-001) selecciona modificar los datos del paciente, este caso de uso continuará en el caso de uso de “Modificar datos del paciente”.

Tabla 19: UC-005 Modificar datos del paciente

UC-005	**Modificar datos del paciente**
Descripción	El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cuando el actor Terapeuta (ACT-001) desee modificar los datos del paciente.
Precondición	La sesión está iniciada.
Secuencia normal	1. El actor Terapeuta (ACT-001) selecciona modificar los datos del paciente. 2. El sistema muestra los datos del paciente y solicita los datos que se van a modificar. 3. El actor Terapeuta (ACT-001) introduce las modificaciones. 4. El sistema comprueba los campos y almacena las modificaciones.
Postcondición	Las modificaciones se han guardado.
Excepciones	4. El sistema comprueba que se han dejado campos vacíos, informa al actor y vuelve al paso 2. 3. El actor Terapeuta (ACT-001) cancela la operación, el caso de uso finaliza.

Tabla 20: UC-006 Ver resultados

UC-006	**Ver resultados**
Descripción	El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cuando el actor Terapeuta (ACT-001) desee identificar y ver los resultados de un ejercicio.
Precondición	La sesión está iniciada
Secuencia normal	1. El actor Terapeuta (ACT-001) realiza el caso de uso “Consultar datos del paciente”. 2. El sistema muestra la lista de ejercicios realizados por el paciente. 3. El actor Terapeuta (ACT-001) selecciona el ejercicio del cual quiere ver los resultados. 4. El sistema muestra los resultados obtenidos en ese ejercicio del paciente con la sesión iniciada.
Excepciones	3. El actor Terapeuta (ACT-001) cancela la acción y el caso de uso finaliza.

Tabla 21: UC-007 Borrar paciente

UC-007	**Borrar paciente**
Descripción	El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cuando el actor Terapeuta (ACT-001) desee borrar al paciente del sistema.
Precondición	La sesión está iniciada.
Secuencia normal	1. El actor Terapeuta (ACT-001) selecciona eliminar los datos del paciente. 2. El sistema muestra los datos del paciente y solicita confirmación. 3. El actor terapeuta (ACT-001) confirma la eliminación de los datos del paciente. 4. El sistema elimina los datos del paciente y vuelve al menú de login.
Postcondición	El paciente ha sido borrado.
Excepciones	3. El actor Terapeuta (ACT-001) cancela la operación y el caso de uso finaliza.

Tabla 22: UC-008 Seleccionar ejercicio

UC-008	**Seleccionar ejercicio**
Descripción	El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cuando el actor Terapeuta (ACT-001) desee comenzar uno de los ejercicios.
Precondición	La sesión está iniciada
Secuencia normal	1. El actor Terapeuta (ACT-001) marca el brazo con el cual quiera realizar la acción y solicita realizar un ejercicio. 2. El sistema muestra los ejercicios que están disponibles. 3. El actor Terapeuta (ACT-001) selecciona un ejercicio. 4. El sistema inicia el ejercicio.
Postcondición	El ejercicio ha iniciado.
Excepciones	1. El actor Terapeuta (ACT-001) cancela la operación y el caso de uso finaliza. 3. El actor terapeuta (ACT-001) selecciona el ejercicio Actividad de salón, Actividad Baño, o Actividad Cartas, el caso de uso continua en el caso de uso “Realizar Actividad”.

Tabla 23: UC-009 Realizar Actividad

UC-009	**Realizar Actividad**
Descripción	El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cuando el actor Paciente (ACT-002) desee realizar una actividad.
Precondición	El Terapeuta debe haber realizado el caso de uso seleccionar ejercicio.
Secuencia normal	1. El actor Paciente (ACT-002) coge el móvil y se posiciona frente a la pantalla. 2. Punto Extensión: Actividad. 3. El sistema toma las mediciones. 4. El actor Paciente (ACT-002) completa el ejercicio. 5. El sistema almacena el resultado asociado a la fecha actual y al paciente, solicitando si se desea cambiar de brazo. 6. El actor Paciente (ACT-002) selecciona continuar con el mismo brazo. 7. El sistema reinicia él ejercicio.
Postcondición	El ejercicio se ha realizado.
Excepciones	1. ,4., 6. El actor Paciente (ACT-002) decide cancelar la operación y el caso de uso finaliza. 2. El actor Paciente (ACT-002) selecciona pausar actividad y el caso de uso continua en “pausar actividad”. 6. Si el actor paciente (ACT-002) selecciona cambiar de brazo, modifica el brazo y el Caso de Uso continua en el paso 7. 6. Si el actor paciente (ACT-002) selecciona salir, el Caso de uso “Salir” comienza.

Tabla 24: UC-010 Actividad Salón

UC-010	**Actividad 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 salón. Punto de Extensión Actividad salón.
Precondición	El Terapeuta debe haber realizado el caso de uso seleccionar ejercicio.
Secuencia normal	1. El actor Paciente (ACT-002), selecciona un objeto, la taza o el cepillo. 2. El sistema carga el ejercicio seleccionado (ejercicio de peinarse o ejercicio de tomar el té), mostrando una cuenta atrás, tras esta solicita al paciente coger el objeto. 3. El actor paciente (ACT-002) coge el objeto. 4. El sistema solicita al paciente que lleve el objeto hasta la cabeza, (boca si el objeto es la taza o pelo si el objeto es el cepillo) 5. El actor Paciente (ACT-002) lleva el objeto hasta la cabeza.
Postcondición	--
Excepciones	--

Tabla 25: UC-011 Actividad Baño

UC-011	**Actividad Baño**
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 baño. Punto de Extensión Actividad baño.
Precondición	El actor Terapeuta debe haber realizado el caso de uso seleccionar ejercicio.
Secuencia normal	1. El actor Paciente (ACT-002) introduce el número de repeticiones a realizar. 2. El sistema muestra una cuenta atrás, cuando esta finaliza solicita al actor paciente que coja el cepillo de dientes. 3. El actor paciente (ACT-002) coge el cepillo de dientes. 4. El sistema solicita al paciente que moje el cepillo de dientes en el agua. 5. El actor paciente (ACT-002) moja el cepillo de dientes en el agua. 6. El sistema solicita al paciente que lleve el cepillo de dientes hasta la boca. 7. El actor paciente (ACT-002) lleva el cepillo de dientes hasta la boca. 8. El sistema solicita al paciente que se cepille los dientes. 9. El actor paciente (ACT-002) se cepilla los dientes.
Postcondición	--
Excepciones	--

Tabla 26: UC-012 Actividad Emparejar Cartas

UC-012	**Actividad Emparejar Cartas**
Descripción	El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cuando el actor Paciente (ACT-001) desee realizar la actividad cartas. Punto de Extensión Actividad Emparejar cartas.
Precondición	El actor Terapeuta debe haber realizado el caso de uso seleccionar ejercicio.
Secuencia normal	1. El sistema muestra las instrucciones y carga el ejercicio mostrando una cuenta atrás, cuando esta finaliza solicita al actor paciente eliminar las parejas de cartas. 2. El actor Paciente selecciona las parejas de cartas.
Postcondición	--
Excepciones	--

Tabla 27: UC-013 Pausar Actividad

UC-013	**Pausar Actividad**
Descripción	El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cuando el actor Paciente (ACT-002) desee pausar una actividad.
Precondición	El paciente debe de estar realizando una actividad.
Secuencia normal	1. El actor Paciente (ACT-002) pausa la actividad. 2. El sistema muestra diferentes opciones. 3. El actor Paciente (ACT-002) reanuda la actividad. 4. El sistema recupera la actividad en pausa.
Postcondición	El ejercicio se ha pausado.
Excepciones	2. El actor Paciente (ACT-002) selecciona la opción de salir, el caso de uso “salir de la actividad” comienza. 2. El actor Paciente (ACT-002) selecciona la opción de salir y guardar, el caso de uso “salir de la actividad y guardar” comienza. 2. El actor Paciente (ACT-002) selecciona reiniciar ejercicio, el caso de uso “reiniciar” comienza.

Tabla 28: UC-014 Salir de la actividad

UC-014	**Salir de la actividad**
Descripción	El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cuando el actor Paciente (ACT-002) desee salir de una actividad.
Precondición	El paciente debe haber pausado o completado una actividad
Secuencia normal	1. El actor Paciente (ACT-002) selecciona salir de la actividad. 2. El sistema finaliza la actividad en pausa sin guardar datos en la base de datos y carga la escena del menú principal.
Postcondición	El ejercicio ha finalizado.
Excepciones	--

Tabla 29: UC-015 Salir de la actividad y guardar

UC-015	**Salir de la actividad y guardar**
Descripción	El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cuando el actor Paciente (ACT-002) desee salir de una actividad.
Precondición	El paciente debe haber pausado una actividad
Secuencia normal	1. El actor Paciente (ACT-002) selecciona salir de la actividad. 2. El sistema realiza el caso de uso “Finalizar Actividad” y carga la escena del menú principal.
Postcondición	El ejercicio ha finalizado y se han guardado los datos.
Excepciones	--

Tabla 30: UC-016 Reiniciar

UC-016	**Reiniciar**
Descripción	El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cuando el actor Paciente (ACT-002) desee reiniciar una actividad.
Precondición	El paciente debe haber pausado o completado una actividad
Secuencia normal	1. El actor Paciente (ACT-002) selecciona reiniciar la actividad. 2. El sistema finaliza la actividad en pausa y vuelve a cargarla.
Postcondición	El ejercicio se ha vuelto a cargar.
Excepciones	--

4.8 Diagrama de dominio

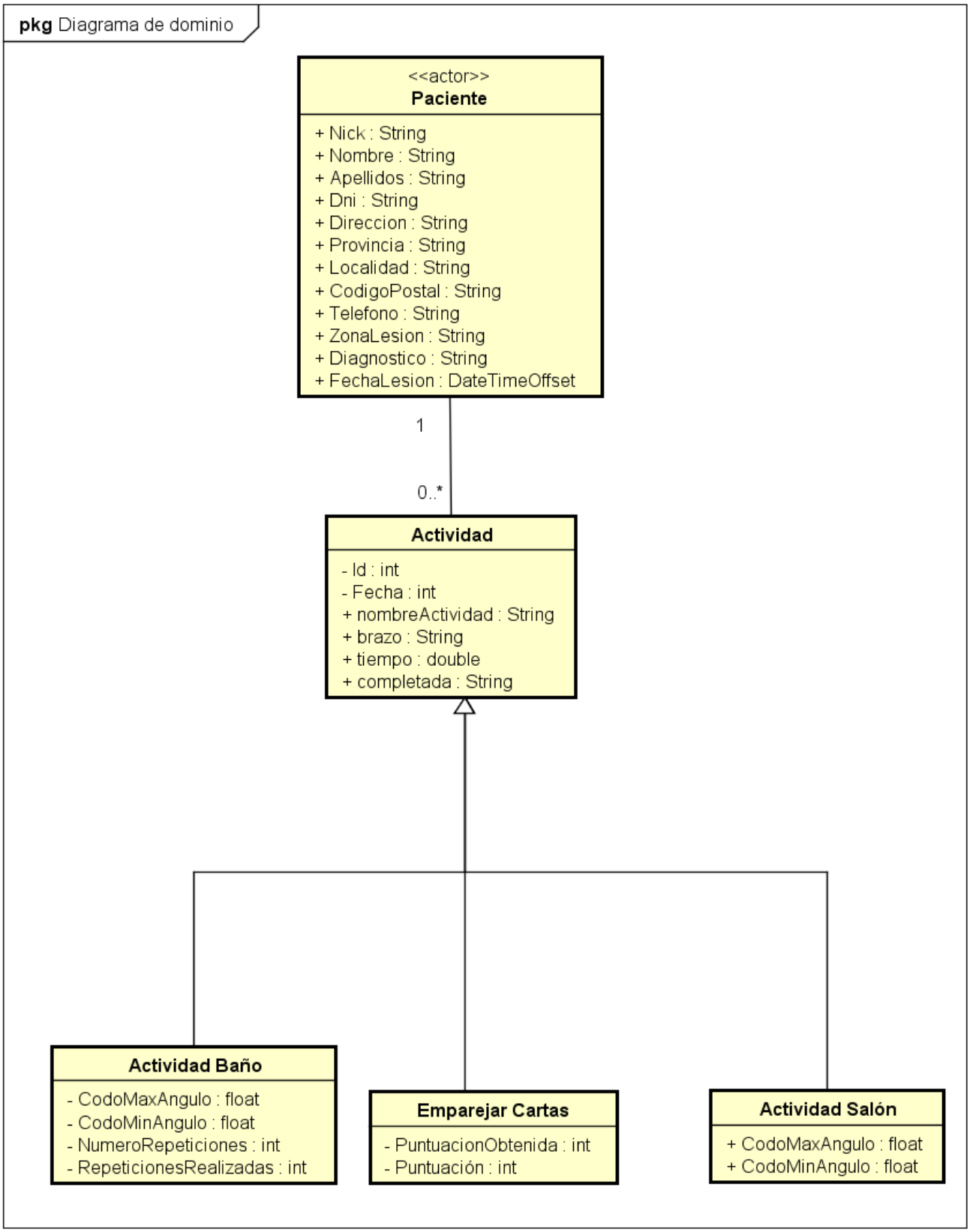

Ilustración 16: Diagrama de dominio

Capítulo 5

5 Diseño

5.1 Introducción

En este capítulo se tratan los pasos seguidos para la elaboración de la aplicación en la parte de Unity y Android, como se muestra en el siguiente diagrama con la arquitectura inicial.

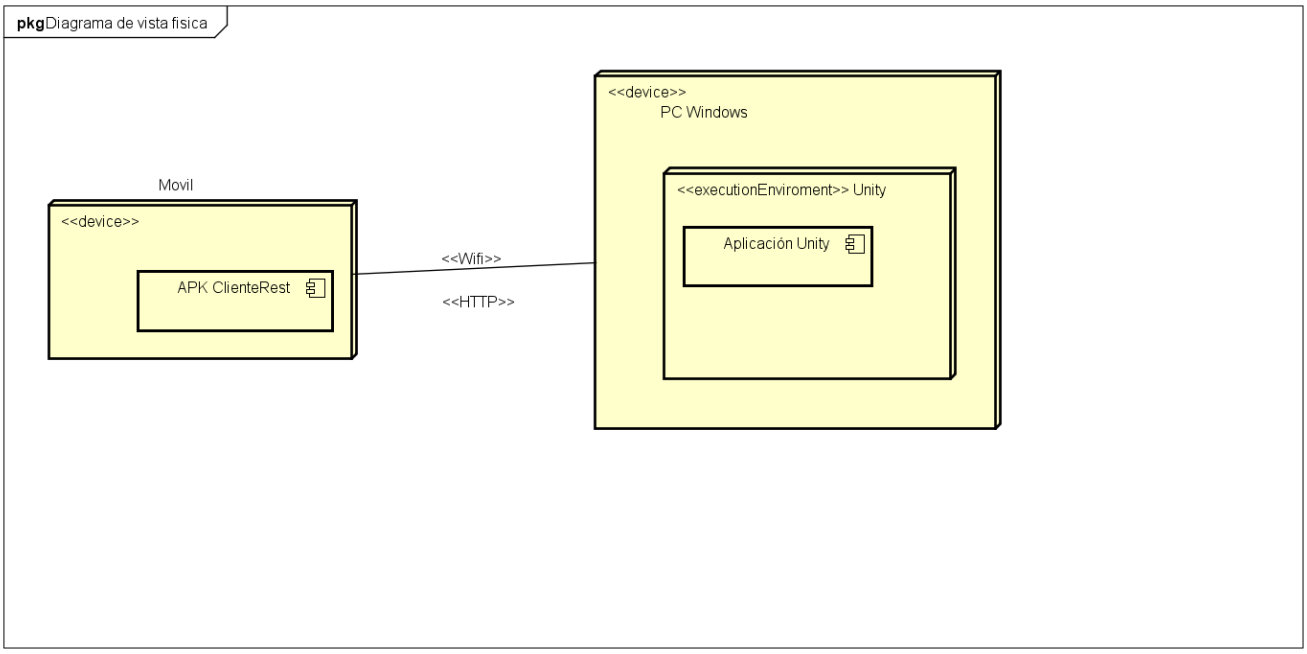

Ilustración 17: Arquitectura inicial

También se mostrarán los diagramas de secuencia, de clases de diseño, de vista física, de clases y el de entidad relación de la base de datos.

5.2 Parte Unity

La mayoría de las secuencias de comandos que se escriben en Unity son clases que heredan del componente MonoBehaviour. Para controlar el objeto, este tipo de scripts pueden reemplazar un conjunto de métodos virtuales predefinidos. Queda claro, que además de todos los componentes definidos, un programador puede desarrollar sus propios componentes y crear funcionalidades adicionales.

Los scripts en todo momento deben heredar de la clase MonoBehaviour, siguiendo un ciclo de vida en el que las fases básicas podrían ser:

- **Start():** se ejecuta una sola vez al habilitar un script (una vez en la vida del script) y antes de que cualquiera de los métodos Update se llame por primera vez.
- **Update():** se ejecuta una vez cada frame. Es la función más utilizada para implementar cualquier tipo de comportamiento del juego.
- **FixedUpdate():** se puede ejecutar más de una vez por cada frame. Se utiliza en lugar de Update() cuando se trata de manejar los componentes relacionados con la física, como podría ser RigidBody .

- **Awake():** exactamente igual que Start(). Inicializa componentes de objetos, este solo se llama una vez durante la vida de la instancia del script(independientemente que esté este habilitado o no) y se llama antes que Start(). Además, es llamado después de que todos los objetos sean inicializados para una comunicación efectiva entre ellos.
- **OnEnable():** es llamado cuando el objeto se habilita/activa.
- **OnGUI():** se puede ejecutar varias veces por frame y se llama para procesar y manejar eventos GUI.
- **LateUpdate():** es llamado en cada frame y después de que sean llamadas todas las funciones de actualización (Update) Ej: Camara: realiza seguimiento de los objetos que podrían haberse movido dentro de un Update.
- **OnDisable():** es llamado cuando el objeto se deshabilita/desactiva.
- **OnDestroy():** es llamado una vez durante la vida del script. Solo podrá ser invocado en objetos que hayan sido previamente activados y este será destruido.

