



Universidad de Valladolid

Escuela de Ingeniería Informática

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

(Mención Ingeniería del Software)

**Medición de los movimientos de las
articulaciones mediante teléfonos
inteligentes**

Autor:

Rafael Matamoros Luque

Tutor:

Miguel Ángel Laguna Serrano

Resumen

El rango de movimiento (ROM) de determinadas articulaciones en pacientes que han sufrido un accidente se puede ver gravemente afectado. En estos casos la rehabilitación de los mismos suele ser dirigida por medio de escalas de evaluación del ROM. Sin embargo, este tipo de escalas son, generalmente, caras y poco prácticas, ya que, requieren de demasiado tiempo para llevarse a cabo. En este documento se presenta el diseño y la implementación de una aplicación para teléfonos móviles inteligentes que utilicen el sistema operativo Android, que, mediante el uso de los sensores incorporados en los mismos, obtiene el ángulo de desplazamiento del dispositivo. El sistema ha sido testado, principalmente, en las articulaciones del hombro y el codo, proporcionando datos significativamente similares a los obtenidos mediante un dispositivo Kinect. Estos resultados muestran la capacidad del sistema para obtener resultados comparables a los que se pueden obtener con un goniómetro tradicional, pero reduciendo el tiempo necesario para llevar a cabo las mediciones. No sólo permite medir el ROM, sino que permite llevar un control de los pacientes, así como sus avances mediante las lecturas del goniómetro, al contar con una base de datos donde poder almacenar toda esta información. La portabilidad y la facilidad de uso del sistema pueden simplificar enormemente las tareas de evaluación del rango de movimiento de los pacientes.

Abstract

The range of motion (ROM) of some joints in patients who have suffered an accident could be severely affected. In these cases, rehabilitation is usually guided through assessment scales of ROM. However, these kinds of scales are, generally, expensive and not so practical, since they take much time to carry out. This document presents the design and implementation of a smartphone application for Android operative system, which, using the incorporated sensors of the smartphone, obtains the angle of shift. The system has been tested, mainly, in shoulder and elbow joints, providing significantly similar data to those obtained by Kinect. These results show the ability of the system to achieve comparably results to those obtained with a traditional goniometer, but decreasing the time necessary to carry out the measurements. It does not only allow to measure the ROM, but it allows to keep track of patients, as well as their progress by goniometer records, as it has at its disposal a database where to place all this information. The portability and ease-of-use of the system would simplify hugely the tasks of ROM assessment of patients.

Tabla de contenidos

1. Introducción.....	1
1.1. Contexto y motivación.....	1
1.2. Objetivos.....	1
1.3. Metodología empleada	2
1.4. Estructura de la memoria	2
1.5. Contenido del CD.....	3
2. Herramientas empleadas	4
2.1. Herramientas software.....	4
2.2. Características de los equipos	4
3. Ángulo de desplazamiento	6
3.1. Cuaterniones	6
3.2. Cálculo del ángulo de desplazamiento	6
3.2.1. Limitaciones.....	8
4. Plan de Desarrollo del proyecto.....	9
4.1. Introducción	9
4.1.1. Propósito	9
4.1.2. Alcance	9
4.2. Visión general del proyecto.....	9
4.2.1. Objetivos.....	9
4.2.2. Suposiciones y restricciones	10
4.2.3. Definiciones y acrónimos.....	11
4.2.4. Artefactos del proyecto	13
4.3. Organización del proyecto.....	13
4.3.1. Interfaces externas	13
4.3.2. Estructura interna.....	14
4.3.3. Roles y responsabilidades.....	14
4.3.4. Eventos para el desarrollo del producto	14
4.4. Planificación del proyecto	15
4.4.1. Estimación temporal del proyecto	15
4.4.2. Estimación de costes	19
4.4.3. Métricas.....	19

4.4.4. Plan de seguimiento	19
4.5. Plan de Gestión de Riesgos.....	19
5. Análisis	25
5.1. Requisitos	25
5.1.1. Ingeniería tradicional vs Desarrollo ágil (Scrum)	25
5.1.2. Requisitos funcionales	26
5.1.3. Requisitos no funcionales	27
5.2. Modelo de Casos de Uso	27
5.2.1. Identificación de actores del sistema	27
5.2.2. Diagrama de casos de uso	28
5.2.3. Especificación de casos de uso	28
5.3. Realización de los Casos de Uso en Análisis	34
5.3.1. Diagramas de actividad.....	34
5.3.2. Modelo de dominio	41
5.3.3. Descripción de las clases del modelo de dominio	42
6. Diseño	45
6.1. Diseño de la arquitectura	45
6.1.1. Patrón arquitectónico utilizado: MVP (Modelo Vista Presentador).....	45
6.1.2. Patrones de diseño utilizados.....	48
6.1.3. Arquitectura general del proyecto	52
6.1.4. Descomposición modular del proyecto.....	53
6.1.5. Diseño detallado de los módulos del proyecto	53
6.2. Modelado de Datos	63
6.2.1. Diseño conceptual de la Base de Datos.....	63
6.2.2. Esquema relacional de la Base de Datos	63
6.2.3. Comentarios con respecto a la Base de Datos	64
6.3. Realización en Diseño de los Casos de Uso	66
6.3.1. UC-0001: Añadir_Medicion. Parte 1.....	66
6.3.2. UC-0001: Añadir_Medicion. Parte 2.....	67
6.3.3. UC-0001: Añadir_Medicion. Parte 3	68
6.3.4. UC-0001: Añadir_Medicion. Parte 4.....	69
6.3.5. Detalle de GetPacienteBy	70
6.3.6. Detalle de StoreMeasurement	70
6.3.7. UC-0002: Visualizar_10_Ultimas_Mediciones.....	71
6.3.8. Detalle de GetUltimasMediciones	72

6.3.9. UC-0003: Borrar_Medicion.....	72
6.3.10. Detalle de OnDeleteMeasurement	73
6.3.11. UC-0004: Visualizar_Perfiles_Pacientes	74
6.3.12. Detalle de GetPacientes	75
6.3.13. UC-0005: Filtrar_Pacientes.....	75
6.3.14. Detalle de GetPacientesFiltrado.....	76
6.3.15. UC-0006: Crear_Perfil_Paciente – Parte 1	77
6.3.16. UC-0006: Crear_Perfil_Paciente – Parte 2	78
6.3.17. Detalle de StorePatient	78
6.3.18. UC-0007: Visualizar_Informacion_Paciente – Parte 1.....	79
6.3.19. UC-0007: Visualizar_Informacion_Paciente – Parte 2.....	80
6.3.20. Detalle de GetMedicionesPacienteBy	81
6.3.21. UC-0008: Borrar_Perfil_Paciente	82
6.3.22. Detalle de OnDeletePatient.....	82
7. Implementación y pruebas	83
7.1. Implementación.....	83
7.1.1. Uso de bibliotecas externas.....	83
7.1.2. Fases de implementación	84
7.2. Pruebas	85
7.2.1. Introducción	85
7.2.2. Niveles de prueba de software.....	86
7.2.3. Pruebas realizadas	87
8. Estudio comparativo Kinect - App.....	102
8.1. Introducción	102
8.2. Mediciones realizadas	102
8.3. Conclusiones.....	104
9. Conclusiones	105
9.1. Objetivos alcanzados	105
9.2. Líneas de trabajo futuras.....	106
9.3. Valoración personal.....	106
Bibliografía	107

Anexo I - Manual de usuario	110
I.1. Introducción.....	110
I.2. Principales funcionalidades	110
I.2.1. Añadir una medición.....	110
I.2.2. Visualizar 10 últimas mediciones almacenadas.....	115
I.2.3. Borrar una medición	116
I.2.4. Visualizar los perfiles de pacientes.....	118
I.2.5. Filtrar el listado de pacientes.....	119
I.2.6. Crear perfil de un paciente	120
I.2.7. Visualizar la información de un paciente.....	121
I.2.8. Borrar el perfil de un paciente.....	122
 Anexo II - Plan de seguimiento	 124
II.1. Introducción.....	124
II.1.1. Propósito.....	126
II.1.2. Alcance.....	126
II.1.3. Códigos de colores.....	126
II.1.4. Estimación del esfuerzo	127
II.2. Pasos previos antes del primer sprint – 4 de abril a 19 de abril.....	128
II.2.1. Comprensión inicial del problema	128
II.2.2. Elección de la metodología de desarrollo	128
II.2.3. Primeros artefactos de Scrum	129
II.3. Sprint 1 – 20 de abril a 4 de mayo	130
II.3.1. Product backlog	130
II.3.2. Descripción del sprint	131
II.3.3. Sprint backlog	131
II.3.4. Comentarios	132
II.3.5. Revisión de Sprint	132
II.4. Sprint 2 – 5 de mayo a 19 de mayo	133
II.4.1. Product backlog	133
II.4.2. Descripción del sprint	134
II.4.3. Sprint backlog	134
II.4.4. Comentarios	135
II.4.5. Revisión de Sprint	135
II.5. Sprint 3 – 20 de mayo a 3 de junio	136
II.5.1. Product backlog	136
II.5.2. Descripción del sprint	137

II.5.3. Sprint backlog	137
II.5.4. Comentarios	138
II.5.5. Revisión de Sprint	138
II.6. Sprint 4 – 5 de junio a 19 de junio	139
II.6.1. Product backlog	139
II.6.2. Descripción del sprint	140
II.6.3. Sprint backlog	140
II.6.4. Comentarios	141
II.6.5. Revisión de Sprint	141
II.7. Sprint 5 – 20 de junio a 4 de julio	142
II.7.1. Product backlog	142
II.7.2. Descripción del sprint	143
II.7.3. Sprint backlog	143
II.7.4. Comentarios	144
II.7.5. Revisión de Sprint	145

Lista de figuras

Capítulo 3. Ángulo de desplazamiento

Figura 1 - Representación de un cuaternión	6
Figura 2 - Sistema de coordenadas empujado por el vector de rotación	7

Capítulo 4. Plan de Desarrollo del proyecto

Figura 3 - Goniómetro tradicional	11
Figura 4 - Planos que dividen el cuerpo humano	12
Figura 5 - Movimiento de flexión y extensión de cadera	12
Figura 6 - Movimiento de abducción y aducción de cadera	13

Capítulo 5. Análisis

Figura 7 - Requisitos ingeniería de software tradicional vs métodos ágiles	25
Figura 8 - Diagrama de casos de uso	28
Figura 9 - Diagrama de actividad UC-0001	34
Figura 10 - Diagrama de actividad UC-0002	35
Figura 11 - Diagrama de actividad UC-0003	36
Figura 12 - Diagrama de actividad UC-0004	37
Figura 13 - Diagrama de actividad UC-0005	37
Figura 14 - Diagrama de actividad UC-0006	38
Figura 15 - Diagrama de actividad UC-0007	39
Figura 16 - Diagrama de actividad UC-0008	40
Figura 17 - Modelo de dominio	41

Capítulo 6. Diseño

Figura 18 - Patrón MVC (Modelo-Vista-Controlador)	45
Figura 19 - Obtención de datos por parte de la Vista (Activity/Fragment)	46
Figura 20 - Patrón MVP (Modelo-Vista-Presentador)	46
Figura 21 - Diferencias entre MVC y MVP	47
Figura 22 - Estructura patrón Builder	48
Figura 23 - Estructura patrón Singleton	49
Figura 24 - Estructura patrón Adaptador	51
Figura 25 - Estructura elementos RecyclerView - Android	51
Figura 26 - Arquitectura general del proyecto	52
Figura 27 - Descomposición modular del proyecto	53
Figura 28 - Módulo View – Parte 1	54
Figura 29 - Módulo View - Parte 2	55
Figura 30 - Módulo Presenter	56
Figura 31 - Módulo Presenter - Parte 2	57
Figura 32 - Módulo Model	58
Figura 33 - Módulo Interfaces - ViewFunctions	58
Figura 34 - Módulo Interfaces - PresenterFunctions - Parte 1	59
Figura 35 - Módulo Interfaces – PresenterFunctions – Parte 2	59

Figura 36 - Módulo Interfaces - ModelFunctions.....	59
Figura 37 - Módulo DependencyInjection.....	60
Figura 38 - Módulo Data - Parte 1	61
Figura 39 - Módulo Data - Parte 2	62
Figura 40 - Diseño conceptual de la Base de Datos	63
Figura 41 - Esquema relacional de la Base de Datos.....	63
Figura 42 - Diagrama de secuencia UC-0001 - Parte 1	66
Figura 43 - Diagrama de secuencia UC-0001 - Parte 2	67
Figura 44 - Diagrama de secuencia UC-0001 - Parte 3	68
Figura 45 - Diagrama de secuencia UC-0001 - Parte 4	69
Figura 46 - Diagrama de secuencia GetPacienteBy.....	70
Figura 47 - Diagrama de secuencia StoreMeasurement	70
Figura 48 - Diagrama de secuencia UC-0002.....	71
Figura 49 - Diagrama de secuencia GetUltimasMediciones.....	72
Figura 50 - Diagrama de secuencia UC-0003.....	72
Figura 51 - Diagrama de secuencia OnDeleteMeasurement	73
Figura 52 - Diagrama de secuencia UC-0004.....	74
Figura 53 - Diagrama de secuencia GetPacientes	75
Figura 54 - Diagrama de secuencia UC-0005.....	75
Figura 55 - Diagrama de secuencia GetPacientesFiltrado.....	76
Figura 56 - Diagrama de secuencia UC-0006 – Parte 1	77
Figura 57 - Diagrama de secuencia UC-0006 – Parte 2	78
Figura 58 - Diagrama de secuencia StorePatient	78
Figura 59 - Diagrama de secuencia UC-0007 – Parte 1	79
Figura 60 - Diagrama de secuencia UC-0007 – Parte 2	80
Figura 61 - Diagrama de secuencia GetMedicionesPacienteBy	81
Figura 62 - Diagrama de secuencia UC-0008.....	82
Figura 63 - Diagrama de secuencia OnDeletePatient.....	82

Capítulo 7. Implementación y pruebas

Figura 64 - Ejemplo de uso biblioteca EnumFormat	84
---	----

Anexo I - Manual de usuario

Figura 65 - Instrucciones para iniciar una medición.....	110
Figura 66 - Instrucciones para iniciar una medición desde el menú lateral.....	110
Figura 67 - Instrucciones para realizar la medición.....	111
Figura 68 - Instrucciones para reiniciar la medición	111
Figura 69 - Instrucciones para almacenar la medición.....	111
Figura 70 - Instrucciones para cancelar el almacenamiento de la medición	112
Figura 71 - Instrucciones para seleccionar paciente para almacenar la medición	112
Figura 72 - Instrucciones para rellenar la información sobre la nueva medición	114
Figura 73 - Instrucciones para ir a Últimas 10 mediciones	115
Figura 74 - Instrucciones para ir a Últimas 10 mediciones a través del menú lateral	115
Figura 75 - Visualizar 10 últimas mediciones almacenadas	115
Figura 76 - Instrucciones para borrar la medición seleccionada.....	116
Figura 77 - Instrucciones para confirmar el borrado de la medición	116
Figura 78 - Instrucciones para cancelar el borrado de la medición	117
Figura 79 - Instrucciones para ir a Perfiles de pacientes.....	118

Figura 80 - Instrucciones para ir a Perfiles de pacientes a través del menú lateral.....	118
Figura 81 – Visualizar los perfiles de pacientes.....	118
Figura 82 - Instrucciones para seleccionar un criterio de filtro de pacientes	119
Figura 83 - Instrucciones para escribir información en el cuadro de búsqueda	119
Figura 84 - Instrucciones para ir a la pantalla con los campos a rellenar sobre el nuevo paciente.....	120
Figura 85 - Instrucciones para rellenar la información sobre el nuevo paciente.....	120
Figura 86 - Instrucciones para almacenar el perfil del paciente	121
Figura 87 - Instrucciones para visualizar la información de un paciente	121
Figura 88 - Instrucciones para borrar el perfil del paciente seleccionado	122
Figura 89 - Instrucciones para confirmar el borrado del perfil del paciente.....	122
Figura 90 - Instrucciones para cancelar el borrado del perfil del paciente.....	123

Anexo II - Plan de seguimiento

Figura 91 - Proceso de desarrollo - Scrum.....	124
Figura 92 - Incremento iterativo vs incremento continuo - Scrum	125
Figura 93 - Ciclo iterativo - Scrum	125

Lista de tablas

Capítulo 3. Ángulo de desplazamiento

Tabla 1 - Cálculo del ángulo de desplazamiento entre dos cuaterniones.....	8
Tabla 2 - Cálculo del producto escalar entre dos cuaterniones.....	8

Capítulo 4. Plan de Desarrollo del proyecto

Tabla 3 - Eventos en Scrum	15
Tabla 4 - Product roadmap del proyecto	17
Tabla 5 - División del proyecto por sprints.....	17
Tabla 6 - Plan de release del proyecto	19
Tabla 7 - Métrica - Velocidad del sprint	19
Tabla 8 - Definición de la exposición al riesgo	20
Tabla 9 - Matriz Impacto/Probabilidad	20

Capítulo 5. Análisis

Tabla 10 - Requisitos funcionales.....	27
Tabla 11 - Requisitos no funcionales.....	27
Tabla 12 - Requisitos de información.....	27
Tabla 13 - Especificación UC-0001	29
Tabla 14 - Especificación UC-0002	30
Tabla 15 - Especificación UC-0003	30
Tabla 16 - Especificación UC-0004	31
Tabla 17 - Especificación UC-0005	31
Tabla 18 - Especificación UC-0006	32
Tabla 19 - Especificación UC-0007	32
Tabla 20 - Especificación UC-0008	33

Capítulo 7. Implementación y pruebas

Tabla 21 - Prueba-0001	88
Tabla 22 - Prueba-0002	88
Tabla 23 - Prueba-0003	88
Tabla 24 - Prueba-0004	89
Tabla 25 - Prueba-0005	89
Tabla 26 - Prueba-0006	89
Tabla 27 - Prueba-0007	90
Tabla 28 - Prueba-0008	90
Tabla 29 - Prueba-0009	90
Tabla 30 - Prueba-0010	91
Tabla 31 - Prueba-0011	91
Tabla 32 - Prueba-0012	91
Tabla 33 - Prueba-0013	92
Tabla 34 - Prueba-0014	92
Tabla 35 - Prueba-0015	92

Tabla 36 - Prueba-0016	93
Tabla 37 - Prueba-0017	93
Tabla 38 - Prueba-0018	93
Tabla 39 - Prueba-0019	94
Tabla 40 - Prueba-0020	94
Tabla 41 - Prueba-0021	94
Tabla 42 - Prueba-0022	95
Tabla 43 - Prueba-0023	95
Tabla 44 - Prueba-0024	95
Tabla 45 - Prueba-0025	96
Tabla 46 - Prueba-0026	96
Tabla 47 - Prueba-0027	96
Tabla 48 - Prueba-0028	97
Tabla 49 - Prueba-0029	97
Tabla 50 - Prueba-0030	97
Tabla 51 - Prueba-0031	98
Tabla 52 - Prueba-0032	98
Tabla 53 - Prueba-0033	98
Tabla 54 - Prueba-0034	99
Tabla 55 - Prueba-0035	99
Tabla 56 - Prueba-0036	100
Tabla 57 - Prueba-0037	100
Tabla 58 - Prueba-0038	100

Capítulo 8. Estudio comparativo Kinect - App

Tabla 59 – Comparación Kinect y App - Abducción hombro.....	103
Tabla 60 - Comparación Kinect y App - Flexión hombro	103
Tabla 61 - Comparación Kinect y App - Flexión codo	103
Tabla 62 - Comparación Kinect y App - Rotación externa/interna hombro.....	103
Tabla 63 - Comparación Kinect y App - Flexión horizontal hombro.....	104

Anexo II - Plan de seguimiento

Tabla 64 - Product backlog - Sprint 1	131
Tabla 65 - <i>Sprint backlog - Sprint 1</i>	131
Tabla 66 - Product backlog - Sprint 2	133
Tabla 67 - Sprint backlog - Sprint 2.....	135
Tabla 68 - Product backlog - Sprint 3	137
Tabla 69 - Sprint backlog - Sprint 3.....	138
Tabla 70 - Product backlog - Sprint 4	140
Tabla 71 - Sprint backlog - Sprint 4.....	141
Tabla 72 - Product backlog - Sprint 5	143
Tabla 73 - Sprint backlog - Sprint 5.....	144

Capítulo 1

Introducción

1.1. Contexto y motivación

Hasta ahora, la utilización de goniómetros tradicionales traía consigo un problema, y es que, si se quería tener un registro para poder ir actualizando el estado de un paciente a lo largo del tiempo y ver si, efectivamente, se estaban produciendo mejoras en las condiciones físicas del mismo, se necesitaban, por lo menos, dos elementos: el goniómetro, y algún tipo de sistema (bien en papel o informático) para poder ir anotando el avance de cada uno de los pacientes.

Sin embargo, con la creciente expansión en prácticamente todos los campos de la vida cotidiana de las personas de los teléfonos inteligentes o smartphones, esto ya no es necesario. Los smartphones, con la capacidad de procesamiento y almacenamiento actuales, son unas herramientas muy potentes que pueden ser explotadas de múltiples formas.

En este contexto, surge la motivación de realizar este Trabajo de Fin de Grado, con el fin de poder incorporar en un mismo dispositivo, un goniómetro que pueda realizar lecturas fiables del movimiento de una articulación, así como llevar un registro de todos los pacientes. De esta manera, se puede tener toda la información centralizada en un mismo dispositivo, evitando, además, la necesidad de utilización de sistemas más grandes o más caros para llevar a cabo este fin.

La solución que se ha propuesto en este Trabajo de Fin de Grado es la realización de una aplicación móvil para el sistema operativo Android que sea capaz de realizar estas tareas que se acaban de nombrar. No sólo se ha buscado que esta aplicación cumpla con los requisitos propuestos para la realización de este TFG, sino que, se ha pretendido que sea útil de verdad para cualquier profesional de la salud que desee utilizarla.

En este documento, se especifica todo el proceso de desarrollo de *TFG-Goniometer* (nombre que se le ha dado a esta aplicación), realizado para la asignatura Trabajo de Fin de Grado, del Grado en Ingeniería Informática por la Universidad de Valladolid.

1.2. Objetivos

Este proyecto tiene dos objetivos principales, el primero de ellos es elaborar una aplicación móvil en el sistema Android, que permita tanto almacenar como recuperar mediciones realizadas mediante el uso del dispositivo, así como perfiles de pacientes introducidos por el Usuario.

En un principio, en cuanto al goniómetro, se intentará que éste sea capaz de medir cualquier tipo de movimiento articular realizado por un paciente; sin embargo, dadas las restricciones presentadas en el capítulo 4 (apartado 4.2.2), es posible que no se logre implementar algo así. En tal caso, se intentará que el goniómetro sea capaz de medir el mayor número de movimientos articulares posibles.

El segundo objetivo principal es realizar un estudio comparativo de la precisión de la aplicación con el controlador de movimiento desarrollado por *Microsoft* para su consola *Xbox 360 (Kinect)*, dispositivo con el cual, se había desarrollado ya previamente un software de reconocimiento del movimiento de las articulaciones.

1.3. Metodología empleada

Para el desarrollo de este proyecto se ha decidido utilizar la metodología de desarrollo ágil Scrum.

El principal motivo de elección de esta metodología es que se caracteriza por un desarrollo iterativo e incremental con una entrega temprana y continuada de software funcional. Esto lo que permite es, disponer de piezas de software de la aplicación final desde fases tempranas del desarrollo de este proyecto, evitando así, demorar el comienzo del proceso de implementación en exceso (de haber seguido un desarrollo tradicional en cascada).

A lo largo del capítulo 3 (Plan de Desarrollo del proyecto) se explican más en detalle las características de la metodología Scrum.

1.4. Estructura de la memoria

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

- **Capítulo 1: Introducción.** Este capítulo incluye el contexto, la motivación y los objetivos de este proyecto. Además, incluye la metodología empleada para el desarrollo del mismo, 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.** Este capítulo incluye una breve descripción del software empleado para desarrollar la aplicación y las características de los equipos (tanto en el que se ha desarrollado como en los que se han hecho las pruebas).
- **Capítulo 3: Ángulo de desplazamiento.** Este capítulo incluye los conceptos matemáticos empleados a la hora de obtener el ángulo de desplazamiento del dispositivo (goniómetro), así como las limitaciones que tiene el método escogido para este cálculo.
- **Capítulo 4: Plan de Desarrollo del proyecto.** Este capítulo incluye toda la información relativa a cómo se ha desarrollado este proyecto, además de todo lo referente a los distintos artefactos y eventos de Scrum. Por último, incluye el Plan de Gestión de Riesgos.
- **Capítulo 5: Análisis.** Este capítulo incluye toda la información relativa a la fase de análisis que se ha llevado a cabo para desarrollar *TFG-Goniometer*. Incluye la elicitación de requisitos, modelo de los casos de uso, realización de los mismos en análisis (diagramas de actividad). Por último, incluye el modelo de dominio junto con una descripción de cada una de las clases que lo forman.

- **Capítulo 6: Diseño.** Este capítulo incluye toda la información relativa a la fase de diseño que se ha llevado a cabo para desarrollar *TFG-Goniometer*. Incluye el diseño de la arquitectura (patrón arquitectónico y patrones de diseño empleados), arquitectura general del proyecto, descomposición modular del mismo, relaciones entre los distintos módulos, modulado de datos (todo lo referente a la Base de Datos) y realización de los casos de uso en diseño (diagramas de secuencia).
- **Capítulo 7: Implementación y pruebas.** Este capítulo incluye todas aquellas cuestiones relativas a la implementación y pruebas que se han realizado tras el desarrollo de *TFG-Goniometer*. Éstas incluyen las bibliotecas externas que se han empleado, fases de implementación, tipos de pruebas, así como niveles de software sobre los que se pueden realizar y, por último, las pruebas realizadas con el resultado obtenido para cada una de ellas (y correcciones aplicadas en caso de obtener resultados incorrectos).
- **Capítulo 8: Estudio comparativo Kinect – App.** Este capítulo incluye todos los datos obtenidos de las mediciones realizadas tanto con Kinect como con la aplicación, los resultados obtenidos a partir de esos datos y, finalmente, las conclusiones del estudio comparativo.
- **Capítulo 9: Conclusiones.** Este capítulo incluye los objetivos alcanzados, las líneas de trabajo futuras tras la realización de este Trabajo de Fin de Grado y la valoración personal de todo el trabajo desarrollado.
- **Bibliografía.** En este apartado se incluyen todas las referencias que se han consultado para la realización de este proyecto.
- **Anexos.** En este apartado se incluyen el manual de usuario de la aplicación y el plan de seguimiento del desarrollo del proyecto (con motivo de la utilización de Scrum).

1.5. Contenido del CD

El contenido del CD proporcionado junto con esta memoria es:

- **TFG-Goniometer.apk:** Se trata del ejecutable de la aplicación.
- **codigo:** Contiene todo el código fuente de la aplicación, así como el JavaDoc (en formato HTML) generado por Android Studio.
- **diagramas:** Contiene todos los ficheros con los diagramas UML incluidos en esta memoria.
 - **analisis.asta:** Contiene todos los diagramas referentes a la fase de análisis de la aplicación.
 - **diseño.asta:** Contiene todos los diagramas referentes a la fase de diseño de la aplicación.
 - **bd.asta:** Contiene todos los diagramas referentes a la Base de Datos.
- **memoria.pdf:** Memoria del Trabajo de Fin de Grado.

Capítulo 2

Herramientas empleadas

2.1. Herramientas software

Las herramientas software que se han empleado a lo largo de todo el desarrollo de *TFG-Goniometer* han sido:

- **Android Studio.** Este ha sido el IDE empleado para el desarrollo de toda la parte de implementación en Android.
- **Microsoft Office Word 2016.** Esta ha sido la herramienta empleada para la escritura de toda la memoria del Trabajo de Fin de Grado.
- **Astah Professional.** Esta herramienta ha sido empleada para la realización de todos los diagramas presentes en este documento en relación con las fases de análisis y diseño de la aplicación (casos de uso, diagramas de actividad, modelo de dominio, diagramas de secuencia...).
- **Dropbox.** Esta herramienta ha sido empleada para almacenar las copias de seguridad de toda la documentación del Trabajo de Fin de Grado.
- **GitHub.** Esta herramienta ha sido empleada para alojar todo el código de la aplicación, así como gestionar el control de versiones de la misma.
- **Adobe Photoshop.** Esta herramienta ha sido empleada para desarrollar el manual de usuario, ya que, no se ha podido emplear el emulador de Android Studio debido a un mal funcionamiento del mismo.

2.2. Características de los equipos

El equipo en el que se ha desarrollado todo este proyecto es un portátil adquirido en el año 2012 con las siguientes características:

- **Marca:** Lenovo
- **Modelo:** IdeaPad Z580
- **Procesador:** Intel Core i7-3612QM @2.10 GHz (4 cores)
- **Memoria:** 8 Gb DDR3 SDRAM
- **Disco duro:** 750 Gb 5400 rpm
- **Tarjeta gráfica:** NVIDIA GeForce GT 635M (1Gb DDR3)
- **Tamaño pantalla:** 15.6" (1366x768) HD

- **OS 1:** Windows 7 Home Premium 64-bit
- **OS 2:** Ubuntu 16.04

A la hora de hacer las pruebas pertinentes durante el desarrollo de *TFG-Goniometer* se han utilizado varios dispositivos de distintos tamaños y APIs para verificar el correcto funcionamiento de la aplicación.

El principal dispositivo en el que se han realizado estas pruebas ha sido:

- **Marca:** Huawei
- **Modelo:** P8
- **Procesador:** Hisilicon Kirin 930, 4 núcleos a 2 GHz y 4 núcleos a 1.5 GHz
- **Memoria:** 3 Gb
- **Almacenamiento:** 16 Gb memoria interna + 16 Gb tarjeta SD
- **Tamaño pantalla:** 5.2" IPS LCD
- **Resolución:** 1080 x 1920
- **OS:** Android 5.0.2 Lollipop (API 21) con EMUI 3.1

Además de este dispositivo, se han empleado otros 3 dispositivos para realizar las pruebas, a saber:

- **Samsung Galaxy S II plus:** Dispositivo con un tamaño de pantalla de 4.3" que ejecuta Android 4.2.2 Jelly Bean (API 17)
- **LG Optimus L4 II E440:** Dispositivo con un tamaño de pantalla de 3.8" que ejecuta Android 4.1.2 Jelly Bean (API 16)
- **Samsung Galaxy Young GT-S6310:** Dispositivo con un tamaño de pantalla de 3.27" que ejecuta Android 4.1.2 Jelly Bean (API 16)

Capítulo 3

Ángulo de desplazamiento

3.1. Cuaterniones

Un cuaternión es un número complejo empleado para representar orientaciones y rotaciones de objetos en 3 dimensiones. En comparación con los ángulos de Euler, son más simples de componer y evitan el problema del *gimbal lock*, y, en comparación con las matrices de rotación, son más compactos y (pueden ser) más eficientes.

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 $\langle w, x, y, z \rangle$ donde w es la rotación alrededor del eje $\langle x, y, z \rangle$. En la siguiente imagen se muestra la representación de un cuaternión:

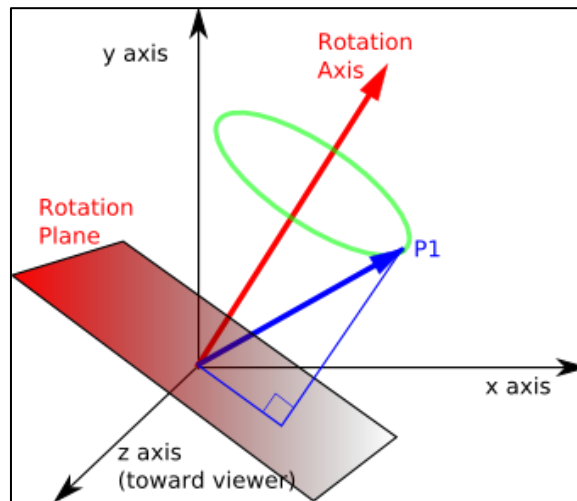


Figura 1 - Representación de un cuaternión

3.2. Cálculo del ángulo de desplazamiento

Para realizar esta tarea se ha decidido emplear un único sensor software, que es el **vector de rotación**. Este sensor utiliza los sensores hardware: acelerómetro, magnetómetro y giroscopio, para representar la orientación del dispositivo como una combinación de un ángulo (θ) y un eje ($\langle x, y, z \rangle$).

Los tres elementos del vector de rotación son: $\langle x \cdot \sin(\theta/2), y \cdot \sin(\theta/2), z \cdot \sin(\theta/2) \rangle$, tal que, la magnitud del mismo es igual a $\sin(\theta/2)$, y la dirección es igual a la dirección del eje de rotación.

Estos tres elementos son iguales a las tres últimas componentes de un **cuaternión unitario** ($\langle \cos(\theta/2), x*\sin(\theta/2), y*\sin(\theta/2), z*\sin(\theta/2) \rangle$).

Este último concepto es importante porque, precisamente para calcular el ángulo de desplazamiento, la información proporcionada por el vector de rotación es convertida a un cuaternión unitario, y es con éstos con los que se trabaja para obtener el valor final del desplazamiento del dispositivo.

Los pasos para la obtención del ángulo de desplazamiento son los siguientes:

En primera instancia, cuando el Usuario pulsa por primera vez en el goniómetro, éste convierte la lectura obtenida por el vector de rotación en ese instante a un cuaternión unitario.

Una vez obtenida la posición inicial, a cada lectura obtenida por el vector de rotación, se convierte, de nuevo, al cuaternión unitario asociado, y se calcula el ángulo que existe entre esos dos cuaterniones mediante una fórmula que se va a explicar a continuación. Este cálculo proporciona el ángulo de desplazamiento entre la posición de origen escogida por el Usuario y la posición del dispositivo a cada instante.

A continuación, se explica esto un poco más en detalle.

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

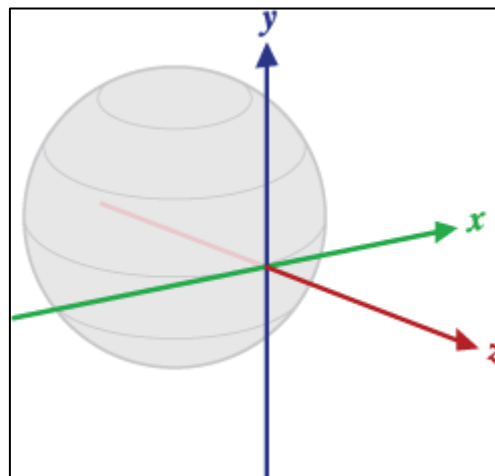


Figura 2 - Sistema de coordenadas empleado por el vector de rotación

Este sistema de referencia se define como una base ortonormal, donde:

- Y es tangencial al suelo en la localización actual del dispositivo y apunta hacia el norte magnético.
- Z apunta hacia el cielo y es perpendicular al suelo.
- X se define como el producto vectorial $Y \times Z$ (es tangencial al suelo en la localización actual del dispositivo y apunta aproximadamente al Este).

Como se puede observar, al emplear un marco de referencia “fijo” (la posición del polo norte magnético varía a diario) a la hora de obtener las lecturas y su posterior conversión a cuaternión, permite que, con la obtención del ángulo entre los cuaterniones inicial y final, se obtenga directamente el ángulo de desplazamiento del dispositivo, de una forma muy sencilla.

Teniendo ya estas posiciones inicial y final del desplazamiento, se emplea la siguiente fórmula para calcular el ángulo entre los dos cuaterniones:

$$\theta = \cos^{-1}(2\langle q_1, q_2 \rangle^2 - 1)$$

Tabla 1 - Cálculo del ángulo de desplazamiento entre dos cuaterniones

Donde $\langle q_1, q_2 \rangle$ denota el producto escalar de los cuaterniones correspondientes:

$$\langle a_1 + b_1\mathbf{i} + c_1\mathbf{j} + d_1\mathbf{k}, a_2 + b_2\mathbf{i} + c_2\mathbf{j} + d_2\mathbf{k} \rangle = a_1a_2 + b_1b_2 + c_1c_2 + d_1d_2$$

Tabla 2 - Cálculo del producto escalar entre dos cuaterniones

Con esta fórmula se obtiene el ángulo entre los cuaterniones inicial y final, o, lo que es lo mismo, se obtiene el ángulo de desplazamiento entre la posición inicial fijada por el Usuario y la del dispositivo en cada momento en referencia a esa posición.

3.2.1. Limitaciones

Este método escogido para el cálculo del ángulo de desplazamiento tiene algunas limitaciones.

La primera de ellas es que el máximo valor que puede tomar es 180, ya que, ese es el máximo ángulo que pueden formar dos cuaterniones (medido en términos absolutos).

La siguiente limitación es que el goniómetro, al utilizar el ángulo entre los dos cuaterniones (representaciones que, de por sí, hacen referencia a las 3 dimensiones), obtiene el ángulo de desplazamiento teniendo en cuenta los 3 ejes tridimensionales (X, Y, Z). Es decir, si, por ejemplo, se quiere realizar un movimiento de aducción de hombro (en el apartado 4.2.3 se da la definición de este tipo de movimiento) y, de forma involuntaria, durante el movimiento se produce una pequeña rotación del dispositivo, esta rotación es contabilizada en el cálculo del ángulo de desplazamiento, afectando a la precisión de la medición (generalmente dando lecturas por encima del valor real de desplazamiento).

Capítulo 4

Plan de Desarrollo del proyecto

4.1. Introducción

En esta parte de la memoria se va a proceder a detallar el Plan de Desarrollo de la aplicación Android, *TFG-Goniometer*.

Como ya se ha comentado previamente, para el desarrollo de esta aplicación se ha utilizado la metodología ágil *Scrum*. A continuación, se detalla toda la información, entre otras cosas, relativas al empleo de este tipo de metodología; así mismo, se incluye en uno de los apéndices de esta memoria, todo el plan de seguimiento que se ha seguido durante el desarrollo de esta aplicación.

4.1.1. Propósito

El propósito de este Plan de Desarrollo es proporcionar toda la información relativa al proceso que se ha seguido a la hora de realizar este proyecto, y que no engloba únicamente la aplicación *TFG-Goniometer*; también incluye un posterior estudio comparativo sobre la precisión de la misma frente a un dispositivo previamente validado por un experto, como es *Kinect*.

4.1.2. Alcance

En esta memoria se incluye una visión general del proyecto, se identifican los participantes en el mismo y sus roles, se detalla un plan de seguimiento del desarrollo del mismo, dividido por iteraciones (sprints), así como todos los artefactos o entregables que se emplean en la gestión ágil (en concreto, *Scrum*), entre otra información relevante.

4.2. Visión general del proyecto

4.2.1. Objetivos

El objetivo de este proyecto es, en primera instancia, desarrollar una aplicación para dispositivos Android (*TFG-Goniometer*), que pueda ser utilizada exactamente igual que un goniómetro tradicional, para medir el movimiento de una articulación en pacientes que tengan ciertas restricciones motoras.

Aunque la principal función de la aplicación sea el poder ser utilizada como un goniómetro normal y corriente, *TFG-Goniometer* posee más funcionalidades, a saber:

- **Almacenamiento y recuperación de las mediciones realizadas:** El usuario de la aplicación podrá almacenar (y, posteriormente, recuperar) todas y cada una de las mediciones que realice. Para poder hacer esto, el usuario deberá indicar a qué paciente corresponde esa medición, así como, también, indicar el movimiento que se ha realizado (flexión, extensión, aducción, abducción, etc.) y a qué articulación se refiere esa medición.
- **Registro de pacientes:** El usuario de la aplicación podrá crear nuevos perfiles (así como recuperar su información) para cada uno de los pacientes. Estos perfiles incluirán información personal del paciente (nombre, edad, sexo, etc.), así como un diagnóstico de su patología o enfermedad y el listado de cada una de las mediciones que se han realizado del paciente.
- **Listado de las 10 últimas mediciones guardadas:** El usuario de la aplicación podrá acceder al listado de las 10 últimas mediciones que haya guardado, así como el paciente al que corresponden.
- **Borrado de una de las 10 últimas mediciones guardadas:** El usuario podrá borrar del sistema una medición, siempre que sea una de las 10 últimas que haya realizado y almacenado.
- **Borrado del perfil de un paciente:** El usuario podrá borrar del sistema el perfil de uno de los pacientes almacenados. Al realizar esta acción, se borrará tanto el perfil del mismo, como todas y cada una de las mediciones que se hayan realizado sobre éste.
- **Filtrado de pacientes:** El usuario podrá filtrar el listado de pacientes que se muestran, según determinados criterios como el nombre, ID o las etiquetas (tags) introducidas al crear el perfil de los mismos.

En segunda instancia, el proyecto consiste, además, en la utilización de la aplicación para obtener una serie de mediciones reales y que se replicarán con el dispositivo Kinect del que se dispone en uno de los laboratorios de la Escuela Técnica Superior de Ingeniería Informática.

Una vez se dispongan de esas mediciones, se realizará un estudio comparativo de la precisión de la aplicación frente a la de Kinect (cuya precisión, se recuerda, ya ha sido validada en una ocasión anterior por un experto).

4.2.2. Suposiciones y restricciones

El proyecto debe cumplir con las siguientes restricciones:

1. Restricciones de presupuesto:
 - Todo el software y los servicios que se empleen para la realización de este proyecto deben ser gratuitos.
2. Restricciones de recursos:
 - Todo el trabajo será realizado única y exclusivamente por una persona, el alumno, bajo la supervisión y tutela del profesor (aunque éste último adoptará el rol de Product Owner durante la realización de este proyecto, como será explicado más adelante).
3. Restricciones de la aplicación:

- La aplicación debe ser desarrollada para el sistema operativo Android.
 - La aplicación debe utilizar un sistema de gestión de bases de datos relacional de tipo SQLite.
4. Restricciones temporales:
- Dada la fecha de inicio del TFG (4 de abril de 2017) y las fechas límite de entrega de la documentación para el curso 2016-2017 (28 de junio y 12 de julio de 2017 para primera y segunda convocatoria, respectivamente), se dispondrá única y exclusivamente de 3 meses para la realización de este proyecto al completo.
5. Restricciones de extensión:
- Según la guía docente del Trabajo de Fin de Grado, se anima a los estudiantes a no rebasar el límite de 150 páginas en lo que respecta a la memoria a presentar, por lo que, se tendrá en cuenta esta medida durante la realización del mismo.

4.2.3. Definiciones y acrónimos

Las definiciones y acrónimos mostrados a continuación pretenden aclarar conceptos que aparecen a lo largo de la memoria.

Goniómetro Instrumento destinado a medir los ángulos. Se utiliza para medir los ángulos de la cara y del cráneo, y para medir la amplitud de los movimientos de las articulaciones.

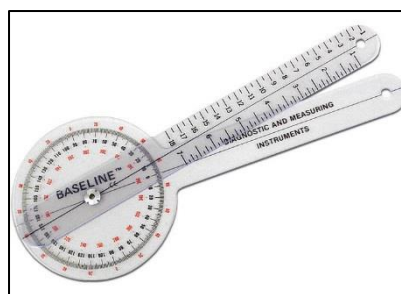


Figura 3 - Goniómetro tradicional

Como es cierto que existen ciertos movimientos articulares que pueden resultar poco familiares, se va a dar su definición también, ilustrándolos con el ejemplo concreto de la articulación de la cadera.

Dado que estos movimientos se producen en relación con un plano del cuerpo humano, se va a explicar también, brevemente, algunos planos para así poder comprender estos movimientos un poco mejor.

Plano sagital Partiendo de la posición de bipedestación, con los brazos extendidos a los lados y girados de forma que las palmas de las manos miren hacia el frente, se define el plano sagital como el plano perpendicular al suelo que corta al cuerpo en la mitad izquierda y derecha.

Plano coronal (frontal) Partiendo de la posición de bipedestación también, se define el plano coronal o frontal como el plano perpendicular al suelo que corta al cuerpo en las secciones ventral y dorsal. Este plano es perpendicular al sagital.

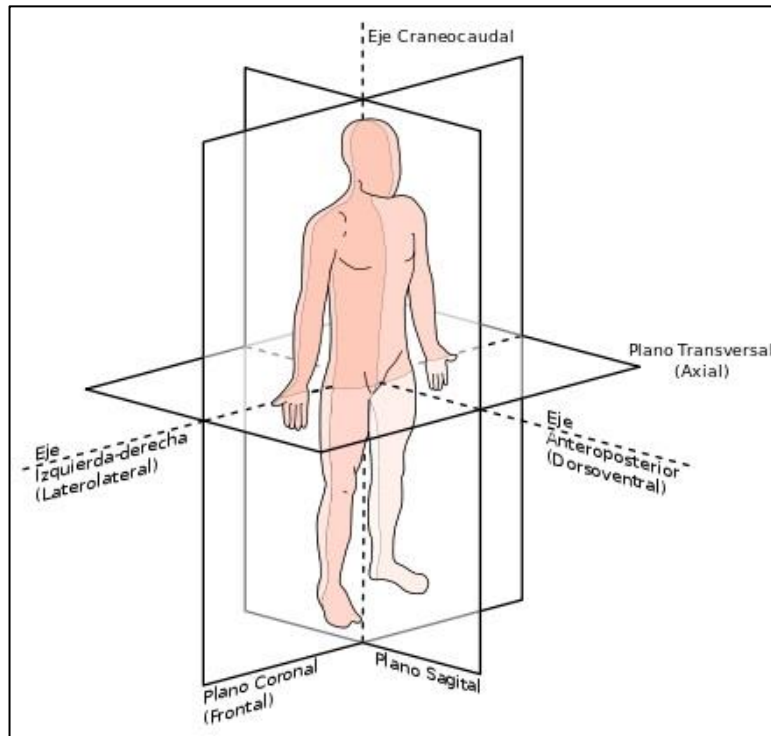


Figura 4 - Planos que dividen el cuerpo humano

Una vez ya se han definido cuáles son algunos de los planos que dividen al cuerpo humano, se pasará a explicar algunos movimientos de las articulaciones, en concreto, la flexión, extensión, aducción y abducción de la cadera.

Flexión Movimiento en el plano sagital por el cual se disminuye el ángulo que forman dos segmentos articulados del cuerpo y que se realiza en dirección anteroposterior (de adelante hacia atrás). En el caso de la cadera estos segmentos articulados harían referencia a la fosa acetabular del coxal (cadera) y al fémur (pierna).

Extensión Movimiento en el plano sagital por el cual se aumenta el ángulo que forman dos segmentos articulados del cuerpo y que se realiza en dirección anteroposterior. Es el movimiento opuesto a la flexión.

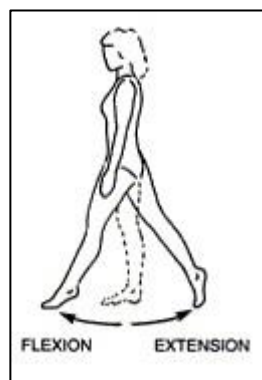


Figura 5 - Movimiento de flexión y extensión de cadera

Aducción Movimiento en el plano frontal por el cual un miembro u órgano se acerca al plano sagital.

Abducción Movimiento en el plano frontal por el cual un miembro u órgano se aleja del plano sagital.

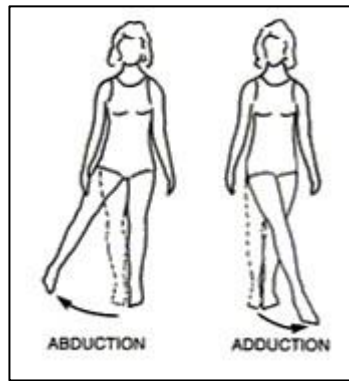


Figura 6 - Movimiento de abducción y aducción de cadera

4.2.4. Artefactos del proyecto

En este apartado de la memoria, se listan cada uno de los entregables del proyecto.

Al utilizarse Scrum como metodología de desarrollo, la versión definitiva de algunos de estos artefactos no se ha alcanzado hasta fechas muy cercanas a la entrega del mismo, tras haber sido modificados y/o revisados en las diversas iteraciones.

Estos artefactos son:

- **Plan de Desarrollo del proyecto**
- **Análisis**
- **Diseño**
- **Implementación y pruebas**
- **Estudio comparativo entre Kinect y TFG-Goniometer**
- **Versión final del producto**
- **Manual de usuario**

Además de estos artefactos propios del proyecto, se van a incluir algunos específicos del empleo de metodologías ágiles (Scrum, en este caso), que son:

- **Product backlog.** Se mostrará cómo va variando la pila del producto conforme avanzan los sucesivos sprints.
- **Product roadmap**
- **Plan de release**
- **Plan de seguimiento.** En esta parte se incluirán, entre otras cosas, el sprint backlog asociado a cada iteración, así como el incremento después de cada una.

4.3. Organización del proyecto

La organización del proyecto que se va a mostrar a continuación está hecha teniendo en cuenta que la metodología de desarrollo escogida ha sido Scrum.

4.3.1. Interfaces externas

En este caso, la interfaz externa del proyecto es el Product Owner o cliente para el que se desarrolla esta aplicación.

4.3.2. Estructura interna

En este caso, la estructura interna del proyecto está compuesta por dos únicos roles, el Scrum Master o facilitador y el Equipo de Desarrollo. En particular para este proyecto, ambos roles son desempeñados por la misma persona.

4.3.3. Roles y responsabilidades

A continuación, se muestran los roles del proyecto, junto con las funciones que cumplen dentro del marco de trabajo:

- **Product Owner:** La persona implicada en establecer un puente entre el usuario final, los responsables del negocio y el equipo de desarrollo. Trabaja día a día para clarificar los requisitos. A veces se le conoce como *customer representative*.
- **Scrum master:** La persona responsable de apoyar al equipo de desarrollo, eliminar las barreras organizativas y mantener la consistencia del proyecto ágil. A veces se le denomina *project facilitator*.
- **Equipo de desarrollo (team):** El grupo de personas que trabajan en la creación de un producto. Diseñadores, programadores, probadores, etc.

Estos roles han sido desempeñados por las siguientes personas:

- **Product Owner:** Rafael Matamoros Luque, Miguel Ángel Laguna
- **Scrum master:** Rafael Matamoros Luque
- **Equipo de desarrollo (team):** Rafael Matamoros Luque

Nota: En los primeros sprints, en los que el trabajo estaba claro, el rol de Product Owner ha sido desempeñado por Rafael Matamoros Luque, mientras que, en el último sprint, para asegurar la corrección del trabajo realizado, este rol ha sido desempeñado por Miguel Ángel Laguna.

4.3.4. Eventos para el desarrollo del producto

En la siguiente tabla se muestran los eventos que se han llevado a cabo durante el desarrollo de este producto, teniendo en cuenta que se ha empleado Scrum para la realización del mismo:

Evento	Descripción
Planificación del proyecto	La planificación inicial. Incluye una visión del producto y un “product roadmap”. Se puede realizar en menos de un día.
Planificación de la release	La planificación del conjunto siguiente de características que va a tener la nueva release del producto. Solo se planifica una release cada vez.
Sprint	Un ciclo de desarrollo corto para crear funcionalidad de producto vendible. A veces se llaman iteraciones, normalmente entre una y cuatro semanas. La duración debe ser la misma durante todo el proyecto.
Planificación de Sprint	Una reunión al comienzo de cada sprint donde se fijan los objetivos del mismo. Se identifican los requisitos y las tareas a realizar para completar cada requisito.

Daily scrum	Una reunión de unos 15 minutos a tener cada día de un sprint donde se indica lo realizado el día anterior y lo que se va a realizar en el actual.
Revisión de Sprint	Una reunión al final de cada sprint, iniciada por el “product owner”, donde se analiza la funcionalidad del producto conseguida durante el sprint.
Sprint retrospective	Una reunión al final de cada sprint, donde el equipo del sprint analiza qué fue bien, qué debe cambiar y cómo realizar los cambios.

Tabla 3 - Eventos en Scrum

4.4. Planificación del proyecto

4.4.1. Estimación temporal del proyecto

A continuación, se presentan algunos entregables de Scrum que ofrecen una estimación de alto nivel poco precisa, pero que han servido como base para la posterior planificación de los sprints (con una estimación mucho más precisa).

4.4.1.1. Product roadmap

En este apartado se muestra el product roadmap. Este artefacto es una vista de alto nivel de los requisitos del producto con un marco temporal poco preciso de cuándo se desarrollarán los mismos.

Para la realización de este product roadmap, se han tomado como fechas de referencia, aquellas que se corresponden con un mes después del inicio del proyecto, dos meses después y, coincidiendo con la entrega final del proyecto, tres meses después del inicio del mismo (04/05, 04/06 y 04/07, respectivamente).

Además, para la realización de estos artefactos, se ha decidido dividir el proyecto en tres grandes bloques.

Por un lado, está el apartado de documentación del mismo, en el que se incluye toda aquella documentación relativa única y exclusivamente al desarrollo de la aplicación *TFG-Goniometer*, además de toda aquella documentación perteneciente al Trabajo de Fin de Grado en particular (Introducción, Plan de Desarrollo del Proyecto, Conclusiones, etc.).

Por otro lado, está el apartado de desarrollo de la aplicación en sí. Es decir, todo el apartado de implementación de la misma, así como las pruebas pertinentes para asegurar su correcto funcionamiento.

Y, por último, se ha decidido separar, también, la parte en la que se hace un estudio comparativo entre la precisión de la aplicación frente a la de Kinect. Si bien este apartado forma parte de la documentación del proyecto en sí, se ha decidido separarlo dado que es uno de los objetivos principales de este proyecto (como ya se ha comentado previamente), junto con el desarrollo de la aplicación propiamente dicho.

Nota importante: Aquellos elementos marcados con un * tanto en el product roadmap como en el plan de release se han realizado de forma parcial. Es decir, se ha obtenido la especificación de requisitos, casos de uso, modelo de dominio, etc. de la funcionalidad que se ha implementado en primera instancia; posponiendo, de esta manera, la finalización de estos elementos hasta que se ha terminado de implementar el resto de la funcionalidad del sistema.

	04/05/2017 (v 0.1)	04/06/2017 (v 0.2)	04/07/2017 (v 1.0)
Documentación	<ul style="list-style-type: none"> - Plan de Desarrollo completo. - Especificación de requisitos del sistema. * - Documento sobre la implementación del sistema y de las pruebas que se han realizado. * 	<ul style="list-style-type: none"> - Especificación de los requisitos del sistema. * - Listado y especificación de los casos de uso del sistema. * - Modelo de dominio y descripción de las clases del sistema (completo) - Documento sobre el diseño de la arquitectura del sistema y descripción de los patrones utilizados. * - Documento sobre la implementación del sistema y de las pruebas que se han realizado. * 	<ul style="list-style-type: none"> - Especificación de los requisitos del sistema (completo). - Listado y especificación de los casos de uso del sistema (completo). - Documento sobre el diseño de la arquitectura del sistema y descripción de los patrones utilizados (completo). - Documento sobre la implementación del sistema y de las pruebas que se han realizado (completo).
Aplicación	<ul style="list-style-type: none"> - Utilización de la aplicación para poder hacer mediciones de los movimientos de las articulaciones. 	<ul style="list-style-type: none"> - Creación de perfiles de pacientes y almacenamiento de esa información en una Base de Datos. - Acceso al perfil de cada uno de los pacientes introducidos previamente. - Inclusión de posibilidad de almacenamiento en una Base de Datos de cada una de las mediciones realizadas. 	<ul style="list-style-type: none"> - Visualización de las últimas 10 mediciones guardadas. - Filtración de pacientes almacenados en el sistema. - Borrado de mediciones del sistema. - Borrado de perfiles de pacientes del sistema. - Creación del menú de información sobre la aplicación. - Traducción de la aplicación al inglés. - Creación del manual de usuario de la aplicación.

Comparación Kinect - App			- Documento sobre los resultados y conclusiones obtenidos de la comparación entre Kinect y la aplicación.
-----------------------------	--	--	---

Tabla 4 - Product roadmap del proyecto

4.4.1.2. División del trabajo

En este apartado, se mostrará la división que se ha hecho de todo el product backlog (mostrado en el apéndice relativo al Plan de seguimiento del proyecto) teniendo en cuenta el tiempo del que se dispone hasta la entrega del proyecto.

La fecha de inicio del proyecto (como ya se ha comentado previamente) ha sido el 4 de abril de 2017 y la fecha de fin estimada es el 4 de julio de 2017. Teniendo esto en cuenta, se dispone de un total de 92 días para la realización de este proyecto.

Dado que se va a trabajar en el proyecto de lunes a sábado (independientemente de si algún día es festivo o no), a ese número hay que restarle los domingos, con lo cual, se queda en 79 días de trabajo.

Teniendo en cuenta que la jornada de trabajo es de 6h diarias, haciendo cuentas da un total de **474 horas-hombre**. Si bien es cierto que este número excede en casi un 60% las horas de trabajo estimadas en la guía docente del Trabajo de Fin de Grado (300 horas), un proyecto de esta importancia y, además, siendo el primero que se realiza de esta índole, es muy probable que esta estimación esté bastante cercana a la realidad al finalizar el mismo.

A continuación, se detalla la división en sprints que se ha hecho del proyecto.

Se ha optado por realizar un total de 5 sprints, con una duración de 13 días de trabajo cada uno de ellos (15 días naturales contando los domingos).

En la siguiente tabla se muestran cada uno de los sprints junto con las fechas de inicio y fin establecidas para cada uno al comienzo del desarrollo del proyecto.

Sprint	Fecha de inicio	Fecha de fin
1	20 de abril de 2017	4 de mayo de 2017
2	5 de mayo de 2017	19 de mayo de 2017
3	20 de mayo de 2017	3 de junio de 2017
4	5 de junio de 2017	19 de junio de 2017
5	20 de junio de 2017	4 de julio de 2017

Tabla 5 - División del proyecto por sprints

4.4.1.3. Plan de release

En este apartado, se muestra el plan de release. Este artefacto es un calendario de alto nivel para el lanzamiento de software completamente operativo.

	Sprint 1	Sprint 2	Sprint 3	Sprint 4	Sprint 5
Documentación	<ul style="list-style-type: none"> - Plan de Desarrollo completo. - Especificación de requisitos del sistema. * - Documento sobre la implementación del sistema y las pruebas que se han realizado. * 	<ul style="list-style-type: none"> - Especificación de los requisitos del sistema. * - Listado y especificación de los casos de uso del sistema. * - Modelo de dominio y descripción de las clases del sistema. * - Documento sobre el diseño de la arquitectura del sistema y descripción de los patrones utilizados. * - Documento sobre la implementación del sistema y las pruebas que se han realizado. * 	<ul style="list-style-type: none"> - Especificación de los requisitos del sistema. * - Listado y especificación de los casos de uso del sistema. * - Modelo de dominio y descripción de las clases del sistema (completo). - Documento sobre el diseño de la arquitectura del sistema y descripción de los patrones utilizados. * - Documento sobre la implementación del sistema y las pruebas que se han realizado. * 	<ul style="list-style-type: none"> - Especificación de los requisitos del sistema. * - Listado y especificación de los casos de uso del sistema. * - Documento sobre el diseño de la arquitectura del sistema y descripción de los patrones utilizados. * - Documento sobre la implementación del sistema y las pruebas que se han realizado. * 	<ul style="list-style-type: none"> - Especificación de los requisitos del sistema (completo). - Listado y especificación de los casos de uso del sistema (completo). - Documento sobre el diseño de la arquitectura del sistema y descripción de los patrones utilizados (completo). - Documento sobre la implementación del sistema y las pruebas que se han realizado (completo).
Aplicación	<ul style="list-style-type: none"> - Utilización de la aplicación para poder hacer mediciones de los movimientos de las articulaciones. 	<ul style="list-style-type: none"> - Creación de perfiles de pacientes y almacenamiento de esa información en una Base de Datos. - Acceso al perfil de cada uno de los pacientes introducidos previamente. 	<ul style="list-style-type: none"> - Inclusión de posibilidad de almacenamiento en una Base de Datos de cada una de las mediciones realizadas. 	<ul style="list-style-type: none"> - Visualización de las últimas 10 mediciones guardadas. - Filtración de pacientes almacenados en el sistema. - Borrado de mediciones del sistema. 	<ul style="list-style-type: none"> - Borrado de perfiles de pacientes del sistema. - Creación del menú de información sobre la aplicación. - Traducción de la aplicación al inglés.

					- Creación del manual de usuario de la aplicación.
Comparación Kinect - App					- Documento sobre los resultados y conclusiones obtenidos de la comparación entre Kinect y la aplicación.

Tabla 6 - Plan de release del proyecto

4.4.2. Estimación de costes

Teniendo en cuenta que la duración del proyecto se ha estimado en 3 meses a jornada completa y que el salario anual de un Ingeniero Informático ronda los 22,000€, se estima el coste de este proyecto en 5,500€. No existe ningún coste derivado de amortización de equipos o licencias, ya que, todos los recursos empleados para la realización del mismo son gratuitos (tal y como se ha explicado en el apartado 4.2.2).

4.4.3. Métricas

En este apartado, se va a comentar brevemente una métrica que se utilizará durante el desarrollo de este proyecto.

Se mide la velocidad de cada uno de los sprints utilizando como medida de ésta, el número de tareas realizadas por sprint.

$$\text{Velocidad (sprint)} = \text{nº tareas realizadas (sprint)}$$

Tabla 7 - Métrica - Velocidad del sprint

4.4.4. Plan de seguimiento

Para consultar el Plan de seguimiento del proyecto al completo, diríjase al Anexo II.

4.5. Plan de Gestión de Riesgos

A continuación, se muestra tanto la matriz de impacto/probabilidad como un listado con todos los riesgos identificados previamente y durante el desarrollo de *TFG-Goniometer*. Para cada uno de estos riesgos, se adjunta una descripción del mismo, el impacto que supondría de llegar a ocurrir, la probabilidad de ocurrencia, la exposición, así como tanto un plan de protección, como uno de contingencia.

En el caso de la exposición a un riesgo, ésta se obtiene de la siguiente manera:

$$\text{Exposición al riesgo} = \text{Probabilidad} \times \text{Impacto}$$

Tabla 8 - Definición de la exposición al riesgo

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

Tabla 9 - Matriz Impacto/Probabilidad

R01 – Pérdida de datos

Pérdida (parcial o completa) de la documentación o código, por alguna causa en concreto, así como de todo el tiempo invertido en la elaboración de los datos perdidos.

Impacto Catastrófico

Probabilidad Bajo

Exposición Moderado

Plan de protección Emplear constantemente herramientas de control de versiones, así como la realización de copias de seguridad a diario.

Plan de contingencia En función de la cantidad de datos perdidos, se procedería a recuperar las últimas versiones disponibles tanto de la documentación como del código y, de ser necesario, se procedería a realizar una replanificación del resto del proyecto.

R02 – Mala estimación de la complejidad de alguna tarea del *sprint backlog*

La estimación de la complejidad de alguna tarea resulta distanciarse demasiado de la complejidad real que ha tenido la realización de la misma (generalmente se ha subestimado su complejidad).

Impacto Crítico

Probabilidad Medio

Exposición Moderado

Plan de protección Prestar especial atención en el scrum daily a tareas que se están demorando más de lo previsto y/o que se prevé que su complejidad sea mayor a la estimada inicialmente.

Plan de contingencia Como primera medida, se aumentará el número de horas de trabajo diarias de 6 a 8 hasta la completitud de la tarea en concreto. Si aún con esa medida, se está empezando a demorar el inicio del resto de tareas, se procederá a trabajar en el proyecto el domingo (día inicialmente pensado como descanso). Y, si ni empleando estas dos medidas se logra que la duración del sprint no aumente, se procederá a realizar una replanificación temporal del resto del proyecto en función de la gravedad del retraso que se haya sufrido debido a esta tarea.

R03 – Cese temporal del trabajo debido a enfermedad

No se puede continuar con el trabajo debido a que el Product Owner, el Scrum Master o el Equipo de Desarrollo se encuentran indispuestos por motivos de salud.

Impacto Marginal

Probabilidad Bajo

Exposición Ninguno

Plan de protección No se dispone de ningún plan de protección ante este tipo de riesgos. En la mayoría de ocasiones simplemente ocurren, sin poder hacer nada al respecto.