Diagrama de flujo MonoBehaviour

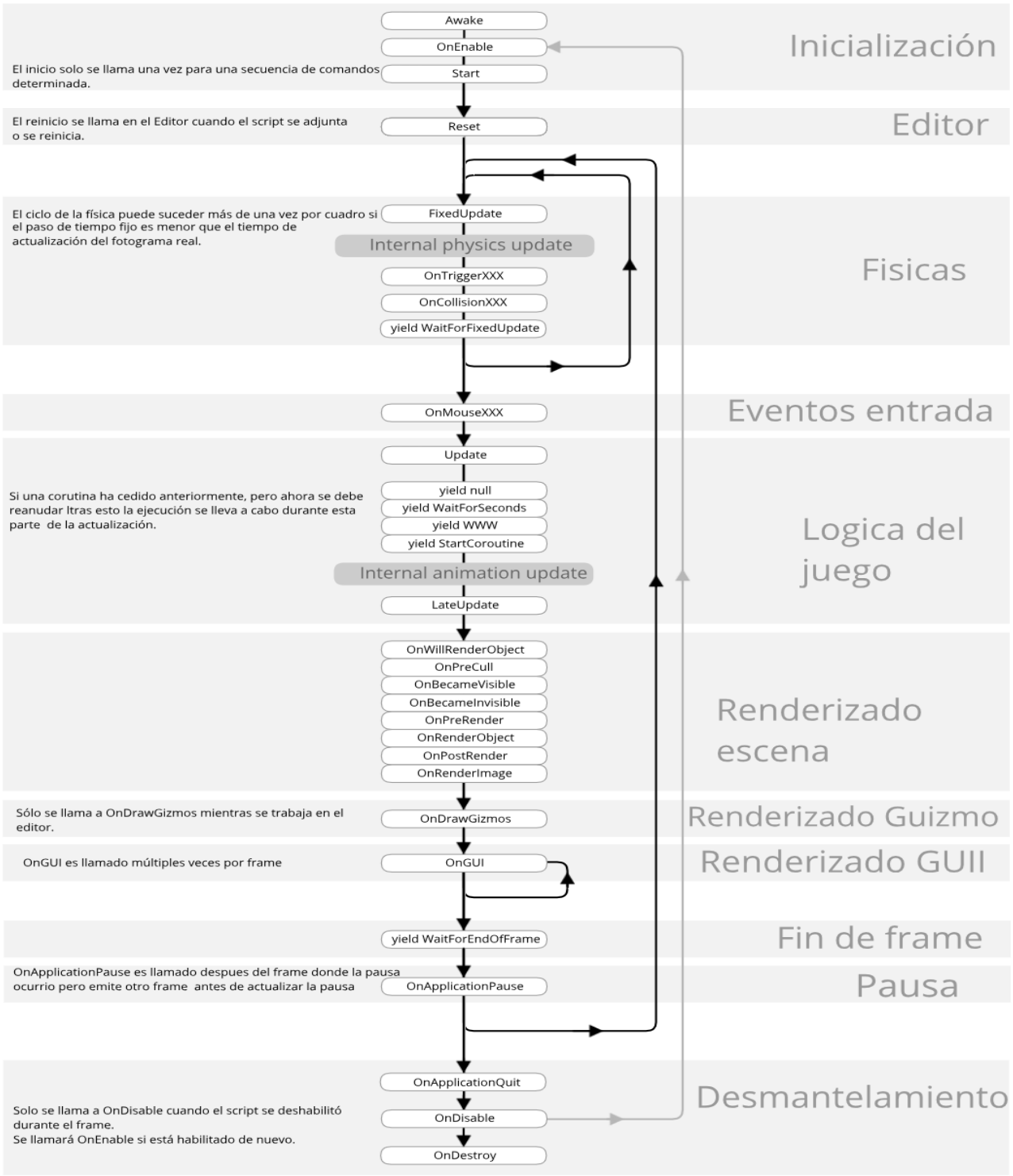

Ilustración 18: Diagrama de flujo MonoBehaviour

5.2.1 Diagramas de Secuencia

Identificar

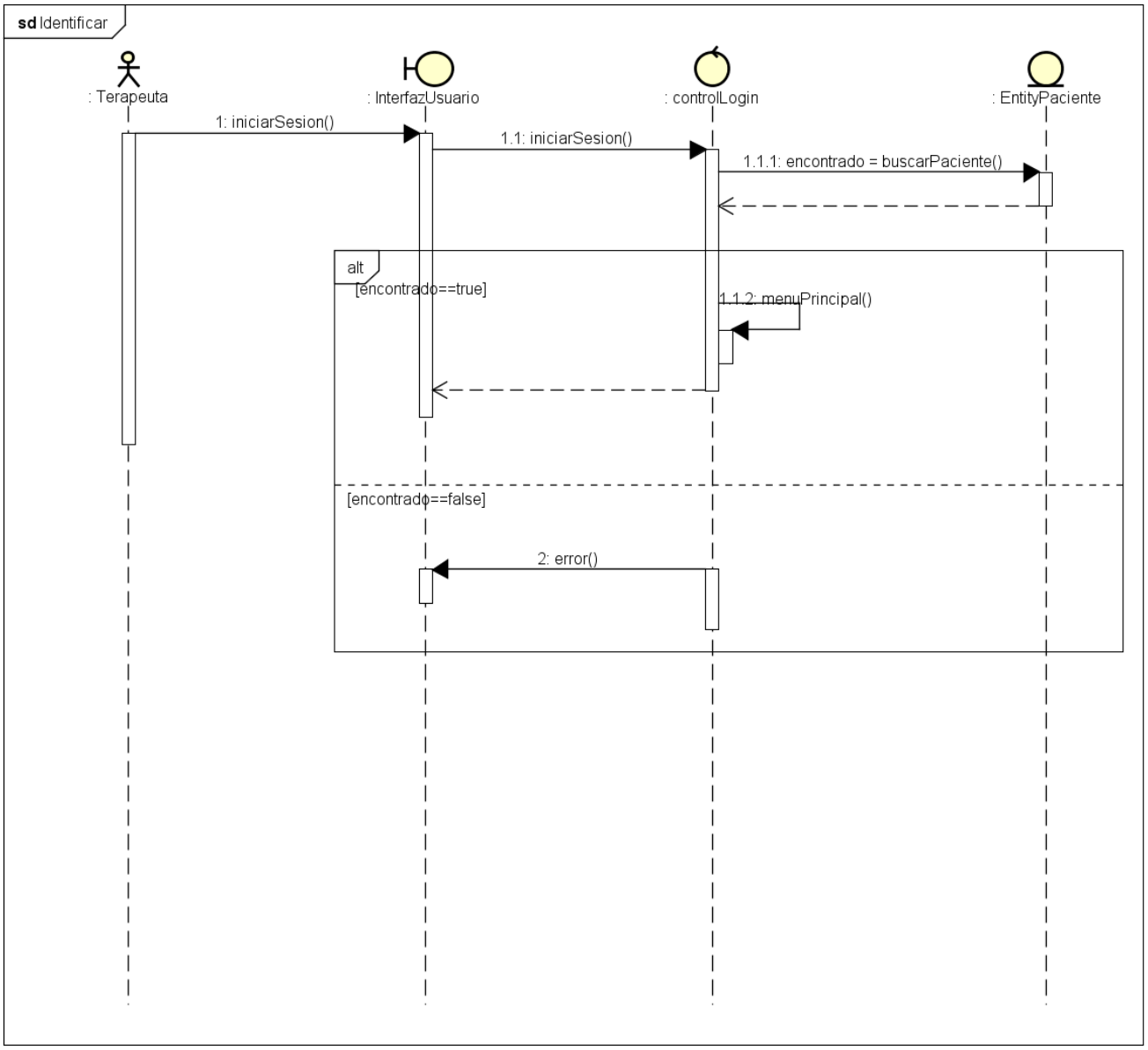

Ilustración 19: Diagramas de Secuencia-Identificar

Registrar

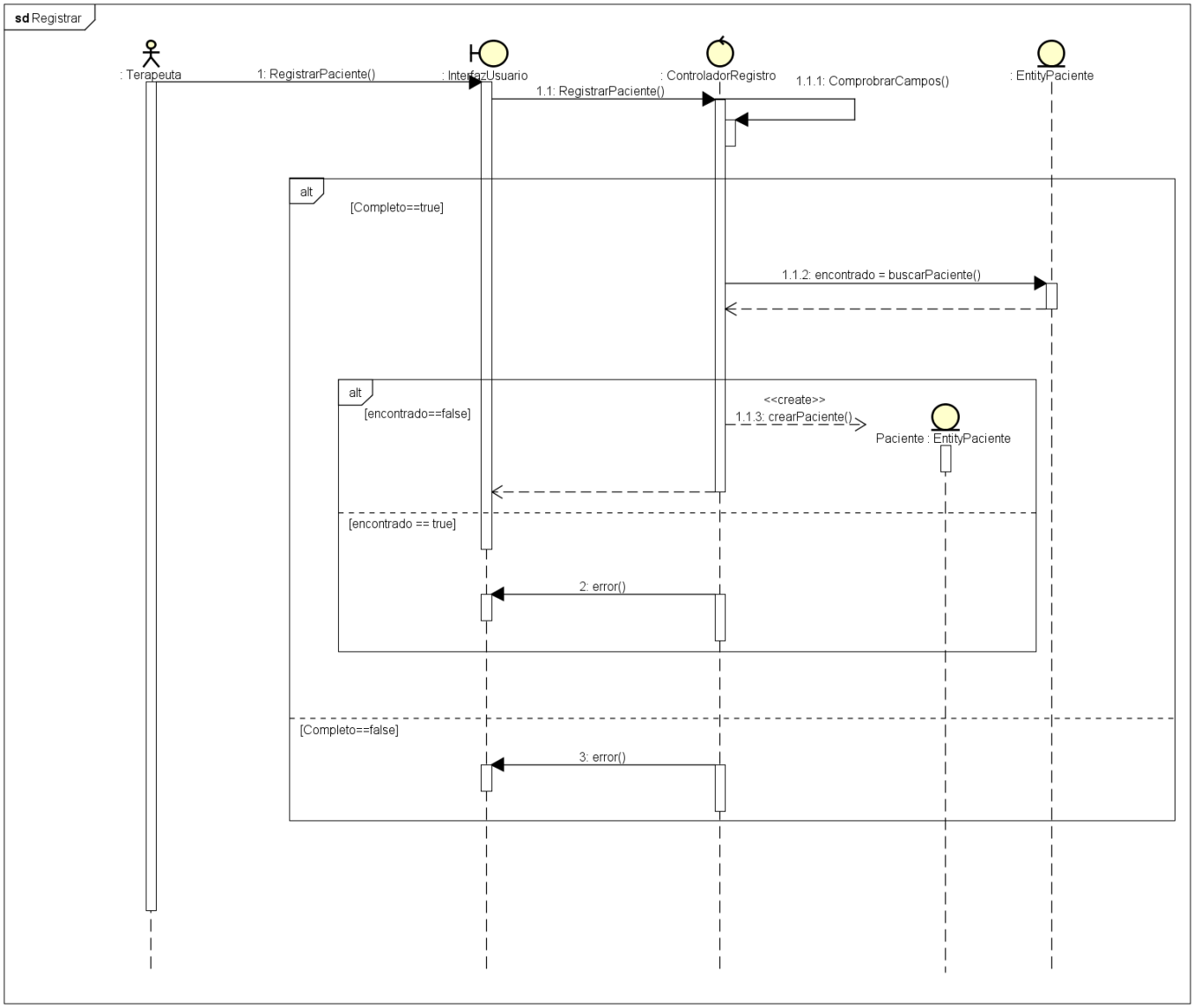

Ilustración 20: Diagramas de Secuencia-Registrar

Modificar Datos Paciente

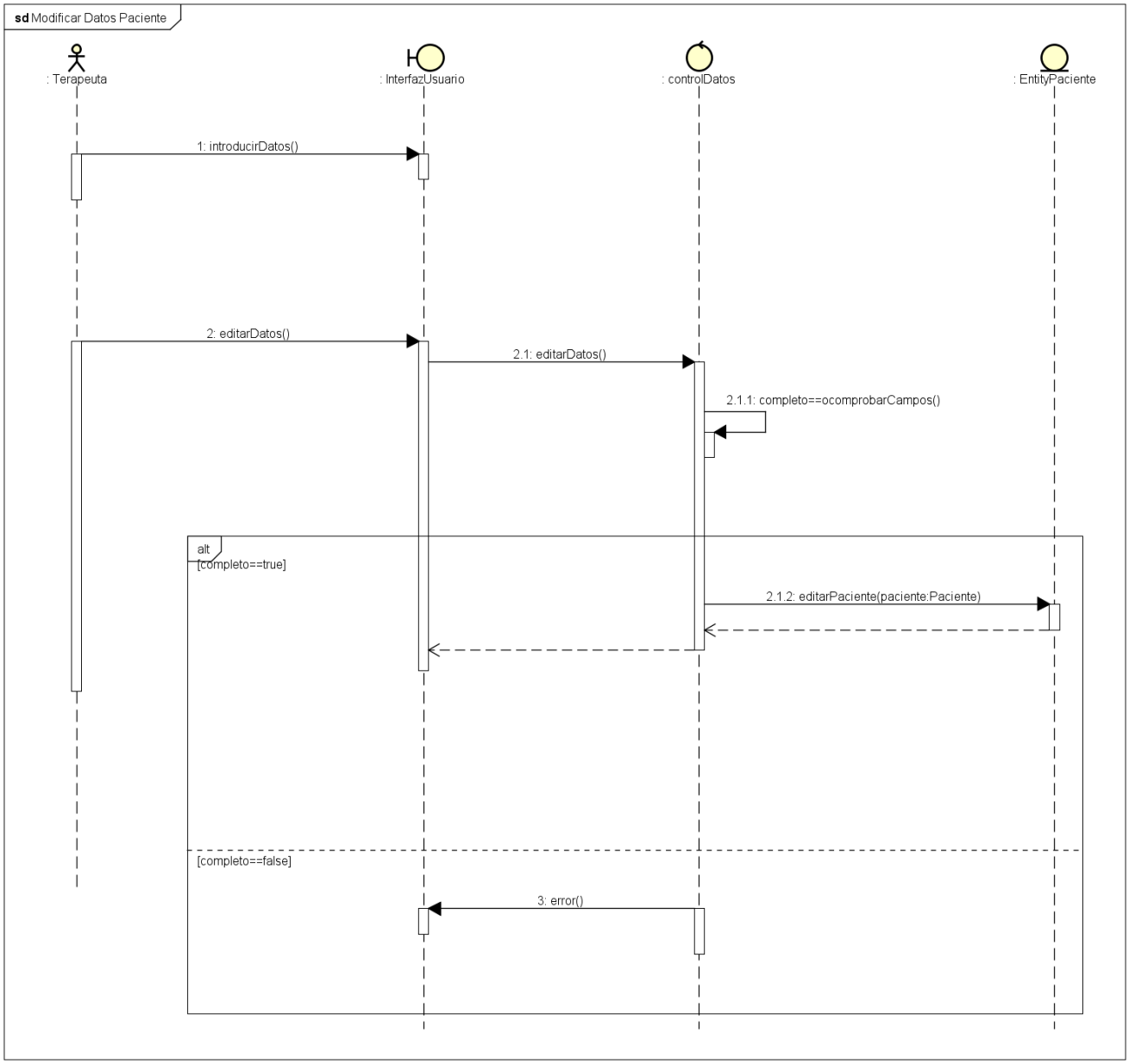

Ilustración 21: Diagramas de Secuencia-Modificar datos pacientes

Eliminar Paciente

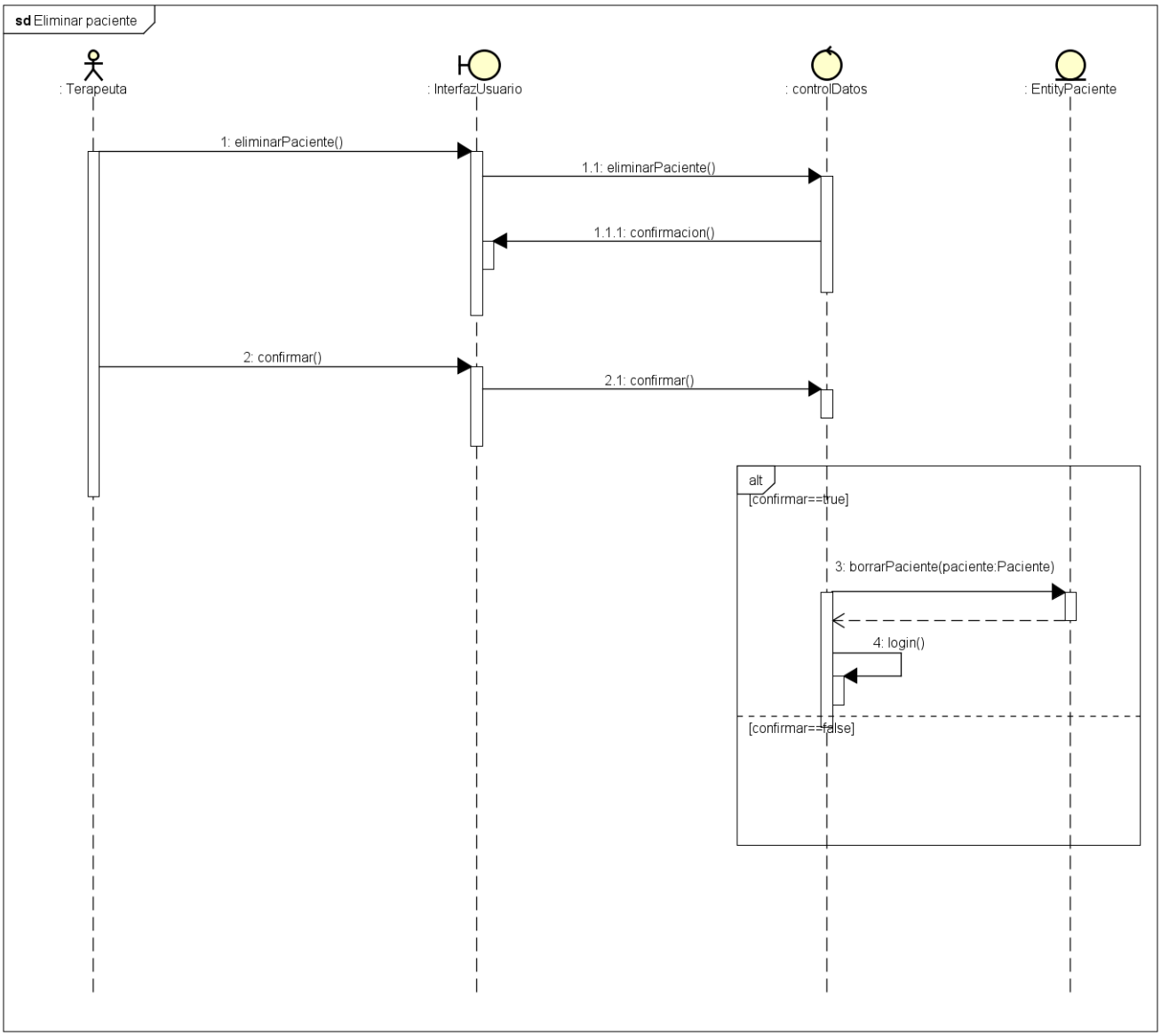

Ilustración 22: Diagramas de Secuencia-Eliminar Paciente

Consultar datos pacientes

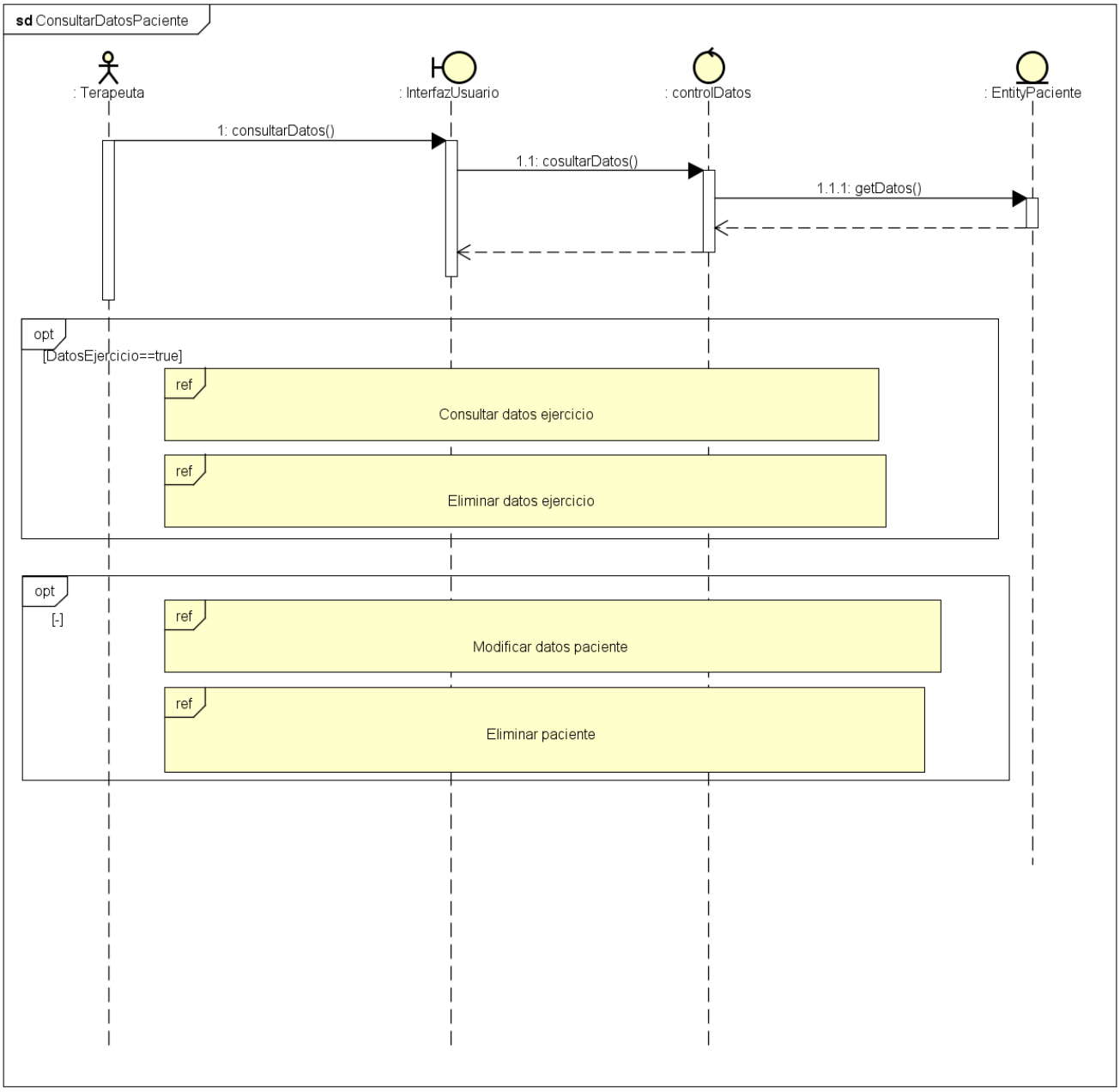

Ilustración 23: Diagramas de Secuencia-Consultar datos

Cerrar sesión

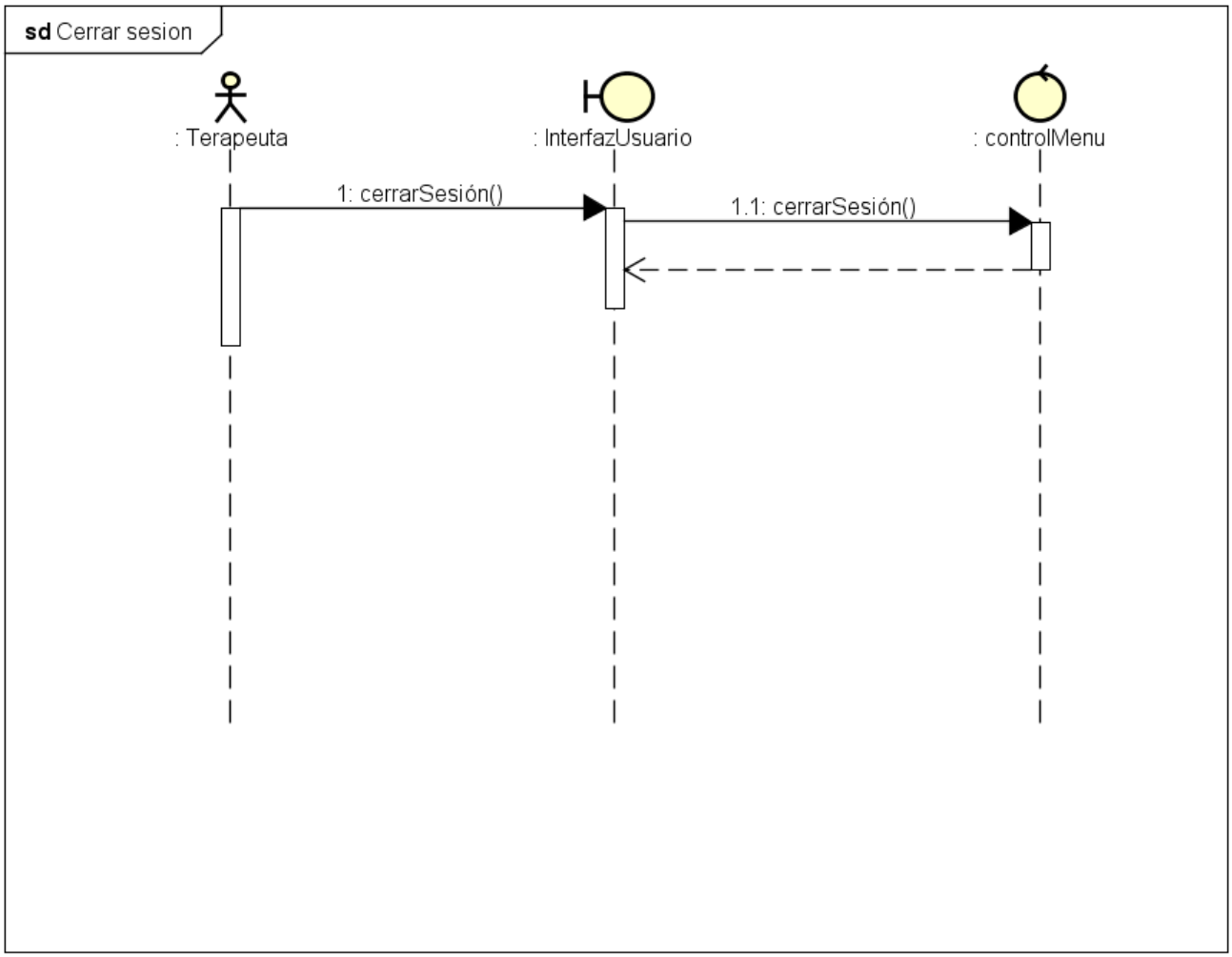

Ilustración 24: Diagramas de Secuencia-Cerrar sesión

Consultar datos ejercicio

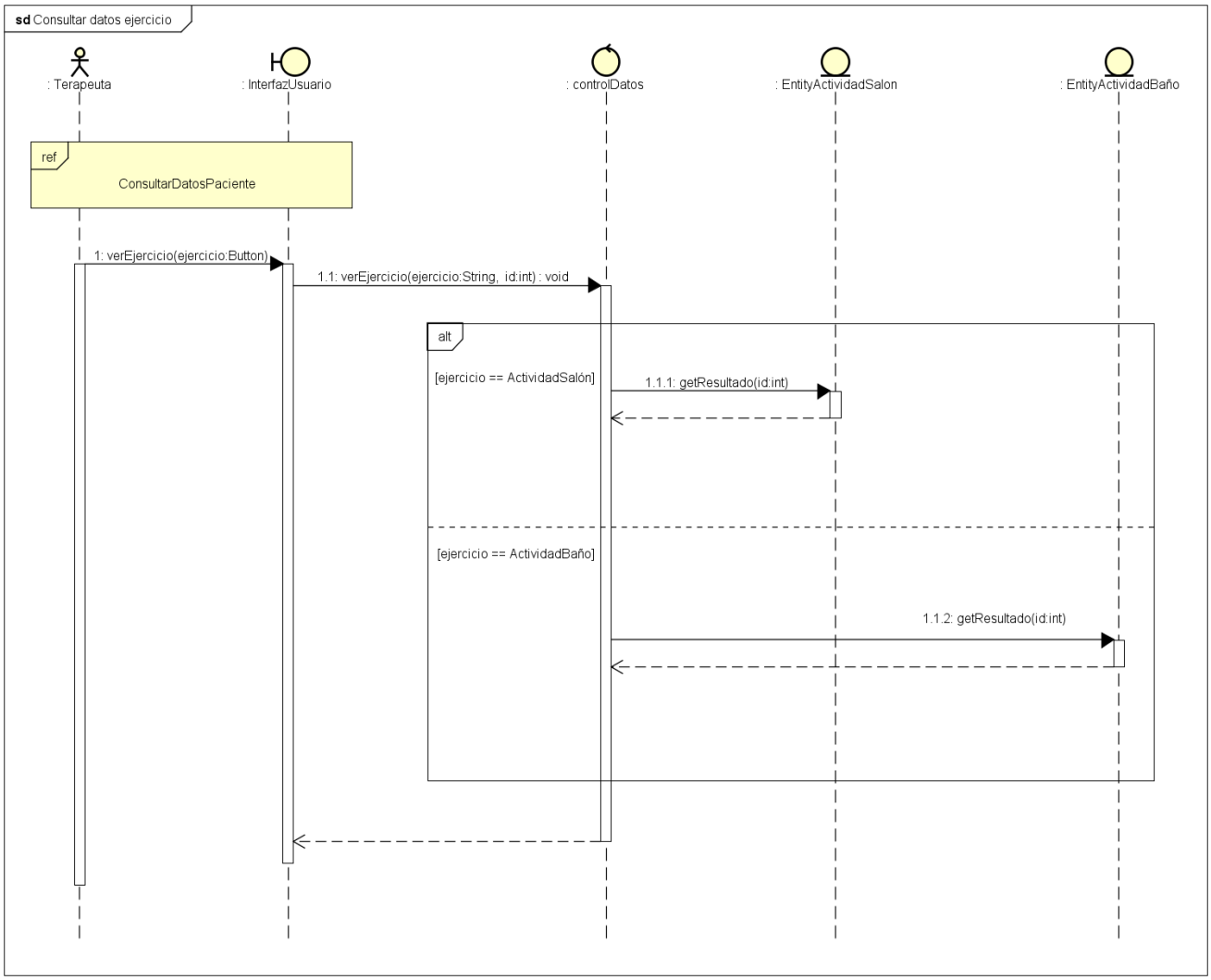

Ilustración 25: Diagramas de Secuencia-Datos ejercicio

Eliminar datos ejercicio

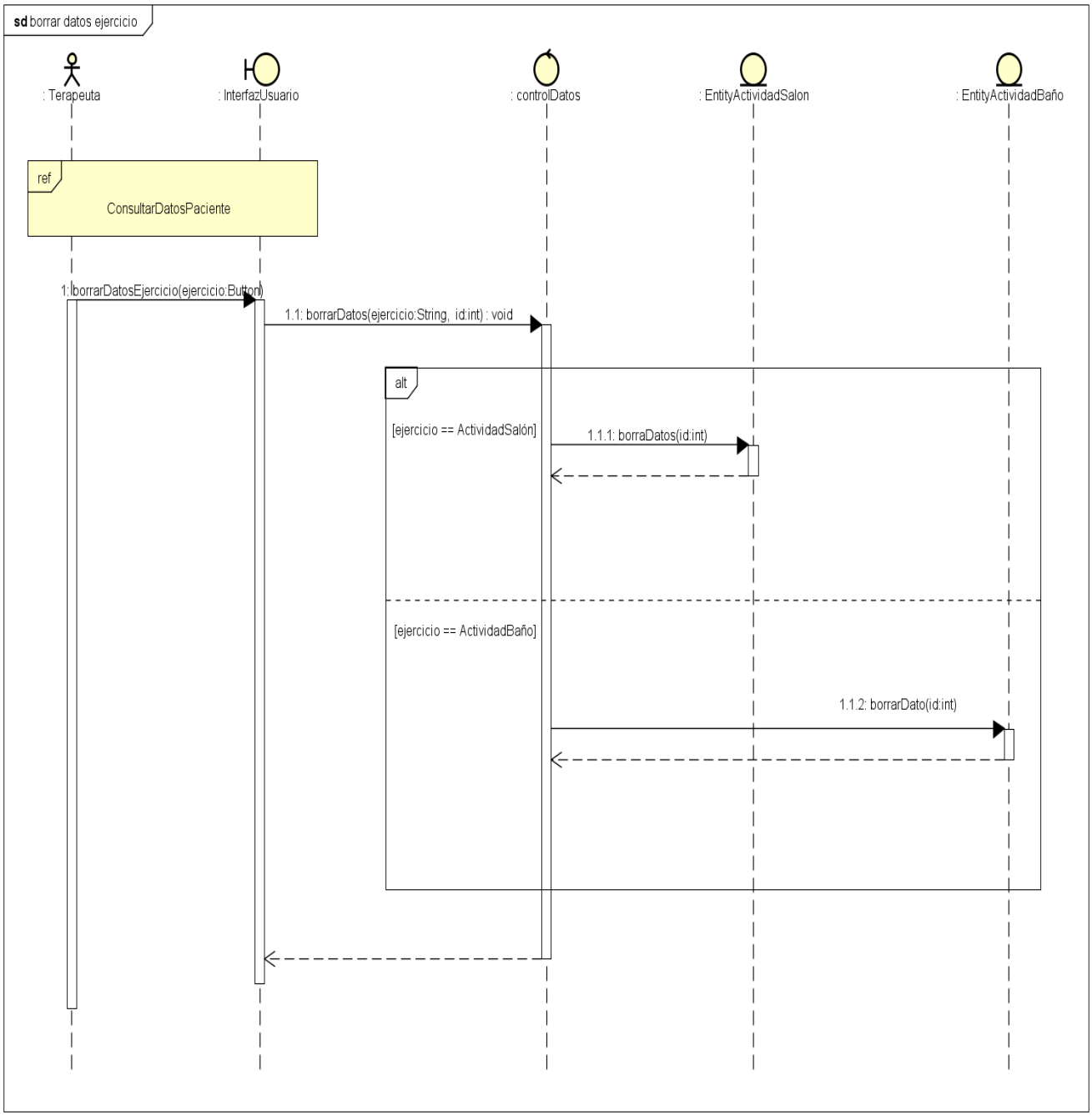

Ilustración 26: Diagramas de Secuencia-Eliminar datos ejercicio

Seleccionar ejercicio

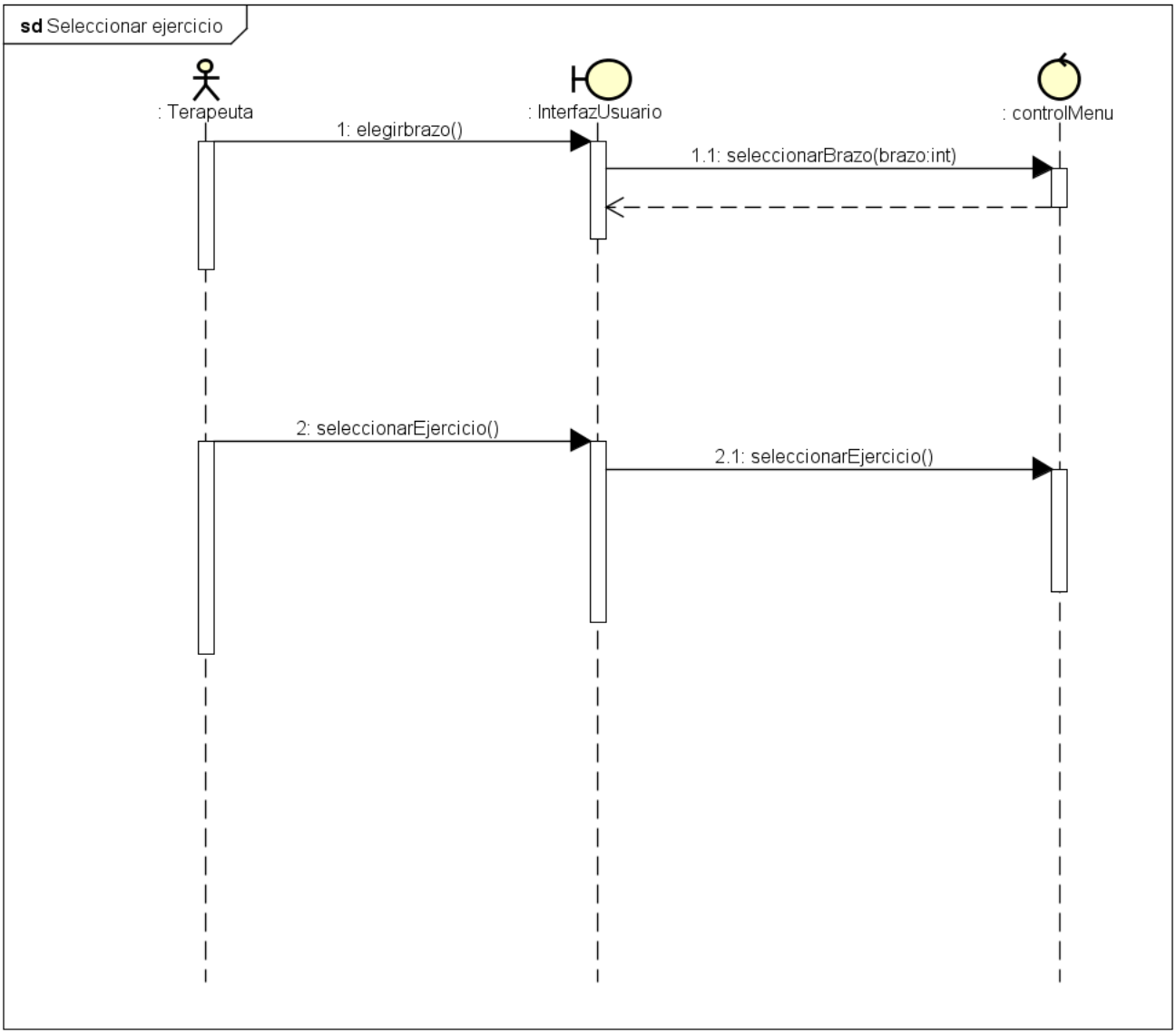

Ilustración 27: Diagramas de Secuencia-Seleccionar ejercicio

Realizar Actividad

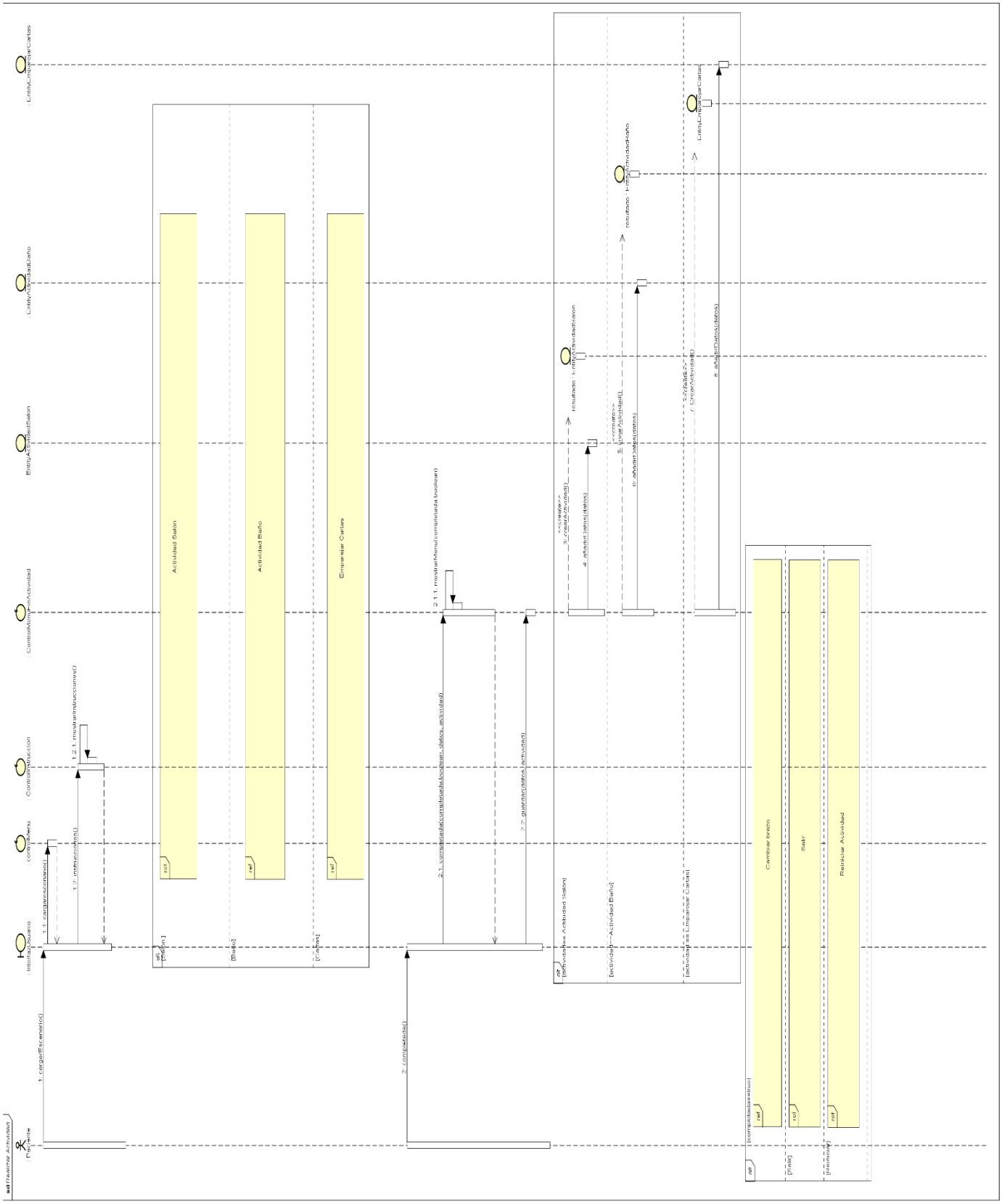

Ilustración 28: Diagramas de Secuencia-Realizar actividad

Actividad salón

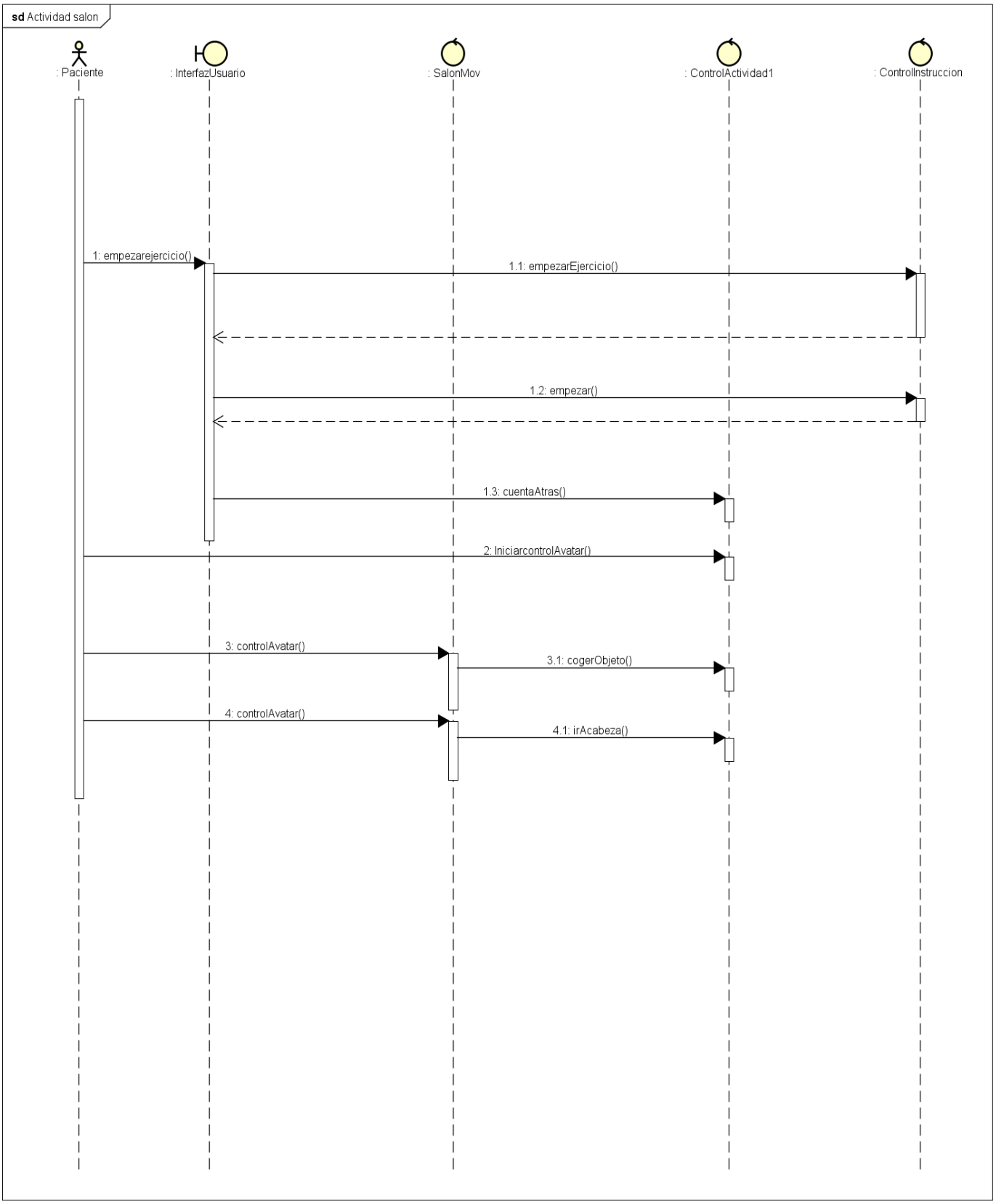

Ilustración 29: Diagramas de Secuencia-Actividad salón

Actividad baño

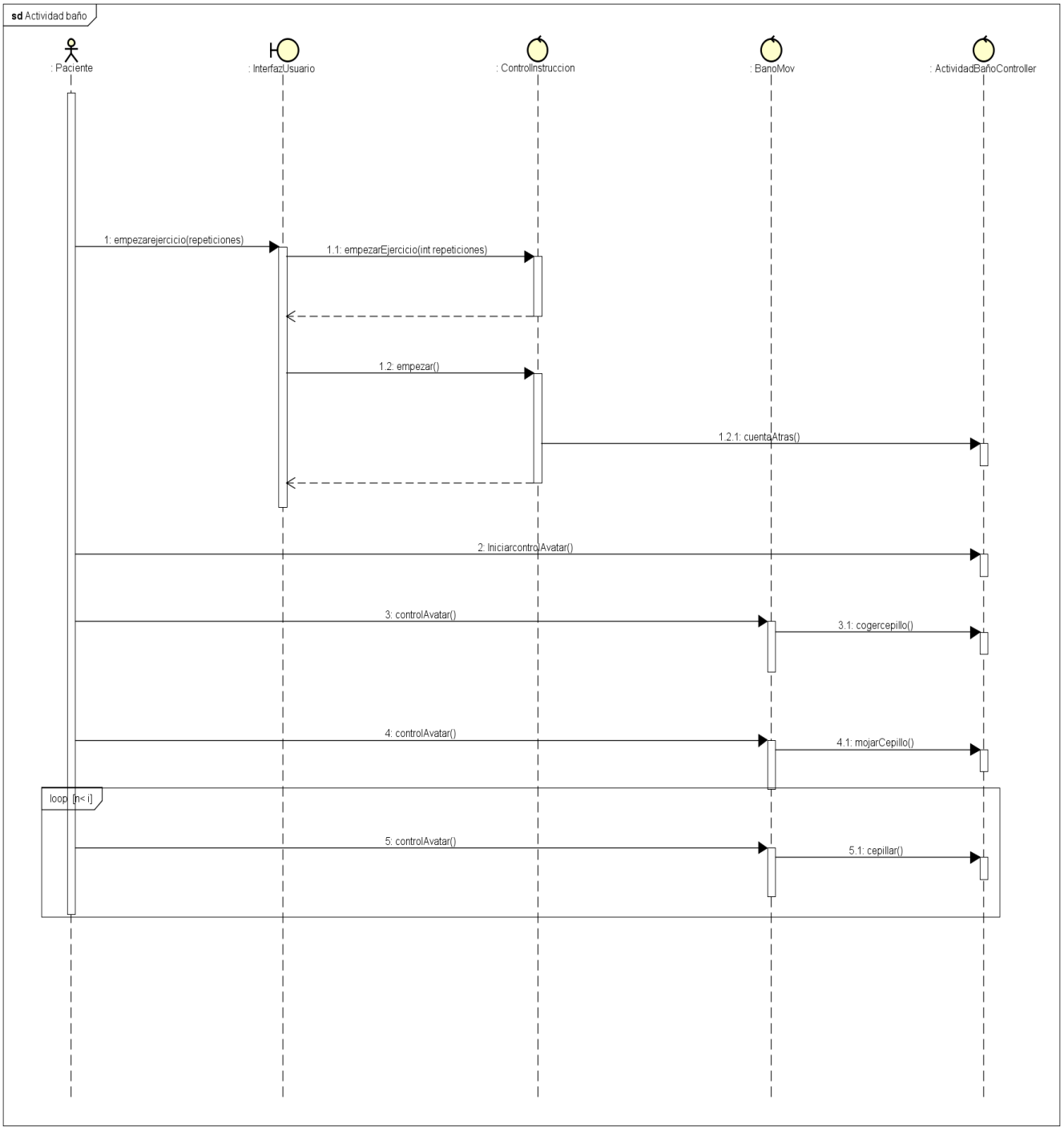

Ilustración 30: Diagramas de Secuencia-Actividad baño

Emparejar Cartas

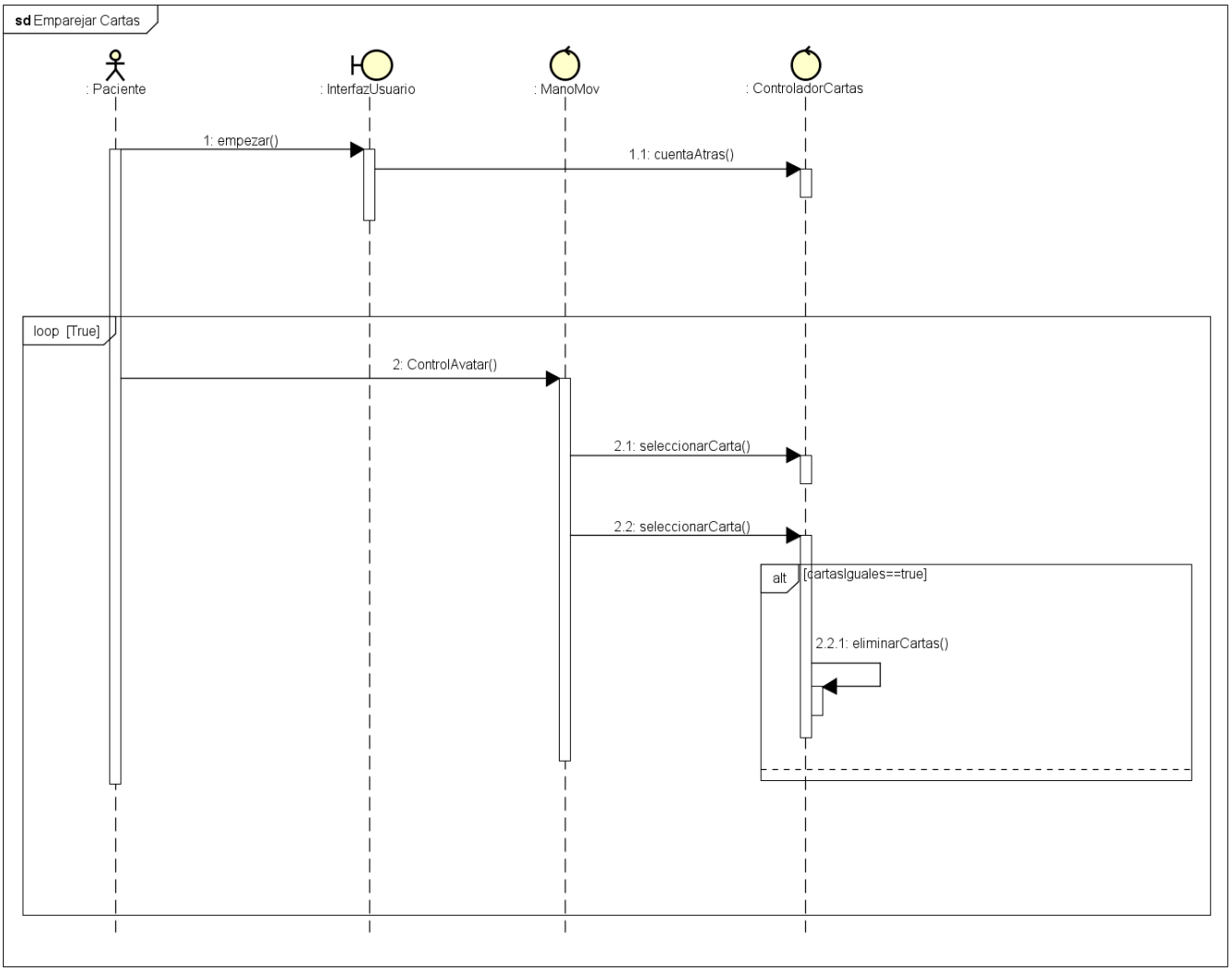

Ilustración 31:Diagramas de Secuencia-Emparejar cartas

Pausar actividad

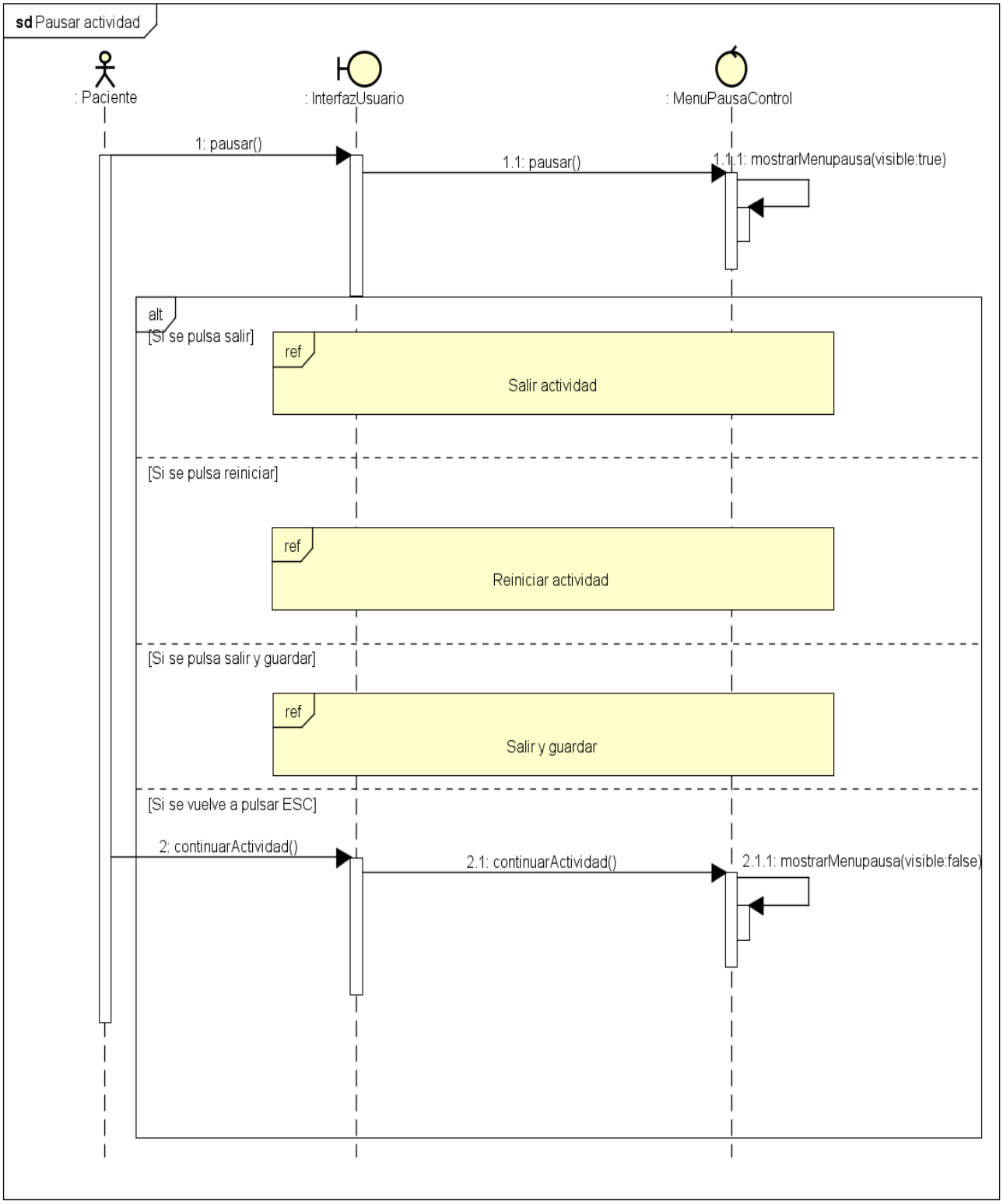

Ilustración 32: Diagramas de Secuencia-Pausar actividad

Salir actividad