Plan de contingencia En la medida de lo posible y según el estado de salud de la persona en concreto, se procedería a trabajar, mientras las condiciones de salud lo permitan, para evitar excesivos retrasos en la entrega del producto. Una vez recuperados, se procedería a realizar una evaluación del retraso y, de ser necesario, una replanificación del trabajo restante.

R04 – Descubrimiento de un nuevo requisito

Durante el desarrollo del proyecto, se ha encontrado un nuevo requisito que debe ser incorporado al mismo.

Impacto Crítico

Probabilidad Bajo

Exposición Bajo

Plan de protección En el momento de realizar la especificación de requisitos que debe cumplir el proyecto, hacer un análisis exhaustivo, asegurándose de que se han abarcado todos los requisitos necesarios.

Plan de contingencia Dado que Scrum sigue un desarrollo iterativo e incremental, se procedería a evaluar si el nuevo requisito debe ser incorporado de inmediato o se puede posponer su inclusión en el proyecto hasta sprints siguientes. Si fuera necesario incorporarlo de inmediato, se pararía el trabajo que se estuviera realizando y se haría una replanificación al final del sprint. Si no fuera demasiado urgente, en la planificación del sprint siguiente se procedería a incorporarse al proyecto.

R05 – Errores en fase de análisis

El documento de análisis elaborado presenta errores que, de no ser subsanados a tiempo, provocarán errores en fases de diseño e implementación posteriores.

Impacto Crítico

Probabilidad Medio

Exposición Moderado

Plan de protección Prestar especial atención al trabajo de análisis que se está realizando, así como consultar con el tutor del TFG con cierta asiduidad para verificar que el trabajo realizado es correcto, y así no propagar errores hasta fases posteriores de desarrollo.

Plan de contingencia Corregir el error tan pronto como sea detectado y proceder a realizar una replanificación del trabajo restante si fuera necesario.

R06 – Errores en fase de diseño

El documento de diseño elaborado presenta errores que, de no ser subsanados a tiempo, provocarán errores en fases de implementación posteriores.

Impacto Crítico

Probabilidad Medio

Exposición Moderado

Plan de protección Prestar especial atención al trabajo de diseño que se está llevando a cabo, procurando ceñirse a toda la documentación de análisis obtenida previamente, así como consultar con el tutor del TFG con cierta asiduidad para verificar que el trabajo realizado es correcto, y así no propagar errores hasta fases posteriores de desarrollo.

Plan de contingencia Corregir el error tan pronto como sea detectado y proceder a realizar una replanificación del trabajo restante si fuera necesario.

R07 – Errores en fase de implementación

El software desarrollado presenta errores que afectan a la funcionalidad del mismo.

Impacto Crítico

Probabilidad Alto

Exposición Alto

Plan de protección Emplear buenas prácticas de programación, procurando hacer un código legible y bien documentado que se ciña al documento de diseño del proyecto. Además de ir realizando pequeñas pruebas de funcionamiento durante el desarrollo del código.

Plan de contingencia Detectar en qué punto se encuentra el error y corregirlo tan pronto como sea posible, para así evitar un retraso en la duración del proyecto. Si fuera necesario, se haría una replanificación del trabajo restante.

R08 – Problemas con el IDE Android Studio

Debido a las continuas actualizaciones que sufren tanto Android como el IDE Android Studio, se pueden producir problemas con el Gradle o en el API de Android, provocando errores en algunas partes del código.

Impacto Marginal

Probabilidad Alto

Exposición Moderado

Plan de protección Antes de realizar una actualización, revisar lo que ofrece y, en el caso de que no suponga un riesgo para el proyecto, actualizar.

Plan de contingencia En primera instancia, se procuraría revisar cuáles son los nuevos errores que han aparecido a raíz de la actualización y, si son fácilmente subsanables, corregir lo que sea necesario. De no ser posible y provocar problemas graves de funcionamiento en la aplicación, volver a una versión anterior del IDE. Si esto supusiera un retraso con respecto a la planificación inicial, replanificar lo que fuera necesario.

R09 – Documentación escasa o mal elaborada

Durante el desarrollo del proyecto se detecta que parte de la documentación elaborada hasta ese momento presenta errores o es demasiado escasa.

Impacto Despreciable

Probabilidad Alto

Exposición Bajo

Plan de protección Releer en varias ocasiones la documentación elaborada con el fin de detectar estos problemas cuanto antes, así como consultar con asiduidad el tutor del TFG para contar con una segunda opinión del trabajo realizado.

Plan de contingencia Dado que un error de este tipo (habiendo contemplado previamente errores en la documentación de análisis y diseño) no se supone un problema grave al desarrollo del proyecto, se procedería a corregir o reescribir la parte de la documentación que sea errónea fuera del horario de trabajo habitual, para así evitar que se puedan producir retrasos con el resto de la planificación.

R10 – Fallo en alguno de los equipos de trabajo

Se produce un fallo en alguno de los equipos de trabajo, lo que los deja inutilizables hasta que se solucione el problema.

Impacto Crítico

Probabilidad Bajo

Exposición Bajo

Plan de protección Procurar que todos los equipos de trabajo tengan un buen mantenimiento, tanto del hardware como del software, así como procurar seguir unas buenas prácticas de navegación a través de Internet, para evitar, en la medida de lo posible, que los equipos se infecten de cualquier tipo de malware.

Plan de contingencia En función de cómo de grave sea el fallo que se ha producido, en primera instancia, se procedería a tratar de recuperar el equipo al estado anterior en el que estaba antes del fallo,

procurando hacer una limpieza exhaustiva tanto del hardware como del software. En segunda instancia se procedería a realizar un formateo del sistema operativo, con la consiguiente reinstalación del software necesario para desarrollar este proyecto. Y, en última instancia, se procedería a reemplazar el equipo por otro del que ya se disponga, o adquirir uno nuevo si no se pudiera disponer de otro.

Capítulo 5

Análisis

En este capítulo, se obtendrá primero la especificación de requisitos de *TFG-Goniometer*. Posteriormente, se identificarán los actores del sistema, así como los casos de uso del mismo, y se obtendrá el diagrama de casos de uso y una descripción detallada de cada uno de ellos, junto con su diagrama de actividad. Y, por último, se adjuntará el modelo de dominio del sistema, así como una descripción de las clases que lo forman.

5.1. Requisitos

5.1.1. Ingeniería tradicional vs Desarrollo ágil (Scrum)

La ingeniería del software tradicional diferencia dos ámbitos de requisitos:

- Requisitos del sistema
- Requisitos del software

Los requisitos del sistema forman parte del proceso de adquisición, y por tanto es responsabilidad del cliente la definición del problema y de las funcionalidades que debe aportar la solución.

Independientemente de que se esté hablando de ingeniería del software tradicional o de desarrollo ágil, la pila del producto (product backlog) es responsabilidad del cliente.

En la siguiente imagen, se muestra la diferencia entre ambos, en cuanto a requisitos se refiere:

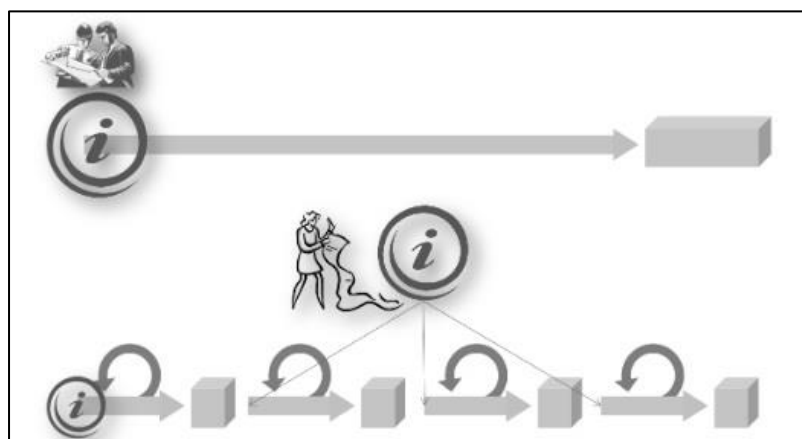


Figura 7 - Requisitos ingeniería de software tradicional vs métodos ágiles

- En los proyectos predictivos, los requisitos del sistema suelen especificarse en documentos formales; mientras que en los proyectos ágiles toman la forma de pila de producto (product backlog) o lista de historias de usuario.
- Los requisitos del sistema formales se especifican de forma completa y cerrada al inicio del proyecto; sin embargo, una pila de producto es un documento vivo, que evoluciona durante el desarrollo.

- Los requisitos del sistema los desarrolla una persona o equipo especializado en ingeniería de requisitos a través del proceso de obtención (elicitación) con el cliente. En Scrum la visión del cliente es conocida por todo el equipo (el cliente colabora con el equipo de desarrollo) y la pila del producto se realiza y evoluciona de forma continua con los aportes de todos.

Scrum, aplicado al software, emplea dos formatos para registrar los requisitos:

- Pila de producto (Product Backlog)
- Pila del sprint (Sprint Backlog)

La pila del producto registra los requisitos vistos desde el punto de vista del cliente. Un enfoque similar al de los requisitos del sistema de la ingeniería tradicional. Está formada por la lista de funcionalidades o historias de usuario que desea obtener el cliente, ordenadas por la prioridad que el mismo le otorga a cada una.

La pila del sprint refleja los requisitos vistos desde el punto de vista del equipo de desarrollo. Está formada por la lista de tareas en las que se descomponen las historias de usuario que se van a llevar a cabo en el sprint.

5.1.2. Requisitos funcionales

ID	Nombre	Descripción
RF01	Medición de los movimientos articulares	El sistema deberá permitir al usuario realizar mediciones del movimiento de las articulaciones.
RF02	Almacenamiento de las mediciones realizadas	El sistema deberá permitir al usuario almacenar todas y cada una de las mediciones que realice.
RF03	Creación y almacenamiento de perfiles de pacientes	El sistema deberá permitir al usuario crear y almacenar un perfil para cada uno de los pacientes.
RF04	Acceso a perfiles de pacientes	El sistema deberá permitir al usuario acceder al perfil de cada uno de los pacientes que ha almacenado previamente.
RF05	Acceso a las 10 últimas mediciones guardadas	El sistema deberá permitir al usuario acceder a la información de las 10 últimas mediciones guardadas, así como a los pacientes a los que se corresponden.
RF06	Filtración de perfiles de pacientes	El sistema deberá permitir al usuario filtrar los perfiles de los pacientes almacenados por nombre, id o etiquetas.
RF07	Borrado de perfiles de pacientes	El sistema deberá permitir al usuario borrar de la base de datos toda la información relativa a un paciente, así como todas las mediciones que se hayan realizado sobre él.
RF08	Borrado de mediciones realizadas	El sistema deberá permitir al usuario borrar de la base de datos cualquier medición que estuviera previamente almacenada, siempre que ésta sea una de las 10 últimas mediciones que haya almacenado en la misma.
RF09	Detalle de mediciones almacenadas	El sistema deberá permitir al usuario consultar el detalle de las mediciones almacenadas en la base de datos, mostrando información sobre la fecha en que se realizó, la articulación involucrada, el movimiento de la misma y la lectura del

		goniómetro, además del paciente sobre el que se realizó la medición.
--	--	--

Tabla 10 - Requisitos funcionales

5.1.3. Requisitos no funcionales

ID	Nombre	Descripción
RNF01	Facilidad de instalación	La aplicación debe ser fácilmente instalable por cualquier usuario a partir de su apk.
RNF02	Solicitud de permisos	La aplicación debe solicitar, en versiones superiores a la API 23 de Android, los permisos necesarios para realizar las tareas correspondientes.
RNF03	Sistema de gestión de bases de datos relacional SQLite	La aplicación debe utilizar para la persistencia este tipo de sistema de gestión de bases de datos.
RNF04	Versiones compatibles	La aplicación debe ser compatible con cualquier dispositivo Android con versión comprendida entre 4.0 y 7.1 (97.4% de todos los dispositivos Android)
RNF05	Idiomas disponibles	La aplicación debe estar traducida tanto en castellano como en inglés.

Tabla 11 - Requisitos no funcionales

5.1.3.1. Requisitos de información

ID	Nombre	Descripción
RINF01	Contenido del perfil de un paciente	Nombre, edad, sexo, id, teléfono, dirección, síntomas, diagnóstico, tratamiento previo, tratamiento actual, comentarios adicionales y tags.
RNF02	Contenido de una medición	Fecha y hora, lectura del goniómetro, articulación involucrada, lado del cuerpo, movimiento realizado, modo del movimiento y paciente.

Tabla 12 - Requisitos de información

5.2. Modelo de Casos de Uso

5.2.1. Identificación de actores del sistema

- Usuario: Es el único actor del sistema y representa a la persona que utilizará la aplicación.

5.2.2. Diagrama de casos de uso

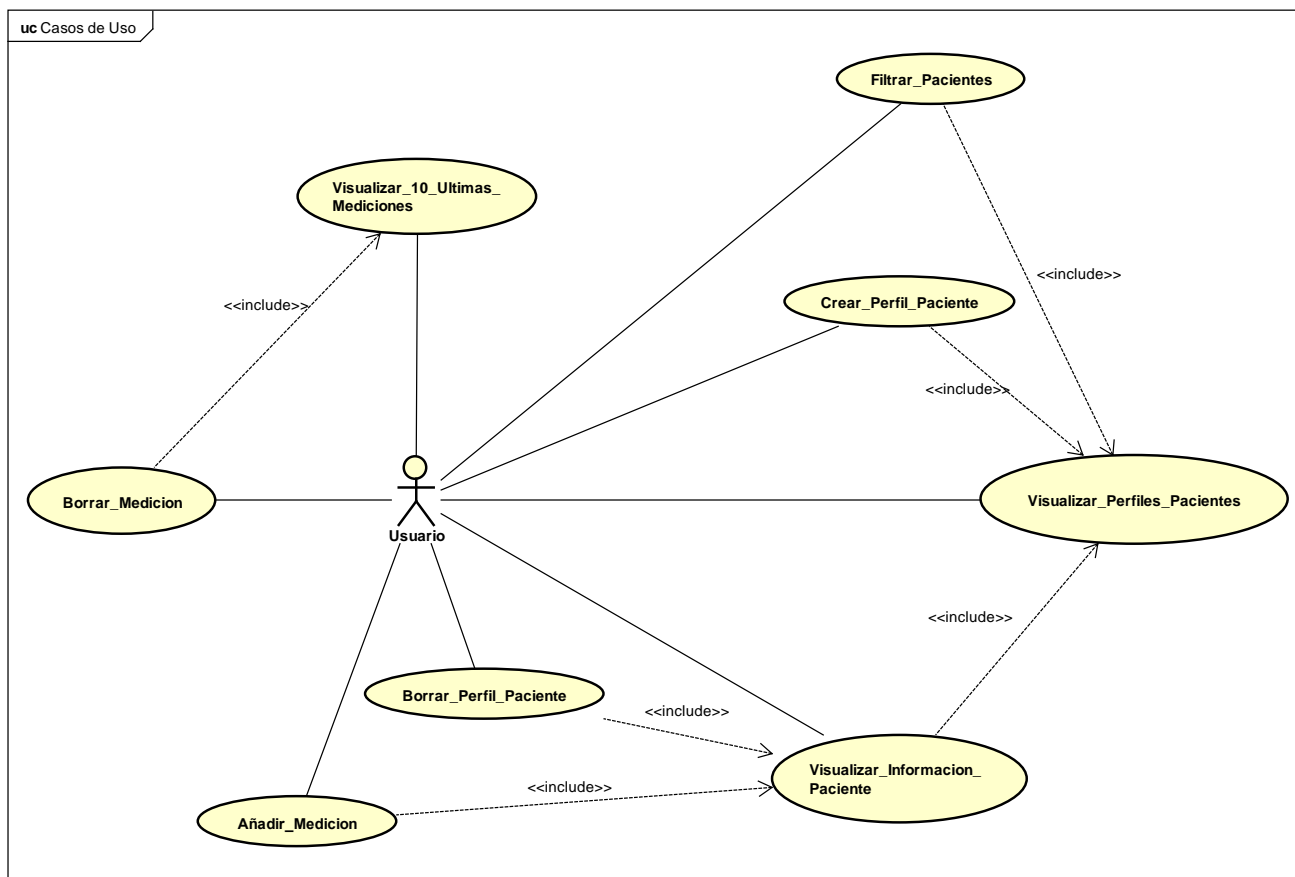


Figura 8 - Diagrama de casos de uso

5.2.3. Especificación de casos de uso

UC-0001	Añadir_Medicion
Descripción	Un Usuario desea añadir una nueva medición que acaba de realizar.
Actor	Usuario
Precondición	1. La aplicación se encuentra iniciada. 2. El Usuario ha almacenado algún paciente en el sistema.
Secuencia normal	1. El caso de uso comienza cuando el Usuario selecciona la opción para iniciar una medición. 2. El sistema muestra la interfaz gráfica asociada a esa opción con el goniómetro inicialmente a 0. 3. El Usuario coloca el dispositivo en la posición que quiere que se tome como referencia para empezar a medir el movimiento y pulsa sobre el goniómetro. 4. El sistema muestra cómo va variando el ángulo en función de la posición del dispositivo en cada momento. 5. El Usuario vuelve a pulsar sobre el goniómetro en el momento en que decide que el movimiento ha finalizado. 6. El sistema muestra el ángulo que se ha desplazado el dispositivo desde el inicio del movimiento hasta el final. 7. El Usuario solicita almacenar la medición.

	<p>8. El sistema permite seleccionar un paciente de los existentes o cancelar el almacenamiento de la medición.</p> <p>9. El Usuario decide seleccionar un paciente de los existentes.</p> <p>10. Se ejecuta el caso de uso <i>Visualizar_Informacion_Paciente</i>.</p> <p>11. El Usuario decide añadir la medición realizada a ese paciente en concreto.</p> <p>12. El sistema muestra de nuevo la información almacenada sobre ese paciente, además de una lista con los siguientes campos para rellenar por el usuario (obligatoriamente) relativos a la medición:</p> <ul style="list-style-type: none"> ▪ Lado (con respecto al plano sagital) ▪ Articulación ▪ Movimiento ▪ Tipo de movimiento <p>Además de estos campos, el sistema muestra dos campos más (no rellenables por el usuario):</p> <ul style="list-style-type: none"> ▪ Fecha y hora ▪ Lectura del goniómetro <p>13. El Usuario introduce la información requerida e indica que ha terminado de introducir información.</p> <p>14. El sistema almacena la información introducida por el usuario, muestra un mensaje de éxito y muestra el perfil del paciente actualizado con esa nueva medición en el listado de mediciones realizadas a ese paciente en concreto.</p> <p>15. El caso de uso finaliza.</p>
Postcondición	La medición ha sido almacenada en el sistema.
Flujo alternativo	<p>3.1., 5.1., 7.1. Si el Usuario decide reiniciar la medición, el caso de uso continúa en el paso 2.</p> <p>9.1. Si el Usuario decide cancelar el almacenamiento de la medición, el caso de uso continúa en el paso 6.</p>
Excepciones	<p>8.1. Si el sistema detecta que no se ha realizado ninguna medición, se muestra un mensaje al usuario indicando este problema y el caso de uso continúa en el paso 2.</p> <p>14.1. Si el sistema detecta que alguno de los campos marcados como obligatorios no han sido rellenados, se muestra un mensaje al usuario indicando este problema y el caso de uso continúa en el paso 12.</p> <p>14.2. Si ocurre algún problema a la hora de almacenar la medición en la Base de Datos, el sistema muestra un mensaje de error indicando esta condición al usuario y el caso de uso continúa en el paso 12.</p>
Frecuencia	Muy alta

Tabla 13 - Especificación UC-0001

UC-0002	Visualizar_10_Ultimas_Mediciones
Descripción	Un Usuario desea ver las 10 últimas mediciones que ha almacenado en el sistema.
Actor	Usuario
Precondición	La aplicación se encuentra iniciada.
Secuencia normal	1. El caso de uso comienza cuando el Usuario selecciona la opción para visualizar las 10 últimas mediciones almacenadas.

	<p>2. El sistema muestra una lista de las 10 últimas mediciones que se han almacenado ordenadas de más a menos reciente, incluyendo la siguiente información de cada una de ellas:</p> <ul style="list-style-type: none"> ▪ Paciente sobre el que se realizó. ▪ Fecha de la medición. ▪ Articulación involucrada. ▪ Movimiento realizado y si fue realizado por el paciente, con ayuda, o por el médico. ▪ Lectura del goniómetro. <p>3. El caso de uso finaliza.</p>
Postcondición	Ninguna.
Flujo alternativo	No existe.
Excepciones	No existen.
Frecuencia	Alta

Tabla 14 - Especificación UC-0002

UC-0003	Borrar_Medicion
Descripción	Un Usuario desea borrar una de las mediciones almacenadas en el sistema.
Actor	Usuario
Precondición	<p>1. La aplicación se encuentra iniciada.</p> <p>2. El Usuario ha guardado alguna medición en el sistema.</p>
Secuencia normal	<p>1. Se ejecuta el caso de uso <i>Visualizar_10_Ultimas_Mediciones</i>.</p> <p>2. El Usuario pulsa sobre el botón correspondiente de una de las mediciones.</p> <p>3. El sistema muestra un mensaje para confirmar el borrado de la medición.</p> <p>4. El Usuario confirma que desea borrar la medición.</p> <p>5. El sistema borra la medición de la Base de Datos, muestra un mensaje al usuario indicando que la medición se ha borrado con éxito, y muestra el listado de las últimas 10 mediciones actualizado después de haber borrado la medición en concreto.</p> <p>6. El caso de uso finaliza.</p>
Postcondición	La medición ha sido eliminada del sistema.
Flujo alternativo	4.1. Si el Usuario cancela el borrado de la medición, el caso de uso finaliza en el paso 6.
Excepciones	5.1. Si ocurre algún problema a la hora de eliminar la medición de la Base de Datos, el sistema muestra un mensaje de error indicando esta condición al usuario y el caso de uso continúa en el paso 1.
Frecuencia	Muy baja

Tabla 15 - Especificación UC-0003

UC-0004	Visualizar_Perfiles_Pacientes
Descripción	Un Usuario desea ver los perfiles de los pacientes que ha almacenado previamente en el sistema.
Actor	Usuario
Precondición	La aplicación se encuentra iniciada.
Secuencia normal	1. El caso de uso comienza cuando el Usuario selecciona la opción para ver los perfiles de los pacientes.

	2. El sistema muestra una lista con todos los nombres de pacientes que se encuentran almacenados en el sistema. 3. El caso de uso finaliza.
Postcondición	Ninguna.
Flujo alternativo	No existe.
Excepciones	No existen.
Frecuencia	Muy alta

Tabla 16 - Especificación UC-0004

UC-0005	Filtrar_Pacientes
Descripción	Un Usuario desea filtrar la lista de pacientes que está visualizando.
Actor	Usuario
Precondición	La aplicación se encuentra iniciada.
Secuencia normal	<ol style="list-style-type: none"> 1. Se ejecuta el caso de uso <i>Visualizar_Perfiles_Pacientes</i>. 2. El usuario selecciona un criterio de filtración (nombre, ID paciente o tags) de los perfiles que se encuentran almacenados en el sistema e introduce información en el cuadro de búsqueda. 3. El sistema muestra una lista de nombres de pacientes en función del filtro establecido y la información introducida por el usuario en tiempo real, realizando la comprobación carácter a carácter. 4. El caso de uso finaliza.
Postcondición	Ninguna.
Flujo alternativo	No existe.
Excepciones	No existen.
Frecuencia	Media

Tabla 17 - Especificación UC-0005

UC-0006	Crear_Perfil_Paciente
Descripción	Un Usuario desea crear un nuevo perfil para un paciente.
Actor	Usuario
Precondición	La aplicación se encuentra iniciada.
Secuencia normal	<ol style="list-style-type: none"> 1. Se ejecuta el caso de uso <i>Visualizar_Perfiles_Pacientes</i>. 2. El usuario pulsa sobre el botón para crear un nuevo perfil. 3. El sistema muestra una lista con los siguientes campos para rellenar (obligatoriamente) por el usuario: <ul style="list-style-type: none"> ▪ Nombre ▪ Edad ▪ Sexo ▪ ID ▪ Diagnóstico ▪ Etiquetas (tags) <p>Además de los siguientes campos (opcionales):</p> <ul style="list-style-type: none"> ▪ Teléfono ▪ Dirección ▪ Síntomas

	<ul style="list-style-type: none"> ▪ Tratamiento previo ▪ Tratamiento actual ▪ Comentarios adicionales <p>4. El Usuario introduce, al menos, la información relativa a los campos obligatorios sobre el paciente y selecciona que quiere guardar el perfil del paciente.</p> <p>5. El sistema almacena el nuevo paciente en el sistema, muestra un mensaje de éxito y muestra la lista de pacientes almacenados en el mismo, actualizada con el nuevo paciente.</p> <p>6. El caso de uso finaliza.</p>
Postcondición	El perfil de un nuevo paciente ha sido almacenado en el sistema.
Flujo alternativo	Si en algún momento el usuario vuelve al listado de pacientes sin haber terminado de almacenar el perfil del paciente, el caso de uso finaliza, quedando sin efecto.
Excepciones	<p>5.1. Si el sistema detecta que alguno (o varios) de los campos que son obligatorios, están vacíos, se muestra un mensaje al usuario indicando esta situación y el caso de uso continúa en el paso 3.</p> <p>5.2. Si ocurre algún problema a la hora de almacenar el perfil del paciente en la Base de Datos, el sistema muestra un mensaje de error indicando esta condición al usuario y el caso de uso continúa en el paso 3.</p>
Frecuencia	Media

Tabla 18 - Especificación UC-0006

UC-0007	Visualizar_Informacion_Paciente
Descripción	Un Usuario desea ver toda la información almacenada en el sistema relativa a un paciente en concreto.
Actor	Usuario
Precondición	<p>1. La aplicación se encuentra iniciada.</p> <p>2. El Usuario ha guardado algún perfil de paciente en el sistema.</p>
Secuencia normal	<p>1. Se ejecuta el caso de uso <i>Visualizar_Perfiles_Pacientes</i>.</p> <p>2. El Usuario pulsa sobre uno de los perfiles de pacientes del listado.</p> <p>3. El sistema muestra toda la información almacenada en el sistema relativa al paciente (el detalle de la misma ha sido detallada en el caso de uso <i>Crear_Perfil_Paciente</i>), además de un listado con todas las mediciones que se han almacenado con relación al paciente.</p> <p>4. El caso de uso finaliza.</p>
Postcondición	Ninguna.
Flujo alternativo	No existe.
Excepciones	No existen.
Frecuencia	Alta

Tabla 19 - Especificación UC-0007

UC-0008	Borrar_Perfil_Paciente
Descripción	Un Usuario desea borrar un perfil de un paciente.
Actor	Usuario
Precondición	<p>1. La aplicación se encuentra iniciada.</p> <p>2. El Usuario ha guardado algún perfil de paciente en el sistema.</p>
Secuencia normal	1. Se ejecuta el caso de uso <i>Visualizar_Informacion_Paciente</i> .

	<p>2. El Usuario pulsa sobre el botón para eliminar el perfil del paciente.</p> <p>3. El sistema muestra un mensaje para confirmar el borrado del perfil.</p> <p>4. El Usuario confirma que desea eliminar el perfil.</p> <p>5. El sistema borra el perfil del paciente junto con todas las mediciones que se han almacenado con relación al mismo, muestra un mensaje al usuario indicando que el paciente ha sido eliminado con éxito y muestra la lista de perfiles de pacientes almacenados en el mismo con el perfil del paciente eliminado.</p> <p>6. El caso de uso finaliza.</p>
Postcondición	El perfil del paciente ha sido eliminado del sistema.
Flujo alternativo	4.1. Si el Usuario decide cancelar el borrado del perfil del paciente, el caso de uso queda sin efecto y finaliza en el paso 6.
Excepciones	5.1. Si ocurre algún problema a la hora de eliminar el perfil del paciente de la Base de Datos, el sistema muestra un mensaje de error indicando esta condición al usuario y el caso de uso continúa en el paso 1.
Frecuencia	Muy baja

Tabla 20 - Especificación UC-0008

5.3. Realización de los Casos de Uso en Análisis

5.3.1. Diagramas de actividad

5.3.1.1. UC-0001: Añadir_Medicion

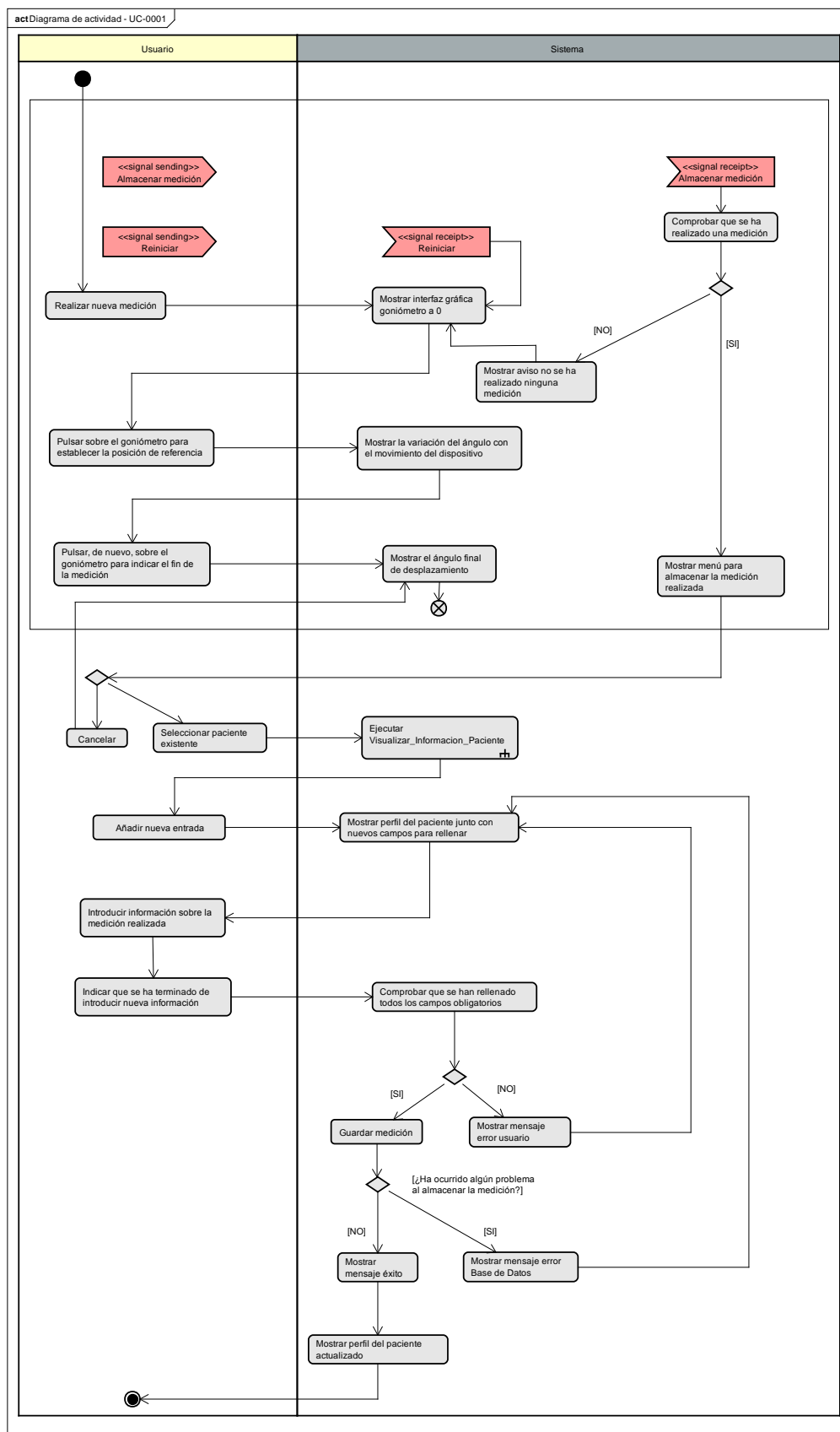


Figura 9 - Diagrama de actividad UC-0001

5.3.1.2. UC-0002: Visualizar_10_Ultimas_Mediciones

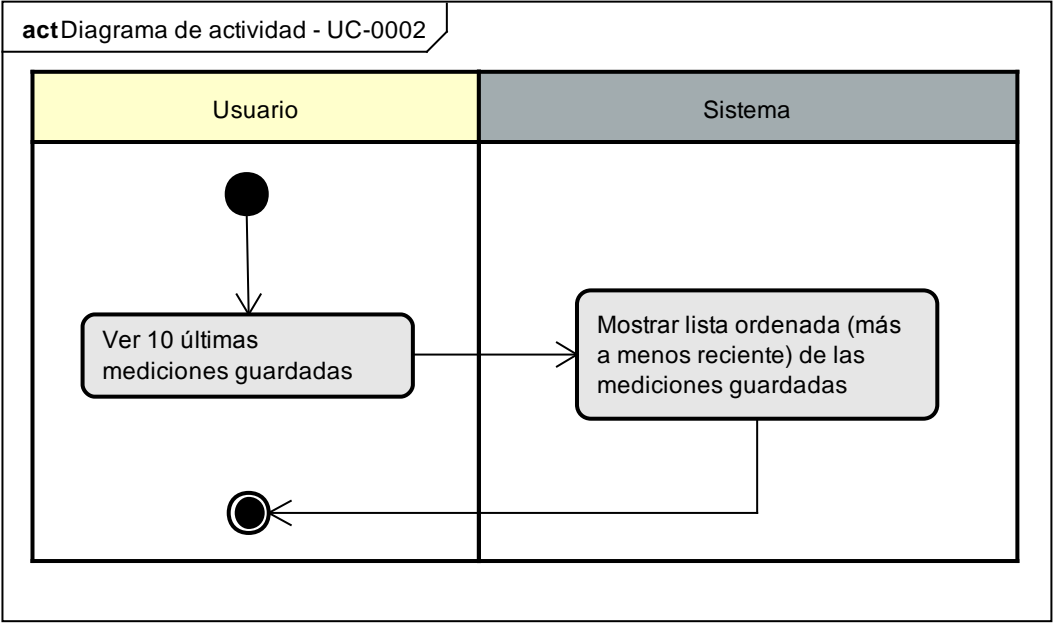


Figura 10 - Diagrama de actividad UC-0002

5.3.1.3. UC-0003: Borrar_Medicion

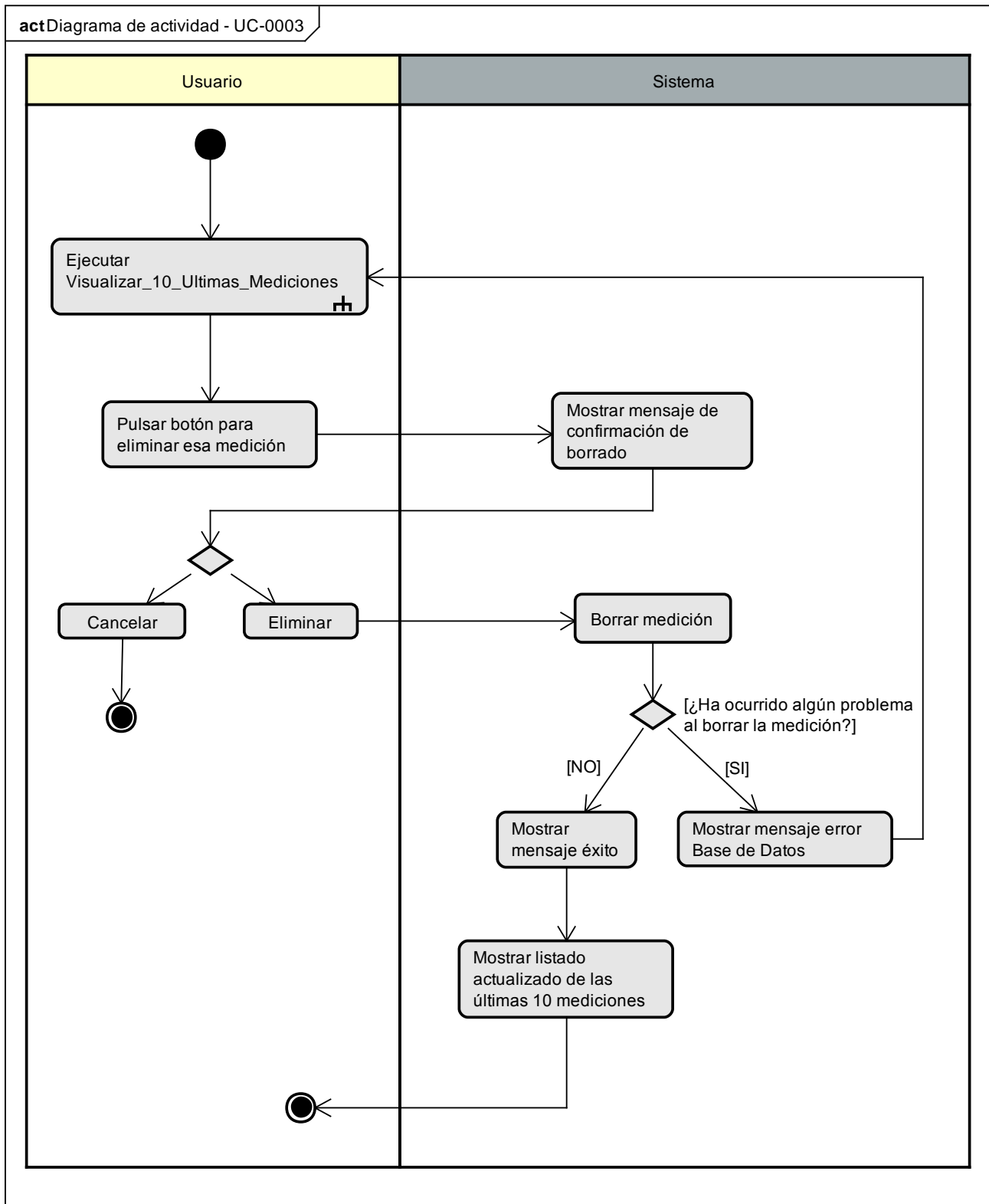


Figura 11 - Diagrama de actividad UC-0003

5.3.1.4. UC-0004: Visualizar_Perfiles_Pacientes

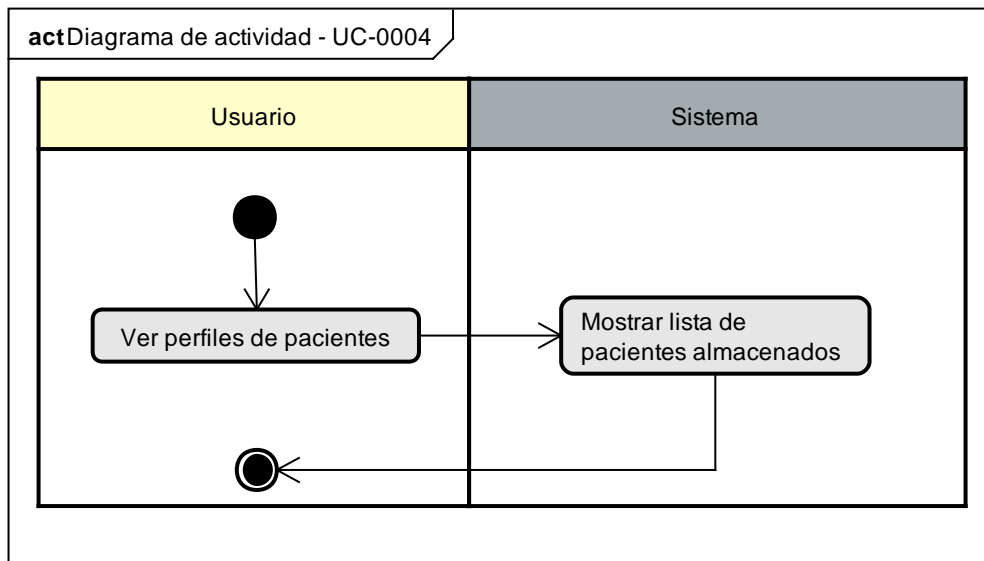


Figura 12 - Diagrama de actividad UC-0004

5.3.1.5. UC-0005: Filtrar_Pacientes

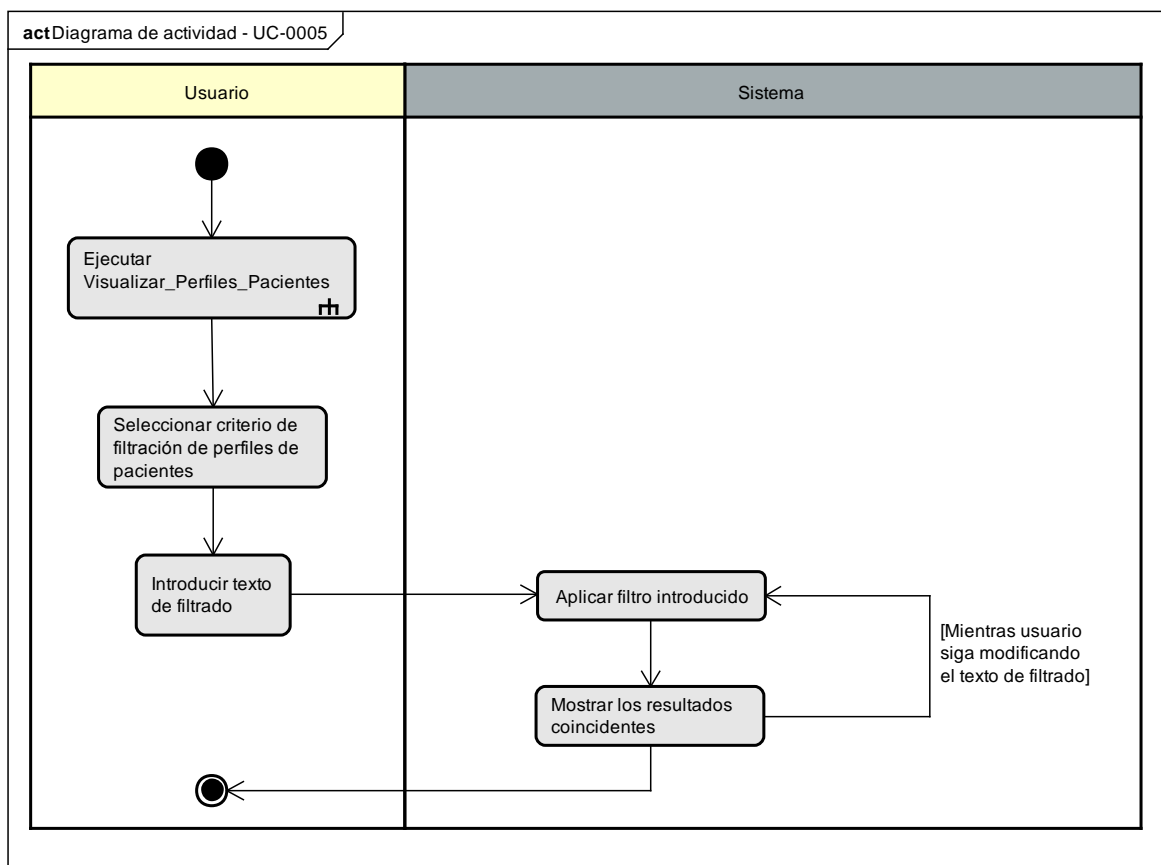


Figura 13 - Diagrama de actividad UC-0005

5.3.1.6. UC-0006: Crear_Perfil_Paciente

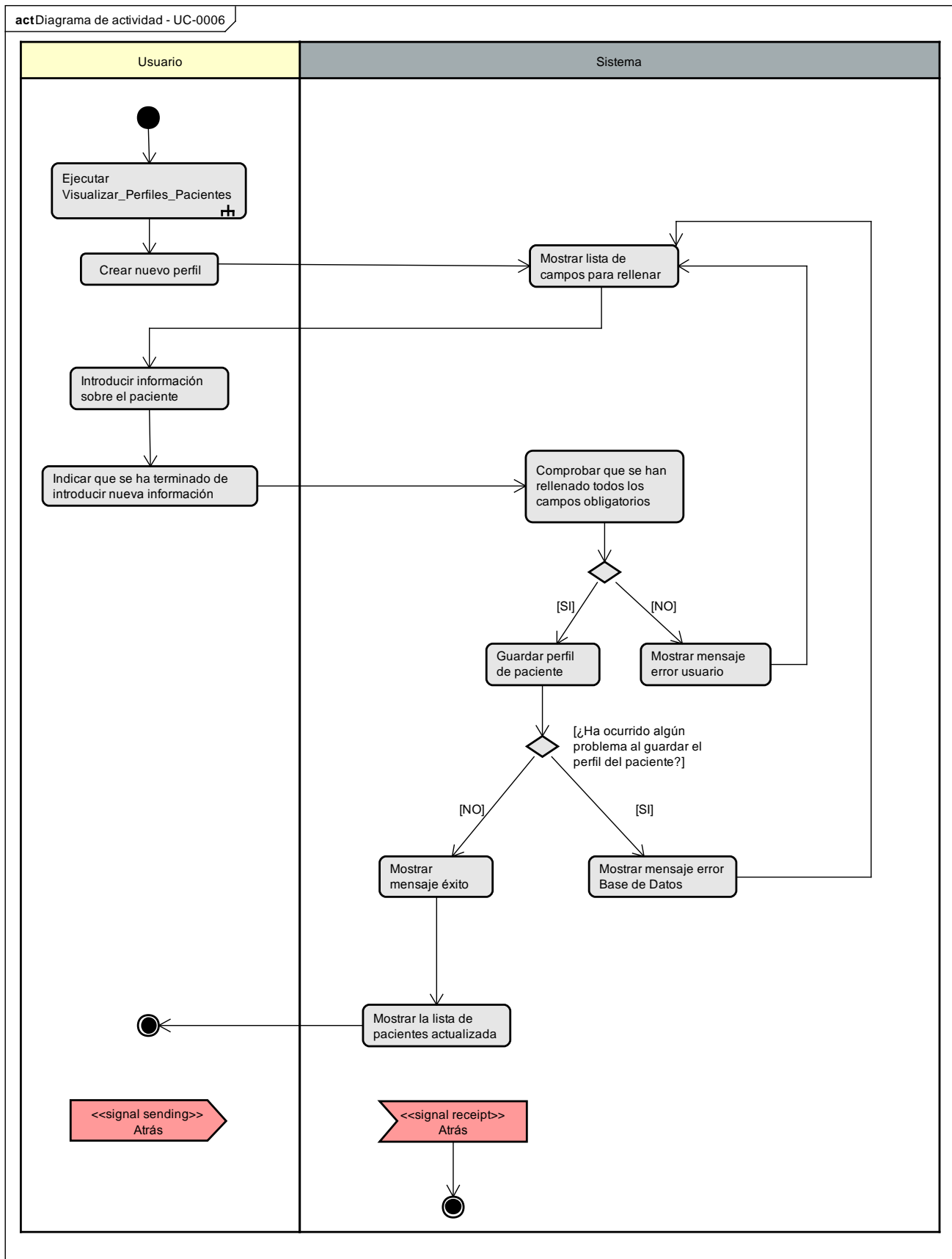


Figura 14 - Diagrama de actividad UC-0006

5.3.1.7. UC-0007: Visualizar_Informacion_Paciente

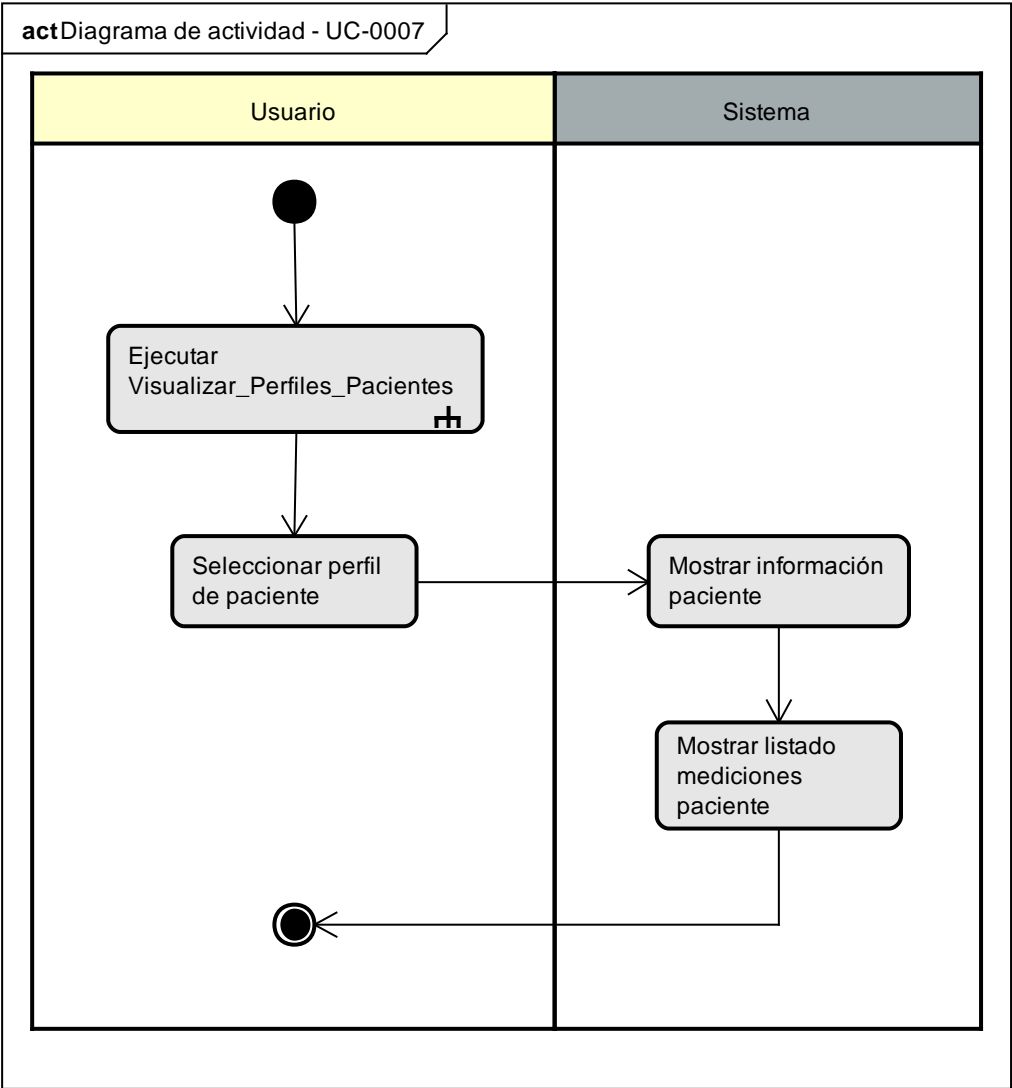


Figura 15 - Diagrama de actividad UC-0007

5.3.1.8. UC-0008: Borrar_Perfil_Paciente

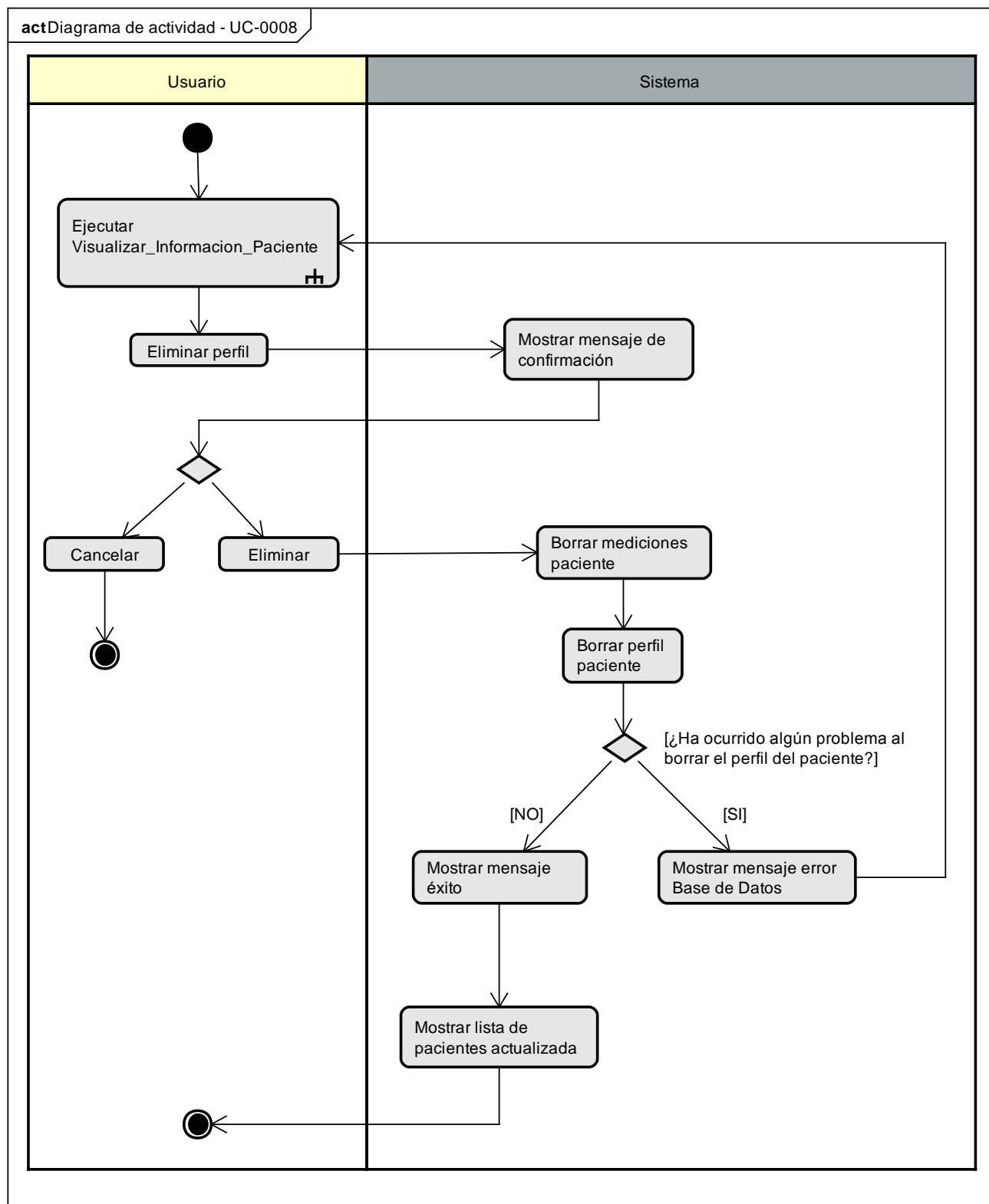


Figura 16 - Diagrama de actividad UC-0008

5.3.2. Modelo de dominio

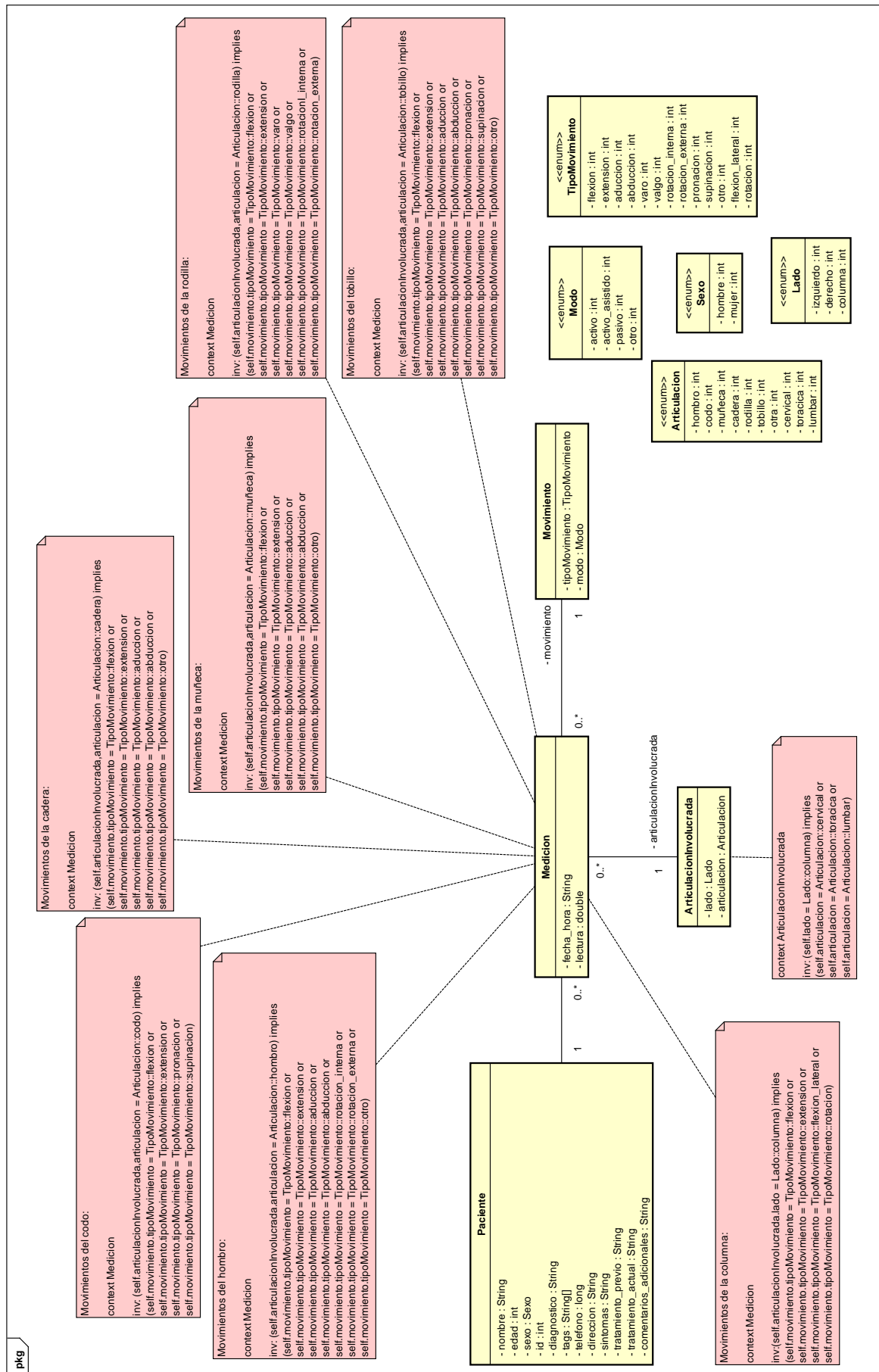


Figura 17 - Modelo de dominio

5.3.3. Descripción de las clases del modelo de dominio

5.3.3.1. Paciente

- **Descripción:** Clase que modela un paciente (perfil).
- **Responsabilidades:** Referenciar el perfil de un paciente dentro de la aplicación.
- **Atributos:**
 - nombre: Nombre del paciente.
 - edad: Edad del paciente.
 - sexo: Sexo del paciente. Los posibles valores de este atributo están especificados en la clase `<<Enum>> Sexo`.
 - id: Identificador numérico del paciente.
 - diagnostico: Diagnóstico del paciente.
 - tags: Lista de etiquetas (tags) que describen el problema del paciente.
 - telefono: Número de teléfono del paciente.
 - direccion: Dirección del paciente.
 - sintomas: Síntomas que tiene el paciente.
 - tratamiento_previo: Tratamiento que ha estado siguiendo el paciente.
 - tratamiento_actual: Tratamiento que está siguiendo el paciente en estos momentos.
 - comentarios_adicionales: Cualquier comentario extra que deba ser añadido sobre el paciente.

5.3.3.2. Medicion

- **Descripción:** Clase que modela una medición realizada por el usuario.
- **Responsabilidades:** Referenciar una medición realizada por el usuario dentro de la aplicación.
- **Atributos:**
 - fecha_hora: Fecha y hora en la que se realizó la medición.
 - lectura: Medición que se ha realizado con el dispositivo empleado como un goniómetro.

5.3.3.3. ArticulacionInvolucrada

- **Descripción:** Clase que modela la articulación involucrada en una medición.
- **Responsabilidades:** Referenciar la articulación involucrada en una medición dentro de la aplicación.
- **Atributos:**
 - lado: Lado del cuerpo en el que se encuentra la articulación involucrada en la medición con respecto al plano sagital, o si se trata de la columna vertebral (ver apartado 4.2.3 para consultar la definición de *plano sagital*). Los posibles valores de este atributo están especificados en la clase `<<Enum>> Lado`.
 - articulacion: La articulación en concreto. Las posibles articulaciones que introducir están especificadas en la clase `<<Enum>> Articulacion`.

5.3.3.4. Movimiento

- **Descripción:** Clase que modela el movimiento realizado por la articulación en una medición.

- **Responsabilidades:** Referenciar, dentro de la aplicación, el movimiento de la articulación medido.
- **Atributos:**
 - tipo: El tipo de movimiento en particular que se ha producido. Los posibles movimientos están especificados en la clase <<Enum>> *TipoMovimiento*.
 - modo: El modo en el que se ha producido este movimiento. Los posibles modos están especificados en la clase <<Enum>> *Modo*.

5.3.3.5. <<Enum>> *Sexo*

- **Descripción:** Clase enumerada que modela el sexo de los pacientes.
- **Posibles valores:**
 - hombre
 - mujer
- **Comentarios:** N/A

5.3.3.6. <<Enum>> *Lado*

- **Descripción:** Clase enumerada que modela el lado en el que se encuentra la articulación involucrada en una medición (con respecto al plano sagital), o bien, si se trata de la columna vertebral.
- **Posibles valores:**
 - izquierdo
 - derecho
 - columna
- **Comentarios:** N/A

5.3.3.7. <<Enum>> *Articulacion*

- **Descripción:** Clase enumerada que modela la articulación en particular involucrada en una medición.
- **Posibles valores:**
 - hombro
 - codo
 - muñeca
 - cadera
 - rodilla
 - tobillo
 - otra
 - cervical
 - toracica
 - lumbar
- **Comentarios:** Los valores *cervical*, *toracica* y *lumbar* sólo se pueden dar cuando en *Lado* se ha seleccionado el valor *columna*.

5.3.3.8. <<Enum>> *TipoMovimiento*

- **Descripción:** Clase enumerada que modela los posibles movimientos que puede realizar una articulación.
- **Posibles valores:**
 - flexion
 - extension
 - aduccion
 - abduccion
 - varo
 - valgo
 - rotacion_interna
 - rotacion_externa
 - pronacion
 - supinacion
 - otro
 - flexion_lateral
 - rotacion
- **Comentarios:** Los posibles valores de *TipoMovimiento* están restringidos en función de la articulación que se haya seleccionado en *Articulacion*, de acuerdo con los movimientos que puede realizar cada articulación. Estos movimientos están especificados en OCL en el *Modelo de Dominio* (apartado 5.3.2).

5.3.3.9. <<Enum>> *Modo*

- **Descripción:** Clase enumerada que modela los modos de movimiento de una articulación.
- **Posibles valores:**
 - activo
 - activo_asistido
 - pasivo
 - otro
- **Comentarios:** N/A

Capítulo 6

Diseño

6.1. Diseño de la arquitectura

6.1.1. Patrón arquitectónico utilizado: MVP (Modelo Vista Presentador)

Este primer apartado referente al diseño de la aplicación se va a centrar en explicar este patrón que se ha utilizado, aplicado al desarrollo de aplicaciones Android.

6.1.1.1. Arquitectura Android

Las herramientas ofrecidas por Android, como *layouts*, *Activities* y estructuras de datos, parece que tratan de guiar a los desarrolladores hacia el patrón *Modelo Vista Controlador (MVC)*. Se trata de un patrón bastante robusto y asentado en el desarrollo de aplicaciones Android, que busca aislar los diferentes roles dentro de una aplicación. Esto es lo que se conoce como *Separación de intereses*.

Esta arquitectura (MVC) crea tres capas:

- Modelo
- Vista
- Controlador

Cada capa es responsable de un aspecto de la aplicación. *Modelo* responde a la lógica de negocio, *Vista* a la interfaz de usuario y *Controlador* se encarga de gestionar el acceso de la *Vista* al *Modelo*.