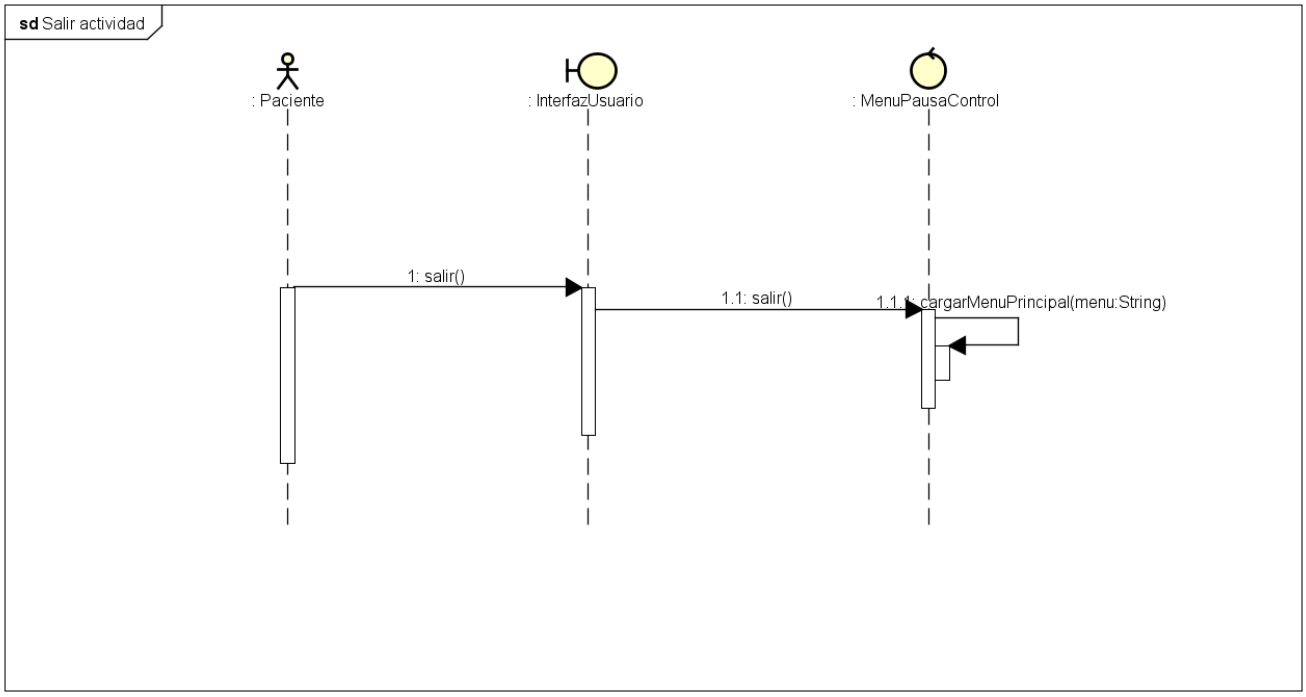

Ilustración 33: Diagramas de Secuencia-Salir actividad

Salir y Guardar

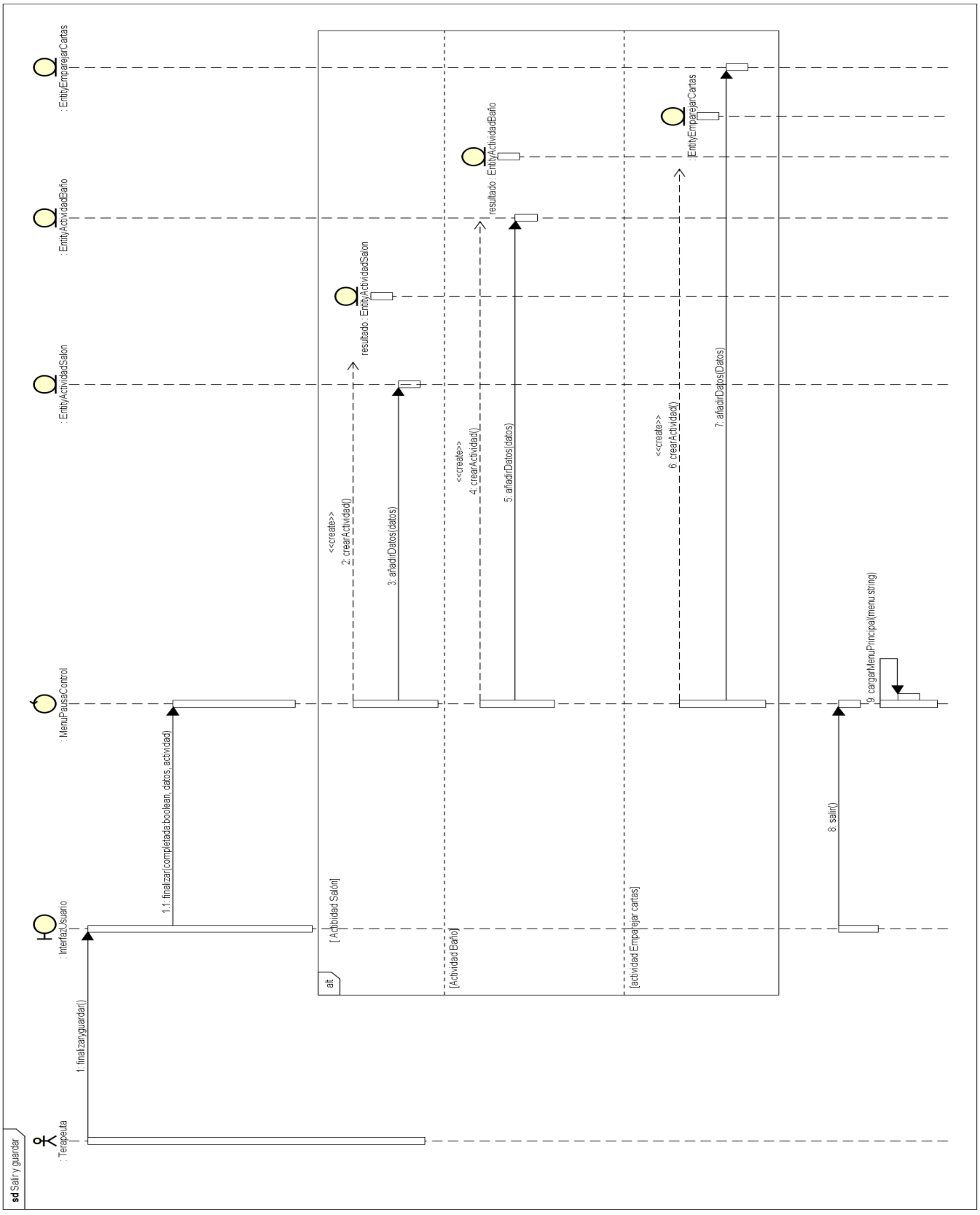

Ilustración 34: Diagrama de secuencia-Salir y Guardar

Reiniciar actividad

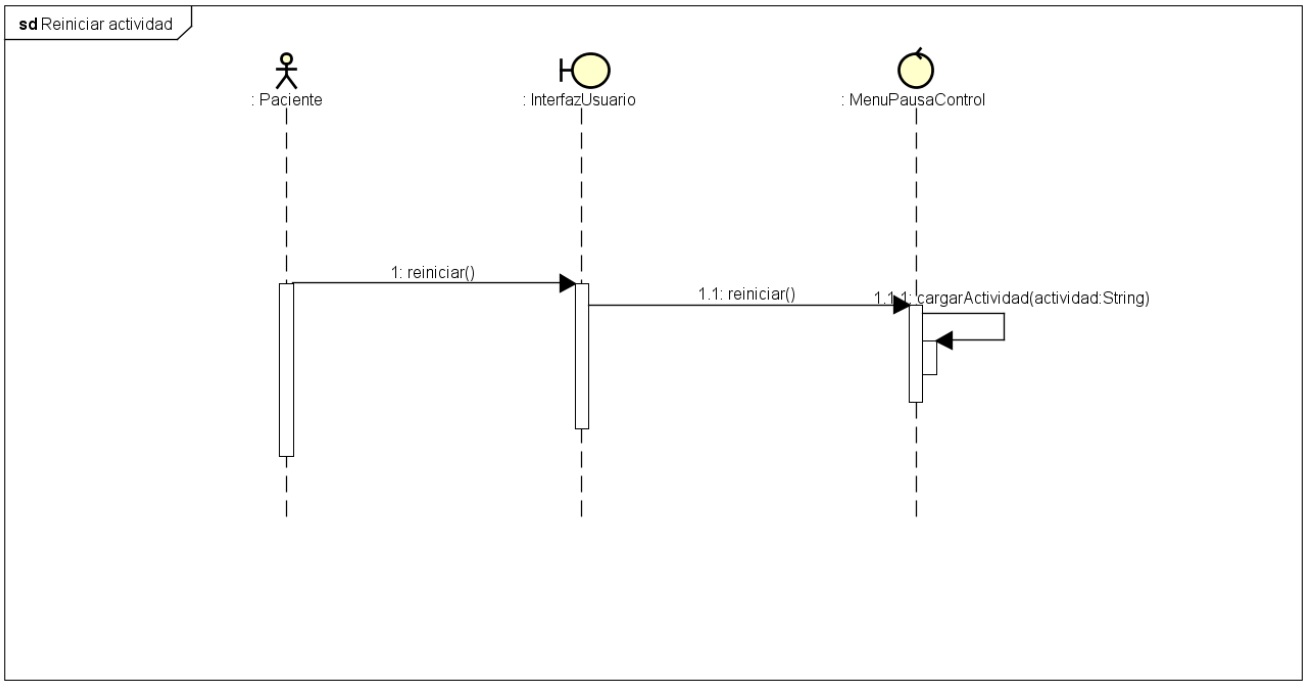

Ilustración 35: Reiniciar actividad

Cambiar brazo

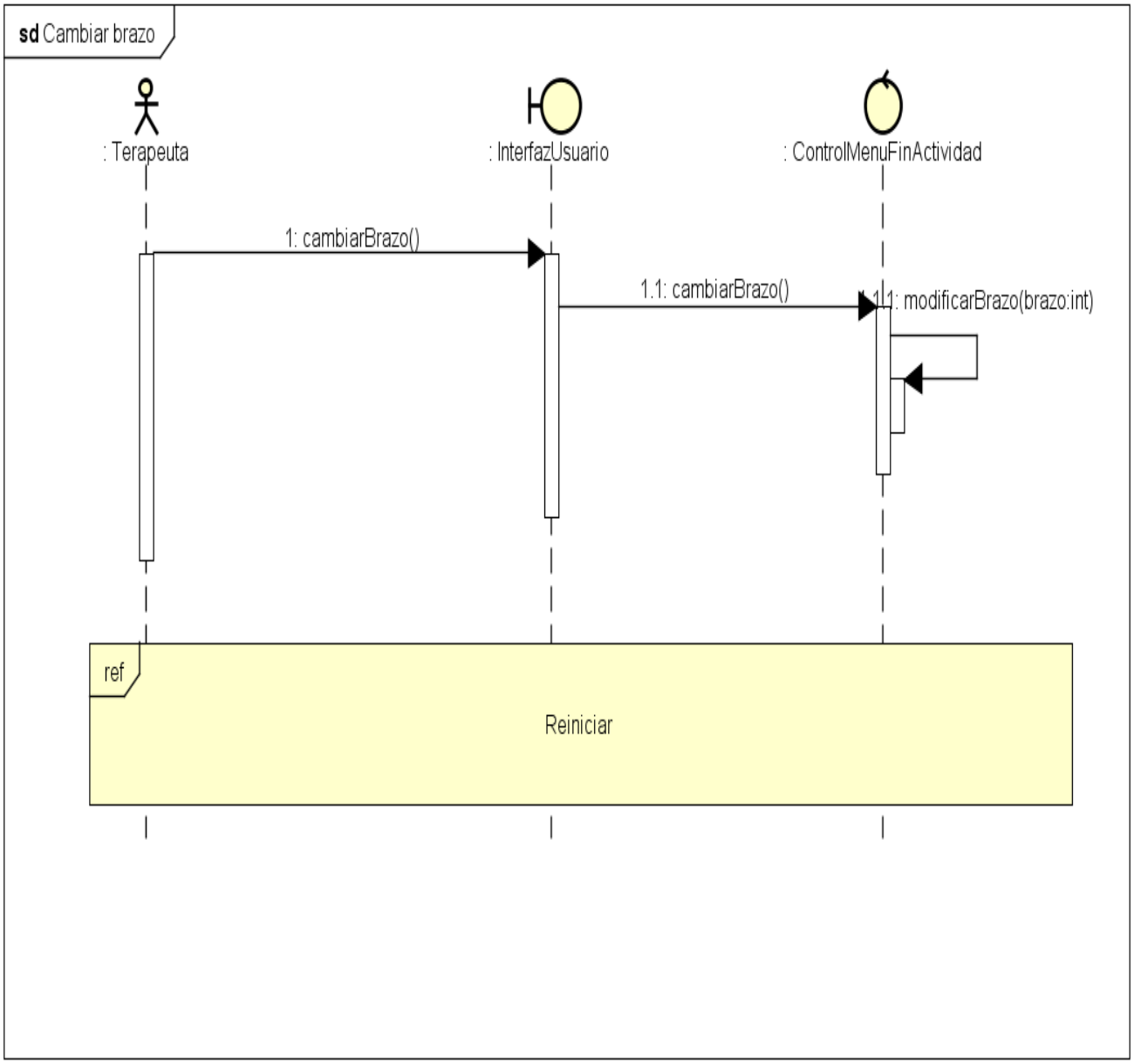

Ilustración 36: Diagramas de Secuencia-Cambiar brazo

5.2.2 Diagrama de clases de diseño

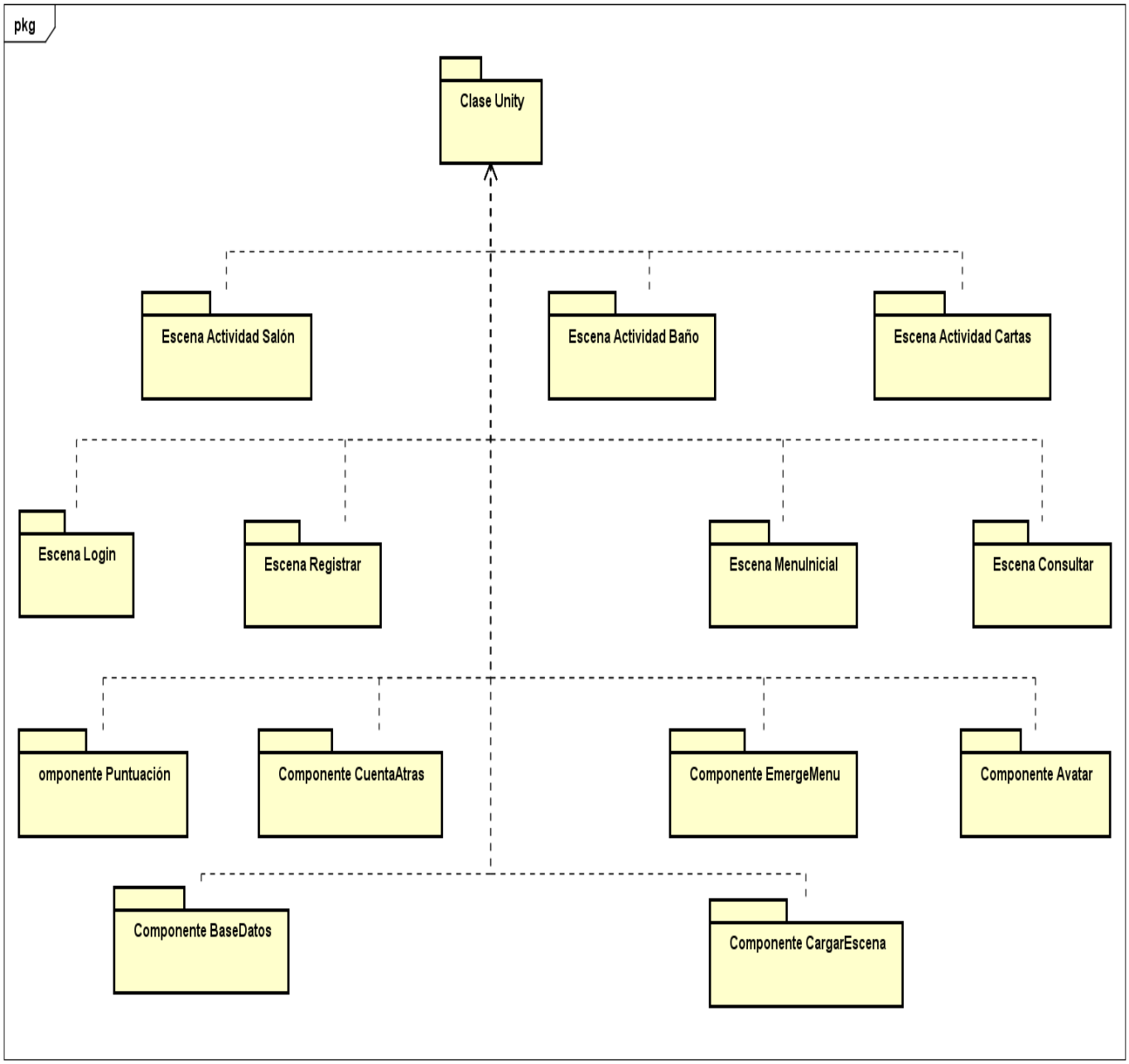

Ilustración 37: Diagrama de clases de diseño

5.2.3 Clases de la Aplicación Unity

A continuación, se describen las clases utilizadas en el desarrollo de la aplicación descendiendo estas de la clase MonoBehavior.

Controlador para cargar escenas

- **LoadSceneController:** clase encargada de controlar el paso de una escena de la aplicación a otra.

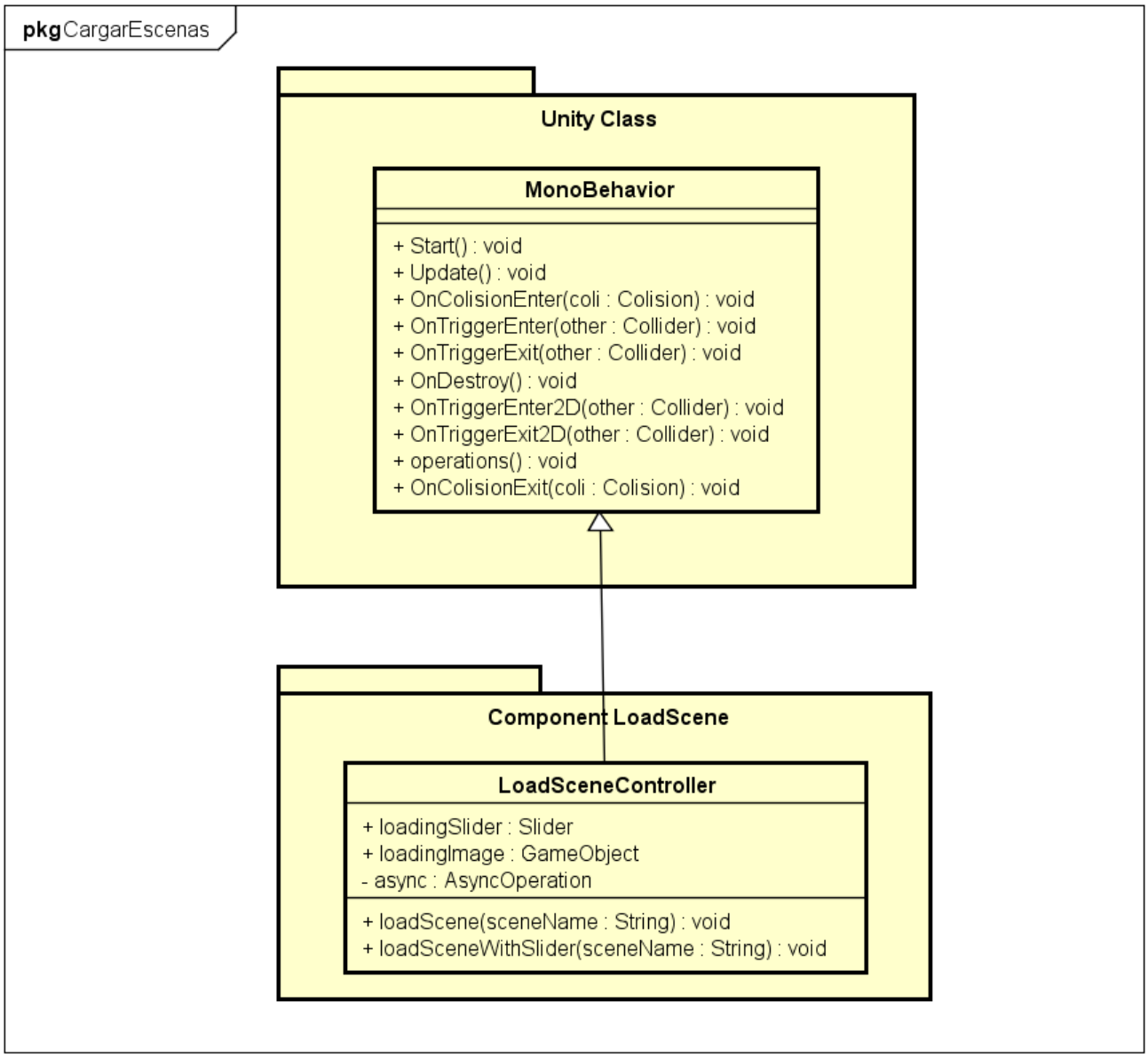

Ilustración 38: Diagrama de clases de diseño-Cargar escena

Controladores de la actividad del Salón

- **ControlActividad1:** Esta clase es la encargada de llevar a cabo el control de lo que ocurre en la actividad, así como su progreso.

- **ControlColisionesOBJ:** Esta se encarga de detectar cuando el objeto colisiona con elementos de la escena.

- **ControlColisionesManos:** Esta clase se encarga de controlar cuando las manos del personaje entran en contacto con un objeto de la escena.

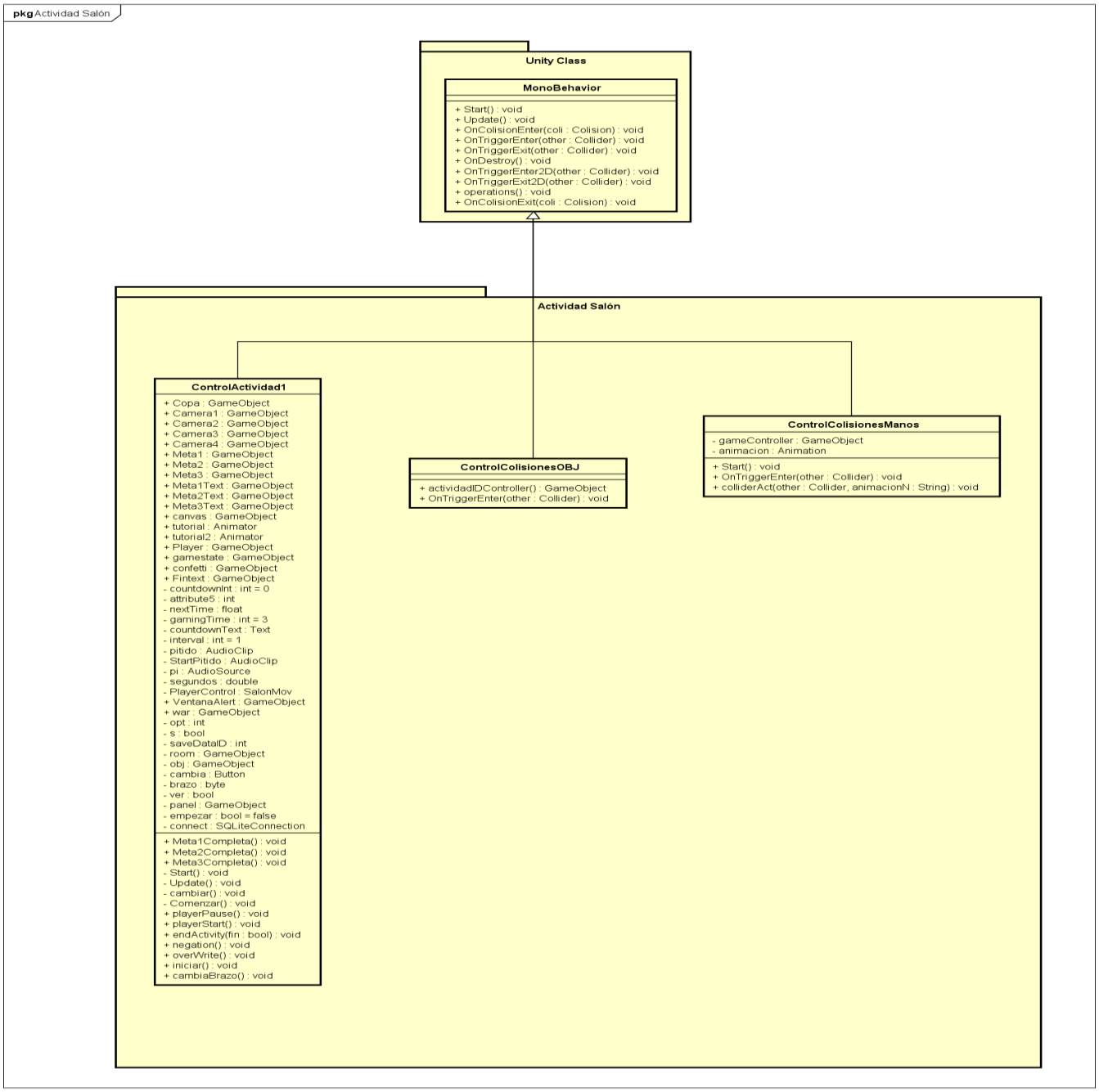

Ilustración 39: Diagrama de clases de diseño-Actividad Salón

Consultar información del paciente

- **ControlInfor:**

  Clase encargada de:

  - Controlar la escena de información del paciente, llevando a cabo la función de obtener los datos del mismo.
  - Modificar los datos del paciente.
  - Obtener los datos de la actividad de la que se quiera ver información
  - Borrar actividades.
  - Borrar el usuario del paciente.