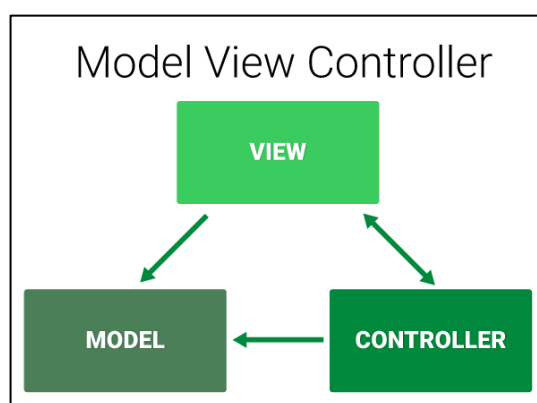


Figura 18 - Patrón MVC (Modelo-Vista-Controlador)

Pero, si se analiza más de cerca la arquitectura Android, especialmente la relación entre *Vista* (*Activities*, *Fragments*, etc.) y *Modelo* (estructuras de datos), se puede concluir que no se trata de MVC.

El problema radica en la conexión simbiótica que se da entre *Loaders* y *Activities/Fragments*. Una *Activity* o *Fragment* es responsable de llamar al *Loader*, el cual se encarga de buscar los datos y devolverlos. Su existencia

está completamente ligada y no hay una separación entre el rol de la *Vista (Activity/Fragment)* y la lógica del negocio (llevada a cabo por el *Loader*).

Este problema complica enormemente la realización de tests unitarios (debido, precisamente, al acoplamiento entre el *Loader* y la *Vista (Activity/Fragment)*).

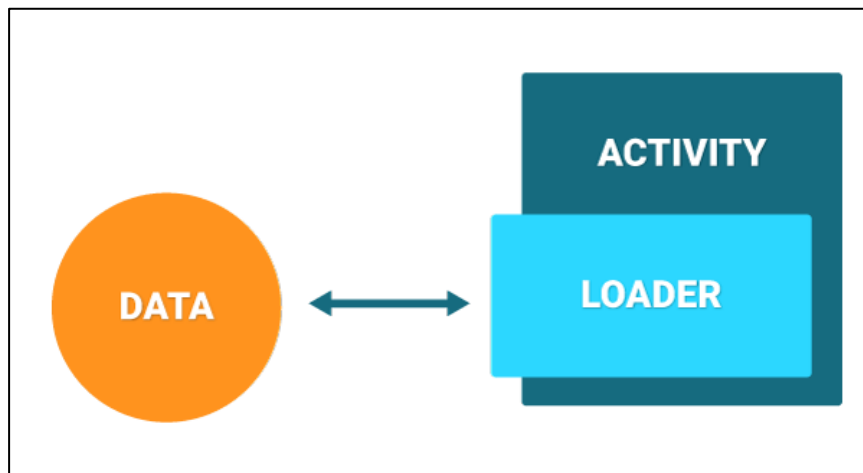


Figura 19 - Obtención de datos por parte de la Vista (Activity/Fragment)

En este contexto, y a pesar de que el patrón MVC es una buena opción, una idea mejor es utilizar el patrón MVP (*Modelo Vista Presentador*). El patrón MVP fue desarrollado bajo las mismas premisas que el MVC, pero siguiendo un paradigma más moderno que crea una mejor separación de intereses y mejora la testeabilidad de la aplicación.

6.1.1.2. Modelo Vista Presentador en Android

Como se ha mencionado en el apartado anterior, la separación de intereses no es el punto fuerte de Android. Afortunadamente, el Modelo Vista Presentador lo mejora significativamente. MVP divide la aplicación en tres capas:

- Modelo
- Vista
- Presentador

Cada una de ellas tiene sus responsabilidades y la comunicación entre estas capas está gestionada por el *Presentador*. El Presentador trabaja como mediador entre las diferentes partes.

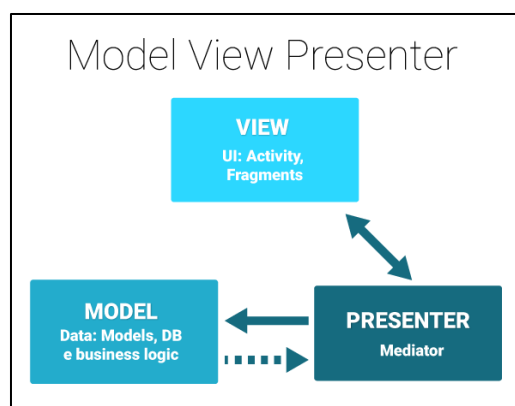


Figura 20 - Patrón MVP (Modelo-Vista-Presentador)

- El Modelo se ocupa de la lógica del negocio de la aplicación. Controla cómo se crean, almacenan y modifican los datos.
- La Vista es una interfaz pasiva que muestra los datos y comunica las acciones del usuario al Presentador.
- El Presentador actúa como intermediario. Recibe los datos del Modelo y los muestra en la Vista. También se encarga de procesar las acciones del usuario comunicadas por la Vista.

6.1.1.3. Diferencias entre MVC y MVP

El patrón Modelo Vista Presentador está basado en el patrón Modelo Vista Controlador, y, dado que comparten algunas características, puede resultar complicado diferenciarlos. El Presentador y el Controlador tienen un rol similar. Ambos son responsables de la comunicación entre el Modelo y la Vista. Dicho de otra manera, el Controlador no gestiona el Modelo y la Vista de manera tan estricta como lo hace el Presentador.

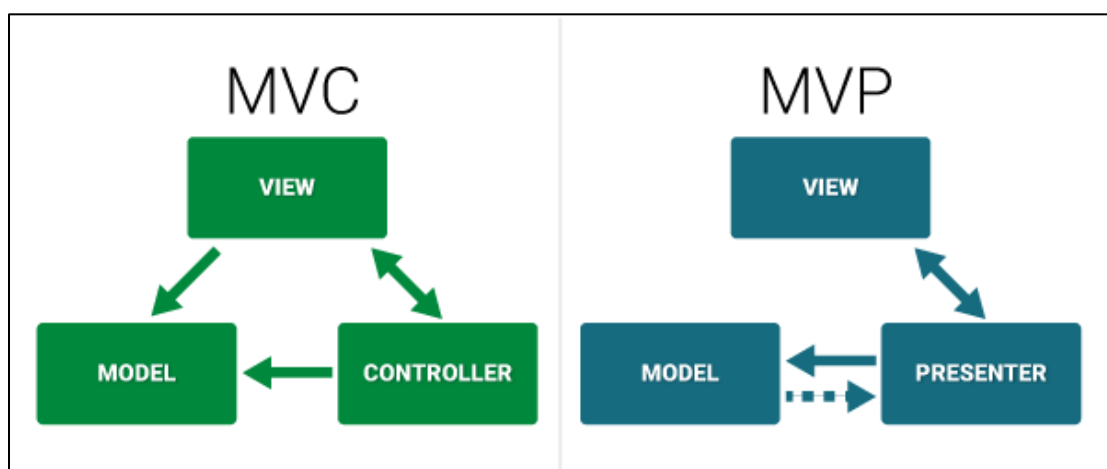


Figura 21 - Diferencias entre MVC y MVP

En el patrón MVC, la Vista puede recuperar datos directamente del Modelo. En el patrón MVP, la Vista es completamente pasiva y los datos se envían siempre a la Vista a través del Presentador. Los Controladores en MVC pueden, además, ser compartidos por múltiples Vistas. En MVP, la Vista y el Presentador tienen una relación 1-a-1, es decir, el Presentador está ligado a una Vista.

En resumen:

- En MVP, la Vista no tiene acceso al Modelo.
- El Presentador está ligado a una única Vista.
- La Vista es completamente pasiva en el patrón MVP.

Estas diferencias conceptuales hacen que el patrón MVP garantice una mejor separación de intereses y aumente considerablemente la testeabilidad de la aplicación promoviendo una mayor separación entre estas tres capas centrales.

6.1.1.4. Activity, Fragment y objetos View

Existen varias interpretaciones de cómo el patrón MVP se puede implementar en Android. En general, las Activities y Fragments asumen el rol de Vista y son responsables de la creación del Presentador y el Modelo. La

Vista también es responsable de mantener la consistencia del Modelo y el Presentador entre cambios en la configuración, informándolos sobre la eventual destrucción de la Vista.

Otros objetos View, como los RecyclerView se pueden considerar también parte de la capa Vista en MVP. Existe una relación 1-a-1 entre la Vista y el Presentador, pero, en situaciones complejas, es posible que se requieran de múltiples Presentadores.

6.1.2. Patrones de diseño utilizados

6.1.2.1. Patrón Builder

El patrón *Builder* es un patrón de diseño creacional cuya finalidad es separar la construcción de un objeto complejo de su representación, de tal manera, que el mismo proceso de construcción puede crear diferentes representaciones.

La estructura de este patrón es la siguiente:

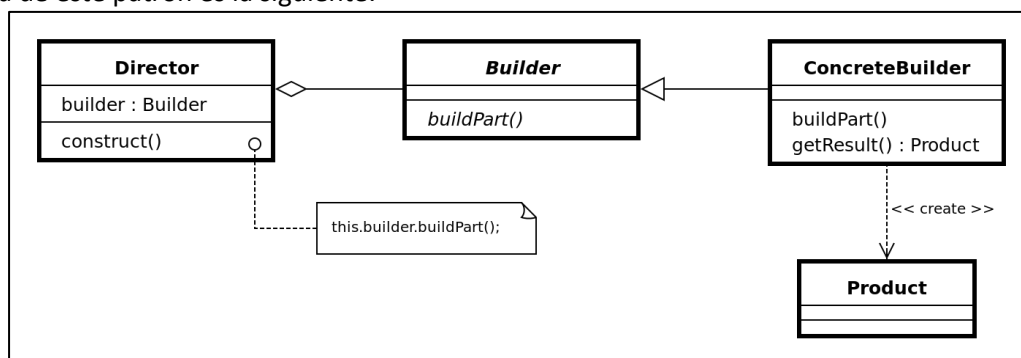


Figura 22 - Estructura patrón Builder

La utilización de este patrón tiene varias ventajas, que son las siguientes:

- Reduce el acoplamiento.
- Permite variar la representación interna de estructuras complejas, respetando la interfaz común de la clase *Builder*.
- Se logra una separación entre la construcción y la representación. Las clases concretas que tratan las representaciones internas no forman parte de la interfaz *Builder*.
- Cada *ConcreteBuilder* tiene el código específico para crear y modificar una estructura interna concreta.
- Distintos *Director* con distintas utilidades (visores, parsers, etc.) pueden utilizar el mismo *ConcreteBuilder*.
- Permite un mayor control en el proceso de creación del objeto. El *Director* controla la creación paso-a-paso; sólo cuando el *Builder* ha terminado de construir el objeto, lo recupera el *Director*.

En el caso de Android, este patrón aparece (entre otras ocasiones) cuando se utilizan objetos como *AlertDialog.Builder*. Este constructor sigue un proceso paso-a-paso y permite especificar solo aquellas partes del objeto *AlertDialog* que se necesiten.

Para el desarrollo de esta aplicación, es precisamente donde se ha utilizado este patrón de diseño. En concreto se emplea para mostrar un objeto de tipo *AlertDialog* cuando el usuario intenta almacenar una medición antes de haber hecho una lectura con el goniómetro, y cuando el usuario intenta eliminar el perfil de

un paciente o una medición; en el primer caso, para avisar al usuario de esa situación incorrecta, y en el segundo caso con el fin de confirmar que el usuario desea realizar esa acción.

6.1.2.2. Patrón Singleton

El patrón *Singleton* es un patrón de diseño creacional cuya finalidad es asegurarse que una clase tiene una sola instancia, y proporcionar un único punto de acceso global a ella.

Las situaciones más habituales de aplicación de este patrón son aquellas en las que dicha clase controla el acceso a un recurso físico único (como puede ser el ratón o una Base de Datos) o cuando cierto tipo de datos debe estar disponible para todos los demás objetos de la aplicación.

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.

La estructura de este patrón es la siguiente:

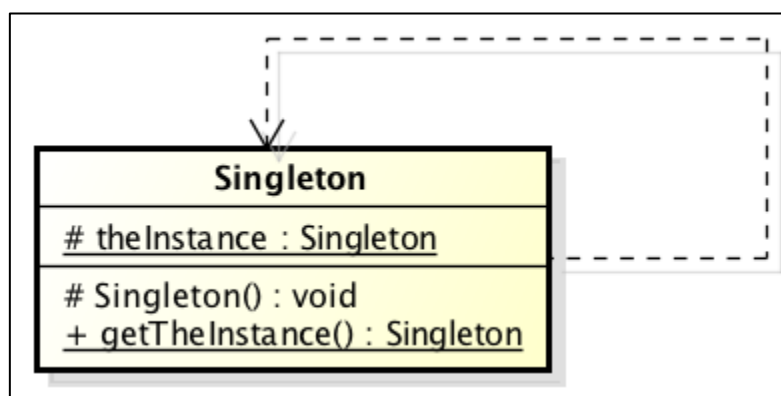


Figura 23 - Estructura patrón Singleton

Para el desarrollo de esta aplicación, este patrón se ha utilizado para disponer de una única instancia del Modelo (clase encargada de acceder y modificar la Base de Datos de la aplicación), y así, controlar que sólo esta clase es la que interactúa con la Base de Datos (a través de la clase *Helper* correspondiente).

6.1.2.3. Patrón Inyección de dependencias

La *inyección de dependencias* (en inglés Dependency Injection, DI) es un patrón de diseño creacional en el que se suministran objetos a una clase en lugar de ser la propia clase la que cree el objeto.

Esto no es más que extraer responsabilidades a un componente para delegarlas en otro, estableciendo un mecanismo a través del cual el nuevo componente pueda ser cambiado en tiempo de ejecución.

Entre los beneficios de utilizar la inyección de dependencias se pueden encontrar:

- **Reducción de las dependencias:** Mediante este patrón se pueden eliminar o, al menos reducir, las dependencias innecesarias de los componentes. Un componente es vulnerable a los cambios en sus

dependencias. Si una dependencia cambia, es probable que el componente deba adaptarse a esos cambios; sin embargo, gracias a este patrón de diseño, al reducir (al menos) las dependencias de un componente, se logra que éste sea menos vulnerable frente a cambios en las mismas.

- **Mayor reutilización:** Como consecuencia del punto anterior, al reducir las dependencias de un componente, también se logra que éste sea más fácilmente reutilizable en un contexto diferente. El hecho de que las dependencias puedan ser inyectadas y, por ende, configuradas de forma externa, incrementan la reutilización de ese componente.
- **Mayor testeabilidad:** Cuando las dependencias pueden ser inyectadas en un componente, es posible inyectar implementaciones de prueba (en inglés, mock) de estas dependencias. Los objetos mock son utilizados para realizar pruebas en sustitución de objetos “reales”. Estos objetos pueden ser configurados como se quiera, lo que permite realizar todo tipo de pruebas en el componente.
- **Mayor legibilidad del código:** La inyección de dependencias traslada las dependencias a la interfaz del componente, de esta manera es más sencillo ver las dependencias que posee éste, lo que se traduce en una mayor legibilidad del código.
- **Reduce el transporte de dependencias:** El transporte de dependencias se da cuando un objeto recibe un parámetro que él mismo no necesita, pero que lo necesita uno de los objetos a los que invoca, por lo que es necesario enviar este parámetro a pesar de no utilizarlo en ningún momento.

A modo de ejemplo, suponer que un programa tiene 4 componentes (**A**, **B**, **C** y **D**). El componente **A** crea un objeto de configuración que sólo necesita el componente **D**; sin embargo, para que éste objeto llegue desde **A** hasta **D** tiene que pasar por **B** y **C** (a pesar de que estos dos componentes no necesitan este objeto). La cadena de llamadas sería algo así: **A** (crea el objeto de configuración) → **B** → **C** → **D** (utiliza el objeto de configuración). Esto es lo que se conoce como transporte de dependencias.

Este problema se soluciona utilizando un contenedor de inyección de dependencias. El contenedor conoce a todos los componentes de la aplicación, por lo que, puede conectarlos perfectamente sin tener que pasar las dependencias de un componente a otro.

En el contexto de la aplicación, se ha intentado seguir lo más fielmente posible el patrón arquitectónico *MVP* (Modelo-Vista-Presentador), por lo que la Vista (Activity, Fragments y Views) debía ser completamente pasiva, es decir, ni siquiera debía encargarse de crear su propio Presentador. Para esta tarea se ha decidido utilizar este patrón de diseño de inyección de dependencias, quitando, de esta manera, la responsabilidad de crear el Presentador a la Vista.

En este caso, se ha decidido implementar este patrón mediante el framework *Dagger2*. En el apartado 7.1.1.1 (Implementación y Pruebas) se explica en detalle en qué consiste *Dagger2*.

6.1.2.4. Patrón Adaptador

El patrón *Adaptador* es un patrón de diseño estructural cuya finalidad es convertir la interfaz de una clase en otra esperada por los clientes, es decir, permitir que clases con interfaces incompatibles puedan comunicarse.

Este patrón se puede utilizar bien cuando se quiera utilizar una clase ya existente y su interfaz no se corresponda con la interfaz que se necesita, o bien, cuando se quiera envolver código no orientado a objeto con forma de clase.

Las ventajas de la utilización de este patrón son:

- Una misma clase adaptadora puede adaptar a una clase adaptada y a todas sus subclases.
- La implementación es directa. Sólo hay que tener en cuenta de cuál de las alternativas se trata (utilizar una clase ya existente o envolver código no orientado a objetos).

La principal desventaja de este patrón es:

- La adaptación no siempre consiste en cambiar el nombre y/o los parámetros de las operaciones, sino que puede ser más complejo que eso.

La estructura de este patrón es la siguiente:

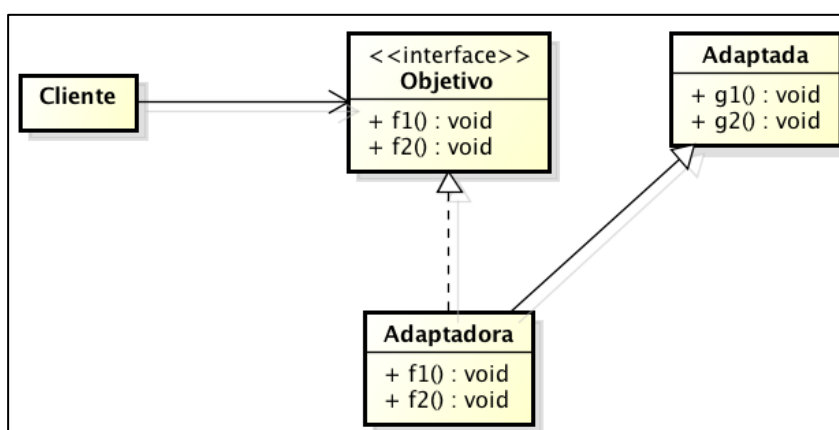


Figura 24 - Estructura patrón Adaptador

Para el desarrollo de esta aplicación, este patrón se ha utilizado cuando se han utilizado los elementos *RecyclerView* para mostrar el listado de pacientes, y el listado de las últimas 10 mediciones almacenadas en el sistema.

La estructura de clases de los elementos *RecyclerView* es la siguiente:

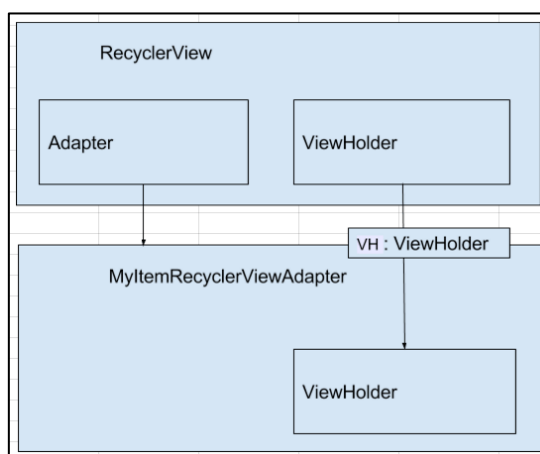


Figura 25 - Estructura elementos RecyclerView - Android

Siguiendo este esquema, se han definido dos clases para este propósito, *PacienteListAdapter* y *MedicionListAdapter* (ambas heredan de *RecyclerView.Adapter*).

6.1.3. Arquitectura general del proyecto

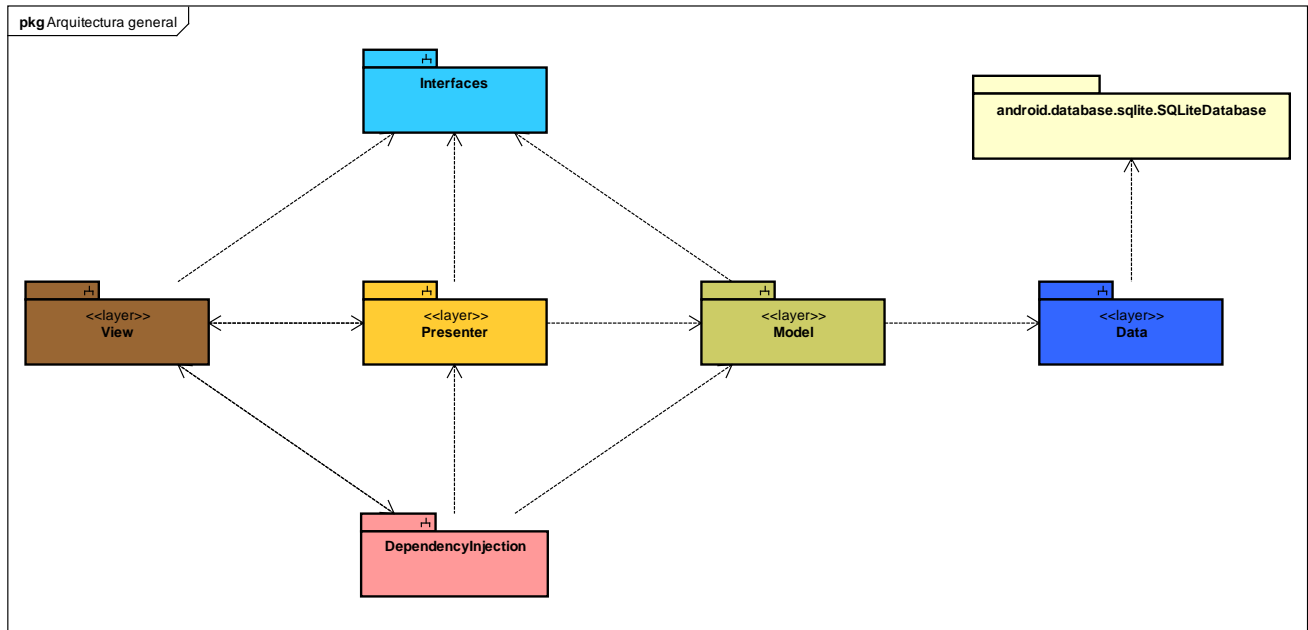


Figura 26 - Arquitectura general del proyecto

6.1.3.1. Descripción de la arquitectura general del proyecto

Para el desarrollo de esta aplicación, como se puede apreciar, se ha realizado una arquitectura con cuatro capas:

- **View:** Capa correspondiente a la *Vista* en el patrón MVP.
- **Presenter:** Capa correspondiente al *Presentador* en el patrón MVP.
- **Model:** Capa correspondiente al *Modelo* en el patrón MVP.
- **Data:** Capa de acceso a la Base de Datos.

En este caso, al utilizar el patrón MVP, la lógica operacional de la aplicación debería realizarse en el *Modelo*, y que el *Presentador* simplemente se encargara de hacer de intermediario entre la *Vista* y el *Modelo*; sin embargo, al desarrollar una aplicación en Android aplicando este principio, se observó que el código se complicaba en exceso, así como algunas dependencias entre las distintas capas de la aplicación; por lo que, finalmente, se ha optado por dejar la lógica de la aplicación en la capa *Presenter* y que la capa *Model* contenga únicamente el dominio de la aplicación.

Además de las cuatro capas explicadas, se han representado dos paquetes extras:

- **Interfaces:** En este paquete se incluyen las interfaces correspondientes a las capas *View*, *Presenter* y *Model*, donde se recogen todos los métodos públicos que emplean para comunicarse estas tres capas entre ellas.
- **DependencyInjection:** En este paquete se incluyen las clases necesarias para implementar la inyección de dependencias en la aplicación mediante el uso de la biblioteca *Dagger2* (como se ha explicado previamente).

Por último, también se representa la dependencia de la capa de acceso a datos con una base de datos SQLite.

6.1.4. Descomposición modular del proyecto

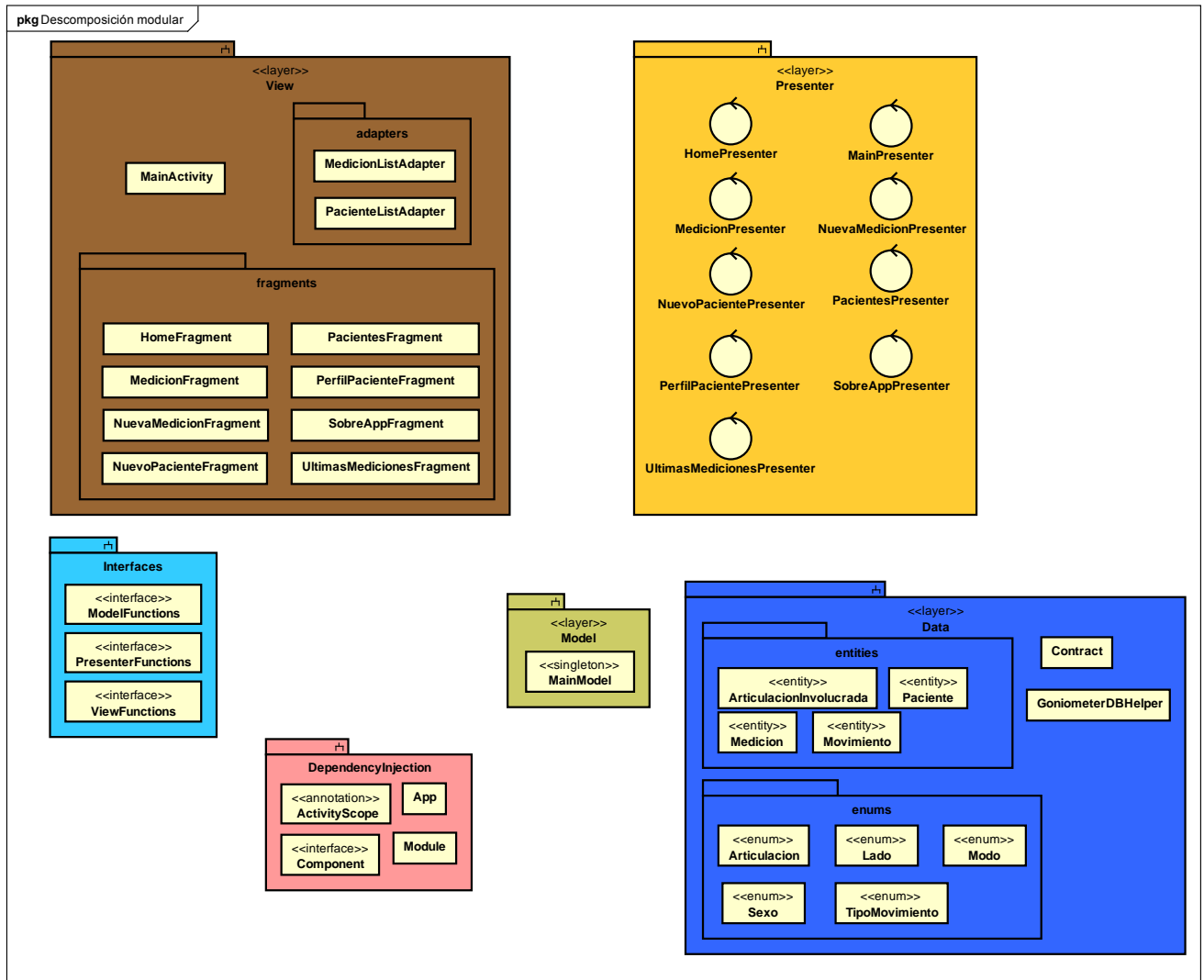


Figura 27 - Descomposición modular del proyecto

6.1.4.1. Descripción de la descomposición modular del proyecto

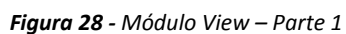
El objetivo de este diagrama es mostrar a un nivel un poco más alto, la descomposición de cada uno de los módulos, atendiendo a los paquetes, clases e interfaces que componen cada uno de ellos.

En los próximos apartados, se detallará cada uno de estos módulos, con las relaciones que existen tanto dentro de cada uno de ellos, como de las relaciones entre ellos.

6.1.5. Diseño detallado de los módulos del proyecto

En este primer apartado, se van a detallar cada uno de los módulos con las relaciones existentes entre las clases dentro de los mismos.