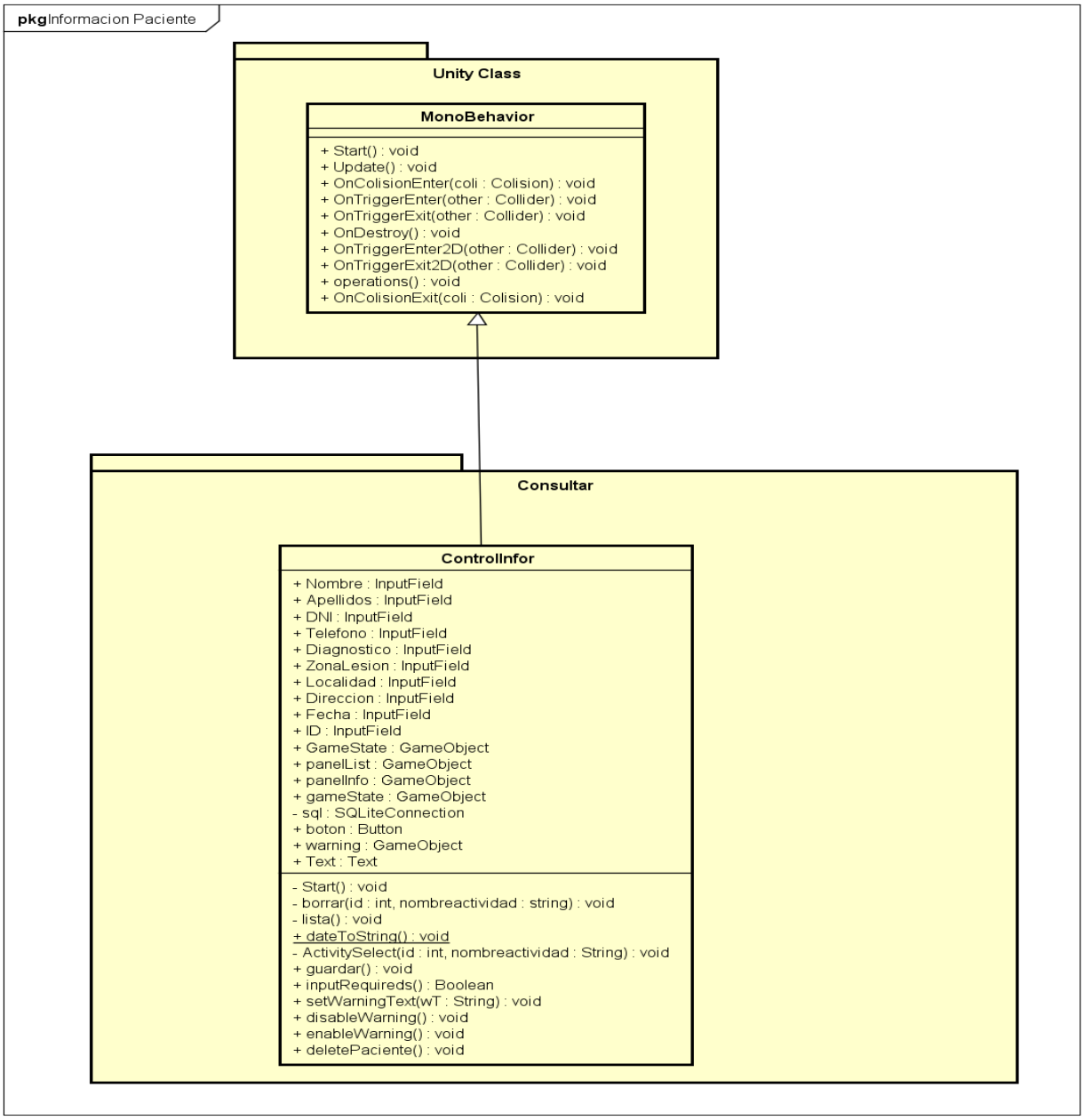

Ilustración 40: Diagrama de clases de diseño-Consultar Información

Inicio de sesión

- **LoginContr:** clase que se encarga de controlar el inicio de sesión, comprobando que el id introducido existe. Para ello conecta con la base de datos y comprueba que el id introducido es igual que el que se encuentra en la base de datos.

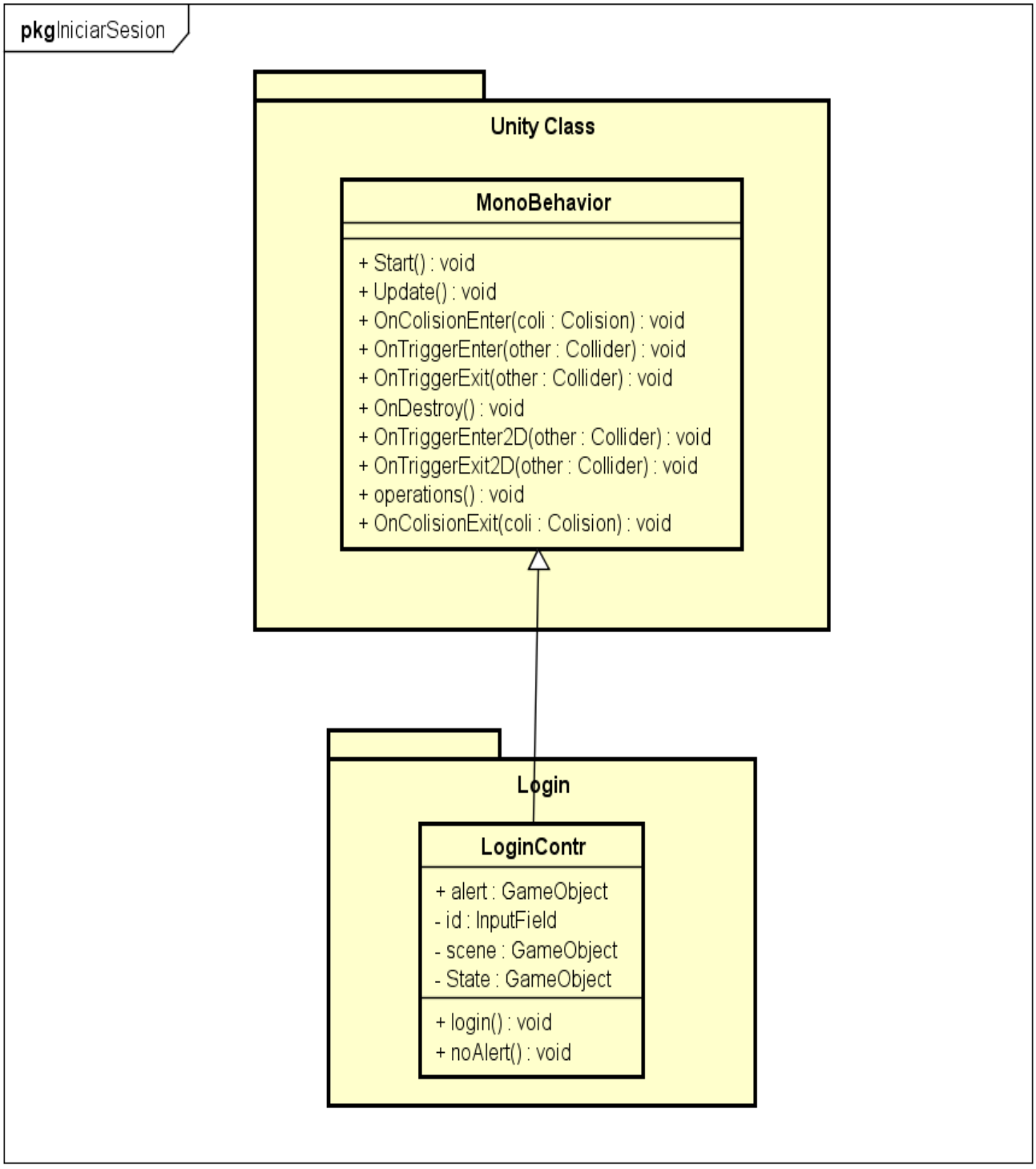

Ilustración 41:Diagrama de clases de diseño-Inicio de sesión

Control menú inicial

- **UIControlMenu:** clase que se encarga del control de la navegación entre las diferentes fases del menú inicial, mostrando las diferentes zonas del menú.

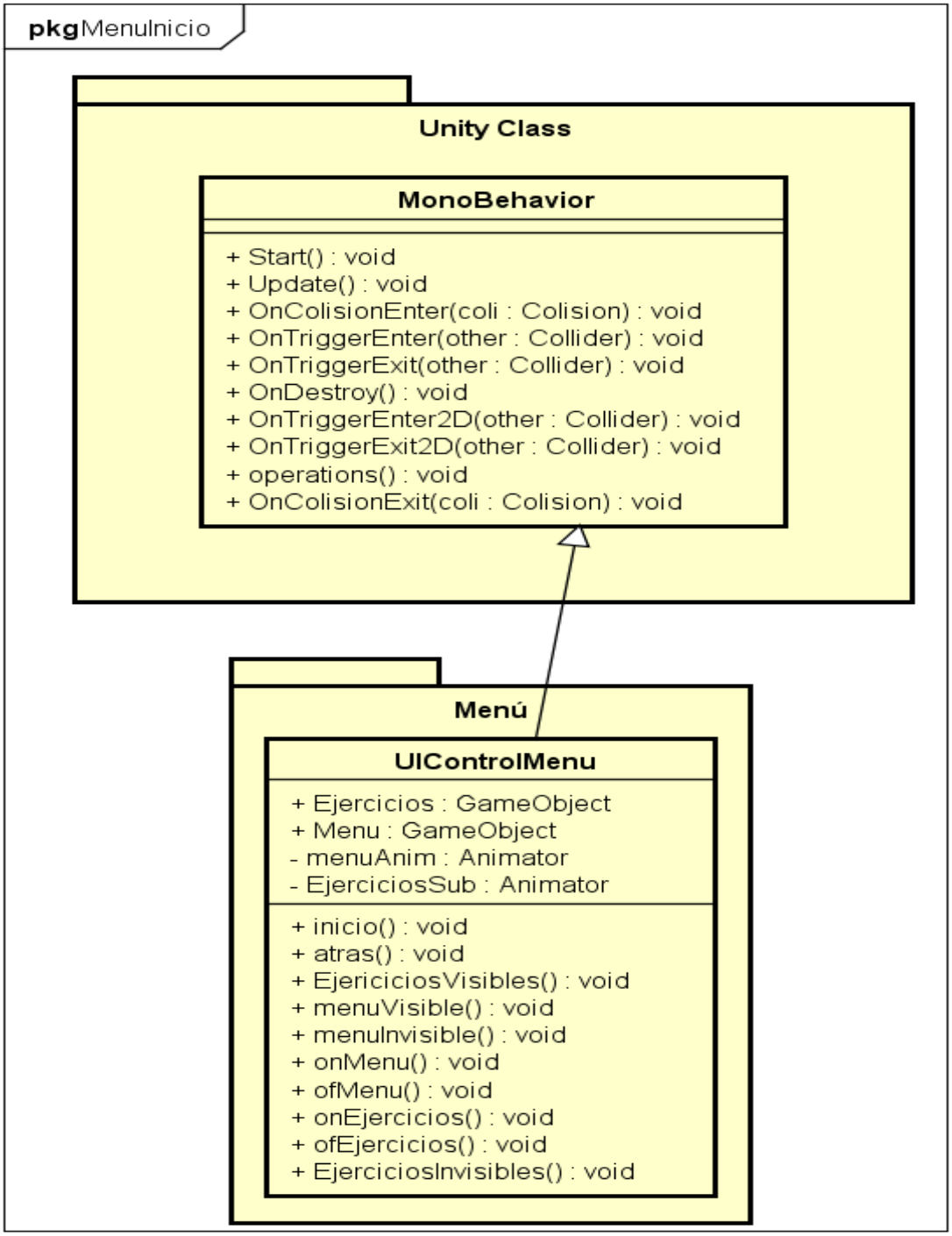

Ilustración 42:Diagrama de clases de diseño-Menú inicial

Registrar

- **DayControl:** Clase encargada de mostrar los días correctos que tiene un mes.
- **YearControl:** Clase encargada de mostrar los años.
- **Registrar:** Clase encargada de registrar nuevos usuarios en el sistema, obteniendo una conexión con la base de datos y validando que todos los campos obligatorios están completos.

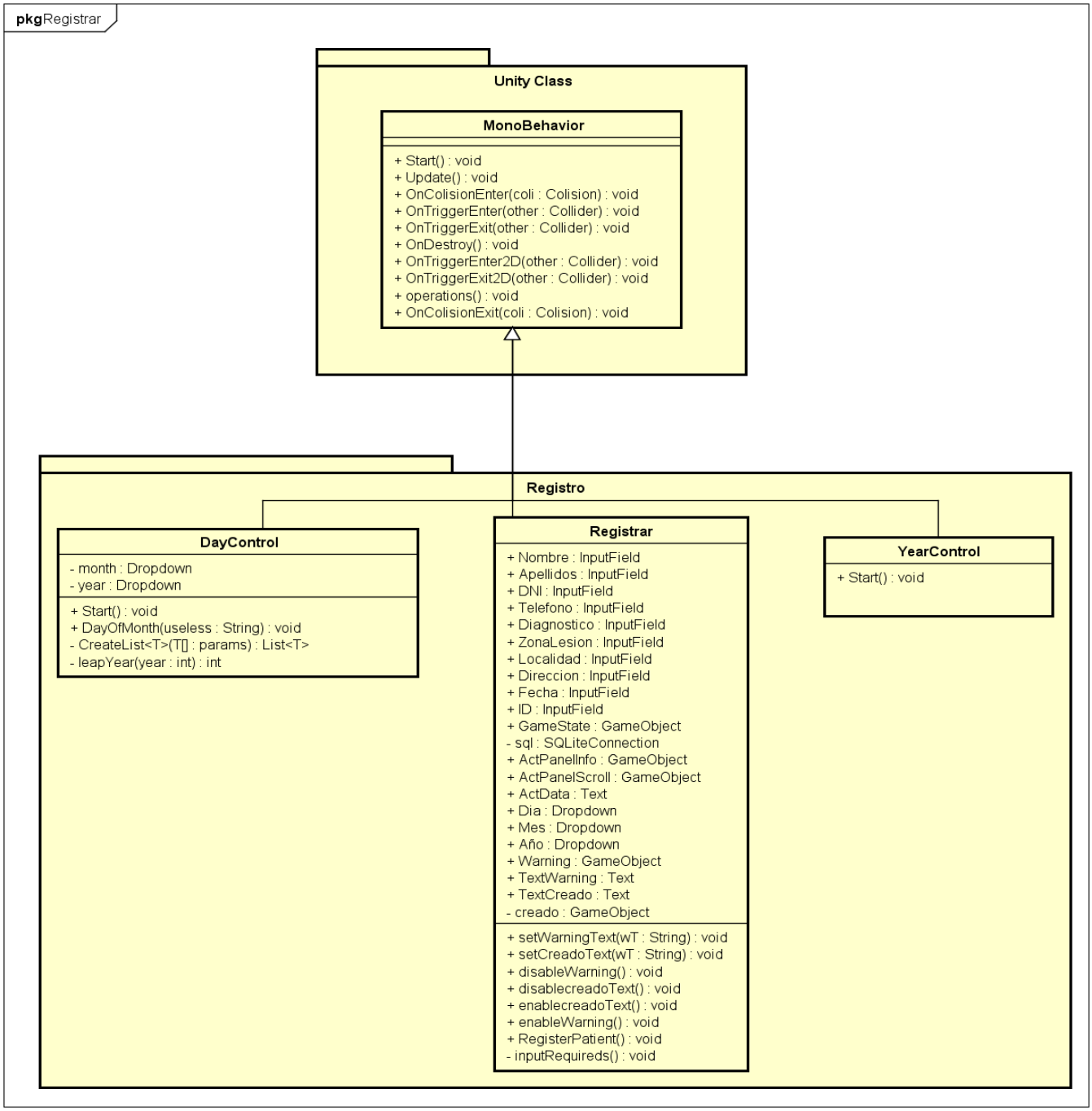

Ilustración 43:Diagrama de clases de diseño-Registrar

Control Actividad Baño

- **ActividadBañoController**: clase encargada de manejar los eventos que ocurren en la actividad.

- **ManoIzquierdaControl:** clase encargada de manejar las colisiones y activar los eventos de la mano izquierda cuando se activa un colisionador.

- **ManoDerechaControl:** clase encargada de manejar las colisiones y activar los eventos de la mano derecha cuando se activa un colisionador.

- **ControlPastaDientes:** clase encargada de controlar las colisiones y activar los eventos cuando la pasta de dientes entra en contacto con un colisionador.

- **CepilloControl:** clase encargada de activar los eventos necesarios cuando el cepillo entra en contacto con un colisionador.

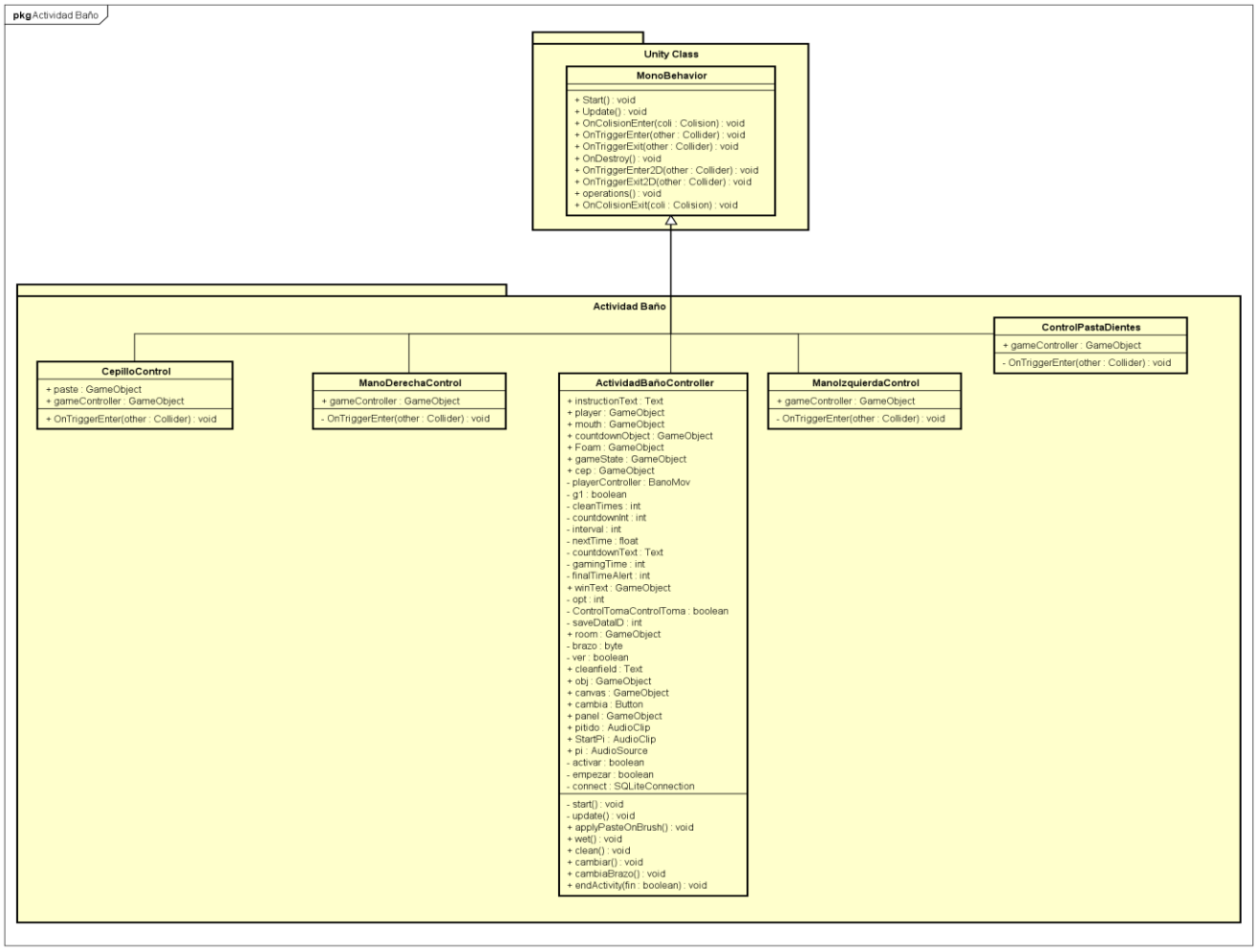

Ilustración 44: Diagrama de clases de diseño-Actividad Baño

Emparejar Cartas

- **ColisionManoController:** clase controlador que gestiona las colisiones entre la mano y los objetos de la escena, teniendo como principal función la selección de las cartas.

- **ControladorCartas:** clase encargada de controlar los eventos que ocurren durante la ejecución de la actividad. También se ocupa de conectar con la base de datos para almacenar la información del ejercicio.

- **InstructionColliderController:** clase controladora encargada de manejar la aparición y transacción de las instrucciones, en colisión con la mano del avatar.

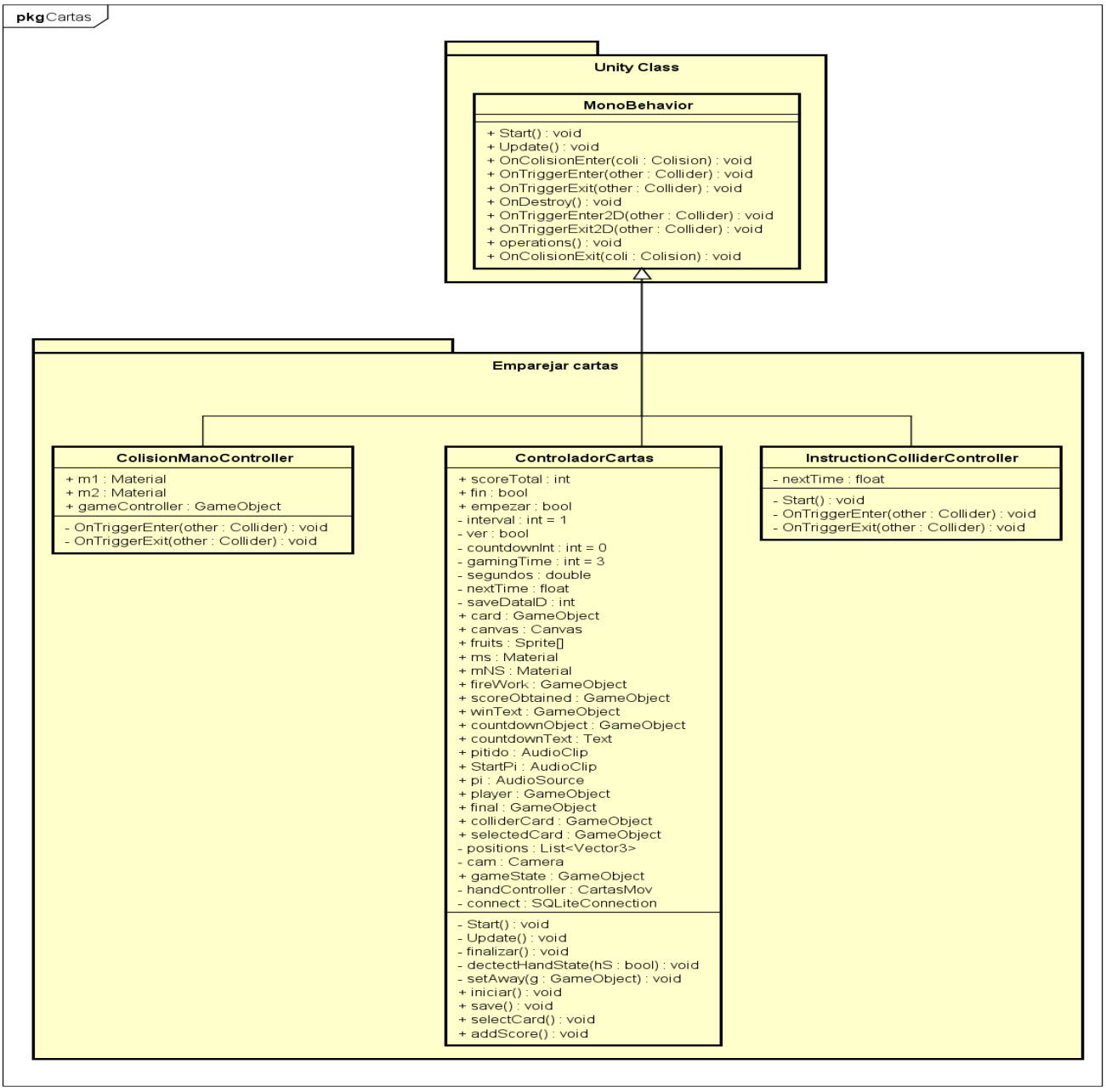

Ilustración 45:Diagrama de clases de diseño-Emparejar Cartas

Menú Escape

- **MenuEsc:** Clase encargada de controlar las operaciones que se realizan con el menú de escape dentro de una actividad.

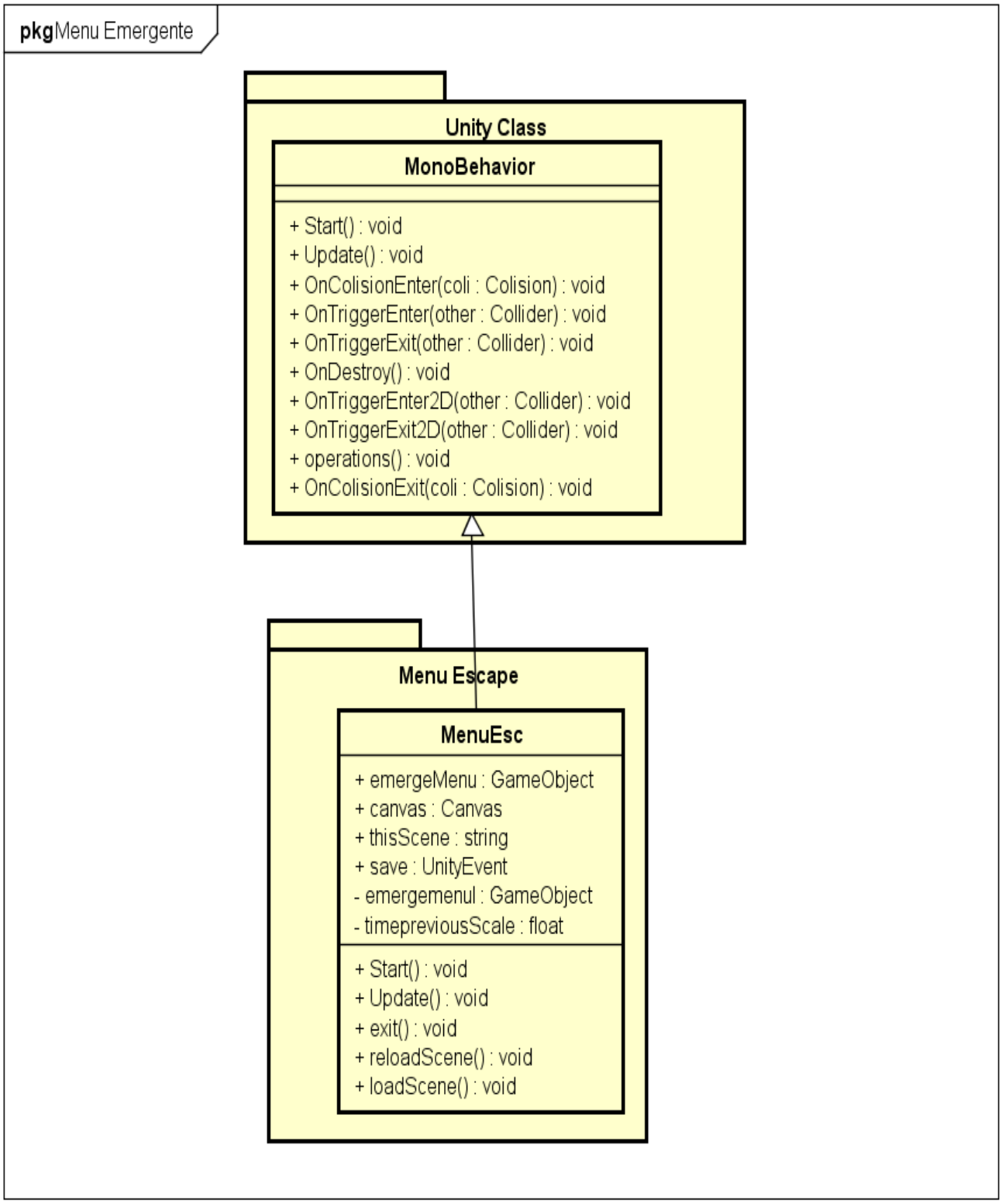

Ilustración 46: Diagrama de clases de diseño-Menú Escape

Creador de base de datos SQLite

- **DBCreater:** Clase encargada de crear una base de datos SQLite en el caso de que no detecte ya una creada previamente con el nombre asignado.