54



6.1.5.2. Módulo *View* – Parte 2

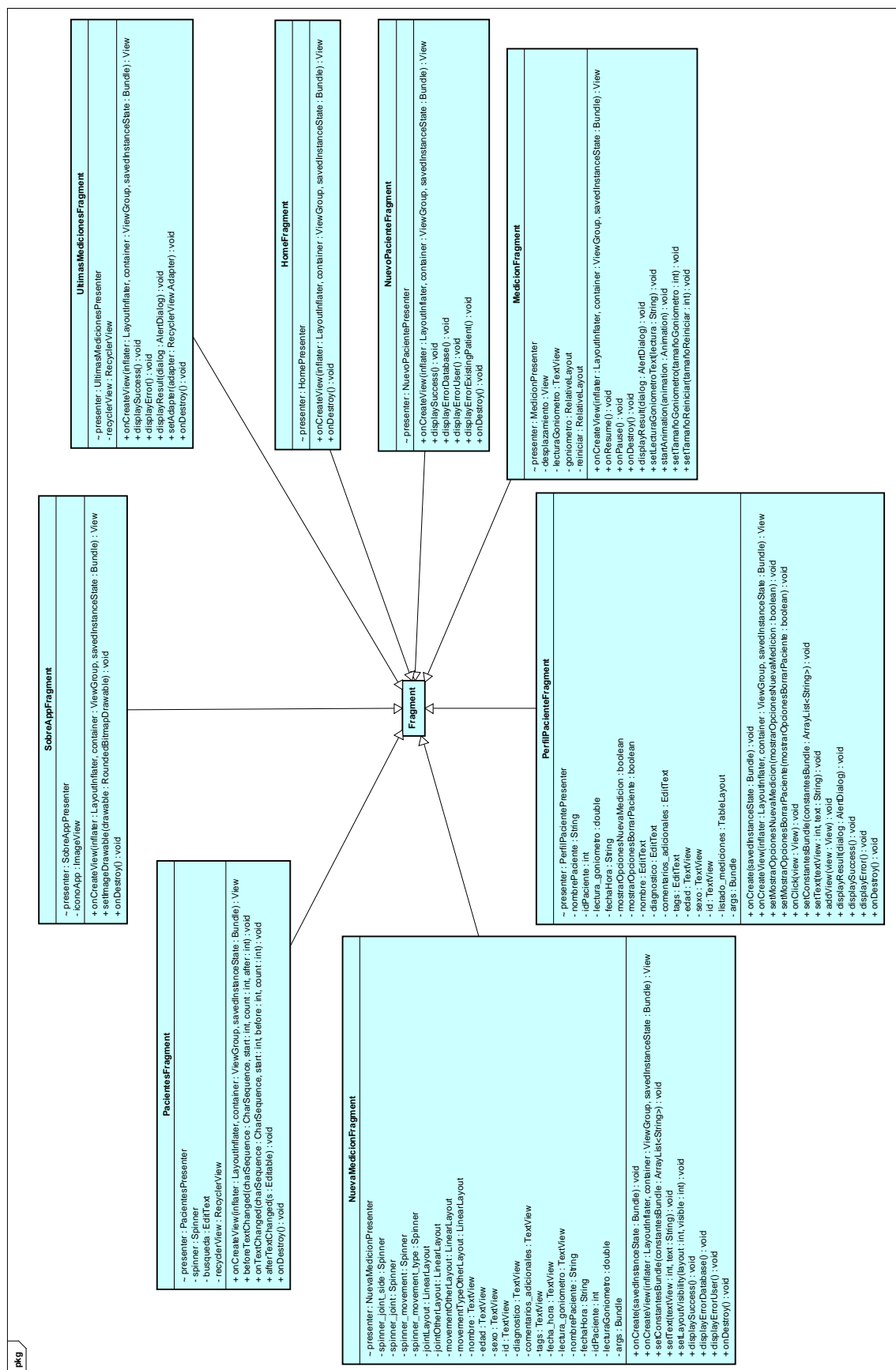


Figura 29 - Módulo View - Parte 2

6.1.5.3. Módulo Presenter – Parte 1

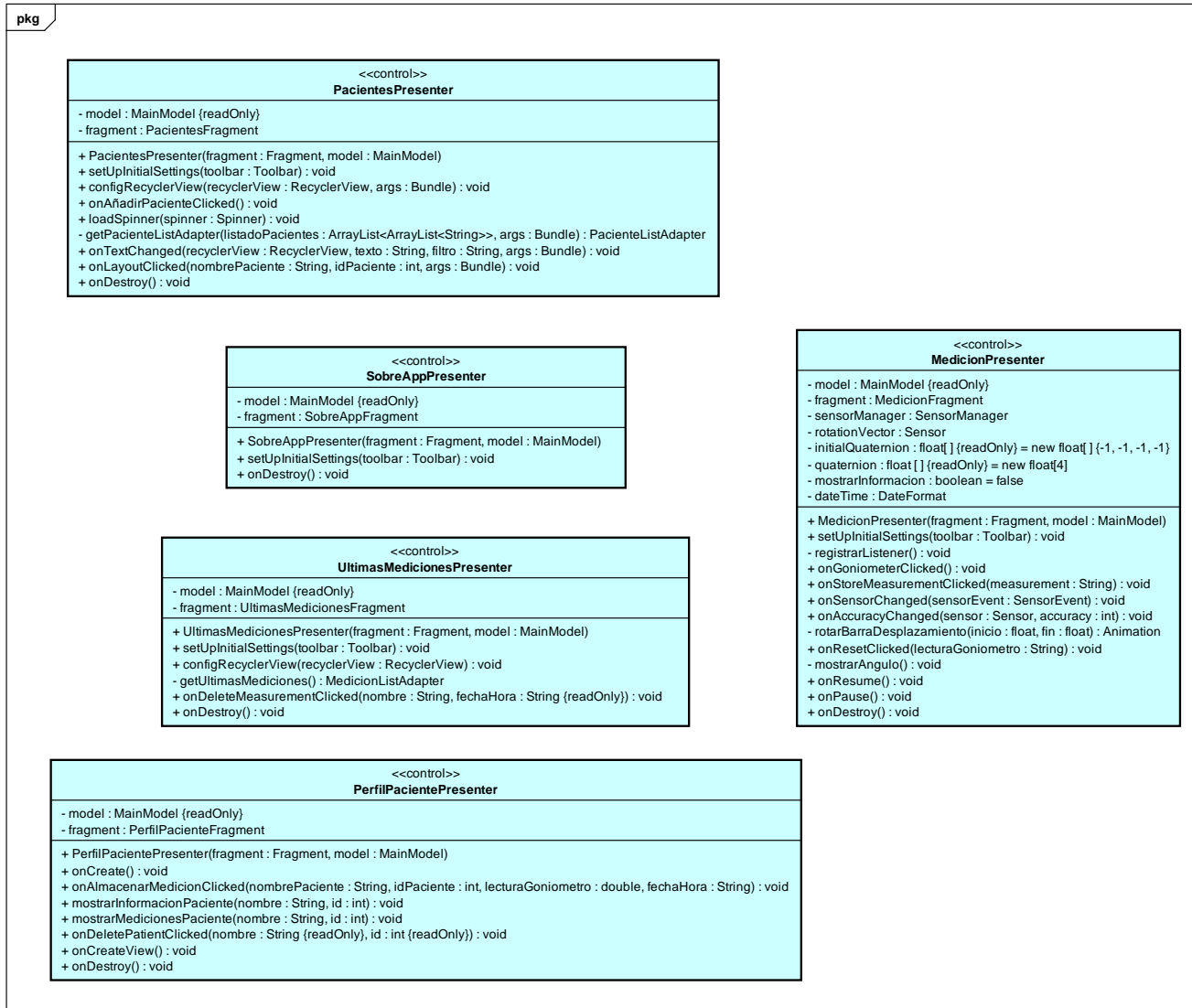
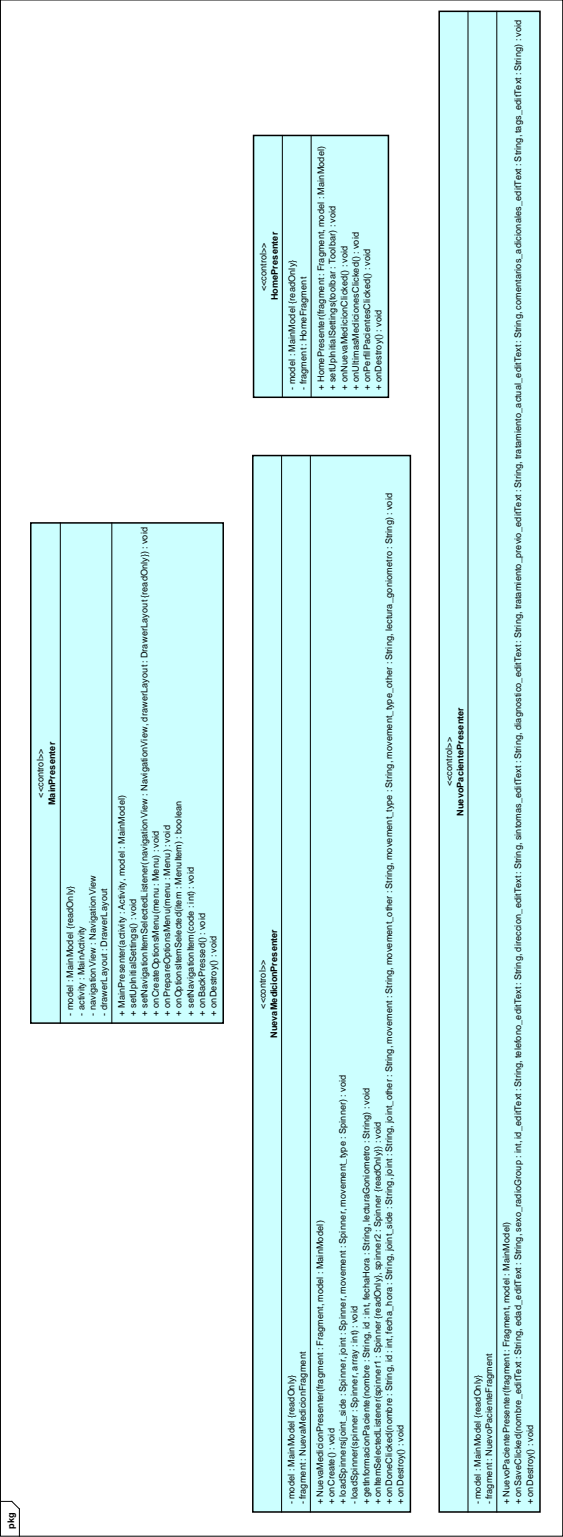


Figura 30 - Módulo Presenter

6.1.5.4. Módulo *Presenter* – Parte 2



6.1.5.5. Módulo *Model*

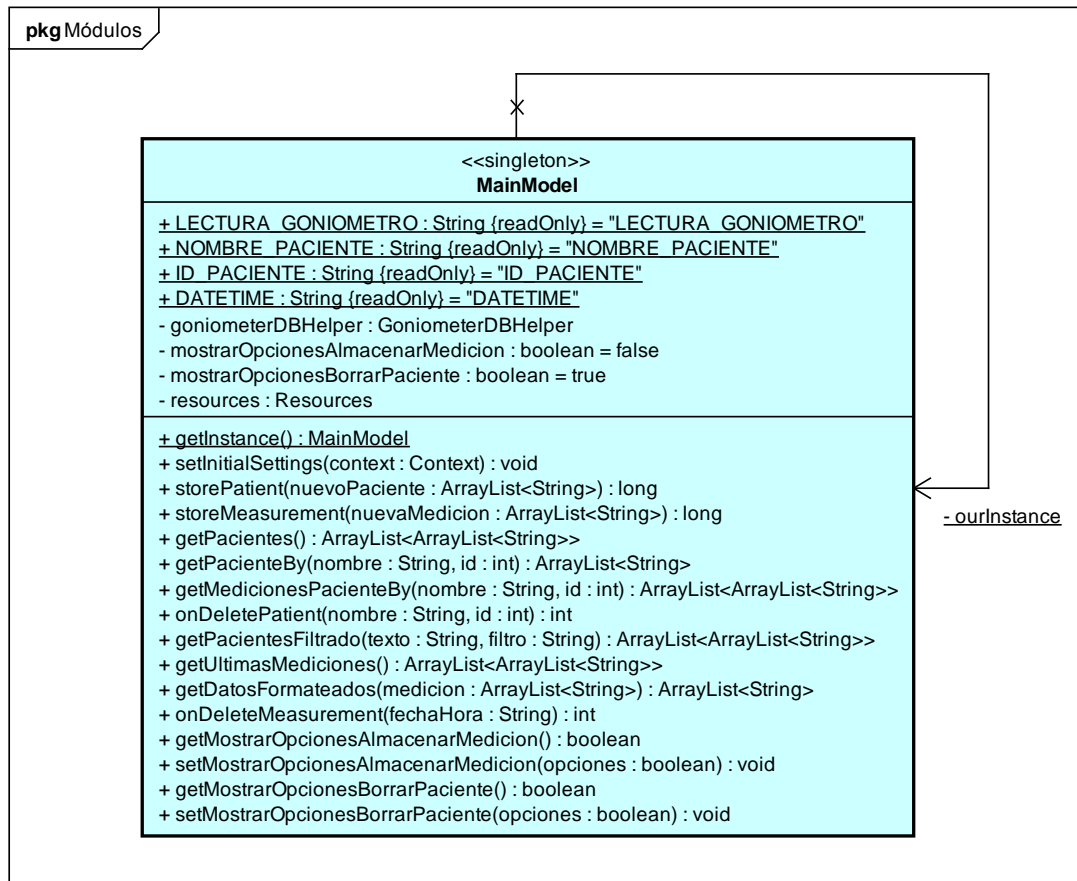


Figura 32 - Módulo *Model*

6.1.5.6. Módulo *Interfaces*

Para este módulo en concreto, dado el tamaño de los diagramas para cada una de las interfaces que lo componen, se va a separar en 3 diagramas distintos, cada uno de ellos representando una interfaz.

6.1.5.6.1. Interfaz *ViewFunctions*

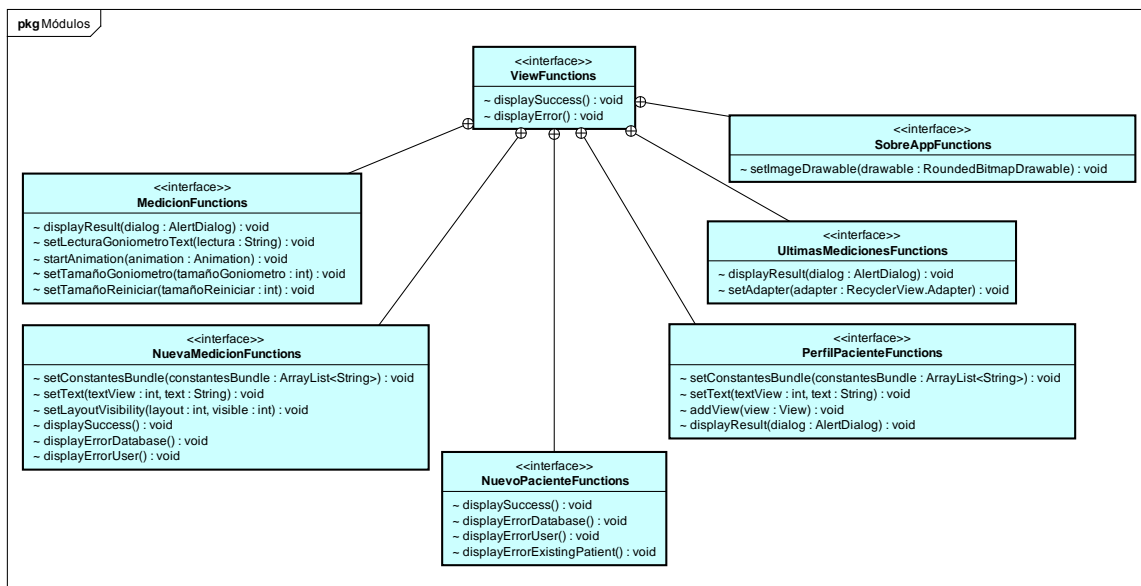


Figura 33 - Módulo *Interfaces* - *ViewFunctions*

6.1.5.6.2. Interfaz *PresenterFunctions* – Parte 1

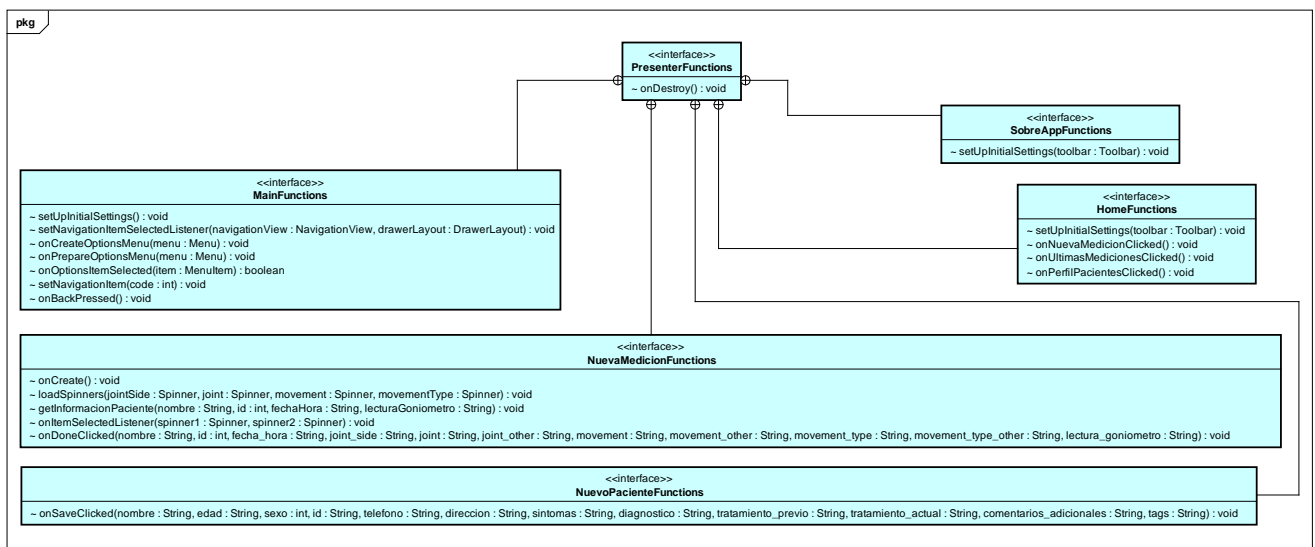


Figura 34 - Módulo Interfaces - *PresenterFunctions* - Parte 1

6.1.5.6.3. Interfaz *PresenterFunctions* – Parte 2

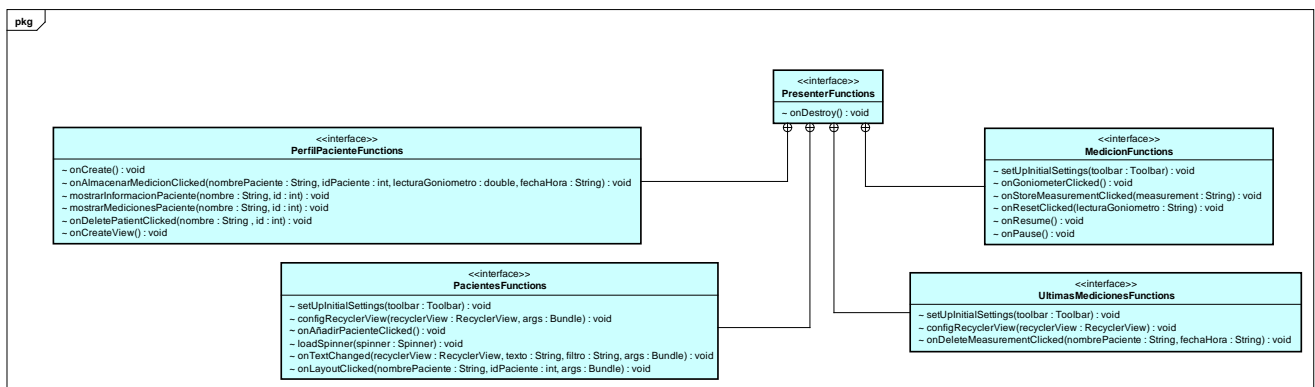


Figura 35 - Módulo Interfaces – *PresenterFunctions* – Parte 2

6.1.5.6.4. Interfaz *ModelFunctions*

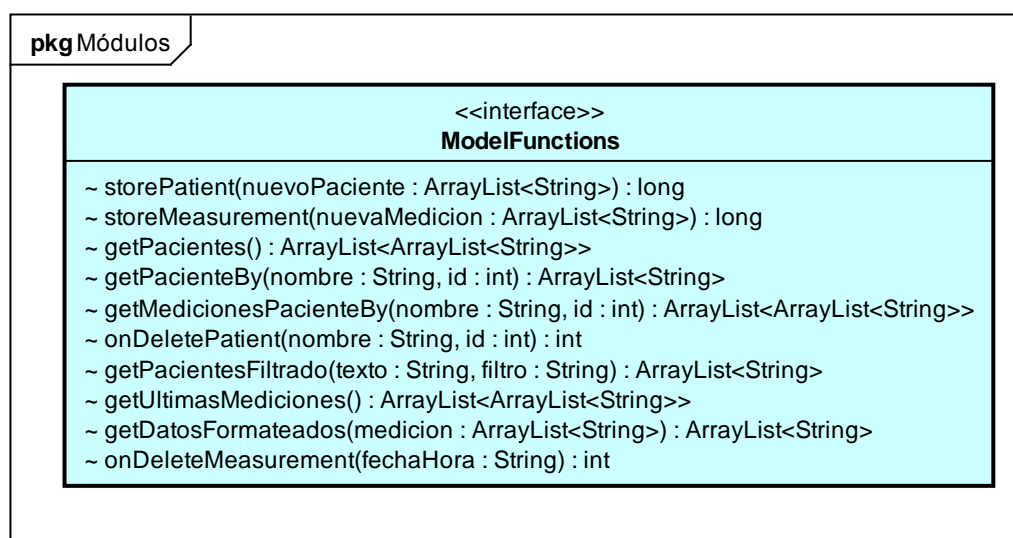


Figura 36 - Módulo Interfaces - *ModelFunctions*

6.1.5.7. Módulo *DependencyInjection*

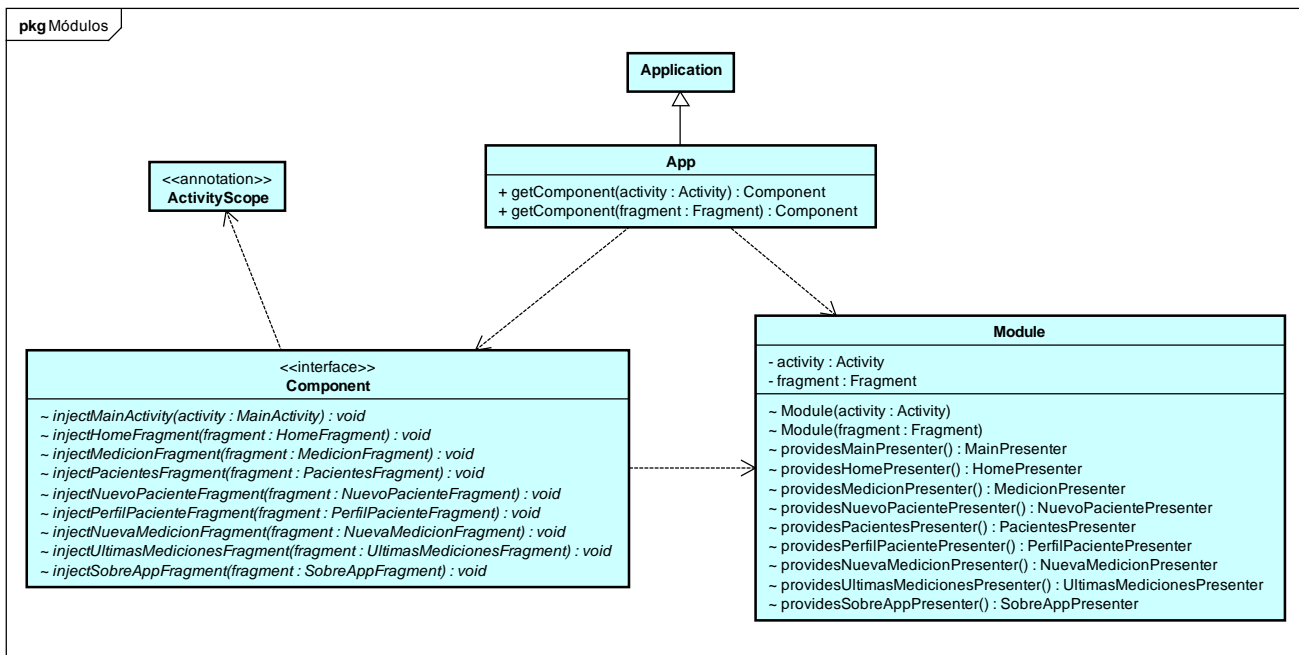


Figura 37 - Módulo *DependencyInjection*

6.1.5.8. Módulo *Data*

En el caso de este diagrama, es necesario comentar que, dada la complejidad en cuanto a las dependencias existentes entre las diferentes clases del módulo, ha sido necesario tener que dividir el diagrama en dos partes. En la primera se muestran todas las relaciones entre las clases “Entities”, “Enums” y “Contract” (esquema de la Base de Datos relacional) y en la segunda, se muestra a la clase manejadora de la Base de Datos (*GoniometerDBHelper*) junto con las relaciones con el resto de elementos del módulo *Data*.