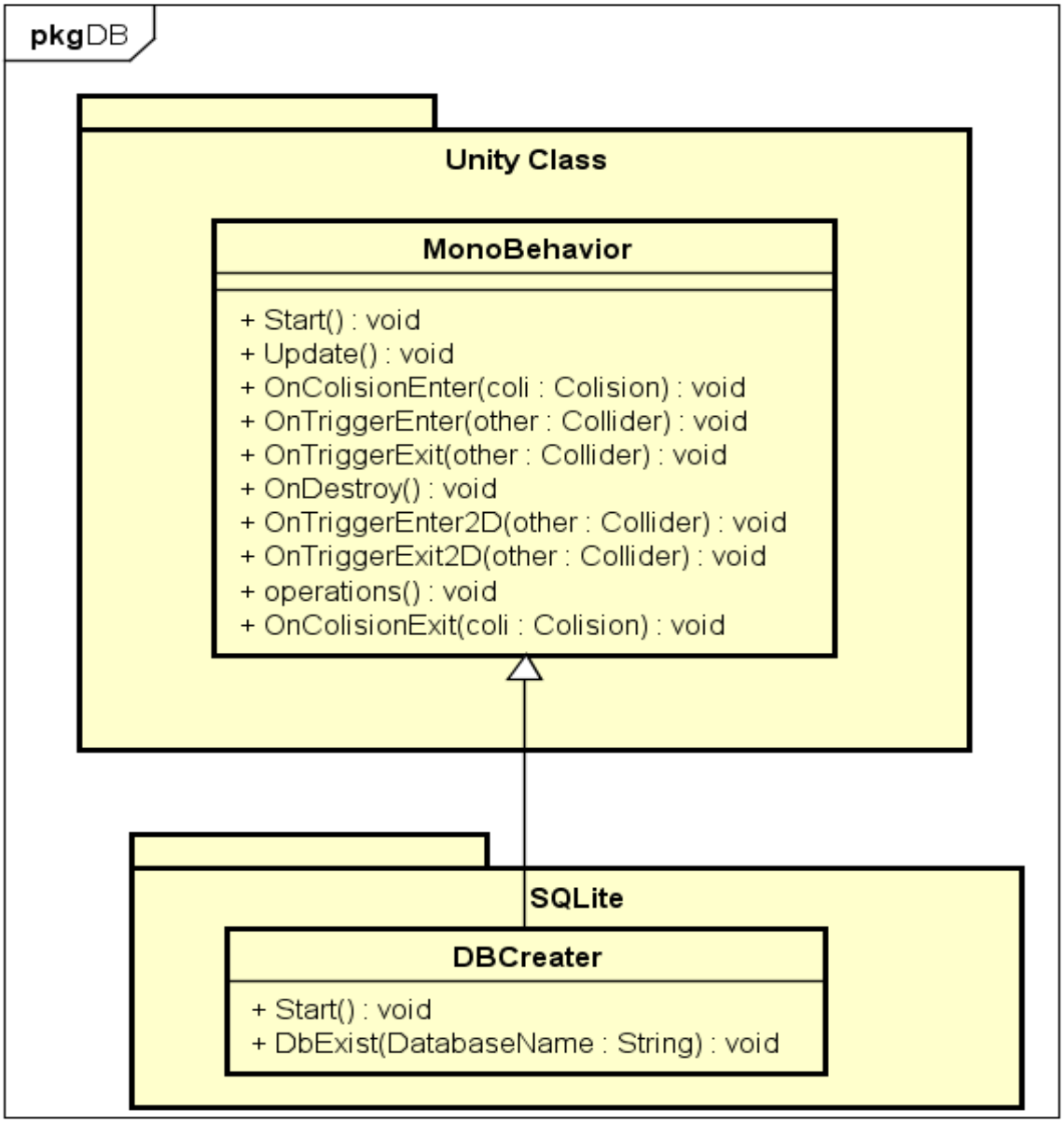

Ilustración 47:Diagrama de clases de diseño-Crear SQlite

5.3 Parte Android

5.3.1 Diagrama de secuencia Android

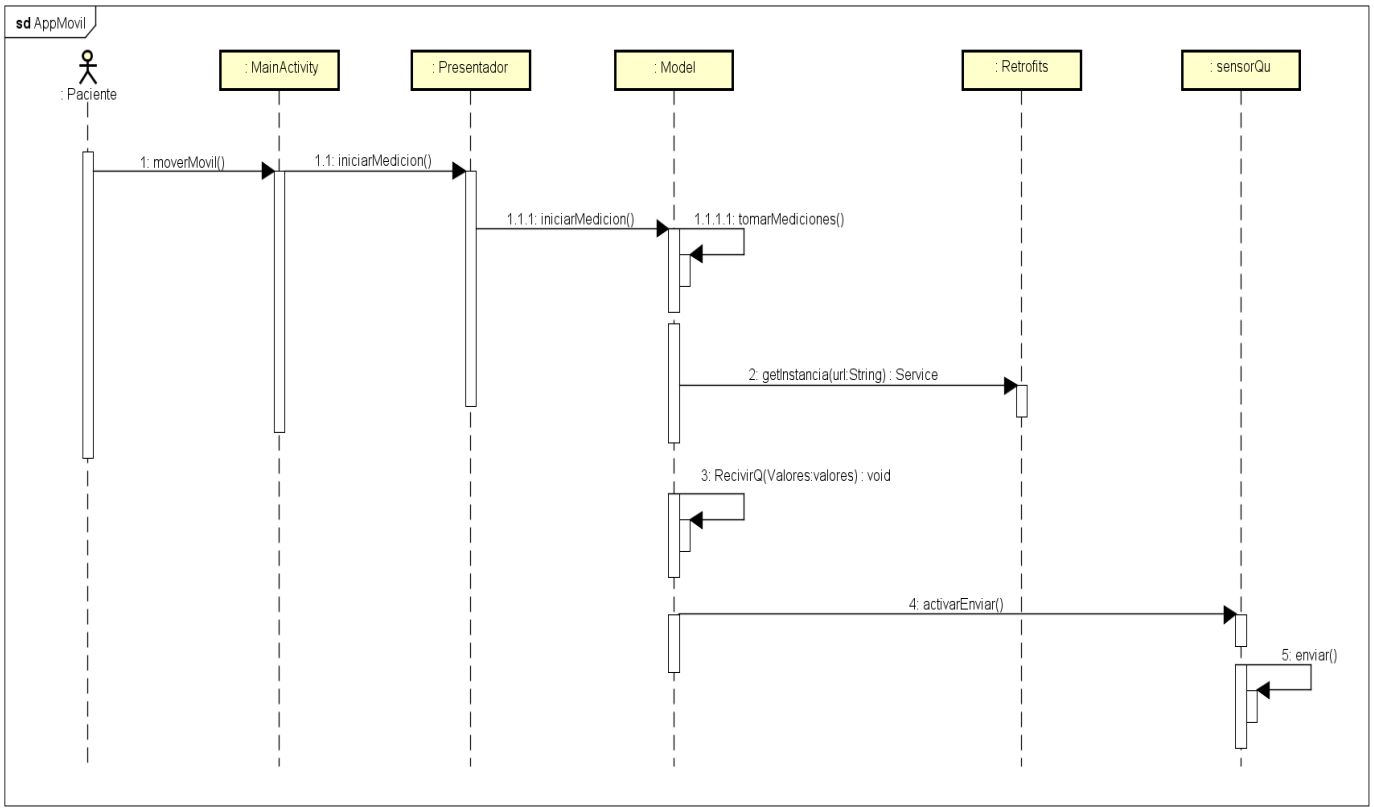

Ilustración 48: Diagrama de secuencia Android

5.3.2 Diagrama de clases de la aplicación Android

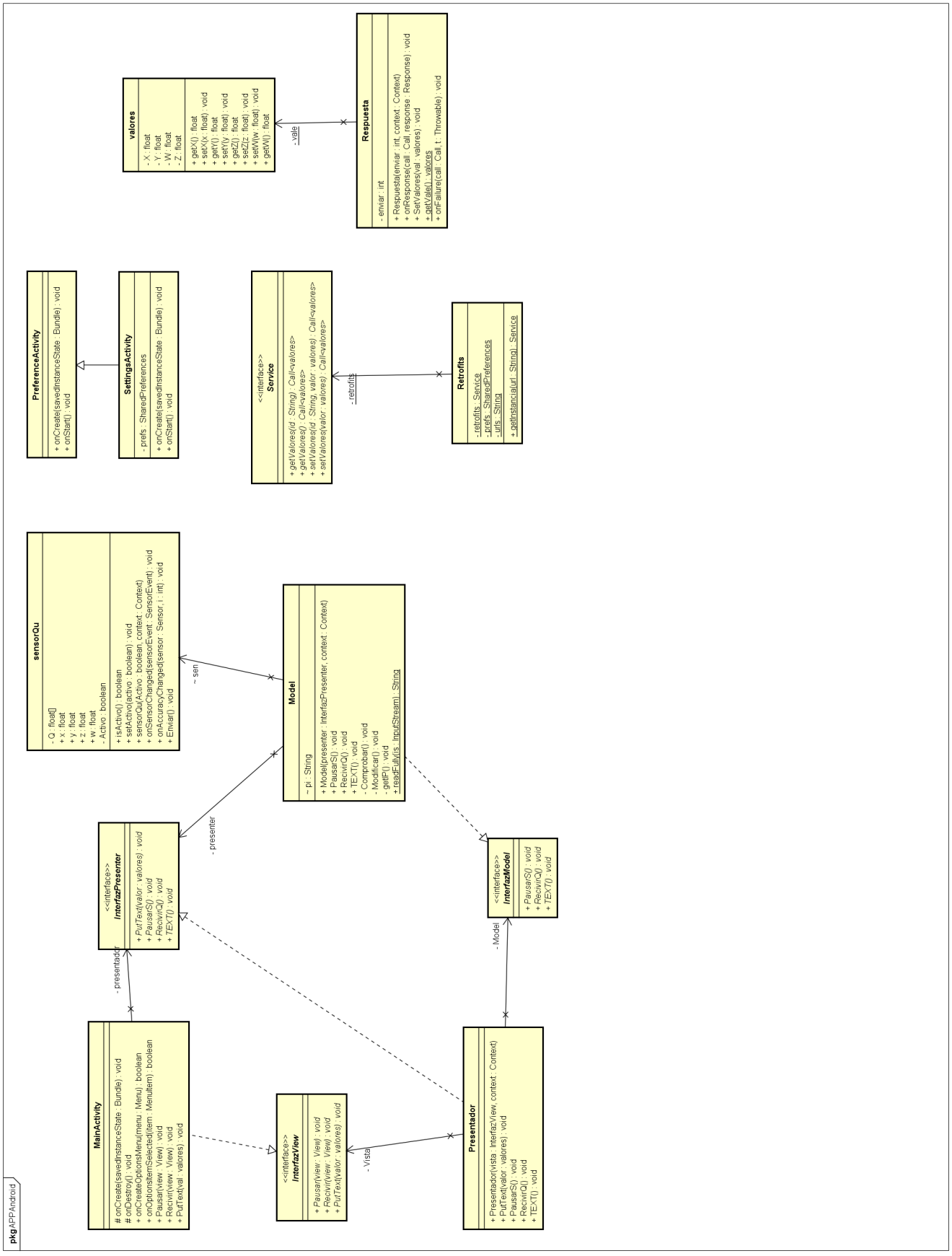

Ilustración 49: Clases de la aplicación Android

En este diagrama se puede ver cómo está estructurada la aplicación Android y que clases la forman, así como la aplicación del Patrón arquitectónico (Modelo Vista Presentador (MVP)), del cual se hablará más adelante en el punto 6.2.3 del capítulo 6 (Implementación de la aplicación).

5.3.3 Paquetes que componen la aplicación Android

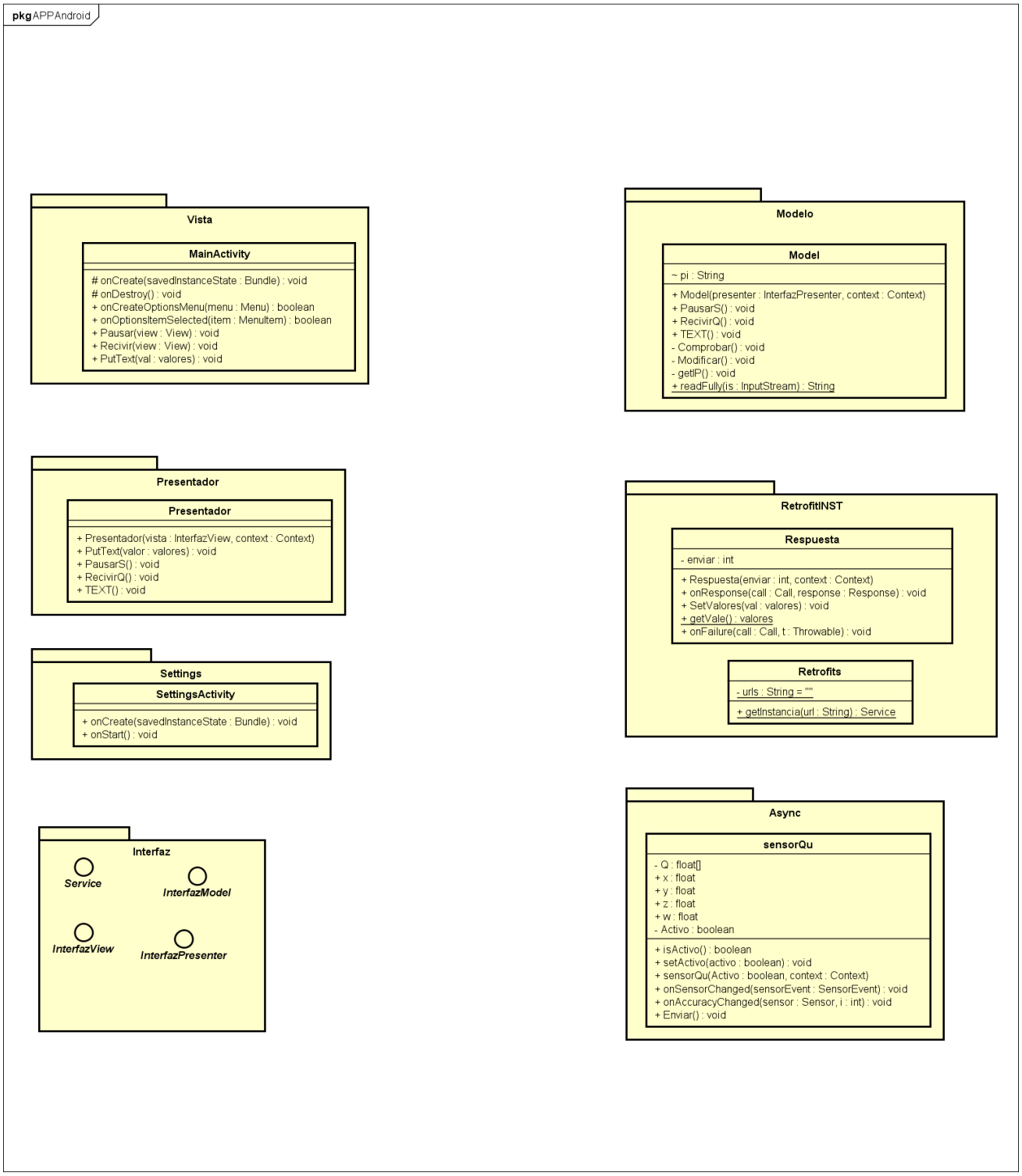

Ilustración 50: Paquetes

Para su desarrollo, la aplicación se ha dividido en 7 paquetes:

- **Paquete Vista:** contiene las clases que controlan la vista de la aplicación, usando el MVP.
    - MainActivity

- **Paquete Presentador:** contiene la clase que actúa como presentador entre la vista y el modelo, utilizando el MVP.
    - Presentador

- **Paquete Modelo:** contiene la clase que actúa como modelo, aplicando el MVP.
    - Modelo

- **Paquete Settings:** contiene la clase que controla los ajustes de la aplicación.
    - SettingsActivity

- **Paquete Interfaz:** contiene las interfaces usadas para aplicar el MVP y para realizar las consultas rest.
    - InterfazView
    - InterfazPresenter
    - InterfazModel
    - Service

- **Paquete RetrofitINST:** contiene las clases que se utilizan para realizar las operaciones rest y para comunicarse con el servidor.
    - Respuesta
    - Retrofits

- **Paquete Async:** contiene la clase que se usa para obtener los datos de los sensores del dispositivo Android.
    - sensorQu

5.4 Diagrama de vista Física

En este diagrama se puede ver la distribución y conexión entre los distintos elementos que forman el sistema.

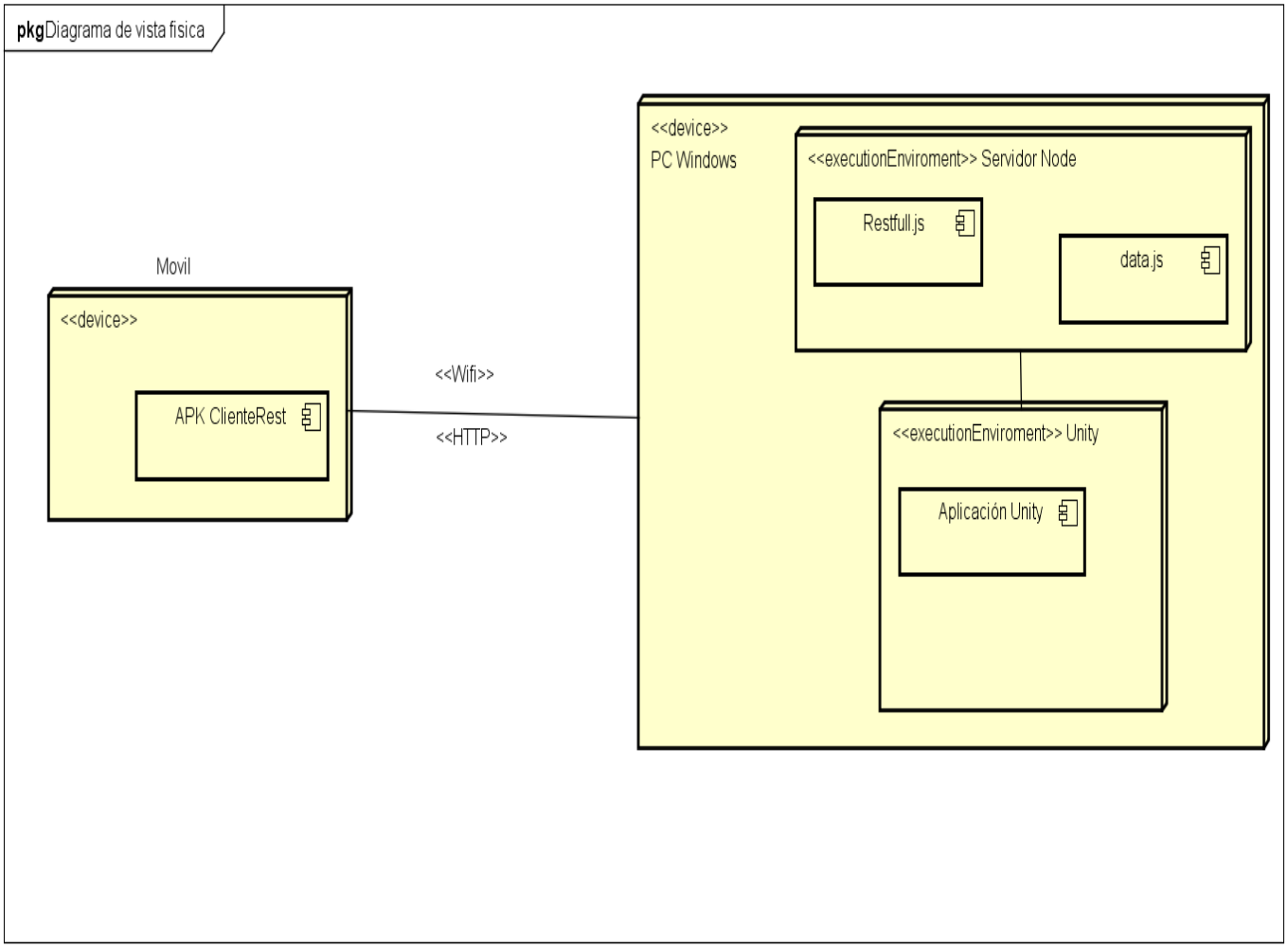

Ilustración 51: Vista física

5.5 Diseño de la base de datos

Diagrama entidad relación de la base de datos:

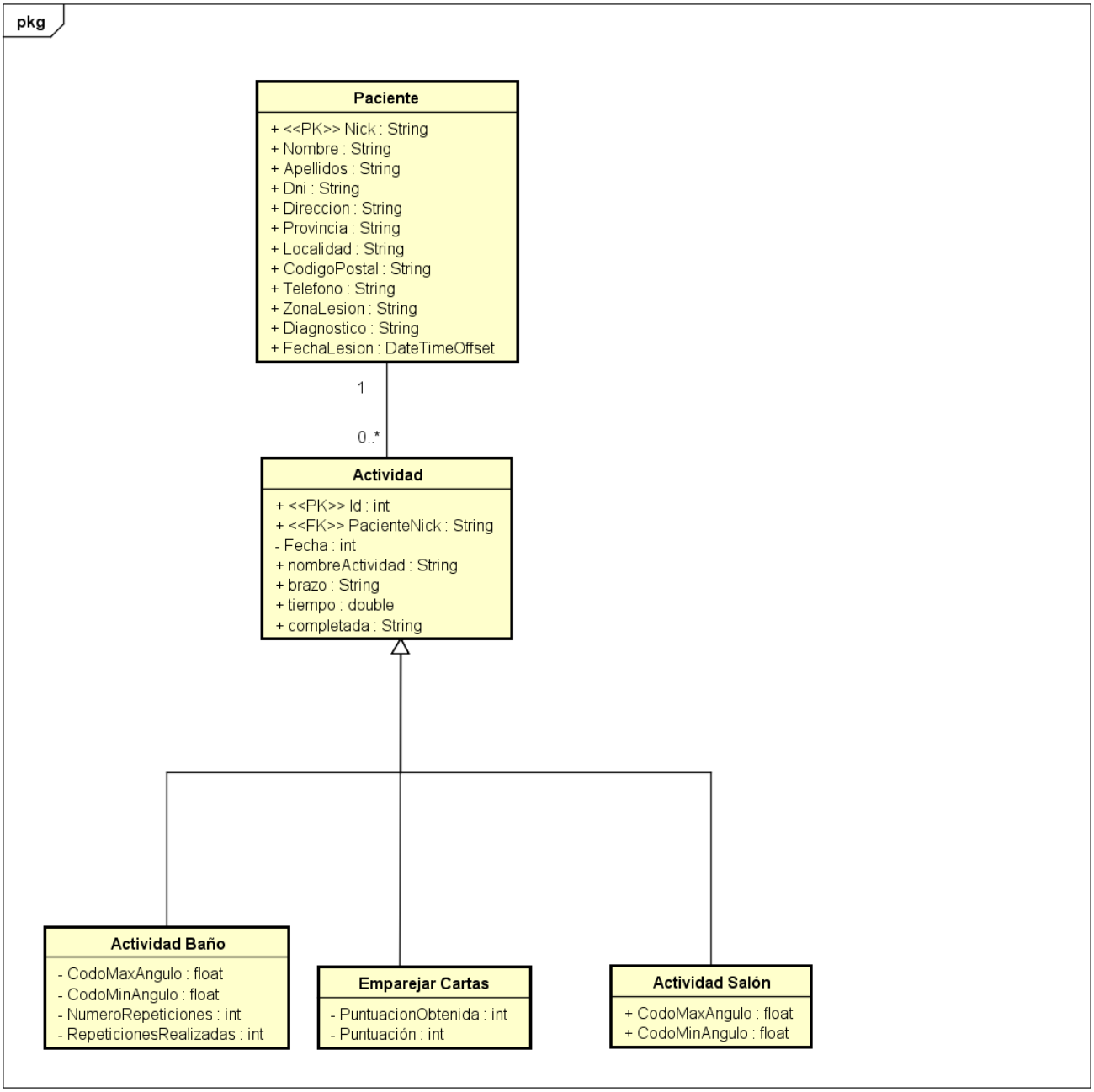

Ilustración 52: Diagrama Entidad Relación Base de datos

5.6 Diseño de la interfaz

5.6.1 Aplicación Unity

Para la elaboración de la interfaz se han usado los recursos proporcionados por Unity, lo que significa que la interfaz está dividida en diferentes escenas. En el desarrollo se han utilizado los diseños de un proyecto anterior, del que se hizo mención en la introducción.

A petición de cliente todos los diseños de las escenas están preparados para la pantalla con una resolución 1366x768.

Diagrama navegación

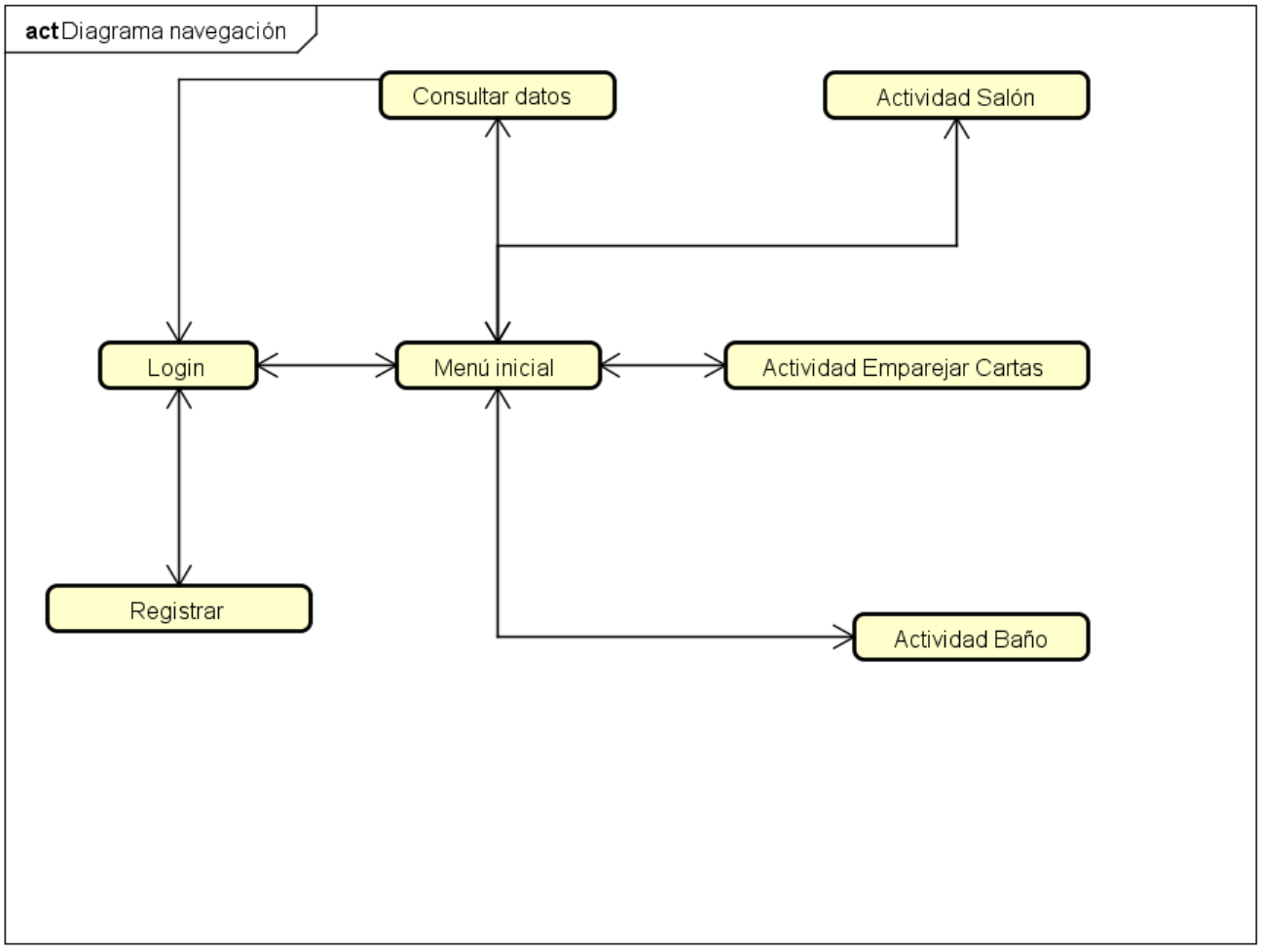

Ilustración 53: Diagrama navegación Escenas

A continuación, se mostrará la Interfaz de cada escena.

Escena Login:

Es la primera escena que se muestra al arrancar la aplicación, desde ella podemos navegar hacia el Menú inicial o hacia la escena de Registrar.

En la esquina inferior derecha encontramos un botón que sirve para cerrar la aplicación.

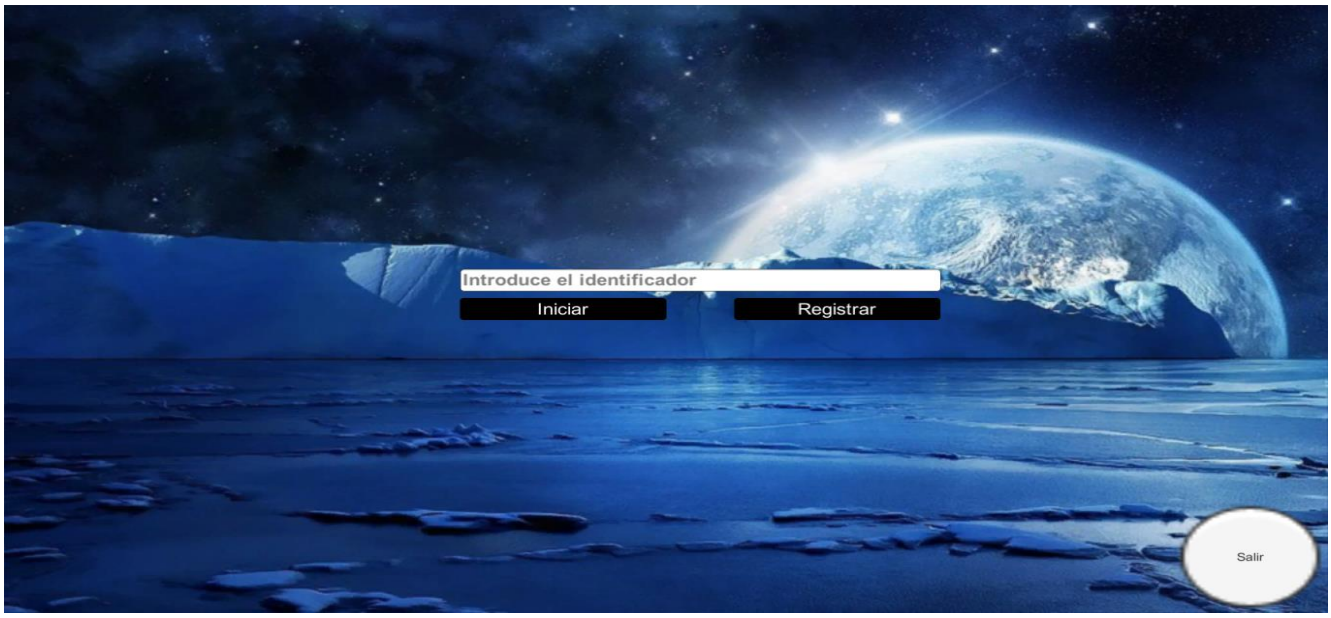

Ilustración 54: Escena Login

Escena Registrar:

Esta solo es accesible desde la escena Login, y solo puede navegar hacia esta.

En ella se muestra el formulario de Registrar y los dos botones Cancelar y Crear en la esquina inferior derecha.

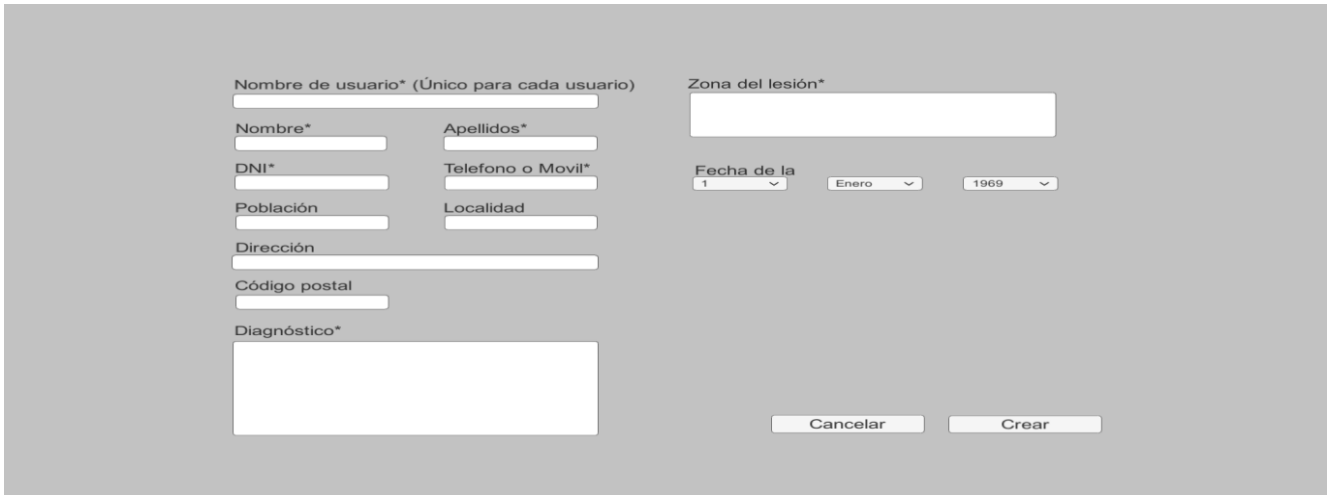

Ilustración 55: Escena Registrar

Escena Menú inicio:

Desde esta escena se puede navegar hacia todas las demás a excepción de la de Registrar.

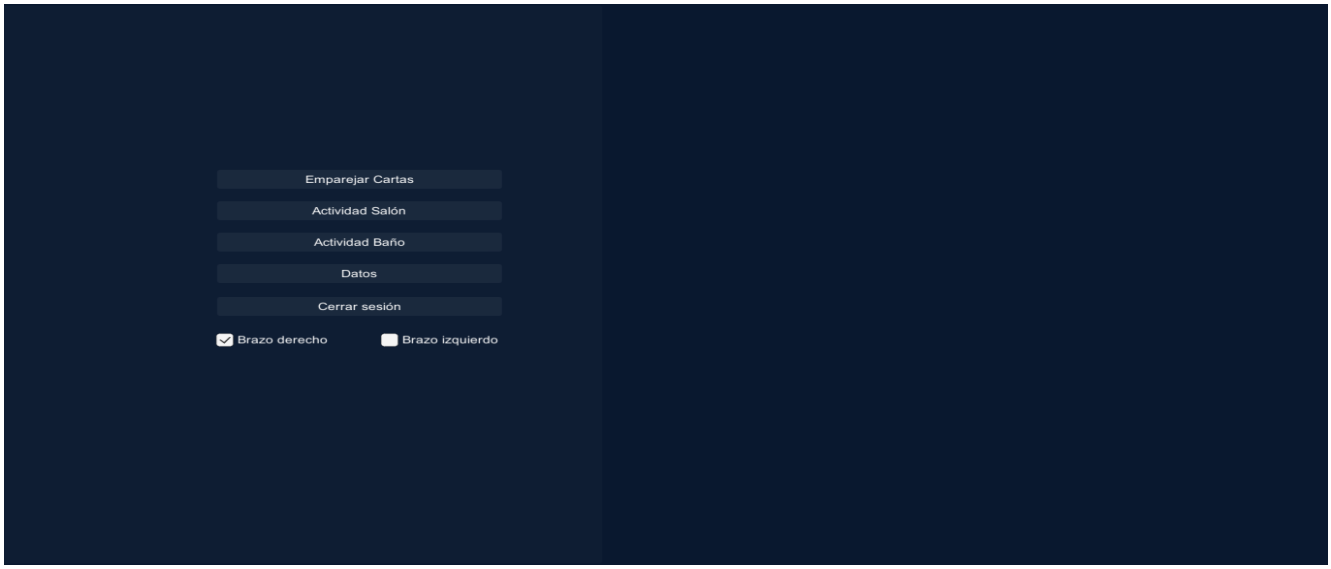

Ilustración 56: Escena Menú inicial

Escena Consultar datos:

En esta escena nos encontramos en la parte izquierda un formulario con los datos del paciente, al igual que una caja donde se muestran las actividades que ha realizado.

En la parte derecha aparece la zona donde se muestra la información sobre las actividades.

En la parte inferior de la pantalla encontramos el botón de volver y el de eliminar.

Desde esta escena se puede navegar hacia el Menú inicial y el Login.

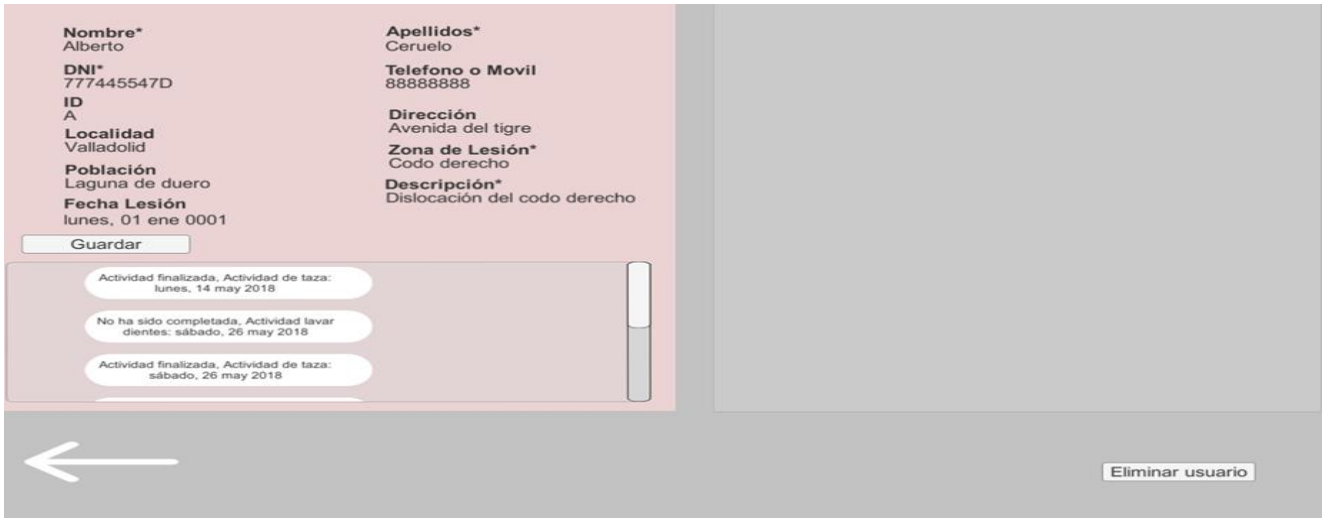

Ilustración 57: Escena Consultar datos

Escena Actividad Salón:

Representa una interfaz sencilla de un salón, teniendo dos vistas disponibles como se puede ver en la **Ilustración 58**, una trasera y otra frontal. Dentro de esta cabe destacar tres objetos, el avatar, el objeto sobre la mesa y la pizarra con las instrucciones.

Esta escena permite navegar al menú inicial.

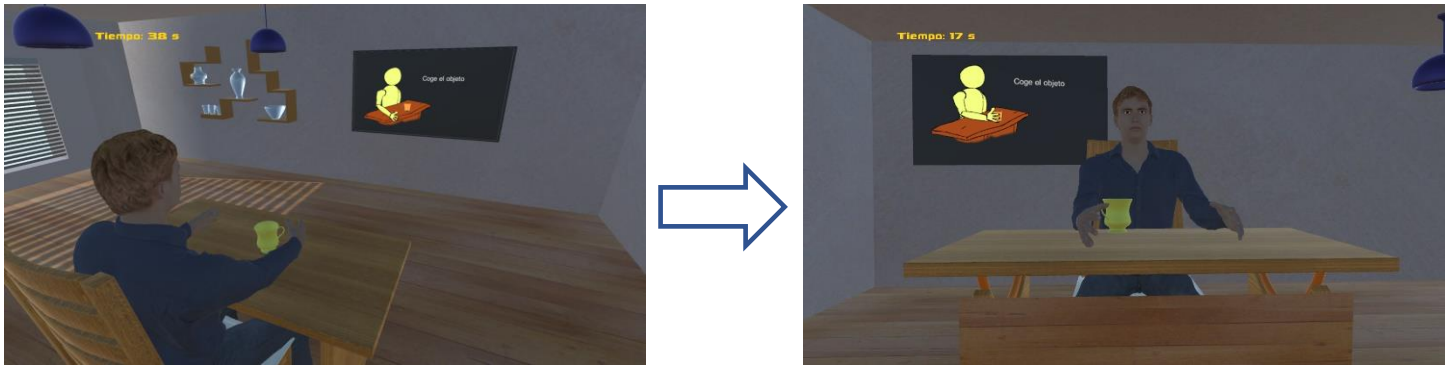

Ilustración 58: Escena Actividad Salón

Escena Actividad Baño:

Esta muestra el ambiente de un baño que modifica la vista conforme se va realizando la actividad, como se muestra en la **Ilustración 59**.

Esta escena permite navegar al menú inicial.

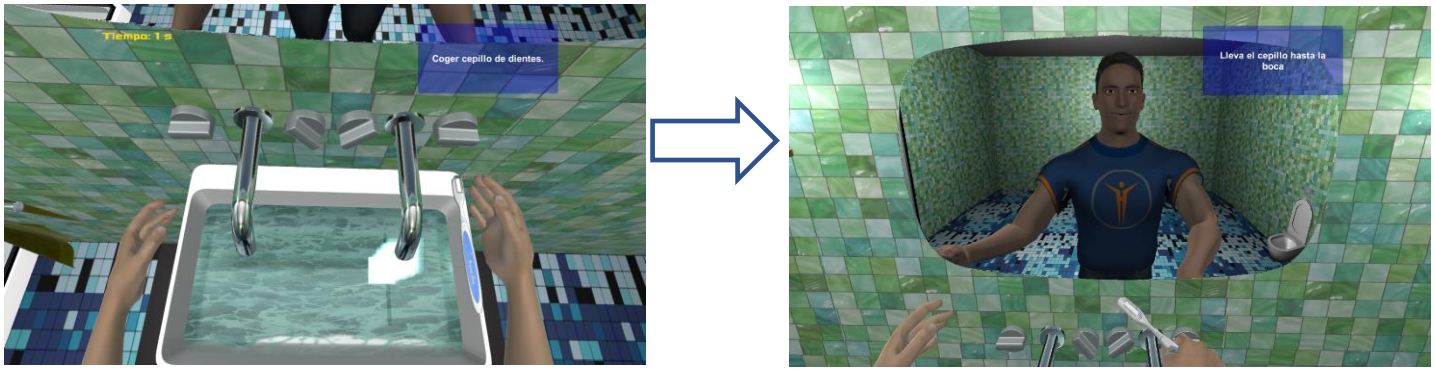

Ilustración 59: Escena Actividad Baño

Escena Actividad 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.

Esta escena permite navegar al menú inicial.

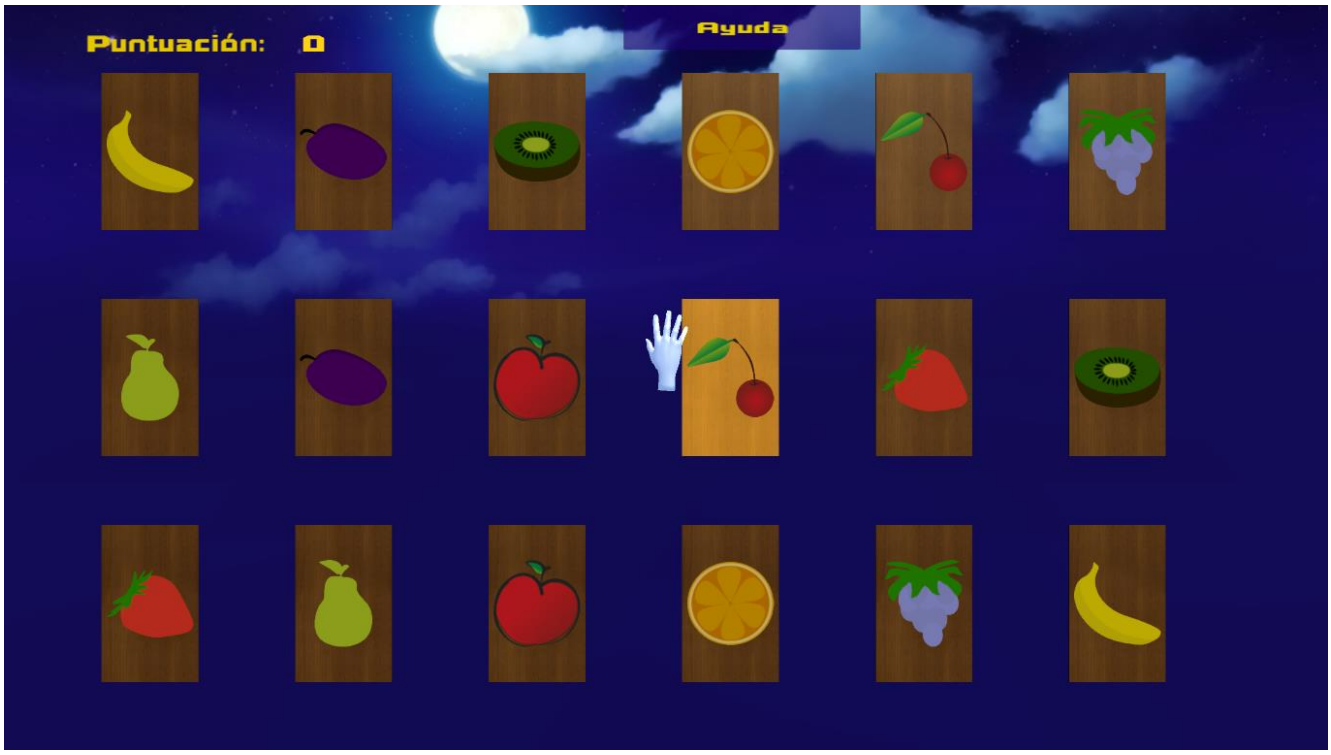

Ilustración 60: Escena Actividad cartas

5.6.2 Aplicación Android

Las interfaces disponibles en la aplicación Android son muy sencillas, siendo solamente dos:

- La interfaz principal formada por dos botones, cuatro cuadros de texto y un menú en la parte superior derecha.
- La segunda interfaz es accedida mediante el menú y muestra la configuración disponible.

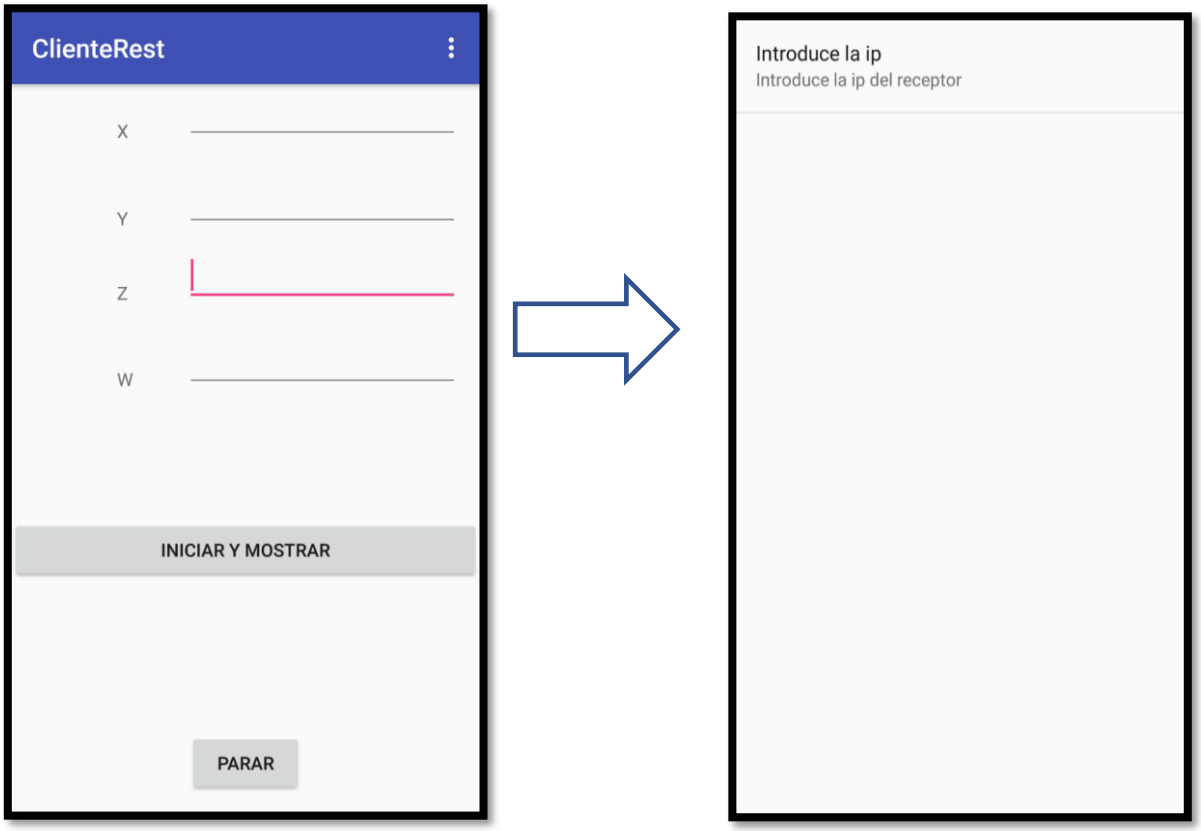

Ilustración 61: Interfaz Android

Capítulo 6

6 Implementación de la aplicación

6.1 Introducción

En los sucesivos apartados se explicarán algunas cuestiones relativas a la implementación del proyecto.

6.2 Paso de datos del dispositivo móvil Android a Unity.

En las fases iniciales del proyecto existía el riesgo de que el paso de los datos (Cuaterniones) entre el dispositivo móvil y la plataforma Unity no fuera estable, provocando que el movimiento del avatar fuera a tirones, haciendo inviable la realización del proyecto.

Tras varias pruebas, se ha logrado que el paso de los datos sea estable y que el movimiento que se realiza con el móvil se represente de forma fluida en el avatar de Unity.

Como solución al problema de la fluidez del paso de los datos se ha procedido a la creación de un servidor que haga de enlace entre las dos plataformas.

En la creación del servidor se ha usado la tecnología Node js, con la que consigue un servidor rest para el paso de los datos de forma directa.

En la comunicación y paso de los datos al servidor se implementó una aplicación para la plataforma móvil Android, que realiza las operaciones get y put. Al igual se ha tenido que realizar un script en Unity, para que pueda actuar como cliente rest obteniendo los datos del servidor.

6.3 App Android

Para el desarrollo del proyecto se ha requerido la creación de una aplicación para Smartphones Android.

Esta aplicación es la encargada de obtener los datos de orientación y rotación del Smartphone, los cuales son representados en formato de cuaterniones. A su vez, se ocupa de enviarlos a un servidor para que posteriormente sean usados en la aplicación de Unity que representa los movimientos.

6.3.1 Obtención de la posición y la rotación del móvil.

Para la obtención de los datos se hace uso de tres sensores del Smartphone, acelerómetro, magnetómetro y giroscopio, mediante el **sensor Vector de rotación**. Este combina los datos de los tres sensores anteriormente mencionados para obtener un resultado más preciso, pudiendo así representar la orientación del dispositivo como una combinación de un ángulo (θ) y un eje (<x, y, z>).

Los tres elementos del vector de rotación son: <x*sin(θ/2), y*sin(θ/2), z*sin(θ/2)>, tal que la magnitud de este es igual a sin(θ/2) y la dirección es igual a la dirección del eje de rotación.

Estos tres elementos son iguales a los tres últimos componentes de un cuaternión unitario (<cos(θ/2), x*sin(θ/2), y*sin(θ/2), z*sin(θ/2)>).

El sistema de coordenadas empleado por el vector de rotación como referencia a la hora de calcular la orientación del dispositivo es como se muestra en la siguiente imagen:

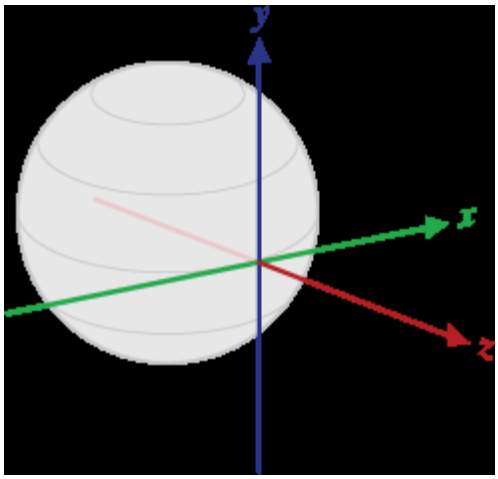

Ilustración 62: Android-Ejes

El cuaternión unitario obtenido es el dato que se usa para que Unity represente el movimiento.

6.3.2 Librería externa

En el desarrollo de la aplicación se ha hecho uso de una biblioteca externa llamada Retrofit2 para la conexión y el paso de los datos al servidor.

Retrofit2 es un cliente REST para Android y Java, pudiendo realizar las operaciones GET, POST, PUT, PATCH, DELETE y HEAD.

Las características que nos ofrece son:

- Aísla el manejador de peticiones en una interfaz
- Usa anotaciones para definir los componentes de la API (verbo, parámetros, cuerpo, cabeceras, etc.)
- Nos permite enviar las peticiones de manera asíncrona
- Es capaz de integrar múltiples conversores para JSON y XML como Gson y Simple XML.

En este caso solo se han usado las operaciones GET y PUT.

En la siguiente tabla se puede ver el código que se implementa para utilizar las operaciones haciendo uso de esta librería.

Tabla 31: Operaciones GET y PUT Android

```
   Public interface Service {


    @GET("coor/")
    Call<valores> getValores();


    @PUT("coor/")
    Call<valores>setValores( @Body valores valor);
}
```

Para ello se ha creado una interfaz con las operaciones necesarias, indicando a cada operación la ruta de la URL del servidor, marcándolas con la etiqueta **@Path**, y en el caso de la operación put se incluyen a mayores los datos que se envían, representándolo con **@Body**.

Retrofit se encargará de convertir estas operaciones en una consulta de tipo REST.

Para que funcione se ha creado una clase que devuelve una instancia del servicio (Service), creando un objeto Retrofit, pasándole la URL que ha de usar Gson para dar formato a los datos.

Tabla 32: Servicio Retrofit

```
   Retrofit  retrofit = new Retrofit.Builder().baseUrl("http://"+url+":4000/")
         .addConverterFactory(GsonConverterFactory.create())
         .build();
urls=url;


 retrofits = retrofit.create(Service.class)
```

Una vez obtenida la instancia se realiza la llamada

6.3.3 Patrón arquitectónico usado

El patrón arquitectónico usado para el desarrollo de la aplicación Android, es el MVP (Modelo Vista Presentador).

Este es una derivación del MVC (Modelo Vista Controlador), con la principal diferencia de que en el patrón MVP el presentador asume toda la funcionalidad, colocándose toda lógica de presentación en este.

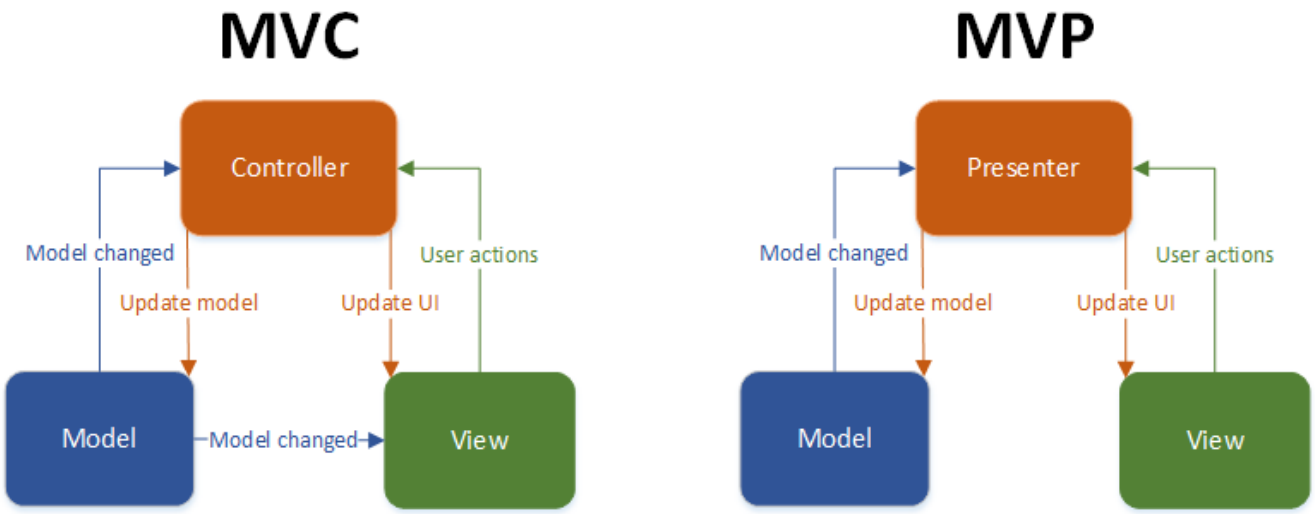

Ilustración 63: MVC y MVP

Las diferentes capas que componen el patrón MVP son las siguientes:

- Modelo: es la capa encargada de la gestión de los datos. En ella se encontrarían las clases ocupadas en la lógica de negocio.
- Vista: se encarga de mostrar los datos. En esta capa encontramos las vistas y los fragmentos.
- Presentador: es la capa encargada de conectar la interfaz gráfica con los datos.

6.3.4 Patrón de diseño usado

En el desarrollo de la aplicación se ha usado el patrón de diseño conocido como patrón Singleton.

Este nos permite restringir la creación de objetos pertenecientes a una clase, garantizando que una clase solo tenga una instancia y proporcionar un acceso global a ella.

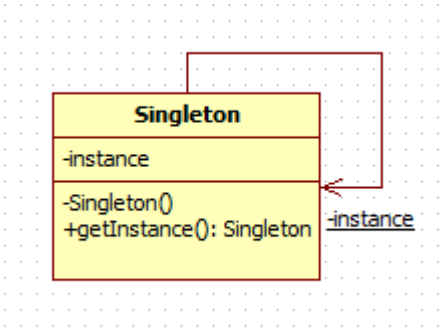

Ilustración 64: Singleton

Este patrón provee una única instancia global gracias a que:

- La propia clase es responsable de crear la única instancia.
- Permite el acceso global a dicha instancia mediante un método de clase.
- Declara el constructor de clase como privado para que no sea instanciable directamente.
- Al estar internamente auto referenciada en lenguajes como Java, el recolector de basura no actúa.

6.4 Servidor Node JS

Para la realización de la comunicación del Smartphone y la aplicación de Unity se ha desarrollado un servidor rest con la tecnología Node JS (explicada en el Capítulo 2) con el cual poder comunicarlos.

El servidor Rest hecho en Node está formado por dos archivos .js, siendo estos restful.js y data.js.

6.4.1 Restful.js

El archivo restful.js es el encargado de arrancar y crear el servidor local al que realizan las peticiones rest, también de llevar a cabo dichas peticiones.

Líneas que arrancan el servidor cuando se ejecuta el archivo restful.js.

```
app.listen(4000,function(){
        console.log("Node server running on http://localhost:4000")

});
```

Líneas que hacen las peticiones get y put, que en este caso son las únicas operaciones rest que se implementan.

```
app.get('/coor',rutas.obtener);
app.put('/coor/',rutas.agregar);
```

Donde rutas es una variable que se usa para ejecutar las operaciones del archivo data. En esta línea se almacena el archivo tipo data en la variable rutas.

```
rutas = require('./data.js')
```

6.4.2 Data.js

El archivo data.js contiene las operaciones que se ejecutan cuando se realiza un get o un put y son llamadas desde el archivo restful.js como se mencionó en el punto anterior. Este fichero indica el formato de los datos que se reciben, y que se envían.

En esta imagen se muestra, como es la variable en la que se almacenan los datos, que se envían y se reciben del servidor. En este caso solo son cuatro, referentes a los cuatro valores que componen un cuaternión.

```
var coordenadas ={
    X:Number,
    Y:Number,
    Z:Number,
    W:Number
}
```

La función “Obtener” es llamada cuando se realiza una petición get y devuelve el cuaternión que está almacenado en ese momento en la variable coordenadas, variable que se mostró en la imagen anterior.

```
exports.obtener=function(req,res){
            res.send(coordenadas);
}
```

La función “agregar” es llamada cuando se realiza la operación put y se encarga de añadir los nuevos valores del cuaternión en la variable coordenadas.

```
exports.agregar=function(req,res){
    coordenadas={
        X:req.body.X,
        Y:req.body.Y,
        Z:req.body.Z,
        W:req.body.W
    }
    res.send(coordenadas);
  }
```

6.5 Unity

6.5.1 Implementación de la base de datos

Para el almacenamiento de los datos se ha optado por el uso de la base de datos SQLite y la biblioteca sqlite-net.

Esta biblioteca nos proporciona las ventajas siguientes:

- Muy fácil de integrar con proyectos existentes pudiéndose ejecutar en todas las plataformas .NET
- Su uso junto con SQLite hace que esta sea rápida y eficiente sin ser un cuello de botella en las consultas.
- Métodos simples para realizar las operaciones CRUD de forma segura y recuperación de los datos fuertemente tipados.
- Trabaja el modelo de datos sin obligar a cambiar las clases.
- Permite realizar operaciones síncronas y asíncronas.

Para ello se ha hecho uso de SQLite4Unity3d, un plugin que facilita el uso de SQLite en proyectos de Unity realizando solo operaciones síncronas.

6.5.2 Atributos y tipos de datos SQLite

Los siguientes atributos son necesarios añadirlos en la clase modelo para que sea reconocida por la base de datos.

Atributos:

- **[PrimaryKey]:** indica la clave primaria.

- **[AutoIncrement]:** sirve para generar e incrementar el contenido de la variable de forma automática en la base de datos. El tipo de propiedad debe ser un número entero y también debe estar marcado con el atributo PrimaryKey.

- **[Indexed]:** esta propiedad debe tener un índice creado para ello.

- **[MaxLength]:** si esta propiedad es una Cadena, MaxLength se usa para especificar el tamaño máximo varchar. La longitud máxima predeterminada es 140.

- **[Ignore]:** esta variable no estará en la tabla.

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

- **[ForeignKey]:** identifica la clave foránea.

Los tipos de datos soportados son:

- **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 y Doubles:** son almacenados como float de SQLite.
- **Strings:** 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.

6.5.3 Consultas y operaciones en SQLite

Las consultas y operaciones que nos ofrece son:

- **CreateTable:** para la generación de tablas.

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

- **Insert:** usado para insertar filas en la base de datos. Si la tabla contiene auto-increment primary key, el valor de la clave primaria estará disponible después de la inserción.

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

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

- **Query:** realizar consultas en la base de datos a bajo nivel.

```
   public static IEnumerable<Valuation> QueryValuations (SQLiteConnection db, Stock stock)

   {
       return db.Query<Valuation> ("select * from Valuation where
   StockId =?", stock.Id);
   }
```

El parámetro genérico del método Query especifica el tipo de objeto para crear en cada fila. Puede ser una clase tabla o cualquier otra clase cuyas propiedades públicas coincidan con la columna devuelta por la consulta. Pudiendo reescribir la consulta anterior como:

```
   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);
   }
```

6.5.4 Comunicación con el servidor

En este punto se muestra cómo realizar la conexión y la posterior obtención de los datos del servidor:

```
public class ClienteU
{
    Coord cordenadas = new Coord();
    byte Error = 0;

    public static ClienteU Instance()
    {
        if (_instance == null)
        {
            lock (typeof(ClienteU))
            {
                if (_instance == null)
                {
                    _instance = new ClienteU();
                }
            }
        }
        return _instance;
    }
    protected ClienteU()
    {

    }
    private static volatile ClienteU _instance = null;
```

En la clase del cuadro anterior se hizo uso del patrón de diseño “patrón Singleton” que ya fue explicado en el punto 6.2.4**,** garantizando que la misma solo tenga una instancia y proporcionando un acceso global a ella.

```
        public IEnumerator getData()
        {

            UnityWebRequest www = UnityWebRequest.Get("http://localhost:4000/coor/");
            yield return www.Send();
            if (www.isError)
            {
                Debug.Log(www.error);
                Error = 1;

                yield return null;
            }
            else
            {


                Error = 0;


                cordenadas = JsonUtility.FromJson<Coord>(www.downloadHandler.text);


                yield return null;
            }

        }
        }
```

En esta ocasión solo usa la operación get, ya que lo único que interesa es la obtención de los datos del servidor.

Para su realización se utilizó la corrutina que en C# se representa mediante el uso de una función **IEnumerator** y la instrucción de retorno **yield**, consiguiendo que solo se ejecute una vez por cada frame e impidiendo así que el juego se bloque en ese punto.

6.5.5 Movimiento del Avatar

En este caso se puede ver como se usan los datos enviados por el móvil para gestionar el movimiento del avatar. Como ejemplo se desarrolla la actividad baño:

```
private void Mover()
    {
        Quaternion rotacion;
        Quaternion cepillar = Quaternion.Euler(0f, 0f, 0f);
        Quaternion x = Quaternion.Euler(0f, 0f, 0f);
        Quaternion y = Quaternion.Euler(0f, 0f, 0f);
        Quaternion z = Quaternion.Euler(0f, 0f, 0f);

        posicionMOV.Set(cordenada.Z, cordenada.Y, cordenada.X, -cordenada.W);

        //Controla si el movimiento del brazo es libre o está limitado a una dirección

        switch (libre)
        {
            case true:
                posicionMOV = Quaternion.Inverse(inicial) * posicionMOV;
                cepillar.Set(0f, 0f, posicionMOV.z * 0.2f, posicionMOV.w);
                break;

            case false:
                posicionMOV = Quaternion.Inverse(inicial) * posicionMOV;
                x.Set(posicionMOV.x, 0f, 0f, posicionMOV.w);
                y.Set(0f, posicionMOV.y, 0f, posicionMOV.w);
                z.Set(0f, 0f, posicionMOV.z, posicionMOV.w);
                posicionMOV = x;
                break;
        }
         //Controla el brazo que se debe mover
        switch (brazo)
        {
            case 0:
                if (tope)
                {
                    RightUpperArm.localRotation = Quaternion.Euler(21.197f, -32.51f, -32.879f);

                    RightElbow.localRotation = Quaternion.Euler(132.611f, -5.122986f, -14.72998f) *
cepillar;


                    Debug.Log(Vector3.Distance(Head.position, LeftHand.position));

                    controlMovHombro = false;
                }
                else if (getDerecha()) { RightElbow.localRotation = Quaternion.Euler(-16.421f,

75.62801f, 52.963f) * x;

                }

                else
                {
                    rotacion = posicionINI[0] * posicionMOV;
                    RightElbow.rotation = rotacion;
                }


                anguloTresPuntos(RightUpperArm, RightElbow, RightHand);
```

```
                if (controlMovHombro)
                {
                    movHombro(0);
                }

                break;

            case 1:
                if (tope)
                {

                    LeftUpperArm.localRotation = Quaternion.Euler(22.557f, 6.522f, 8.053f);

                    LeftElbow.localRotation = Quaternion.Euler(129.059f, 50.05899f, 20.61299f) *
cepillar;


                    controlMovHombro = false;
                }
                else if (getIzquierda()) { LeftElbow.localRotation = Quaternion.Euler(34.701f, -
41.597f, -52.851f) * x; }
                else
                {
                    rotacion = posicionINI[1] * posicionMOV;

                    LeftElbow.rotation = rotacion;
                }


                anguloTresPuntos(LeftUpperArm, LeftElbow, LeftHand);
                if (controlMovHombro)
                {
                    movHombro(1);
                }
                break;
        }
       }
```

Sin embargo, cuando se realiza el movimiento del brazo, el paciente solo puede controlar una articulación a la vez, que en este caso es el codo. Por lo tanto, se ha tenido que crear un controlador que mueva el hombro.

```
private void movHombro(byte hom)
    {

        var grados = 0f;

        var distancia = 0f;
        switch (hom)
        {
            case 0:
                distancia = Vector3.Distance(Head.position, RightHand.position);
                grados = anguloHombro(distancia, velo, Maxi, 0);

                RightUpperArm.rotation = Quaternion.Euler(35, 0, -grados) *
posicionINIRUpper;
                girarmuñeca(0);
                break;

            case 1:
                distancia = Vector3.Distance(Head.position, LeftHand.position);
                grados = anguloHombro(distancia, velo, Maxi, 1);
                LeftUpperArm.rotation = Quaternion.Euler(35, 0, grados) * posicionINILUpper;
                girarmuñeca(1);
                break;
           }
```

Para calcular la rotación del hombro se ha hecho uso de la distancia de la mano del avatar con respecto al cuerpo.

```
private float anguloHombro(float distancia, int vel, int max, int pos)
    {
        var angulomov = 0;
        if (cont < max && distancia < 4)
        {
            cont++; angulomov = cont / vel;
        }

        else if (distancia > 4 && distancia <= 4.2)
        {
            angulomov = cont / vel;
        }
        else if (distancia > 4.2 && cont > 0) { cont--; angulomov = cont / vel; }
        else if (cont == 0) { angulomov = 0; }
        else if (cont >= max)
        {
            angulomov = cont / vel;
        }
        return angulomov;
    }
```

6.5.6 Calculo del ángulo

Se utilizan tres puntos diferentes del cuerpo (hombro, codo, mano):

```
   void anguloTresPuntos(Transform x, Transform y, Transform z)
       {
           Vector3  VPoint1  =  new  Vector3(x.transform.position.x,  x.transform.position.y,
x.transform.position.z);
           Vector3  VPoint2  =  new  Vector3(y.transform.position.x,  y.transform.position.y,
y.transform.position.z);
           Vector3  VPoint3  =  new  Vector3(z.transform.position.x,  z.transform.position.y,
z.transform.position.z);

           double angle = Angulo.AngleBetweenTwoVectors(VPoint2 - VPoint1, VPoint2 - VPoint3);

           if (angle > anguloMaximo)
           {
               anguloMaximo = (float)angle;
           }
           if (angle < anguloMinimo)
           {
               anguloMinimo = (float)angle;

           }

       }
```

Los tres puntos nos sirven en la obtención de los dos vectores, que posteriormente se van a usar para realizar el producto escalar $\vec{x} \cdot \vec{y} = |\vec{x}||\vec{y}|$ y obtener así el ángulo entre ellos:

```
   public static double AnguloEntreVectores(Vector3 vectorA, Vector3 vectorB)
       {
           double dotProduct;
           vectorA.Normalize();
           vectorB.Normalize();
           dotProduct = Vector3.Dot(vectorA, vectorB);

           return (double)Math.Acos(dotProduct) / Math.PI * 180;
       }
```

Capítulo 7

7 Pruebas

7.1 Introducción

En este capítulo se recogerán los casos de prueba que se han realizado sobre el sistema, con el fin de detectar posibles errores.

Las pruebas que se muestran a continuación son de caja negra, ya que las pruebas de caja blanca se han estado realizando durante la implementación del programa. El principal objetivo es comprobar el cumplimiento de los requisitos especificados y la consecución de un funcionamiento correcto.