6.1.5.8.1. Módulo *Data* – Parte 1

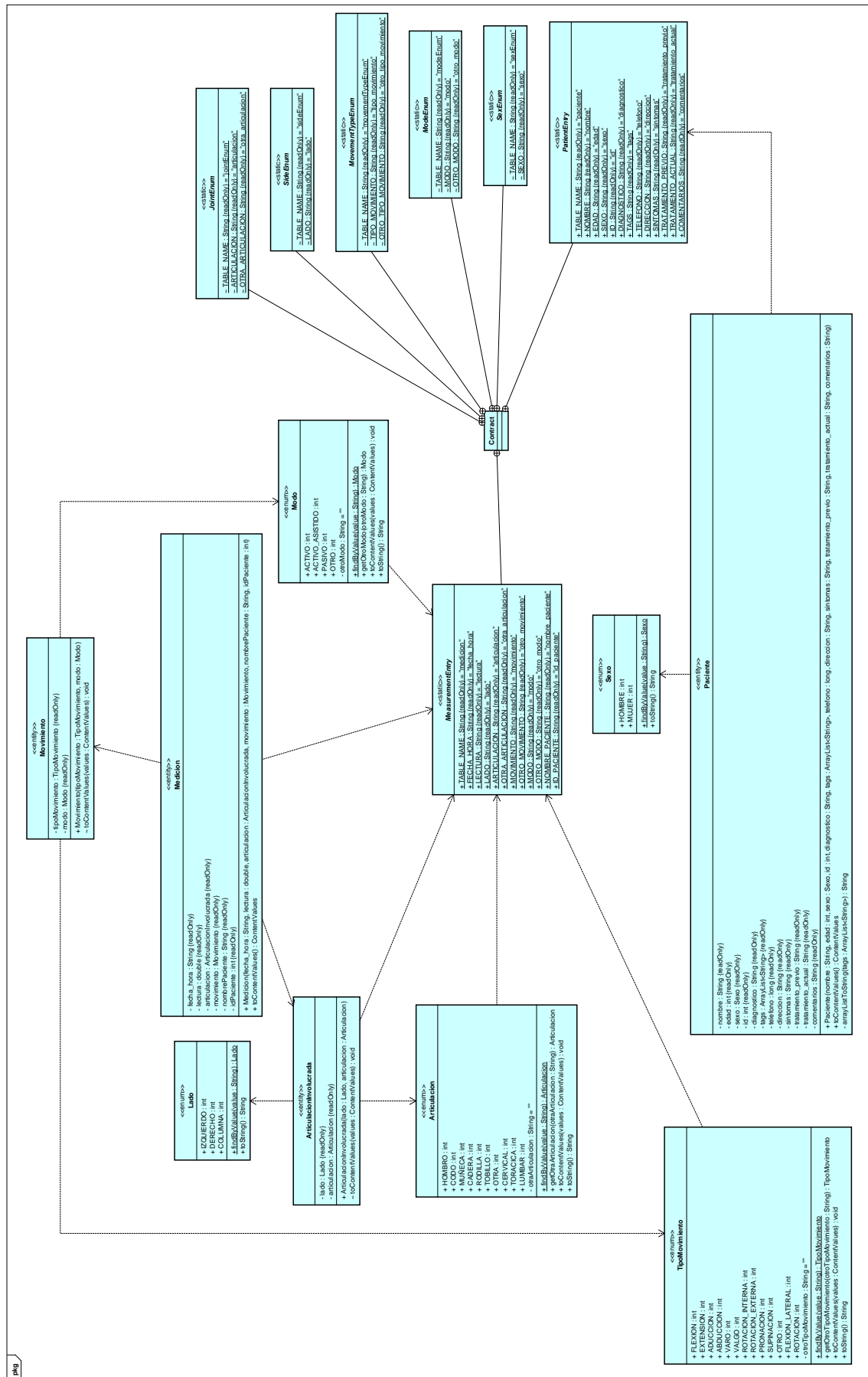


Figura 38 - Módulo Data - Parte 1

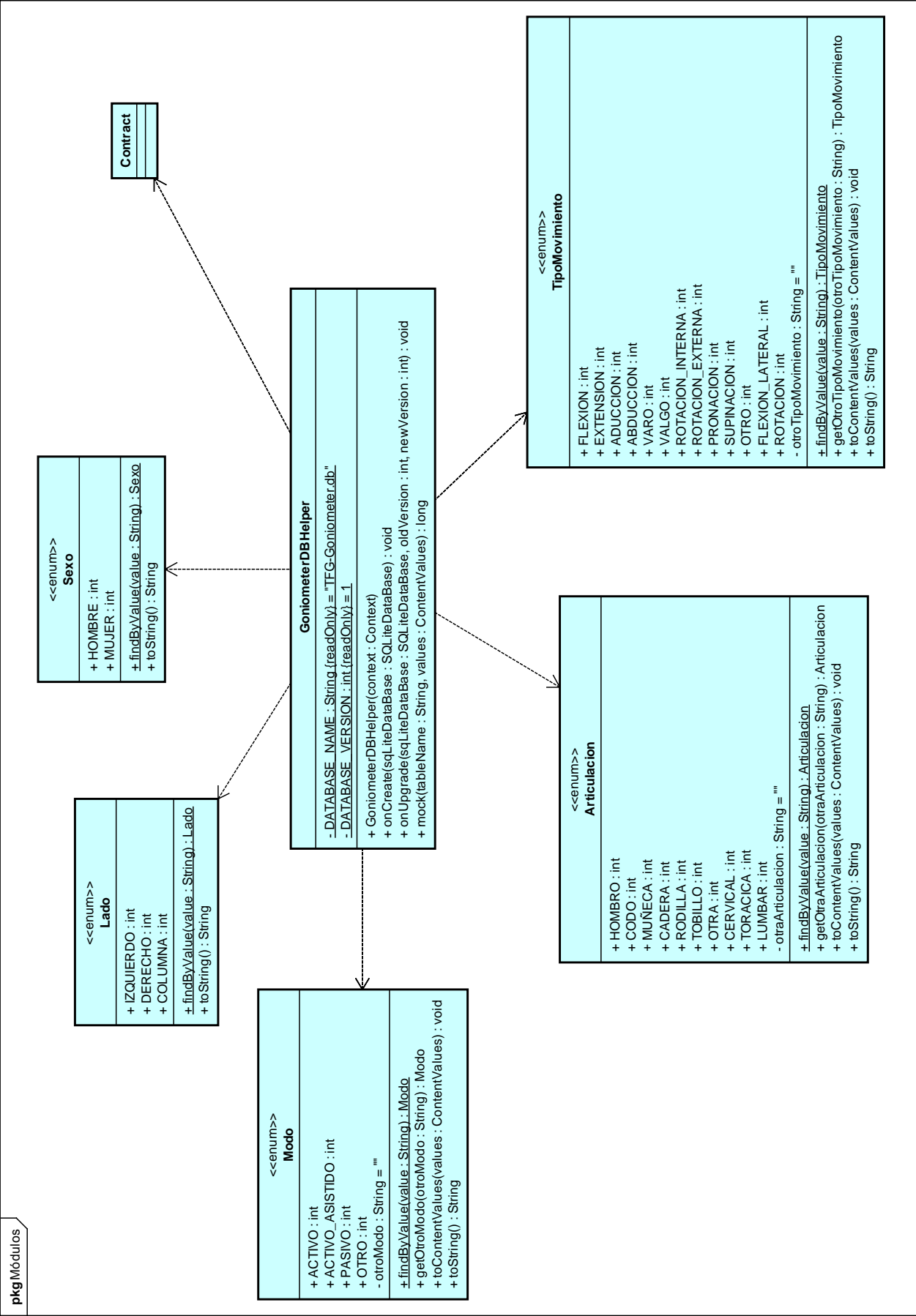


Figura 39 - Módulo *Data* - Parte 2

6.2. Modelado de Datos

6.2.1. Diseño conceptual de la Base de Datos

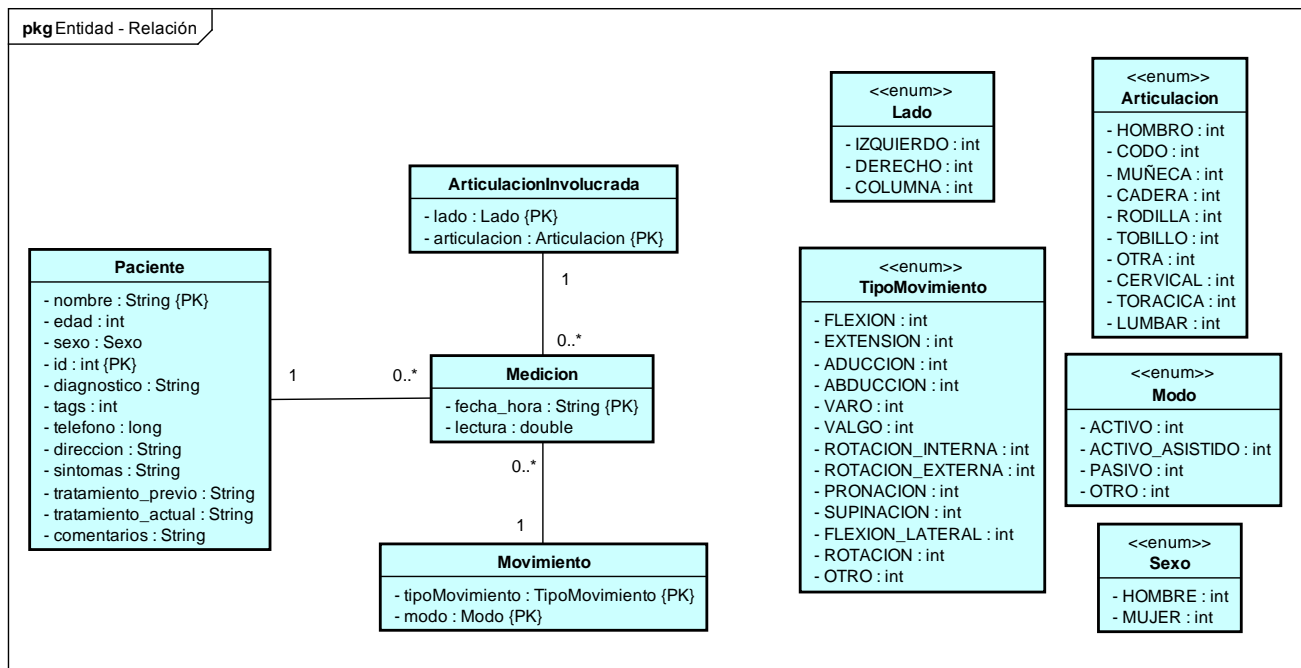


Figura 40 - Diseño conceptual de la Base de Datos

6.2.2. Esquema relacional de la Base de Datos

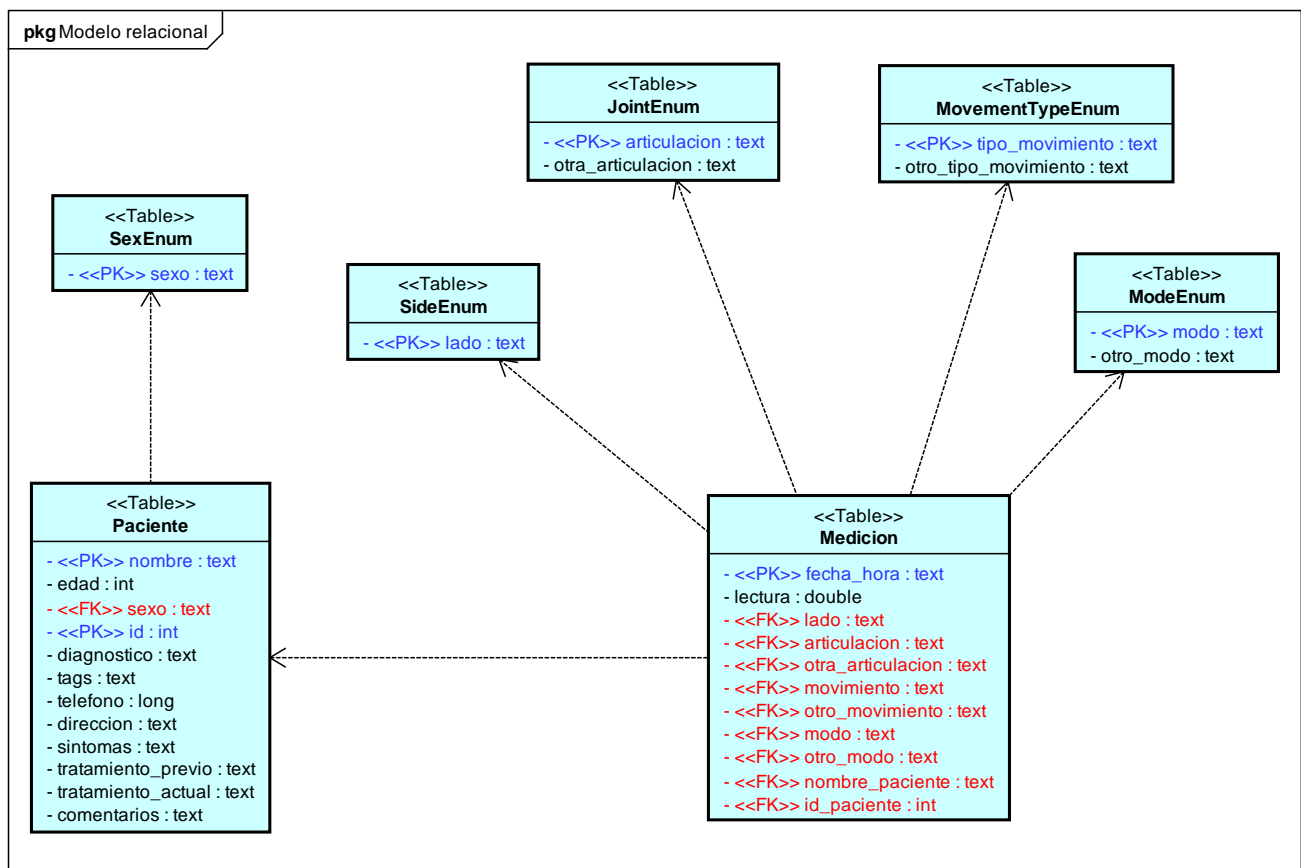


Figura 41 - Esquema relacional de la Base de Datos

6.2.3. Comentarios con respecto a la Base de Datos

En el diseño conceptual aparecen dos entidades “*ArticulacionInvolucrada*” y “*Movimiento*” que, finalmente, no se han incluido en la estructura de la Base de Datos. El motivo de esta decisión es que se consideró importante incluirlas desde un punto de vista conceptual, es decir, realmente aportan cierta claridad al diagrama (una medición está formada, entre otras cosas, por la articulación involucrada en la misma, así como el movimiento que ésta ha realizado); sin embargo, no aportan ninguna información por sí mismas; los únicos atributos que tienen son claves foráneas a otras entidades (de tipo Enum), por lo que, no tenía ningún sentido que formaran cada una de ellas una tabla en la Base de Datos.

6.2.3.1. Paciente

- **Población:** Modificable. Se pueden añadir y borrar pacientes, pero no actualizarlos una vez se han introducido en la Base de Datos.
- **Columnas:** Se utiliza la combinación del nombre del paciente más el ID del mismo para identificarlo de manera única; esto lo que permite es, por un lado, almacenar pacientes que tengan exactamente el mismo nombre y, por otro lado, agrupar bajo un mismo ID a, por ejemplo, pacientes que tengan la misma patología (de cara a poder filtrarlos en el listado de pacientes), pero no almacenar dos pacientes que tengan exactamente el mismo nombre e ID.

De esta manera, el usuario tiene más libertad (al poder almacenar pacientes con el mismo nombre) sin perder la funcionalidad que se quería dar originalmente con el atributo ID (la cual se acaba de explicar).

6.2.3.2. Medicion

- **Población:** Modificable. Se pueden añadir y borrar mediciones, pero no actualizarlas una vez que se han introducido en la Base de Datos.
- **Columnas:**
 - *fecha_hora*: Dado que SQLite no posee un tipo de datos destinado a almacenar fechas, se ha decidido optar por emplear el tipo *text* para esta tarea, lo que, además, ha permitido tener cierta flexibilidad a la hora de manejar este atributo, ya que, no sólo contiene la fecha (día de la medición), sino también la hora en la que se ha realizado (hasta los segundos).

Debido a que almacena tanto el día como la hora (incluyendo segundos) en los que se realiza la medición, es el motivo por el cual se ha decidido utilizar como clave primaria de la tabla. Es lo suficientemente precisa como para asegurar que un usuario no va a poder realizar dos mediciones con exactamente el mismo valor para este atributo.

6.2.3.3. Tablas Enum

- **Población:** No modificable. Los valores que poseen estas tablas están fijados desde antes de su creación e inserción en la Base de Datos. Estos valores son especificados en el diseño conceptual de la misma (apartado 6.2.1.).
- **Columnas:** En las tablas *Jointenum*, *MovementTypeEnum* y *ModeEnum*:
 - *otra_articulacion*, *otro_tipo_movimiento*, *otro_modo*: Estas columnas siempre tienen el valor “NULL”. Se utilizan cuando el usuario ha seleccionado el valor “Otro/a” en alguno de los 3 campos de la medición que representan estas tablas (Articulación, Tipo de movimiento o Modo). En ese caso, se muestra al usuario un cuadro de texto a rellenar. La información que introduzca en ese cuadro de texto es el valor que tomarán estas columnas.

Nota importante: El valor de estas columnas no es modificado en NINGÚN momento. Cuando el usuario introduce esa información, donde se almacena es en los campos *otra_articulacion*, *otro_tipo_movimiento* y *otro_modo* (claves foráneas) de la tabla *MeasurementEntry*, pero los valores de estas columnas en las tablas Enum asociadas son siempre “NULL”.

6.3. Realización en Diseño de los Casos de Uso

6.3.1. UC-0001: Añadir_Medicion. Parte 1

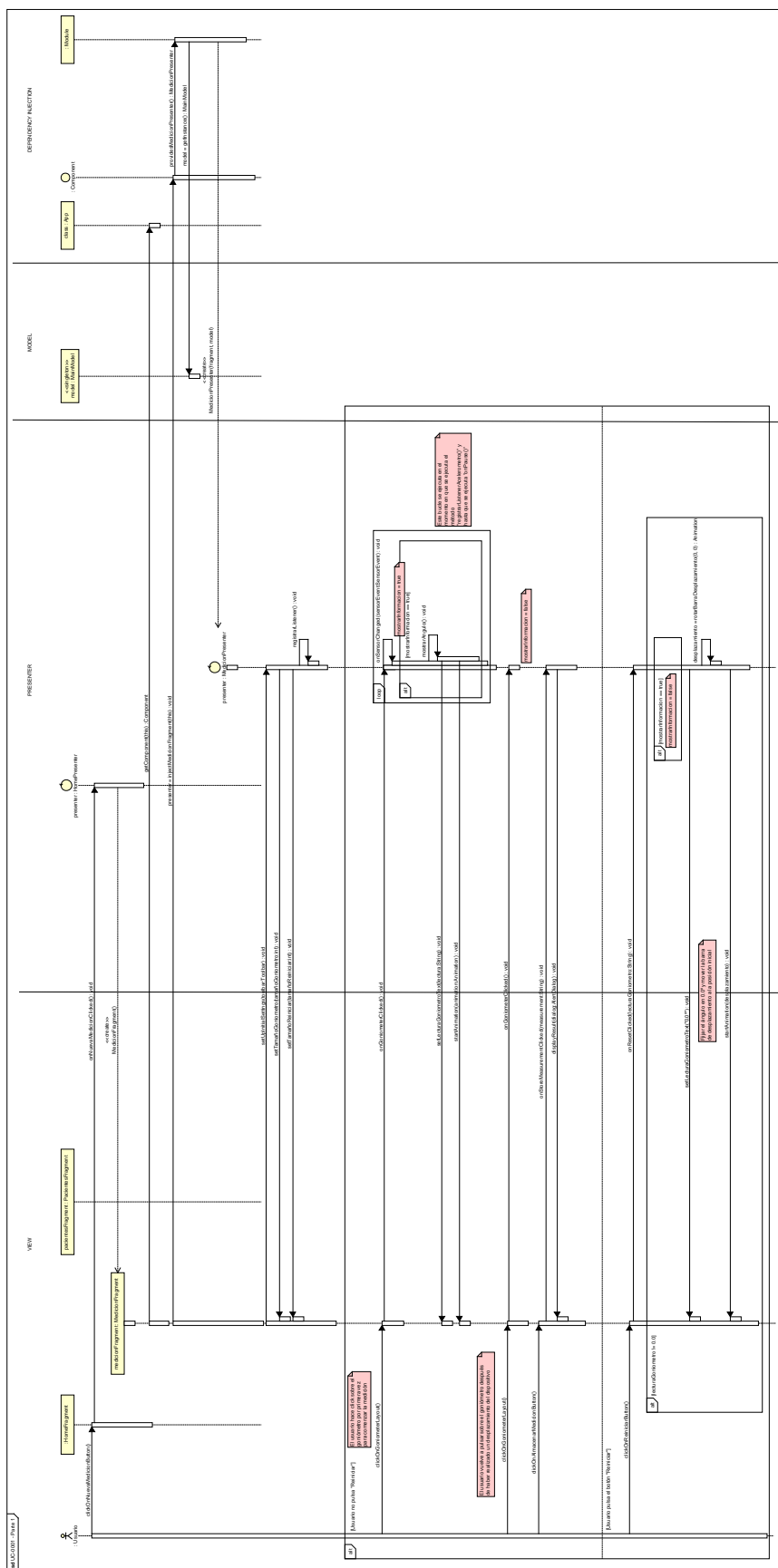


Figura 42 - Diagrama de secuencia UC-0001 - Parte 1

6.3.2. UC-0001: Añadir_Medicion. Parte 2

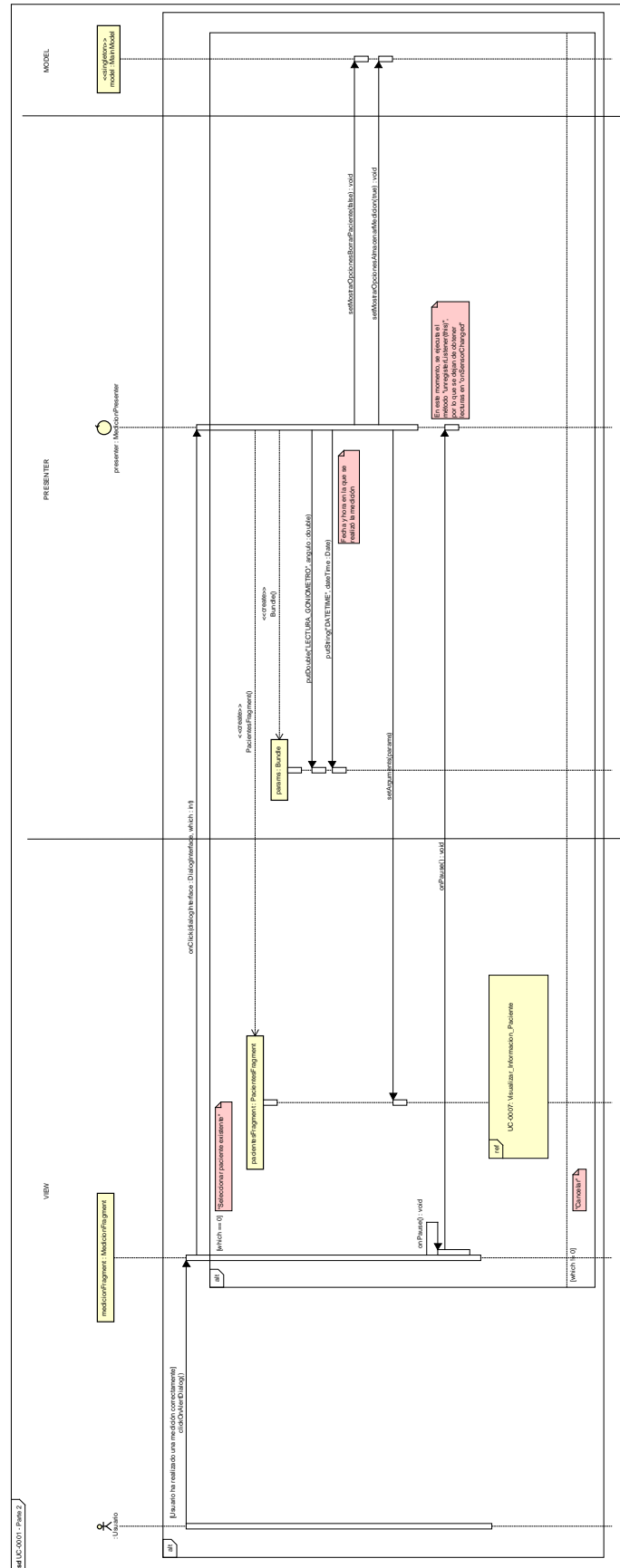
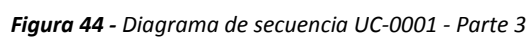


Figura 43 - Diagrama de secuencia UC-0001 - Parte 2

En este diagrama, el cuadro “ref” hace referencia a *UC-0007: Visualizar_Informacion_Paciente*, cuyos diagramas se encuentra en el apartado 6.3.18 y 6.3.19.

68



6.3.4. UC-0001: Añadir_Medicion. Parte 4

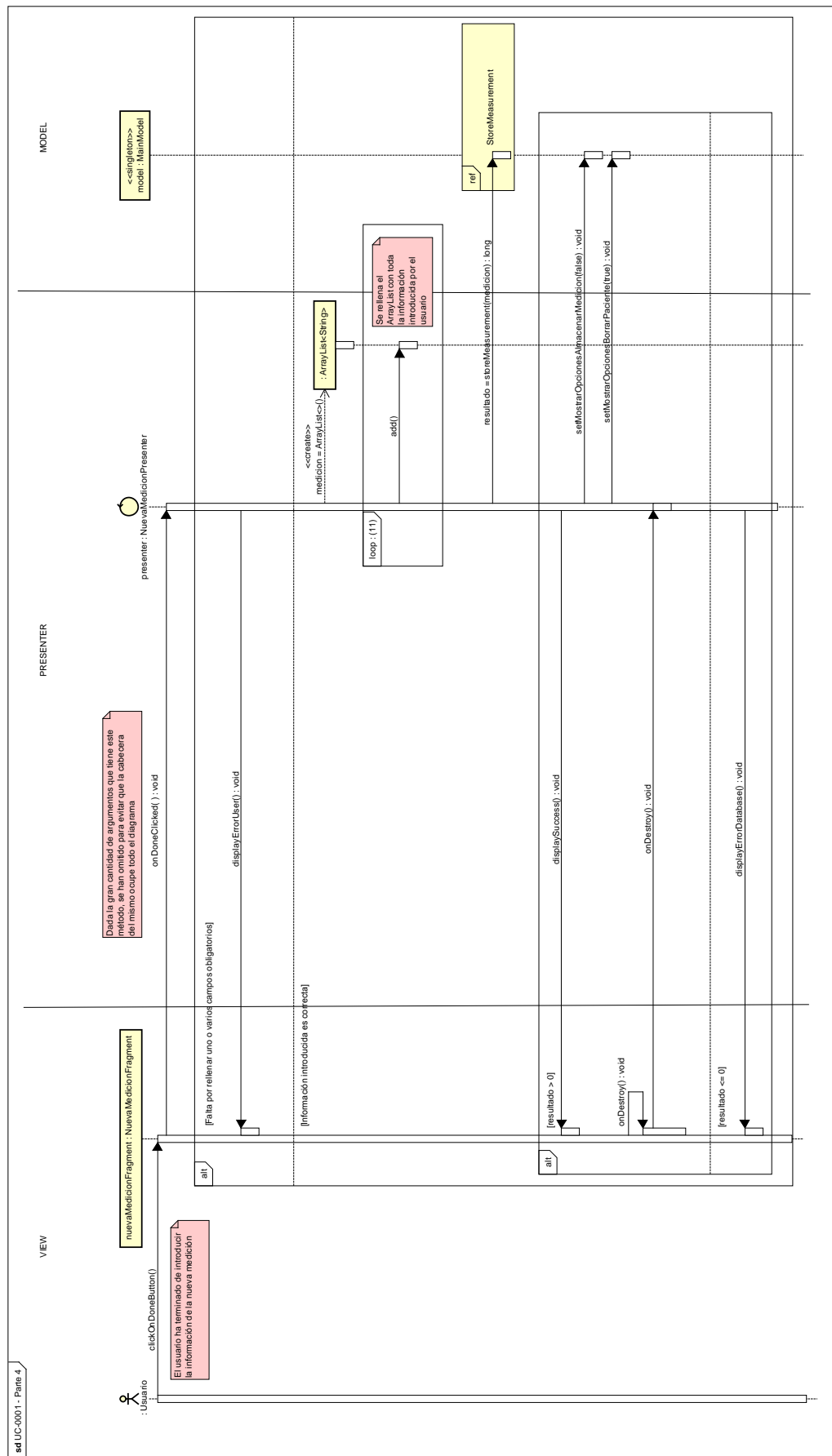


Figura 45 - Diagrama de secuencia UC-0001 - Parte 4

6.3.5. Detalle de *GetPacienteBy*

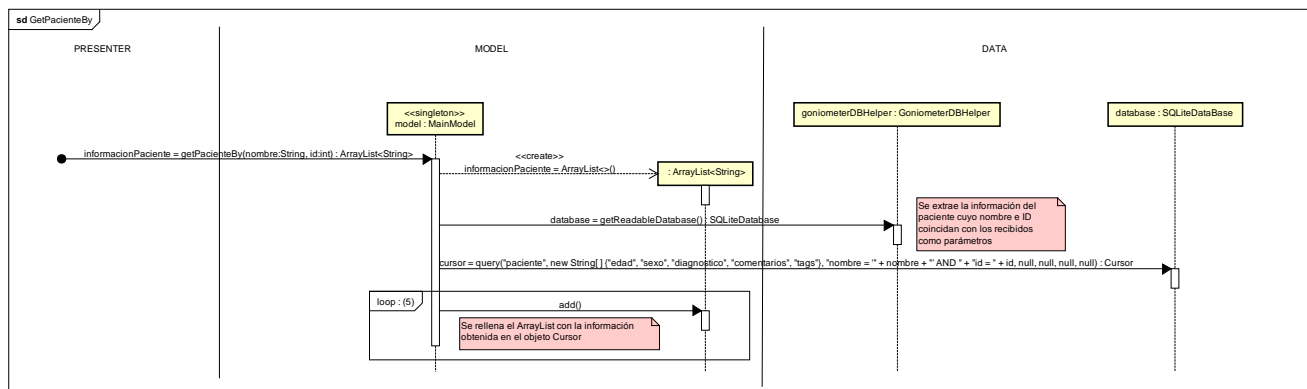


Figura 46 - Diagrama de secuencia *GetPacienteBy*

6.3.6. Detalle de *StoreMeasurement*

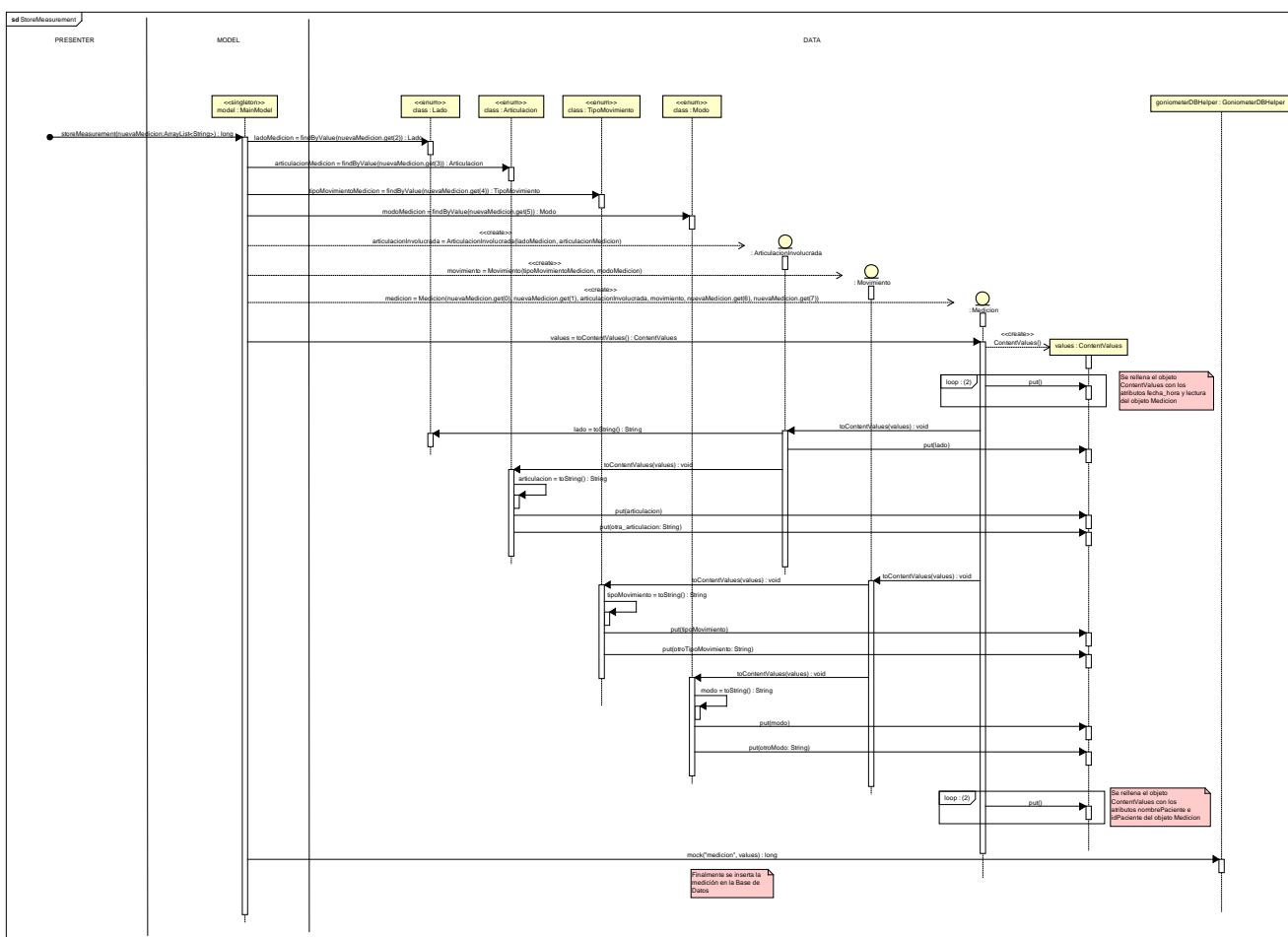


Figura 47 - Diagrama de secuencia *StoreMeasurement*

6.3.7. UC-0002: Visualizar_10_Ultimas_Mediciones

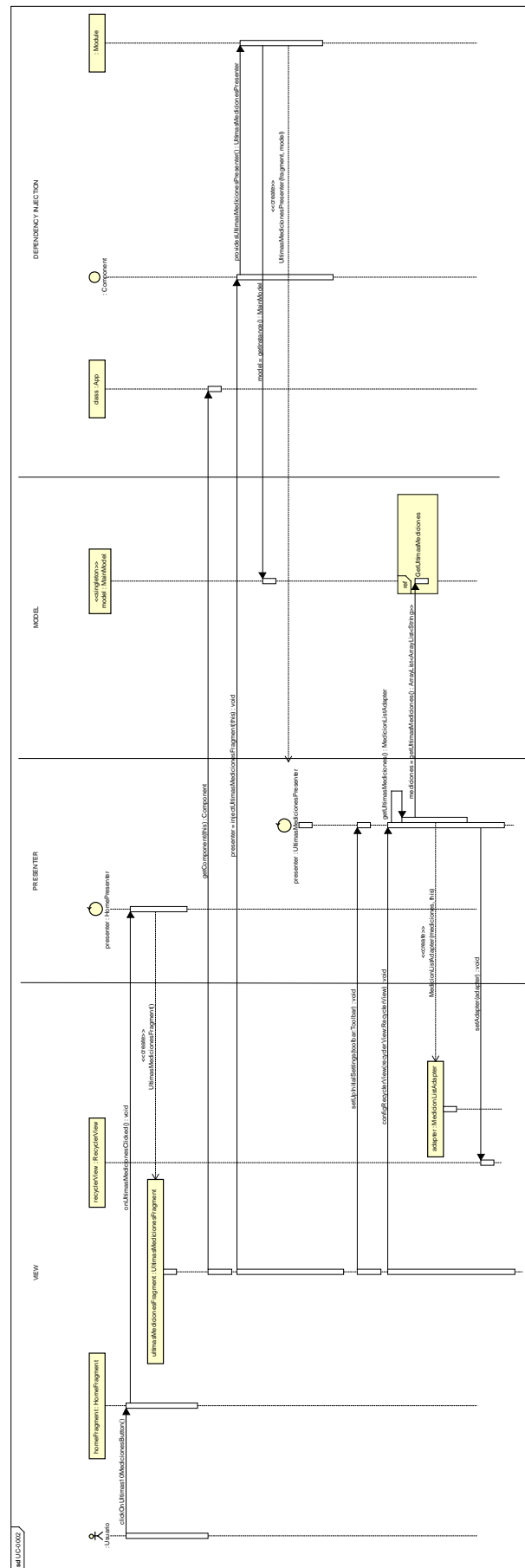


Figura 48 - Diagrama de secuencia UC-0002

6.3.8. Detalle de GetUltimasMediciones

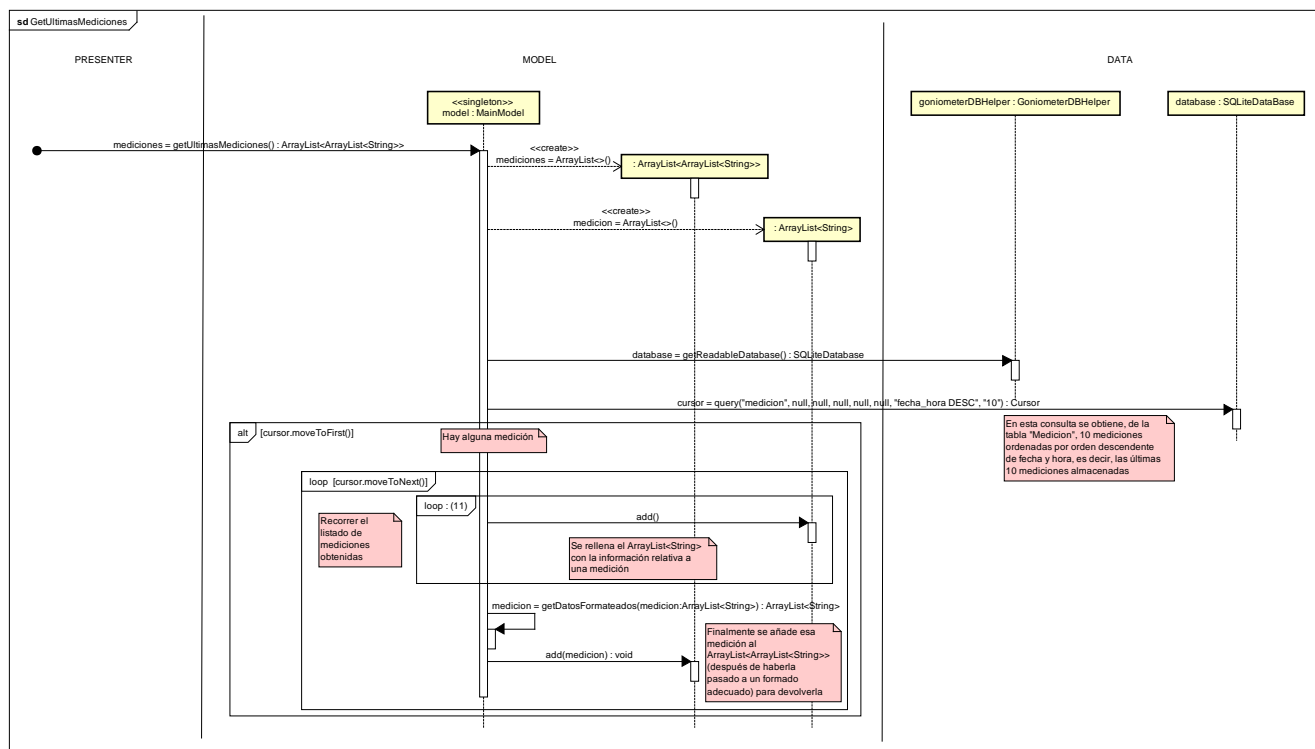


Figura 49 - Diagrama de secuencia GetUltimasMediciones

6.3.9. UC-0003: Borrar_Medicion

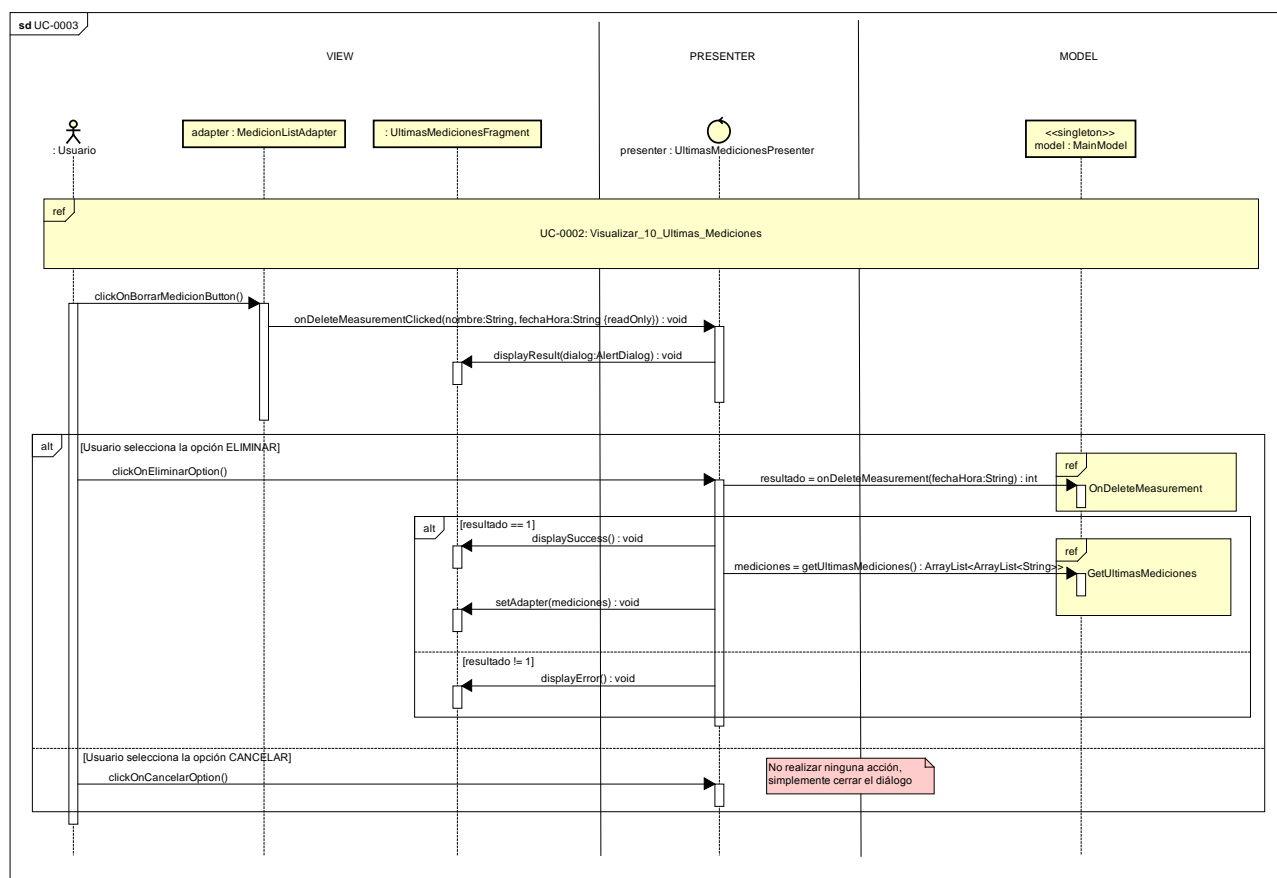


Figura 50 - Diagrama de secuencia UC-0003

En el caso de este diagrama, el segundo cuadro “ref” es una referencia al diagrama mostrado en el apartado 6.3.8.