7.2 Casos de prueba

App Smartphone

Tabla 33: Prueba-App Smartphone

Prueba	Resultado	Valoración
Comenzar medición y paso de los datos.	Tras pulsar Iniciar en la aplicación, esta comienza a medir y enviar los datos al servidor.	OK
Modificar dirección a la que conectarse por una diferente a la del servidor.	El Smartphone no se podrá conectar al servidor y saldrá un error de conexión.	OK
Modificar dirección a la que conectarse por una igual a la del servidor	El Smartphone se conectará con el servidor y comenzará a enviar los datos.	OK

Registrar Paciente

Tabla 34: Prueba-Registrar paciente

Prueba	Resultado	Valoración
Registrar al paciente rellenando todos los campos del formulario.	Los datos se han almacenado correctamente en la base de datos.	OK
Registrar al paciente rellenando solo los campos obligatorios del formulario.	Los datos se han almacenado correctamente en la base de datos.	OK
Registrar al paciente sin rellenar todos los campos obligatorios.	Sale un mensaje de error indicando que no se han completado todos los campos obligatorios y los datos no se guardan.	OK
Registrar al paciente con un nombre de usuario que ya existe.	Sale un mensaje de error indicado que el nombre de usuario no es válido.	OK

Iniciar sesión

Tabla 35: Prueba-Iniciar sesión

Prueba	Resultado	Valoración
Iniciar sesión con un nombre de usuario registrado.	La sesión se inicia correctamente.	OK
Iniciar sesión con un nombre de usuario no registrado.	El sistema no inicia sesión y muestra un mensaje de error.	OK
Iniciar sesión dejando el campo vacío.	El sistema no inicia sesión.	OK

Cerrar sesión

Tabla 36: Prueba-Cerrar Sesión

Prueba	Resultado	Valoración
Se cierra la sesión de un paciente.	El sistema cierra la sesión correctamente y vuelve a la ventana de iniciar sesión.	OK

Consultar datos del paciente

Tabla 37: Prueba-Consultar datos pacientes

Prueba	Resultado	Valoración
Seleccionar la opción de consultar los datos del paciente en el menú inicial.	El sistema muestra de forma correcta los datos personales del paciente, así como sus ejercicios realizados.	OK

Modificar los datos del paciente

Tabla 38: Prueba-Modificar los datos del paciente

Prueba	Resultado	Valoración
Desde la ventana donde se muestran los datos del paciente se modifican los campos sin dejar ninguno vacío.	El sistema almacena los datos modificados de forma correcta en la base de datos.	OK
Desde la ventana donde se muestran los datos del paciente, se modifican los campos dejando alguno no obligatorio vacío.	El sistema almacena los datos modificados de forma correcta en la base de datos.	OK
Desde la ventana donde se muestran los datos del paciente, se modifican los campos dejando alguno obligatorio vacío.	El sistema no almacena los datos modificados y muestra un mensaje de error.	OK

Ver resultado de un ejercicio

Tabla 39: Prueba-Ver resultado de un ejercicio

Prueba	Resultado esperado	Valoración de la prueba
Desde la ventana donde se muestran los datos del paciente, se selecciona la actividad de salón y el ejercicio de la taza.	El sistema muestra los resultados de la actividad en el panel de detalles.	OK
Desde la ventana donde se muestran los datos del paciente, se selecciona la actividad de salón y el ejercicio del peine.	El sistema muestra los resultados de la actividad en el panel de detalles.	OK
Desde la ventana donde se muestran los datos del paciente, se selecciona la actividad de baño y el ejercicio de lavar los dientes.	El sistema no muestra los datos de forma correcta, cortando parte de la información.	Error. Superposición de botón de borrar sobre cuadro de texto. Solucionado. Se ha retocado la posición del botón y el tamaño de este, corrigiendo la visualización.
Desde la ventana donde se muestran los datos del paciente, se selecciona la actividad de emparejar cartas.	El sistema muestra los resultados de la actividad en el panel de detalles.	OK

Borrar paciente

Tabla 40: Prueba-Borrar paciente

Prueba	Resultado	Valoración
Desde la ventana donde se muestran los datos del paciente, se selecciona el botón “eliminar paciente”.	El sistema borra todos los datos del paciente de la base de datos y vuelve a la pantalla de login.	OK

Seleccionar ejercicio

Tabla 41: Prueba-Seleccionar ejercicio

Prueba	Resultado	Valoración
Después de haber seleccionado la opción de “brazo derecho”, seleccionar “Ejercicio salón”.	El sistema carga la actividad del salón para el brazo derecho.	OK
Después de haber seleccionado la opción de “brazo izquierdo”, seleccionar “Ejercicio salón”.	El sistema carga la actividad del salón para el brazo izquierdo.	OK
Después de haber seleccionado la opción de “brazo derecho”, seleccionar “Ejercicio baño”.	El sistema carga la actividad del baño para el brazo derecho.	OK
Después de haber seleccionado la opción de “brazo izquierdo”, seleccionar “Ejercicio baño”.	El sistema carga la actividad del baño para el brazo izquierdo.	OK
Después de haber seleccionado la opción de “brazo derecho”, seleccionar “Ejercicio cartas”.	El sistema carga la actividad de emparejar cartas para el brazo derecho.	OK
Después de haber seleccionado la opción de “brazo izquierdo”, seleccionar “Ejercicio cartas”.	El sistema carga la actividad de emparejar cartas para el brazo izquierdo.	OK

Actividad salón

Tabla 42: Prueba-Actividad salón

Prueba	Resultado	Valoración
Colisiona la mano derecha con la taza.	La taza quedo vinculada con la mano derecha.	OK
Colisiona la mano izquierda con la taza.	La taza quedo vinculada con la mano izquierda.	OK
Una vez agarrada la taza con la mano derecha, llevarla hasta la boca.	La taza colisiona con la boca y el ejercicio finaliza correctamente.	OK
Una vez agarrada la taza con la mano izquierda, llevarla hasta la boca.	La taza colisiona con la boca y el ejercicio finaliza correctamente.	OK
Colisiona la mano derecha con el peine.	El peine quedo vinculado con la mano derecha.	OK
Colisiona la mano izquierda con el peine	El peine quedo vinculado con la mano izquierda.	OK
Una vez agarrado el peine con la mano derecha, llevarlo hasta la cabeza.	El peine colisiona con el pelo y el ejercicio finaliza correctamente.	OK
Una vez agarrado el peine con la mano izquierda, llevarlo hasta la cabeza.	El peine colisiona con el pelo y el ejercicio finaliza correctamente.	OK

Actividad baño

Tabla 43: Prueba-Actividad baño

Prueba	Resultado	Valoración
Colisionar la mano derecha del avatar con el cepillo de dientes.	El cepillo de dientes fue vinculado a la mano derecha una vez han colisionado	OK
Colisionar la mano izquierda del avatar con el cepillo de dientes.	El cepillo de dientes fue vinculado a la mano izquierda una vez han colisionado	OK
Una vez cogido el cepillo de dientes, colisionar el cepillo con el agua.	Tras la colisión el cepillo fue mojado.	OK
Cuando esta mojado el cepillo, colisionar el cepillo con la boca.	La colisión a la altura de la boca fue correctamente detectada.	OK
Cuando el cepillo ha colisionado con la boca.	Las repeticiones de cepillado son detectadas	Error. Las repeticiones no son detectadas de forma correcta. Solucionado. Se ha ajustado mejor la posición del sistema de colisiones.

Actividad emparejar cartas

Tabla 44: Tabla 61: Prueba-Actividad emparejar cartas

Prueba	Resultado	Valoración
Seleccionar dos cartas idénticas	Las dos cartas son eliminadas.	OK
Seleccionar dos cartas distintas.	Las dos cartas seleccionadas son deseleccionadas.	OK
Eliminar todas parejas de cartas.	La actividad fue finalizada correctamente.	OK
Colisionar la mano con el panel de ayuda	Las instrucciones de la actividad son mostradas	OK

Finalizar Actividad

Tabla 45: Prueba-Finalizar Actividad

Prueba	Resultado	Valoración
Tras haber completado una actividad.	El sistema muestra un menú con tres opciones.	OK

Cambiar de brazo

Tabla 46: Prueba- Cambiar de brazo

Prueba	Resultado	Valoración
En el menú de finalizar actividad, se selecciona la opción cambiar de brazo.	El sistema vuelve a cargar la actividad con el brazo contrario.	OK

Salir de la Actividad

Tabla 47: Prueba-Salir de la actividad

Prueba	Resultado	Valoración
Seleccionar la opción de “Salir” del menú Esc.	El sistema vuelve al menú inicial.	OK
Seleccionar la opción de “Salir” del menú de finalizar actividad.	El sistema vuelve al menú inicial.	OK

Reiniciar

Tabla 48:Prueba-Reiniciar

Prueba	Resultado	Valoración
Seleccionar la opción de “Reiniciar” del menú Esc.	El sistema vuelve a cargar la actividad.	OK
Selecciona la opción “Nueva medición” del menú finalizar actividad.	El sistema vuelve a cargar la actividad.	OK

Salir de la Actividad y guardar

Tabla 49: Prueba- Salir de la Actividad y guardar

Prueba	Resultado	Valoración
Seleccionar la opción de “Salir y guardar” del menú Esc.	El sistema almacena el progreso actual y vuelve al menú inicial.	OK

Pausar Actividad

Tabla 50: Prueba-Pausar actividad

Prueba	Resultado	Valoración
Pulsar botón “Esc” del teclado para pausar la actividad de salón.	La actividad es pausada correctamente y el menú Esc es mostrado.	Incompleta. La pausa realizada no era completa, el brazo del avatar se mueve. Solucionado. Se ha añadido un nuevo sistema de control al script que maneja la pausa, permitiendo bloquear los movimientos del avatar.
Pulsar botón “Esc” del teclado para pausar la actividad de baño.	La actividad es pausada correctamente y el menú Esc es mostrado.	El mismo problema que en el caso anterior, con la misma solución.
Pulsar botón “Esc” del teclado para pausar la actividad de emparejar cartas.	La actividad es pausada correctamente y el menú Esc es mostrado.	OK
Pulsar botón “Esc” del teclado para recuperar la actividad de salón después de haber sido pausada.	La actividad se recuperó correctamente de la pausa.	OK
Pulsar botón “Esc” del teclado para recuperar la actividad de baño después de haber sido pausada.	La actividad se recuperó correctamente de la pausa.	OK
Pulsar botón “Esc” del teclado para recuperar la actividad de emparejar cartas después de haber sido pausada.	La actividad se recuperó correctamente de la pausa	OK

Capítulo 8

8 Conclusiones

8.1 Introducción

En este capítulo se tratarán los diferentes objetivos que se han ido alcanzando durante el desarrollo del proyecto, así como se expondrán posibles caminos a seguir en un futuro.

8.2 Objetivos alcanzados

Una vez finalizado el proyecto se puede decir que se han cumplido todos los objetivos inicialmente planteados y en el tiempo previamente establecido. Las metas alcanzadas han sido:

- Se ha creado una aplicación Android, con la que fácilmente se puede captar el movimiento del paciente y trasladarlo a Unity.

- Se ha implementado un servidor que permita la conexión entre la plataforma de Unity y la aplicación Android.

- Se han desarrollado dos escenas con distintas actividades de la vida diaria, y un minijuego como ejercicio para el usuario.

- Se ha conseguido replicar el movimiento del paciente en Unity usando un Smartphone mediante una conexión wifi, logrando que no haya latencia entre el movimiento en el móvil y del avatar.

- Se han implementado controladores del avatar usando el Smartphone, pudiendo agregar nuevos movimientos.

- Se ha creado una base de datos que almacena los datos del usuario, permitiendo llevar un control de sus progresos.

- Se ha construido una sencilla interfaz en ambas aplicaciones para que el usuario pueda usarlas desde un inicio sin ayuda de un profesional.

8.3 Líneas de trabajo futuras

Teniendo en cuenta las diferentes limitaciones, en especial las temporales, se ha tratado de implementar todo lo posible durante el desarrollo de este Trabajo de Fin de Grado, alcanzando los objetivos establecidos. No obstante, es importante resaltar mejoras que podrían ser incluidas en trabajos futuros:

- La aplicación Android podría almacenar información sobre el progreso del paciente y permitir realizar actividades de forma independiente del ordenador, logrando una mayor libertad.

- Se podrían añadir nuevas actividades al programa de Unity relacionadas con diferentes articulaciones del cuerpo, como pueden ser las del tren inferior.

- Trasladar la aplicación de Unity del pc al Smartphone mediante el uso de gafas virtuales y, junto con un dispositivo con los sensores necesarios (como podría ser una pulsera de actividad), realizar los ejercicios.

- Hacer las aplicaciones, tanto la de Unity como la de Android, compatibles con otras plataformas como macOS, Linux en el caso del pc o IOS en el caso de los Smartphones.

- Realizar un análisis estadístico de los datos del paciente mostrando graficas con el progreso de este.

- Trasladar el almacenamiento de los datos a la nube, logrando centralizar los datos del paciente.

Bibliografía

Libros consultados

Arlow, Jim, Neustadt, Ila. “UML 2” Editorial Anaya Multimedia (2006).

Fowler, Martin. “Patterns of Enterprise Application Architecture” Editorial Addison Wesley (2003).

Hughes, Bob & Cotterell, Mike. “Software Project Management” Editorial McGraw Hill (2002).

Jacobson, Ivar & Booch, Grady & Rumbaugh, James. “El Proceso Unificado de Desarrollo de Software” Editorial Addison Wesley (1999).

Kapandji, A. I. “Fisiologia articular 6º edición tomo 1” Editorial Maloine (2006).

Martín Villamor, Pedro Gabriel.” Extremidades superiores: Artrología y Miología”, material de estudio para la asignatura estructura y función del cuerpo humano-1, de la carrera de Enfermería, 2017-2018, Universidad de Valladolid.

Polonio López, Begoña. “TERAPIA OCUPACIONAL EN DISFUNCIONES FÍSICAS Teoría y práctica”, Editorial Panamericana (2015).

Referencias Web

Abernethy, Michael.” ¿Simplemente qué es Node.js?” Disponible en: https://www.ibm.com/developerworks/ssa/opensource/library/os-nodejs/ (Última consulta: 2 de Abril de 2018).

Chen, An “Evaluación del sensor Kinect v2 para rehabilitación funcional” TFG Grado en Ingeniería Informática (2016) Disponible en: http://uvadoc.uva.es/handle/10324/20946 (Última consulta: 6 de mayo de 2018).

codecoding “SQLite4Unity3d” Disponible en: https://github.com/codecoding/SQLite4Unity3d (Última consulta: 3 de marzo de 2018)

Google “Sensors Overview” Disponible en: https://developer.android.com/guide/topics/sensors/sensors_overview (Última consulta: 10 de febrero de 2018)

Hoffmann, Gernot. ”Application of Quaternions” Disponible en: http://docs-hoffmann.de/quater12012002.pdf (Última consulta: 14 de mayo de 2018).

Matamoros Luque, Rafael “Medición de los movimientos de las articulaciones mediante teléfonos inteligentes” TFG Grado en Ingeniería Informática (2017) Disponible en: http://uvadoc.uva.es/handle/10324/27534 (Última consulta: 9 de abril de 2018).

Node. “Node.js v8.11.1 Documentation” Disponible en: https://nodejs.org/docs/latest-v6.x/api/ (Última consulta: 27 de Febrero de 2018).

Npm. “npm” Disponible en: https://docs.npmjs.com/ (Última consulta: 2 de Abril de 2018).

SQLite “About SQLite”, Disponible en: https://www.sqlite.org/about.html (Última consulta: 10 de mayo de 2018).

SQLite “Documentation”, Disponible en: https://www.sqlite.org/docs.html (Última consulta: 5 de mayo de 2018).

Square. “Retrofit”, Disponible en: http://square.github.io/retrofit/ (Última consulta: 6 de febrero de 2018).

Unity “Unity Assert Store”, Disponible en: https://assetstore.Unity.com/ (Última consulta: 3 de mayo de 2018)

Unity “Unity User Manual”, Disponible en: https://docs.Unity3d.com/Manual/ (Última consulta: 12 de mayo de 2018)

Apéndice 1 - Manual de usuario

1. **Requisitos Software**

- Windows 10 (64 bits)
- Node 6.11.4 o superior
- Android 5.0 o Superior
- Directx 11 o Directx 12

2. **Requisitos Hardware**

- Procesador: AMD FX-7500, 2.10 GHz
- Tarjeta gráfica: Radeon R7 APU
- Memoria: 4 GB RAM
- 1 GB de espacio libre en el disco duro

1 Manual de usuario

1.1 Introducción

En este apéndice se detallan cómo utilizar las aplicaciones de Pc y la del móvil. Para ello se van a ilustrar, paso a paso, las principales funcionalidades de ambas, con el fin de despejar todas las posibles dudas que puedan aparecer sobre su utilización.

1.2 Aplicación Smartphone

1. **Configuración conexión con el servidor**

Para realizar la conexión con el servidor es necesario que el Smartphone esté conectado en la misma red que el pc que ejecuta el servidor. Tras esto, hay que introducir en la aplicación la dirección del servidor, que en este caso será la ip que el ordenador tenga asignada en ese momento.

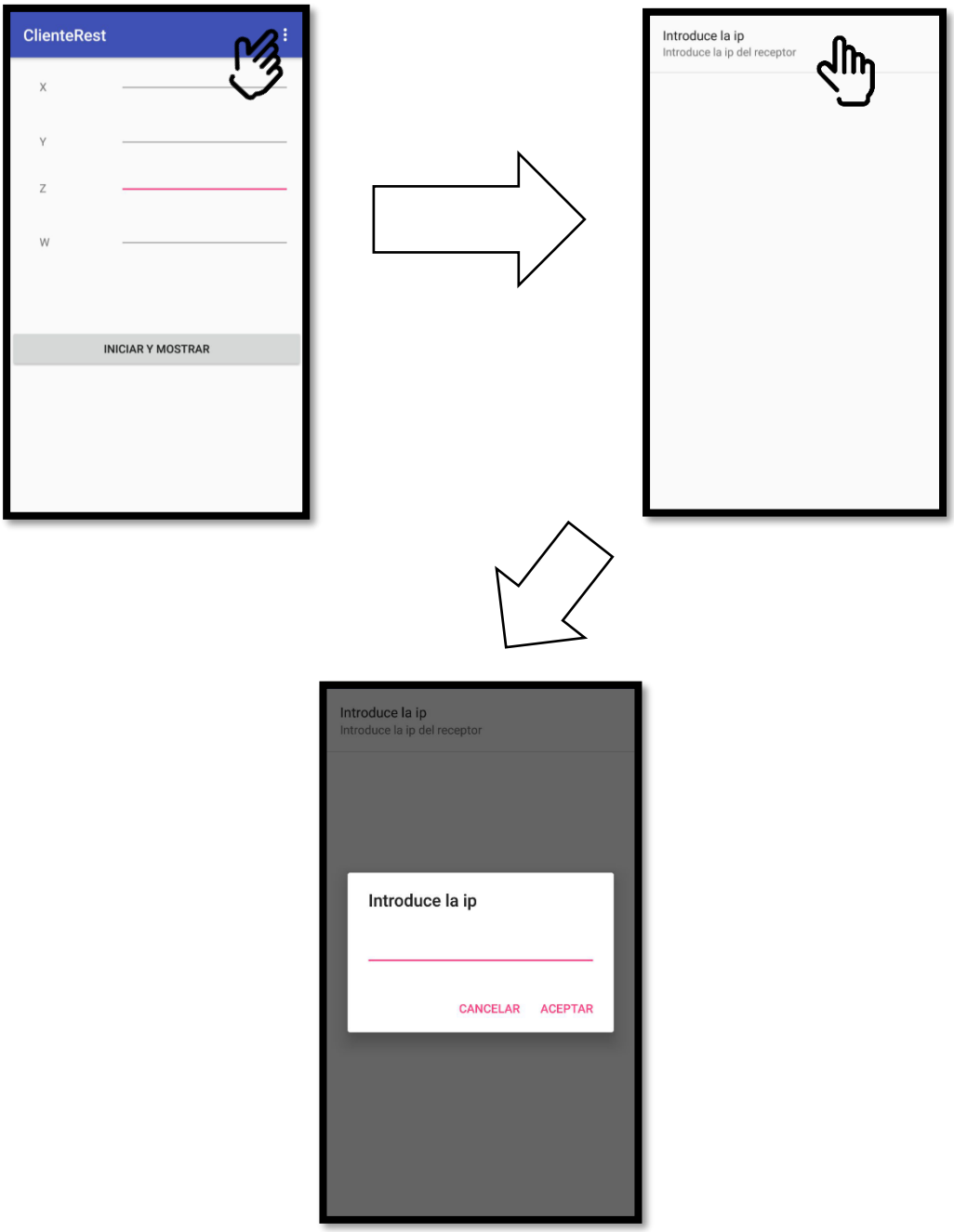

Ilustración 65: Pasos configuración conexión con el servidor

2. **Iniciar toma de mediciones y envió de las mismas al servidor.**

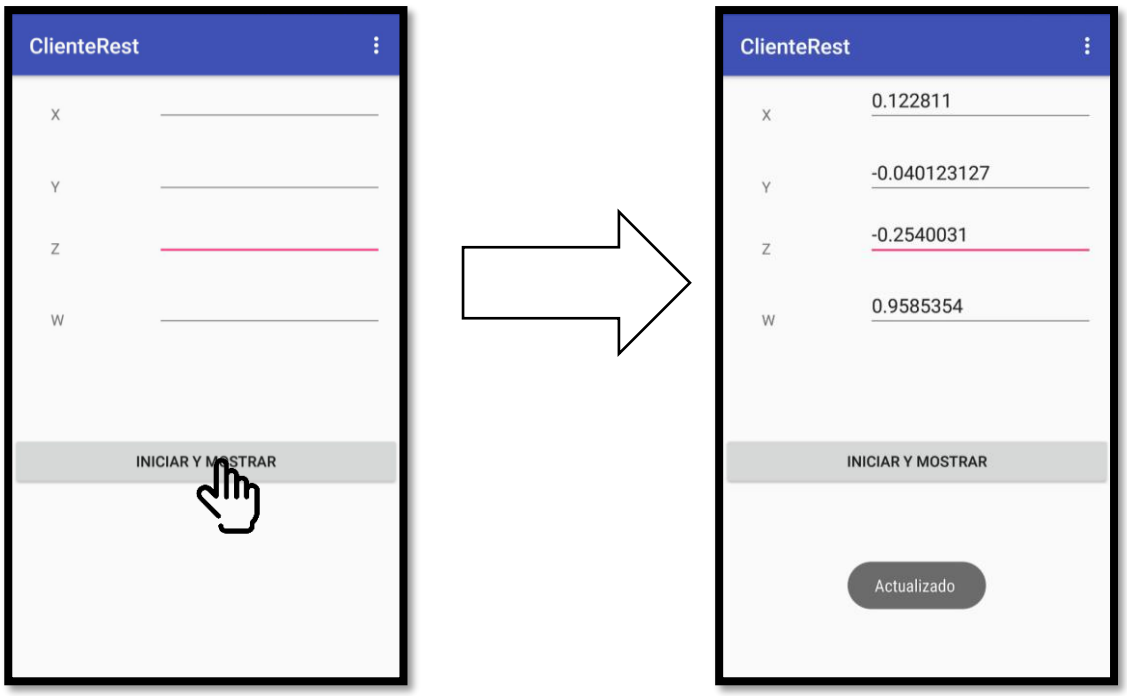

Ilustración 66: Pasos para inicio de mediciones en el Smartphone

3. **Mensajes de información**

- **Error de conexión:** se muestra en la pantalla principal de la aplicación en el cuadro de texto de la X (primera posición). Si al pulsar sobre el botón Iniciar y Mostrar aparecer el mensaje quiere decir que no se ha podido establecer conexión con el servidor.

- **Actualizado:** Se muestra en una burbuja de información en la parte inferior de la pantalla. Indica que las mediciones se están enviando al servidor de forma correcta.

- **Sin conexión con el servidor:** Se muestra en una burbuja de información en la parte inferior de la pantalla. Indica que no se ha podido establecer una conexión con el servidor.

1.3 Aplicación de Unity

1. Inicio

Al ejecutar la aplicación se mostrará la pantalla de inicio. En esta pantalla podemos iniciar la sesión, registrarnos (explicación en el punto 2 registrar) o cerrar la aplicación pulsando el botón salir.

Para iniciar sesión hay que introducir el nombre de usuario con el que se esté registrado y pulsar en el botón iniciar. En caso de que el texto introducido no sea válido saltará un aviso notificando que los datos no son correctos.

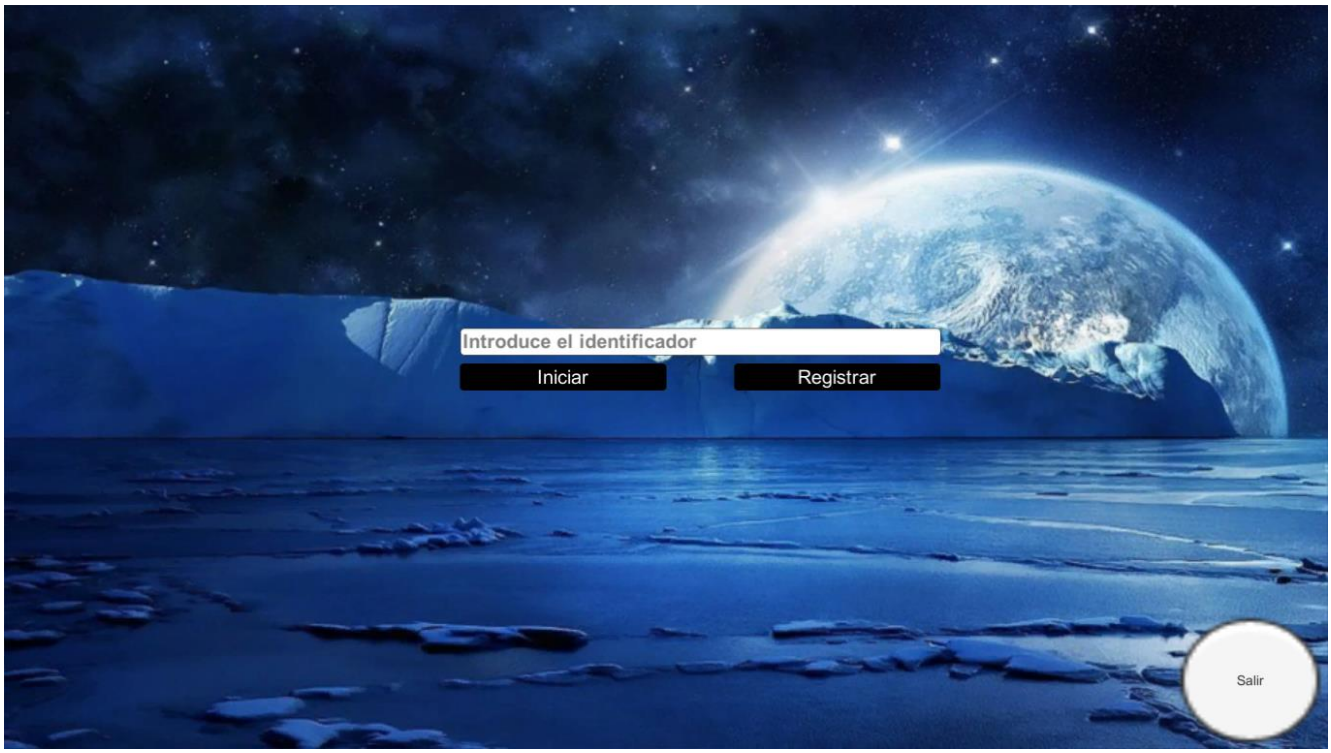

Ilustración 67: Pantalla de login

2. Registrar

Tras pulsar en el botón registrar en la pantalla inicial se nos mostrara un formulario con los datos que tenemos que rellenar, estando marcados con un asterisco (*) aquellos campos que son obligatorios.

Una vez completado el formulario hay que pulsar sobre el botón de crear. Hecho esto se nos mostrará un cuadro con un mensaje informando de la creación correcta del usuario y, al aceptarlo, nos devolverá a la pantalla de inicio.

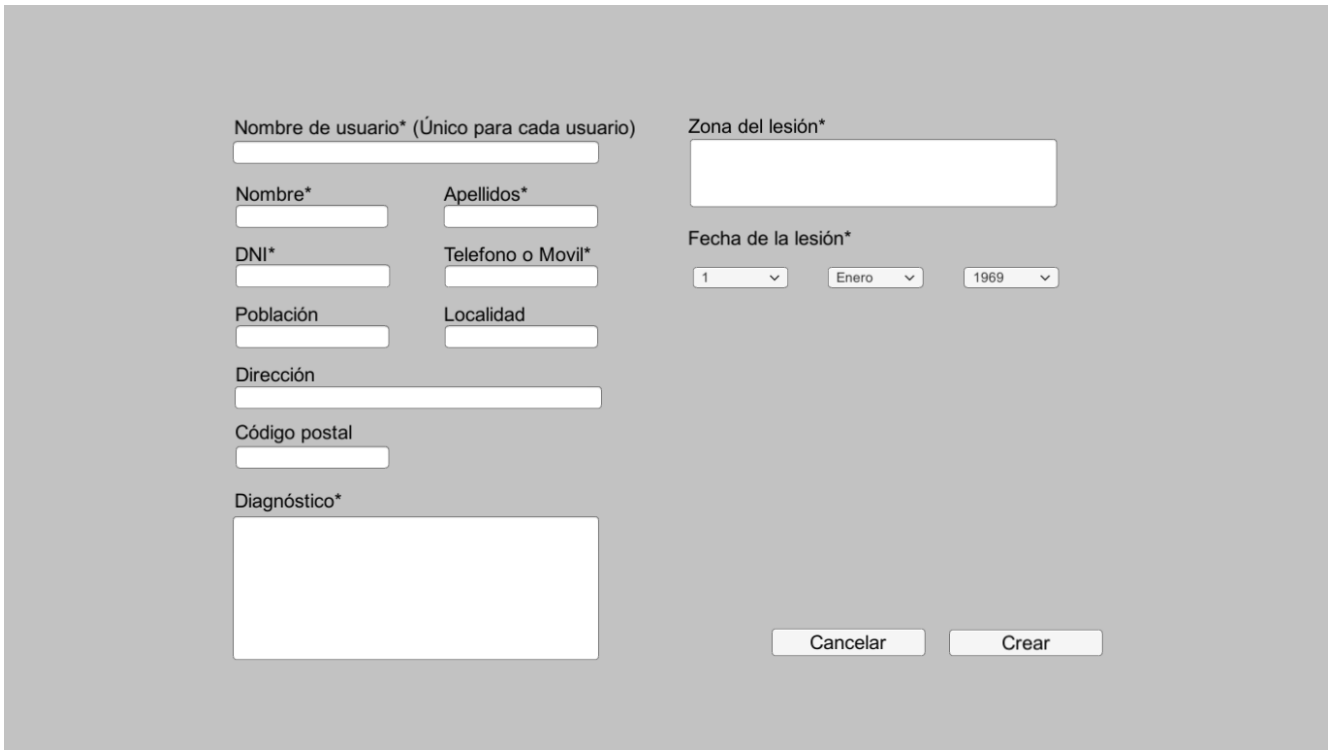

Ilustración 68: Pantalla de registro de pacientes

En el caso de que todos los campos obligatorios no estén completos, o que el nombre de usuario ya exista, se mostrara una alerta cuando se pulse el botón de crear, informando del problema como se muestra en las siguientes imágenes.

Ilustración 69: Mensajes de alerta al registrar un paciente

3. Menú inicial

Una vez se ha iniciado sesión, el menú inicial será cargado mostrando las diferentes opciones: Emparejar Cartas (minijuego), Actividad Salón, Actividad Baño, Datos y Cerrar sesión.

Cada uno de los servicios proporcionados por estas serán detallados en puntos posteriores.

En la parte inferior nos encontraremos dos cuadros de selección que sirven para indicar con que brazo se va a realizar la actividad (al iniciar sesión estará marcada por defecto la opción del brazo derecho), solo se podrá marcar una opción a la vez.

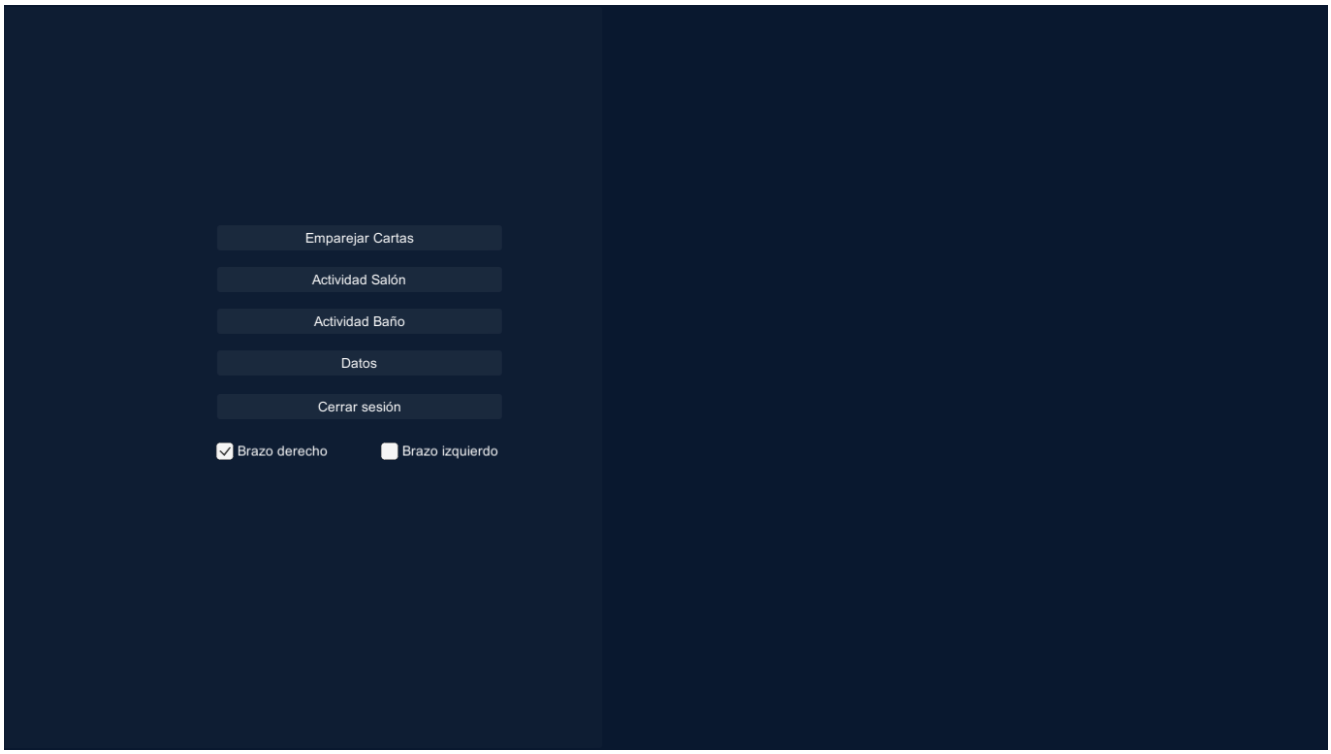

Ilustración 70: Menú inicial

4. Consultar información

Tras pulsar la opción de Datos en el menú inicial se cargará una nueva ventana con la información almacenada del paciente.

En la parte superior del panel izquierdo tenemos los datos personales del paciente. Desde aquí podemos modificar directamente dichos datos a excepción del Nombre de usuario (ID) y la fecha de lesión. Una vez que hayamos realizado los cambios deseados podemos pulsar el botón “Guardar” situado justo debajo para actualizar los cambios en la base de datos.

En la parte inferior del panel izquierdo se encuentran 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.

Los botones que hay en la parte inferior de la pantalla son dos:

- **Eliminar usuario:** se encuentra en la zona derecha y sirve para borrar todos los datos de un usuario. Cuando es pulsado sale una ventana de confirmación y si se confirma, el usuario es borrado y se retorna a la ventana de login.
- **Atrás:** tiene forma de flecha y sirve para volver al menú inicial.

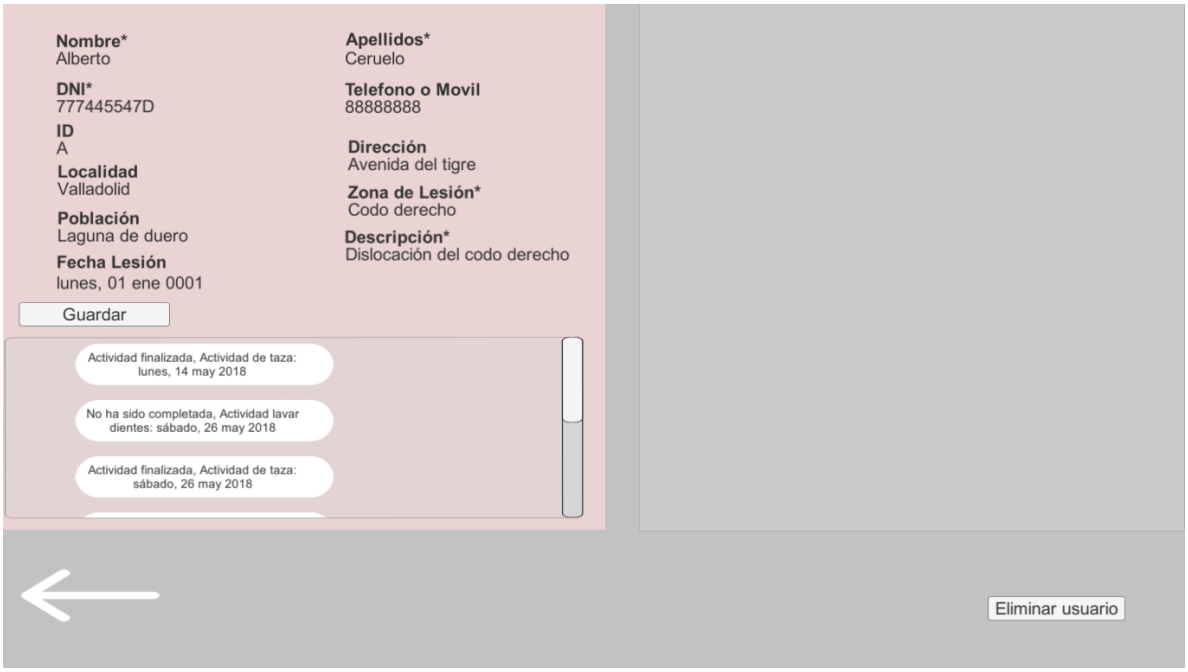

Ilustración 71: Información paciente

Capturas de los resultados de las actividades:

- **Resultados actividad Salón**

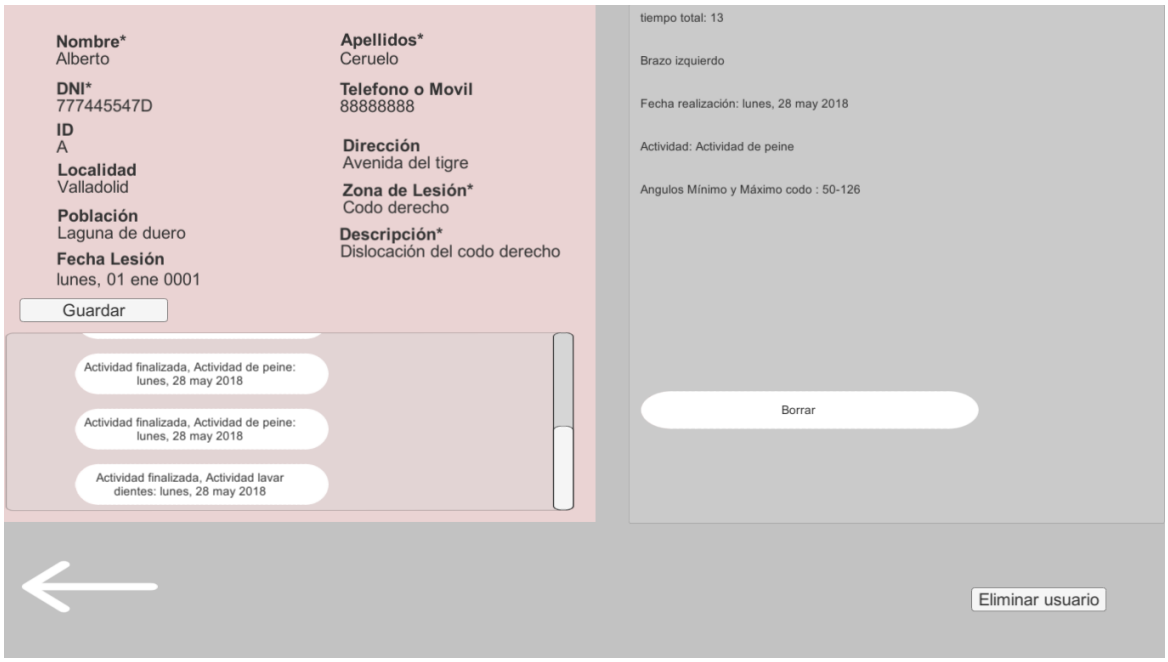

Ilustración 72: Resultados actividad salón

- **Resultados actividad baño**

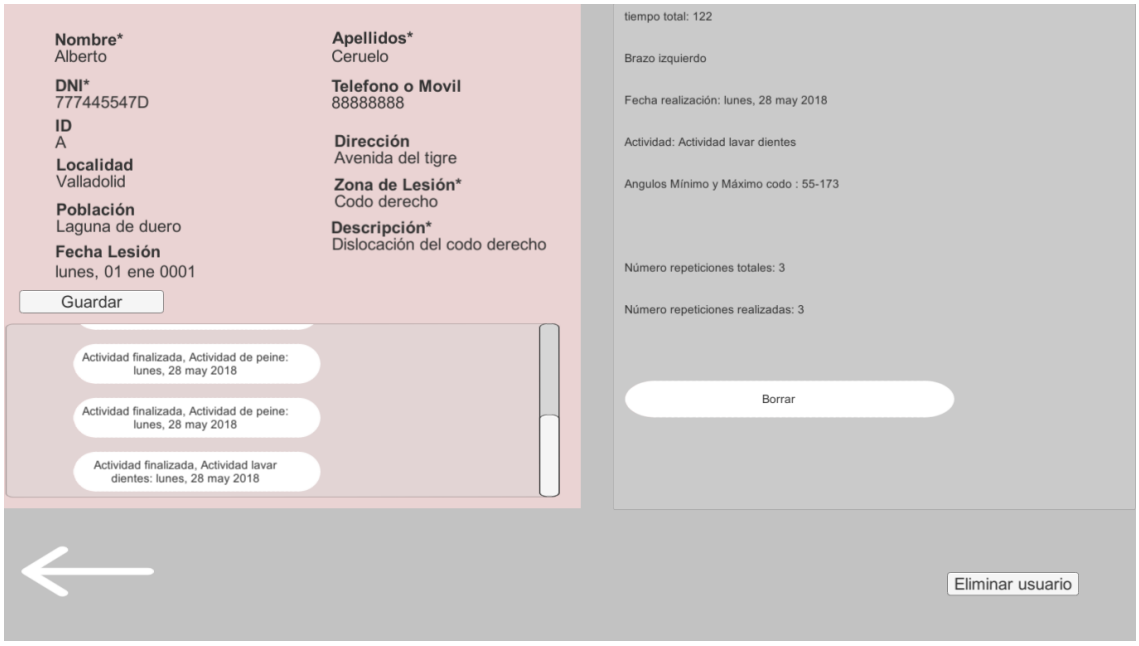

Ilustración 73: Resultados actividad baño

- **Resultados actividad cartas**

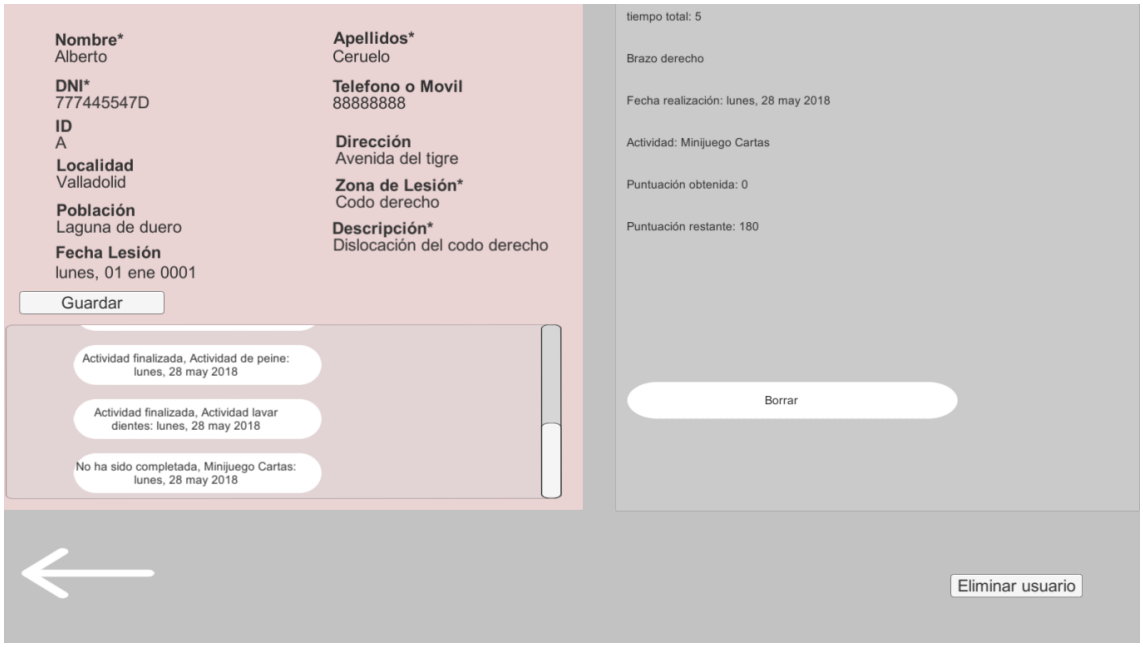

Ilustración 74: Resultados actividad cartas

5. Preparación actividad

La realización de las actividades depende del brazo con el que se vayan a ejecutar:

- Brazo derecho: el móvil hay que cogerlo con la pantalla mirando hacia afuera de la palma de la mano y paralelo al antebrazo, igual que en la ilustración 75 - Izquierda.
- Brazo Izquierdo: en este caso se cogerá con la pantalla mirando hacia la palma de la mano y paralelo al antebrazo, igual que en la ilustración 75 - Derecha.

Ilustración 75: Colocación móvil en mano derecha e izquierda

En todas las actividades habrá una cuenta atrás de tres segundos antes de comenzar. Una vez finalizada, la actividad se iniciará y empezará a contar el tiempo.

Nota*: el tiempo se puede ocultar y desocultar pulsando la tecla (t).

Dentro de las actividades, estas se podrán pausar pulsando la tecla Esc, mostrando un menú con las siguientes opciones:

- Salir: nos sacará de la actividad llevándonos de vuelta al menú inicial.
- Salir y guardar: realizará la misma función que la opción de salir, pero al hacerlo guardará los datos de la actividad.
- Reiniciar: vuelve a cargar la actividad desde el inicio.

Ilustración 76: Menú de escape

Volviendo a pulsar la tecla Esc el menú se cierra y la actividad se reanuda.

Al finalizar una actividad los datos se guardan y se muestran una serie de opciones:

- Cambiar de brazo: reinicia la actividad teniéndose que realizar esta vez con el brazo contrario.
- Salir: vuelve al menú de inicio.
- Nueva medición: carga de nuevo la actividad.

Ilustración 77: Menú finalizar actividad

6. Actividad Emparejar Cartas

Para realizar esta actividad primero hay que ver el punto 4 (Preparación actividad).

Esta consiste en la eliminación de las parejas de cartas iguales, finalizando cuando no quede ninguna. Para seleccionar una carta primero hay que situar la mano del avatar sobre ella, tras esto habrá que girar el móvil ligeramente hacia la derecha y cuando la carta este seleccionada habrá que buscar su pareja y realizar la misma operación. Una vez hecho esto la pareja de cartas será eliminada y la puntuación subirá al marcador.

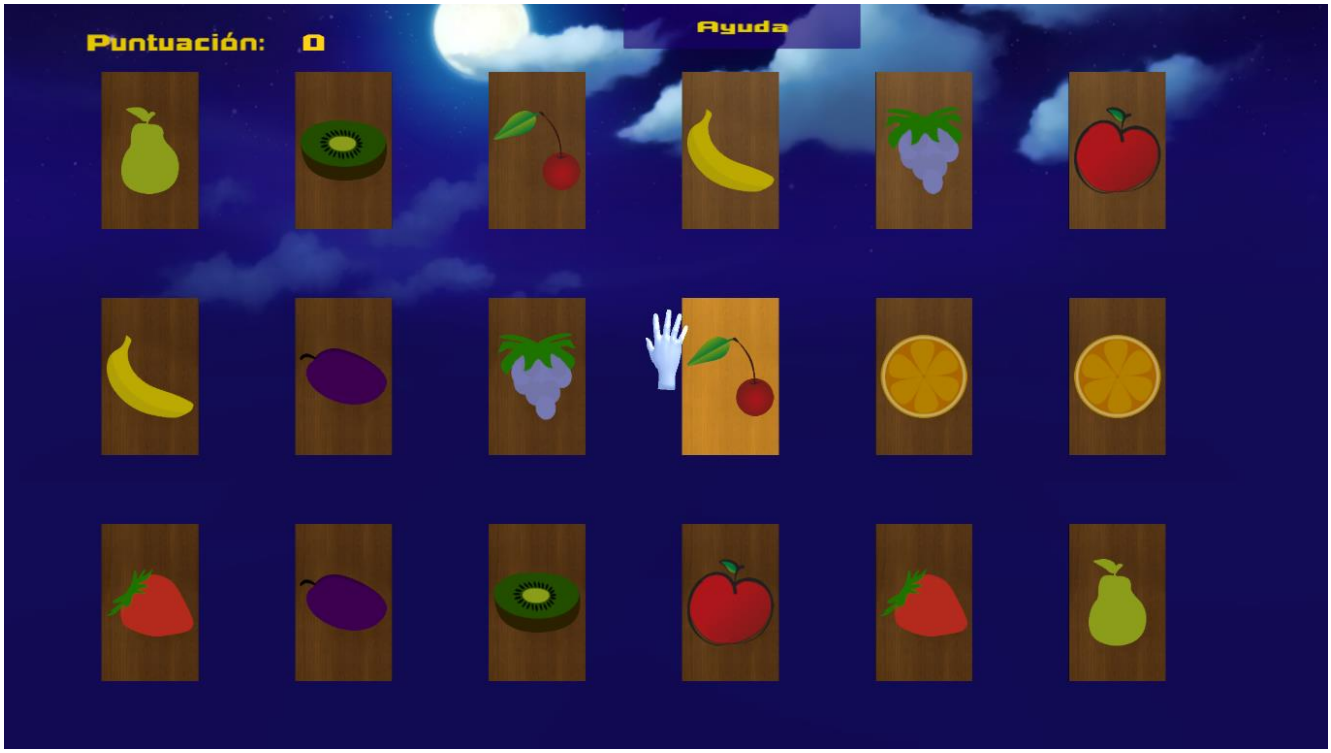

Ilustración 78: Pantalla actividad cartas

7. Actividad Salón

Para realizar esta actividad primero hay que ver el punto 4 (Preparación actividad).

Será necesario sentarse con la espalda recta y con los antebrazos estirados sobre la mesa, como se muestra en la ilustración.

Ilustración 79: Colocación paciente actividad salón

Está compuesta por dos ejercicios diferentes que se eligen al comienzo de la actividad.

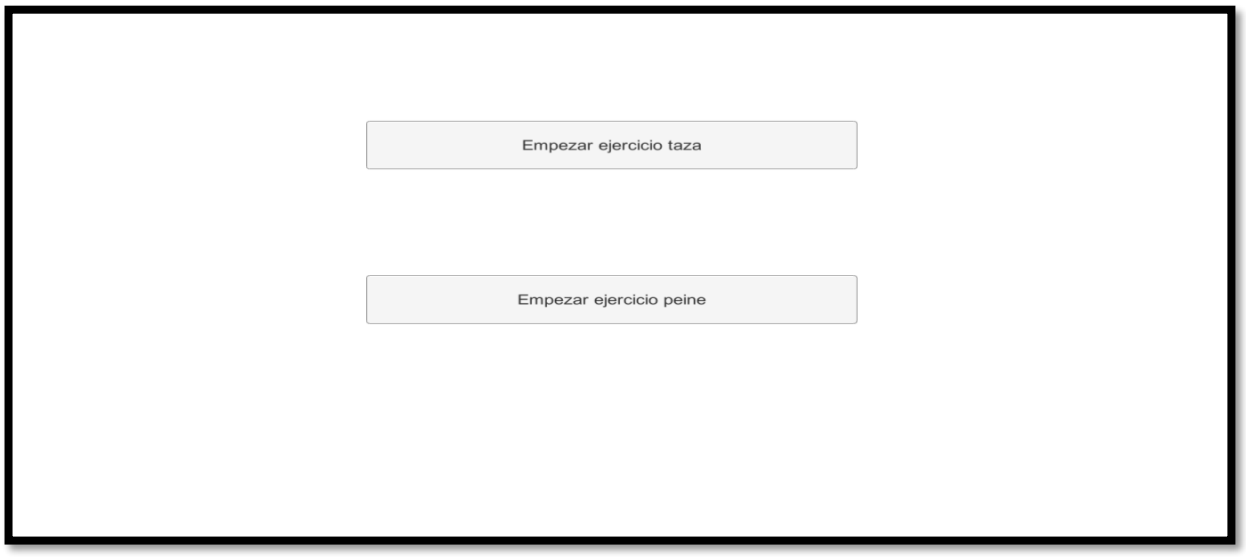

Ilustración 80: Pantalla selección ejercicio actividad salón

El ejercicio de la taza consiste en coger el vaso que hay encima de la mesa y acercarlo a la boca.

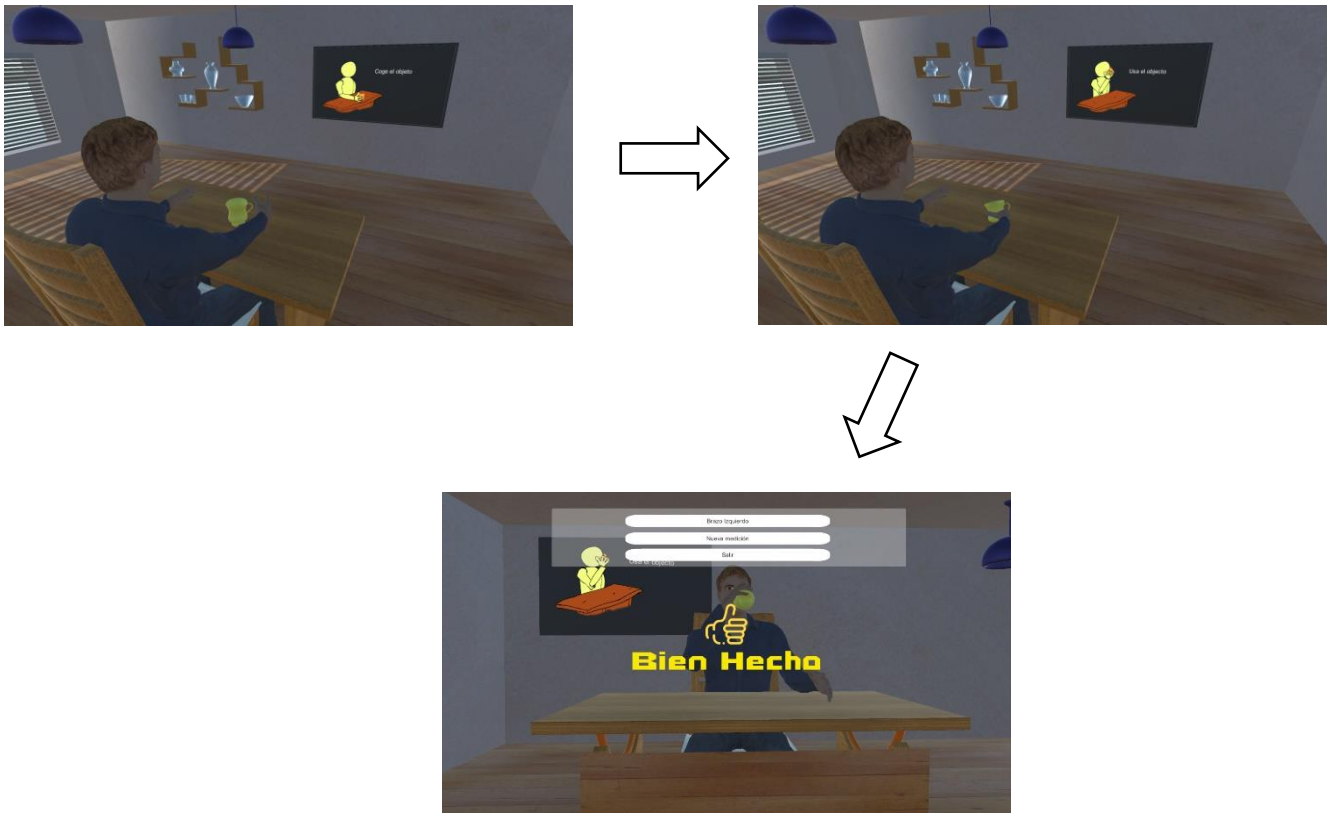

Ilustración 81: Pasos actividad salón-beber

El ejercicio del peine consiste en coger el cepillo que hay encima de la mesa y acercarlo al cuero cabelludo.

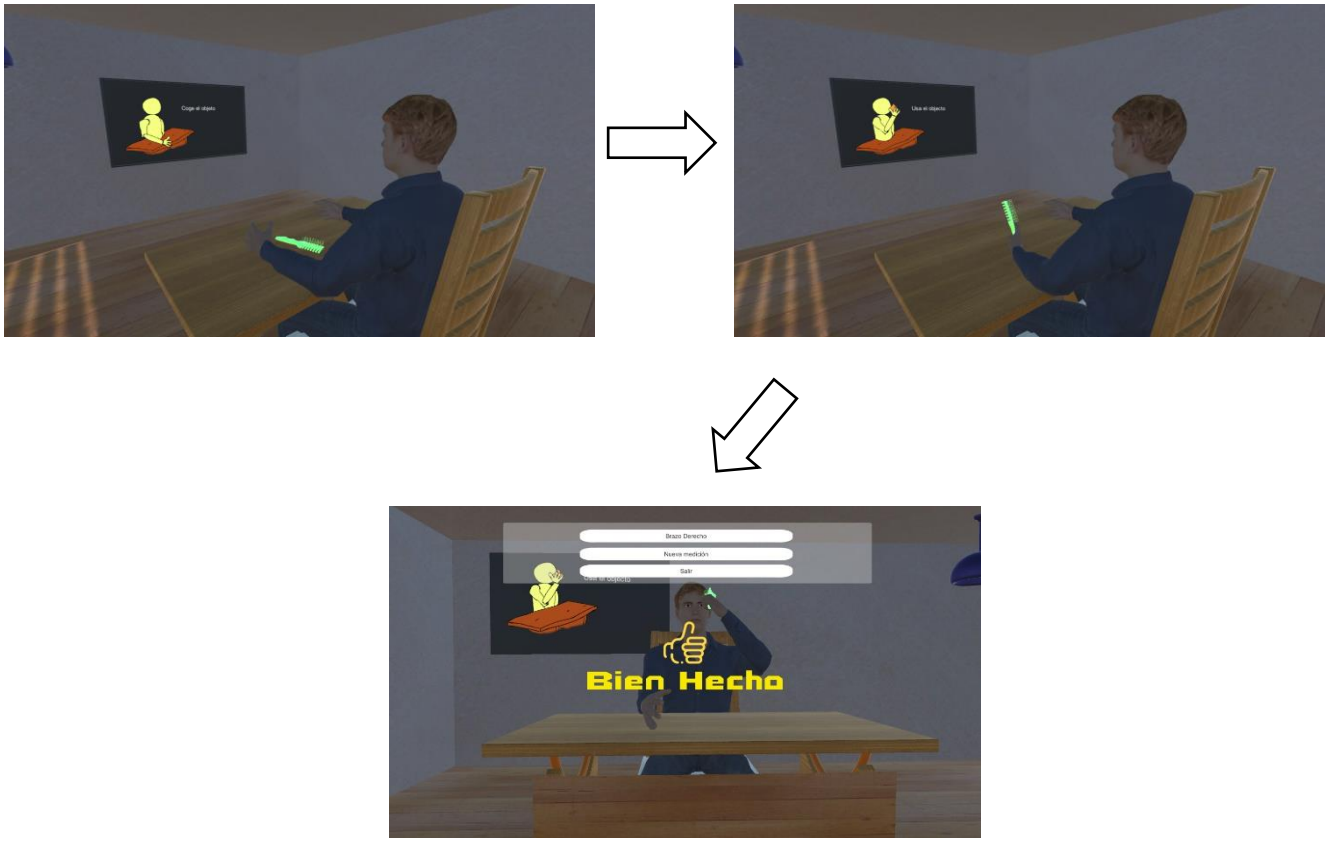

Ilustración 82: Pasos actividad salón-peinarse

La vista de esta actividad depende del brazo con el que se realizará, pudiendo ser observada desde el lateral izquierdo o desde el derecho.

Nota*: la vista frontal es activada pulsando sobre la tecla (d) para el brazo derecho o sobre la tecla (i) para el brazo izquierdo, volviendo a la situación anterior si se teclean de nuevo.

8. Actividad Baño

Para realizar esta actividad primero hay que ver el punto 4 (Preparación actividad).

En su ejecución será necesario situarse de pie con los brazos ligeramente doblados, como se muestra en la ilustración 83.

Ilustración 83: Colocación paciente actividad baño

El primer paso es introducir el número de repeticiones que se quieren realizar. A continuación, pulsar sobre el botón empezar para comenzar el ejercicio.

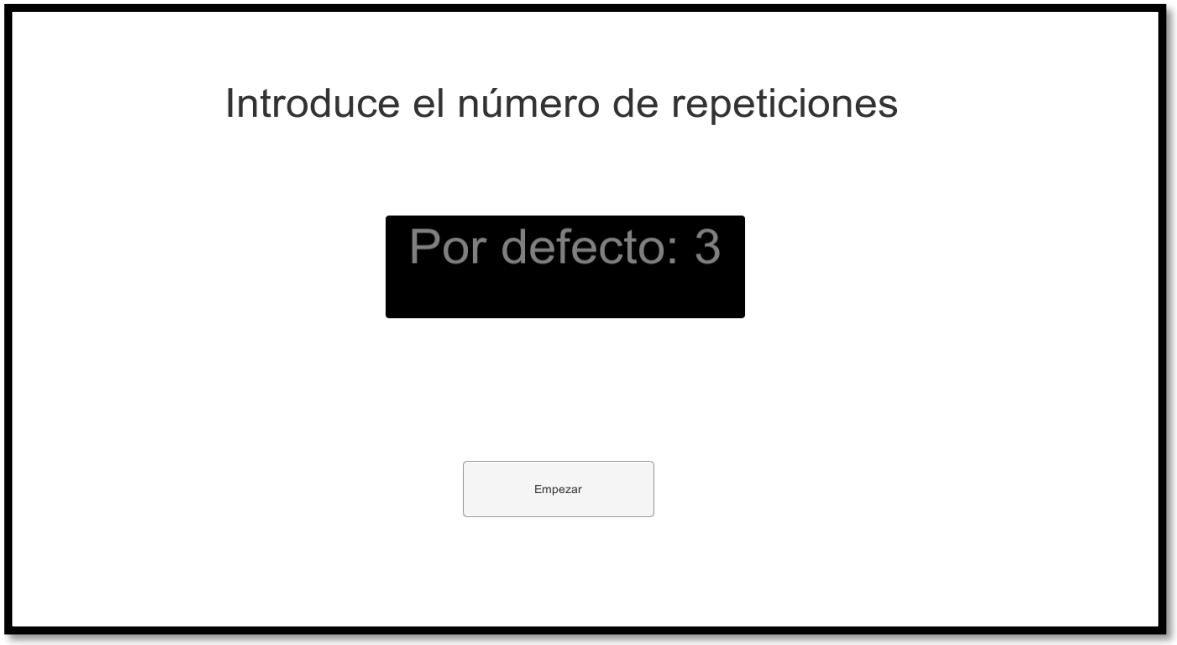

Ilustración 84: Pantalla número de repeticiones

En esta actividad hay que realizar el proceso de cepillado de dientes que está formado por cuatro pasos:

- Coger el cepillo de dientes mediante un movimiento vertical hacia abajo.
- Mojarlo con agua mediante un movimiento horizontal hacia la izquierda si es el brazo derecho o hacia la derecha si es el izquierdo, previamente se ha tenido que girar ligeramente el antebrazo hacía adentro.
- Llevar el cepillo hasta la boca.
- Cepillar los dientes el número de veces establecido.

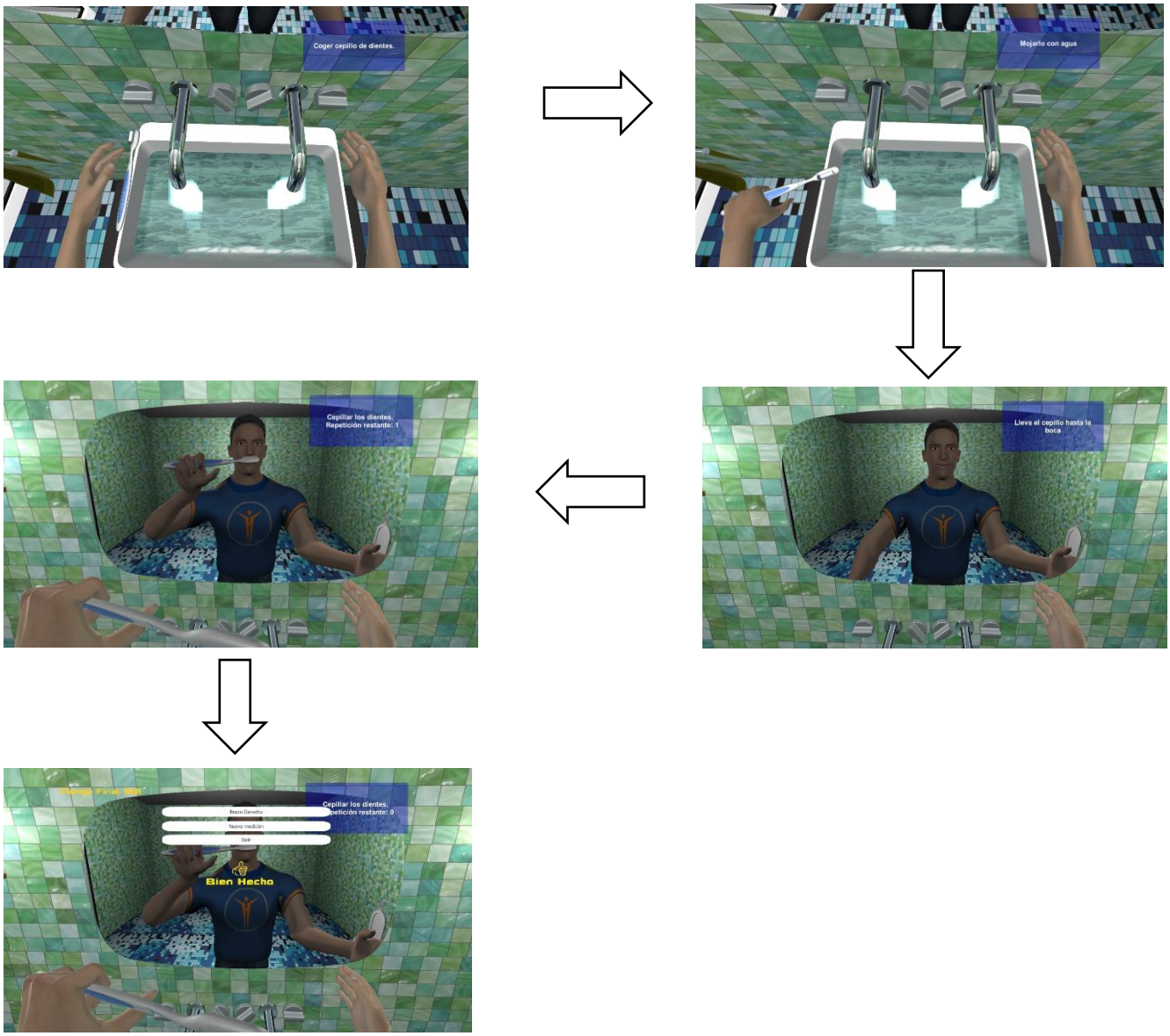

Ilustración 85: Pasos Actividad baño

2 Instalación

Para la ejecución de la aplicación del pc se dispone de un ejecutable (Iniciar.bat) que inicia el servidor y la aplicación de Unity. Para ello todo debe encontrarse en la misma carpeta.

En cuanto a la conexión del móvil con el servidor, se ofrecen una serie de scripts para Windows, con los cuales el ordenador crea un punto wifi personal al que se podrá conectar el Smartphone para la realización del ejercicio. Los scripts son tres:

- Crear el punto wifi (establecer un nombre y una contraseña). Solo haría falta arrancarlo una única vez, siempre que no se borren los ajustes de red de Windows o se quiera modificar el punto wifi.

```
@echo Presiona la tecla Enter para configurar la Red WIFI
pause
netsh wlan set hostednetwork mode=allow ssid=” Nombre de la red” key=” Contraseña”
pause
exit
```

- Arrancar punto wifi.
- Apagar punto wifi.

Estos scripts se han creado para tratar de mejorar y facilitar el paso de datos del móvil a Unity.

Por defecto la ip que se asigna al pc es la 192.168.137.1, aunque se puede modificar desde los ajustes de Windows.

Para poder hacer uso de dichos scripts primero hay que asegurarse de que se dispone de una la tarjeta de red wifi y que esta sea compatible, ya que no todas permiten la creación de un punto wifi. En este caso, los scripts no funcionarían, teniendo que introducir la ip que ha sido asignada a la tarjeta de red del pc.

Una forma de comprobar la ip que se ha asignado al pc es ejecutar el comando ipconfig en la terminal de Windows (CMD), obteniendo como salida las conexiones del pc a la red y pudiendo ver la ip que tiene asignada.