6.3.10. Detalle de OnDeleteMeasurement

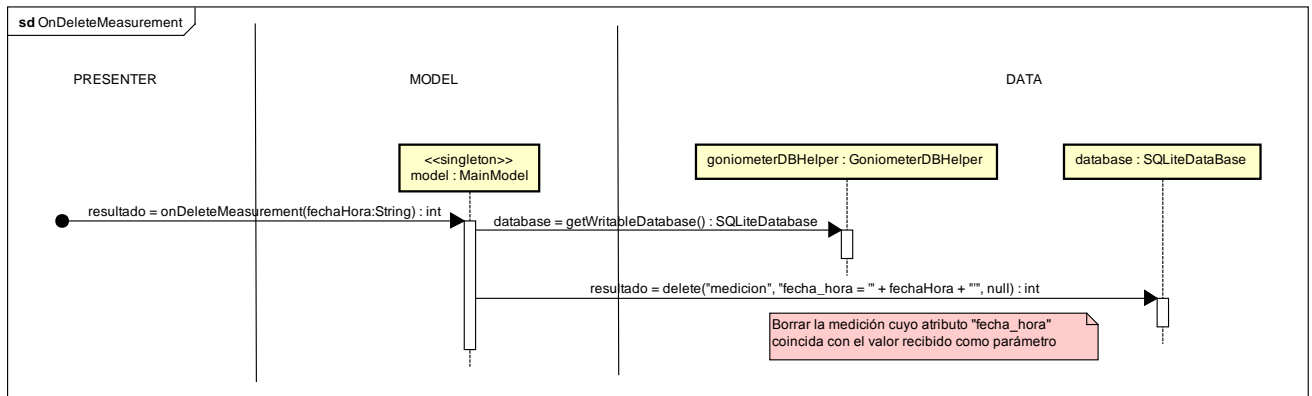
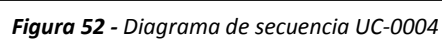


Figura 51 - Diagrama de secuencia OnDeleteMeasurement

74



6.3.12. Detalle de *GetPacientes*

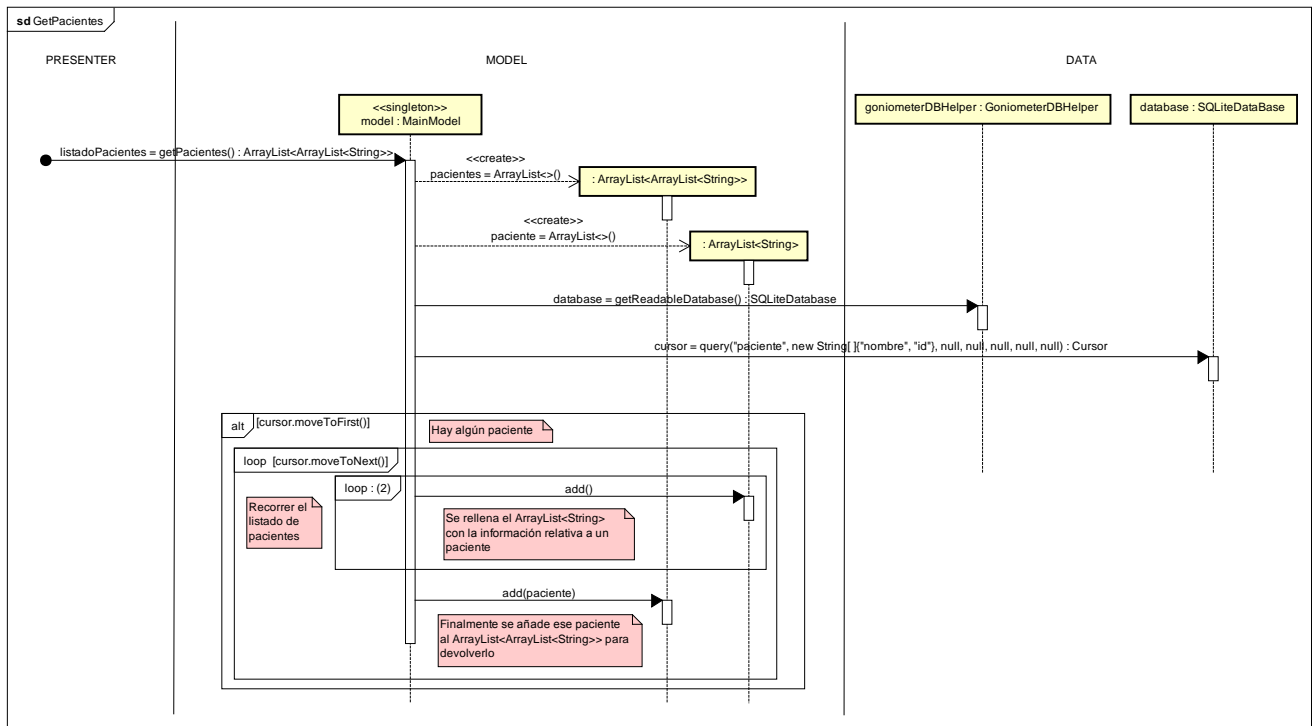


Figura 53 - Diagrama de secuencia *GetPacientes*

6.3.13. UC-0005: Filtrar_Pacientes

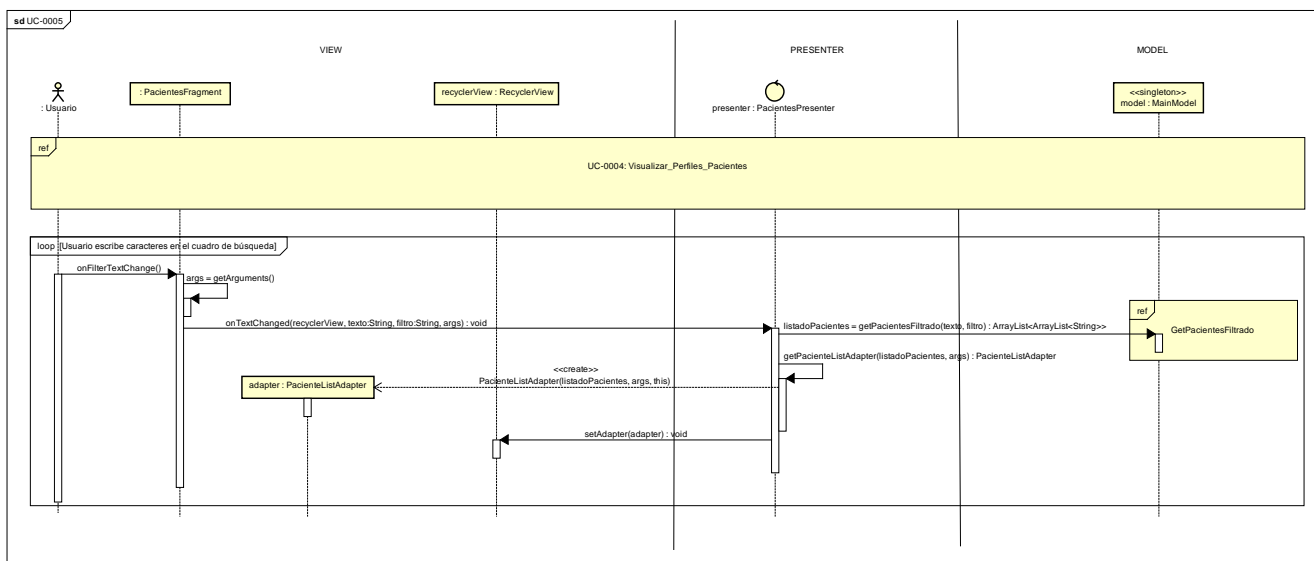


Figura 54 - Diagrama de secuencia UC-0005

6.3.14. Detalle de GetPacientesFiltrado

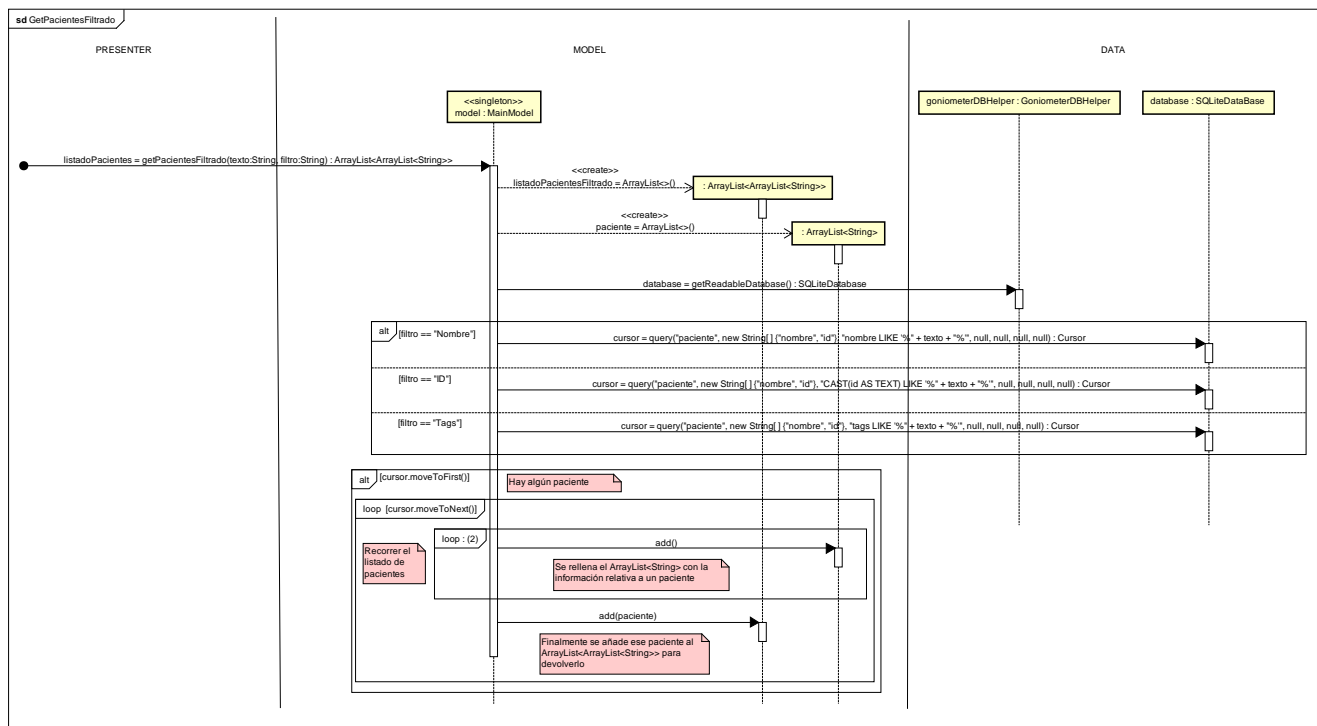


Figura 55 - Diagrama de secuencia GetPacientesFiltrado

6.3.15. UC-0006: Crear_Perfil_Paciente – Parte 1

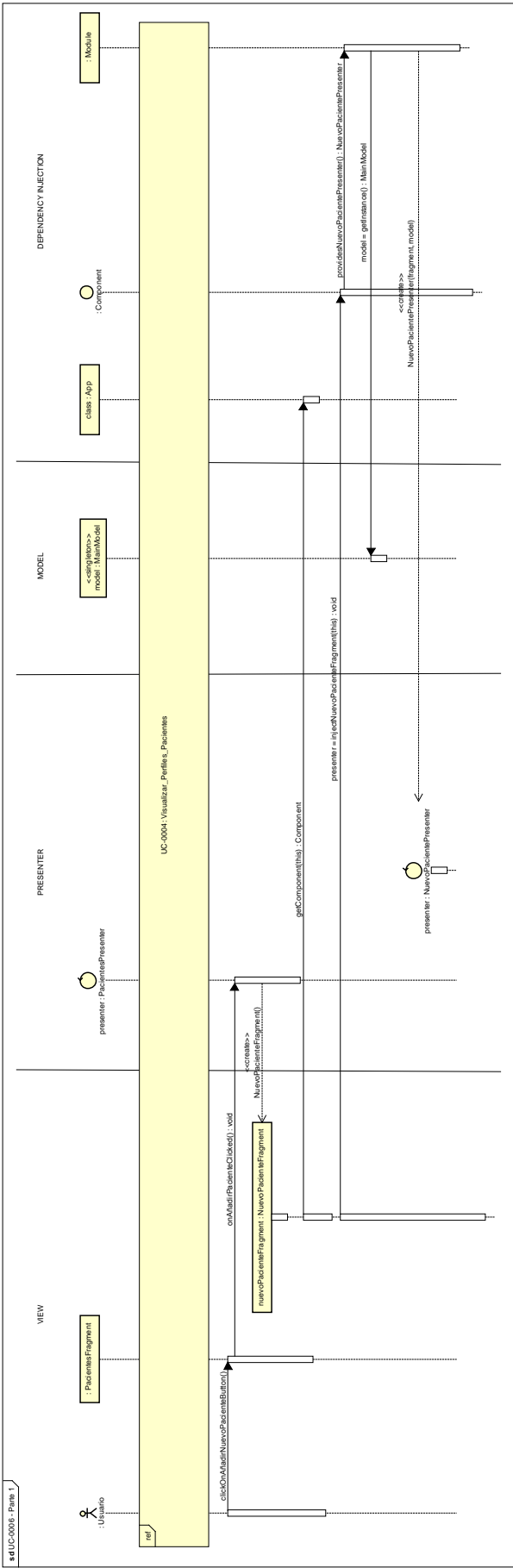


Figura 56 - Diagrama de secuencia UC-0006 – Parte 1

6.3.16. UC-0006: Crear_Perfil_Paciente – Parte 2

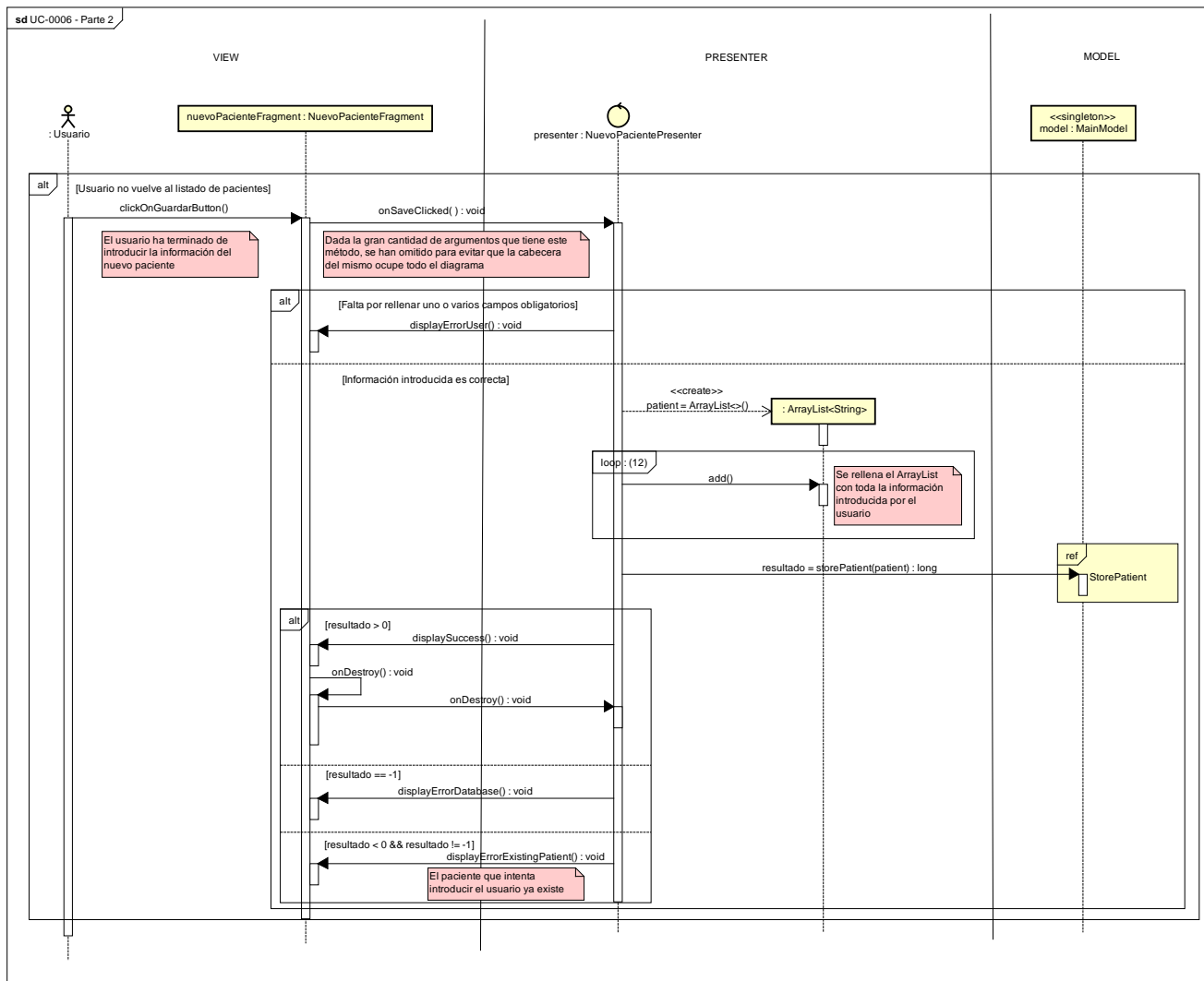


Figura 57 - Diagrama de secuencia UC-0006 – Parte 2

6.3.17. Detalle de StorePatient

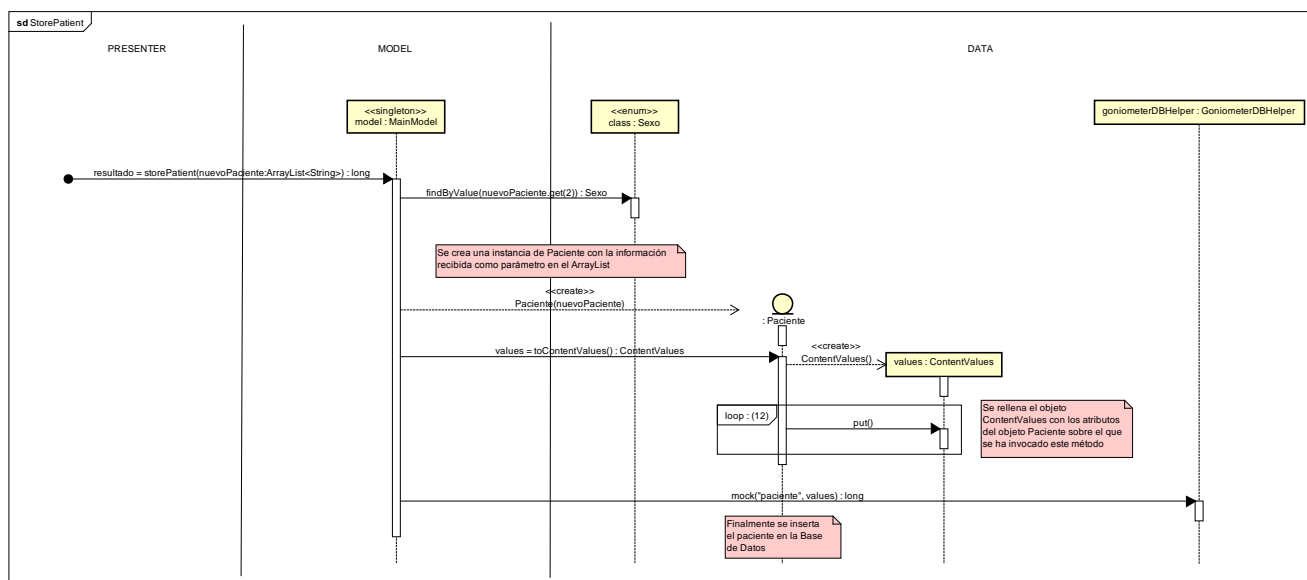


Figura 58 - Diagrama de secuencia StorePatient

6.3.18. UC-0007: Visualizar_Informacion_Paciente – Parte 1

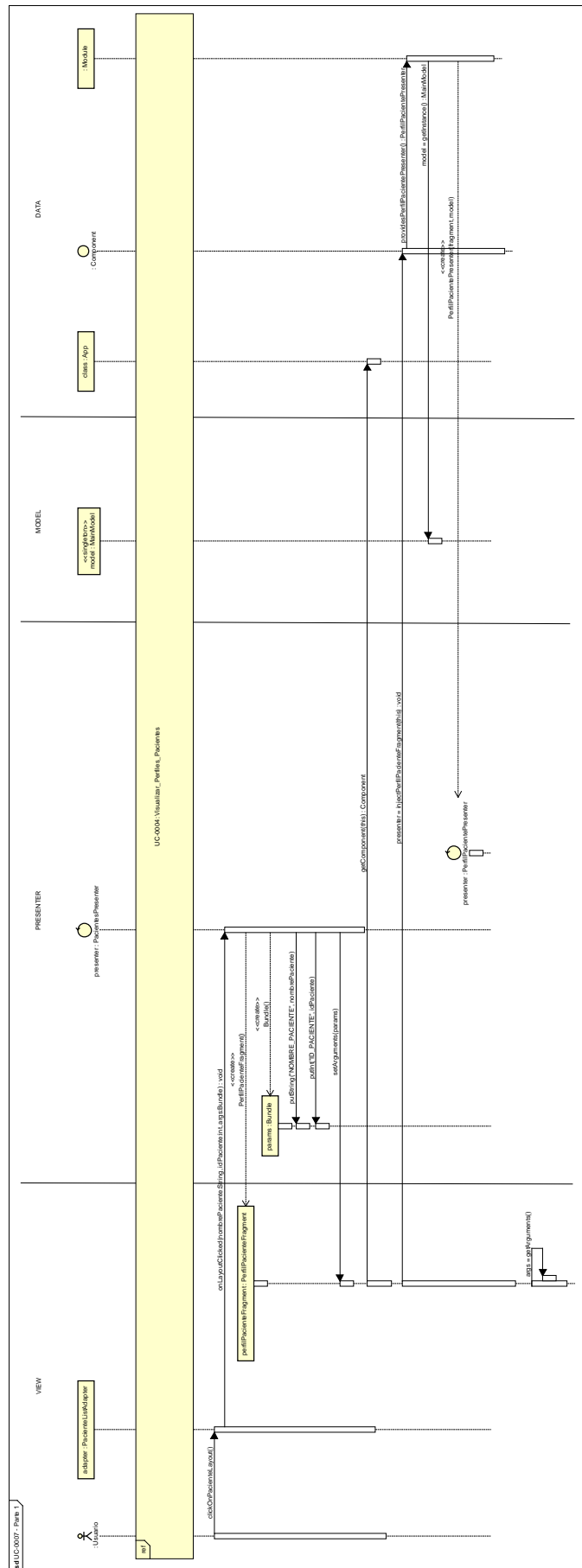


Figura 59 - Diagrama de secuencia UC-0007 – Parte 1

6.3.19. UC-0007: Visualizar_Informacion_Paciente – Parte 2

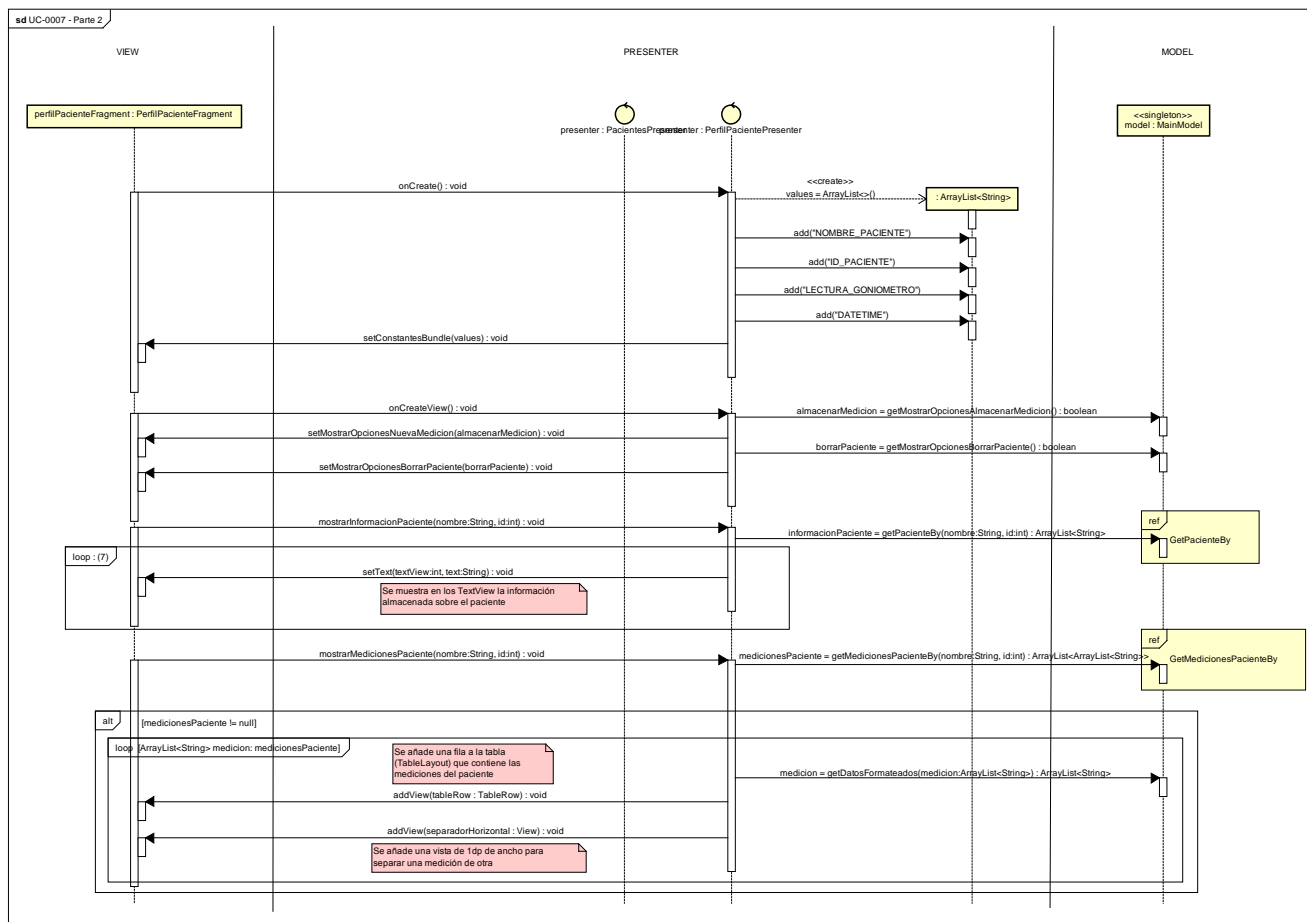


Figura 60 - Diagrama de secuencia UC-0007 – Parte 2

En el caso de este diagrama, el primer cuadro “ref” es una referencia al diagrama mostrado en el apartado 6.3.5.

6.3.20. Detalle de GetMedicionesPacienteBy

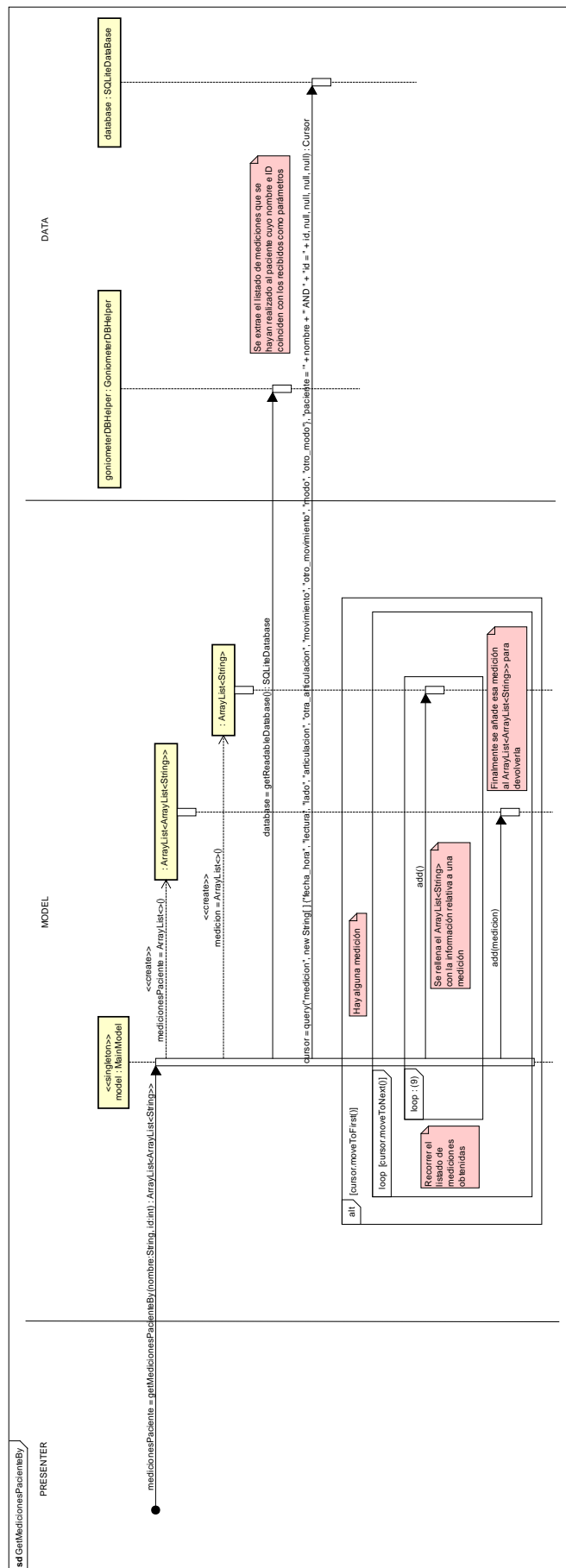


Figura 61 - Diagrama de secuencia GetMedicionesPacienteBy

6.3.21. UC-0008: Borrar_Perfil_Paciente

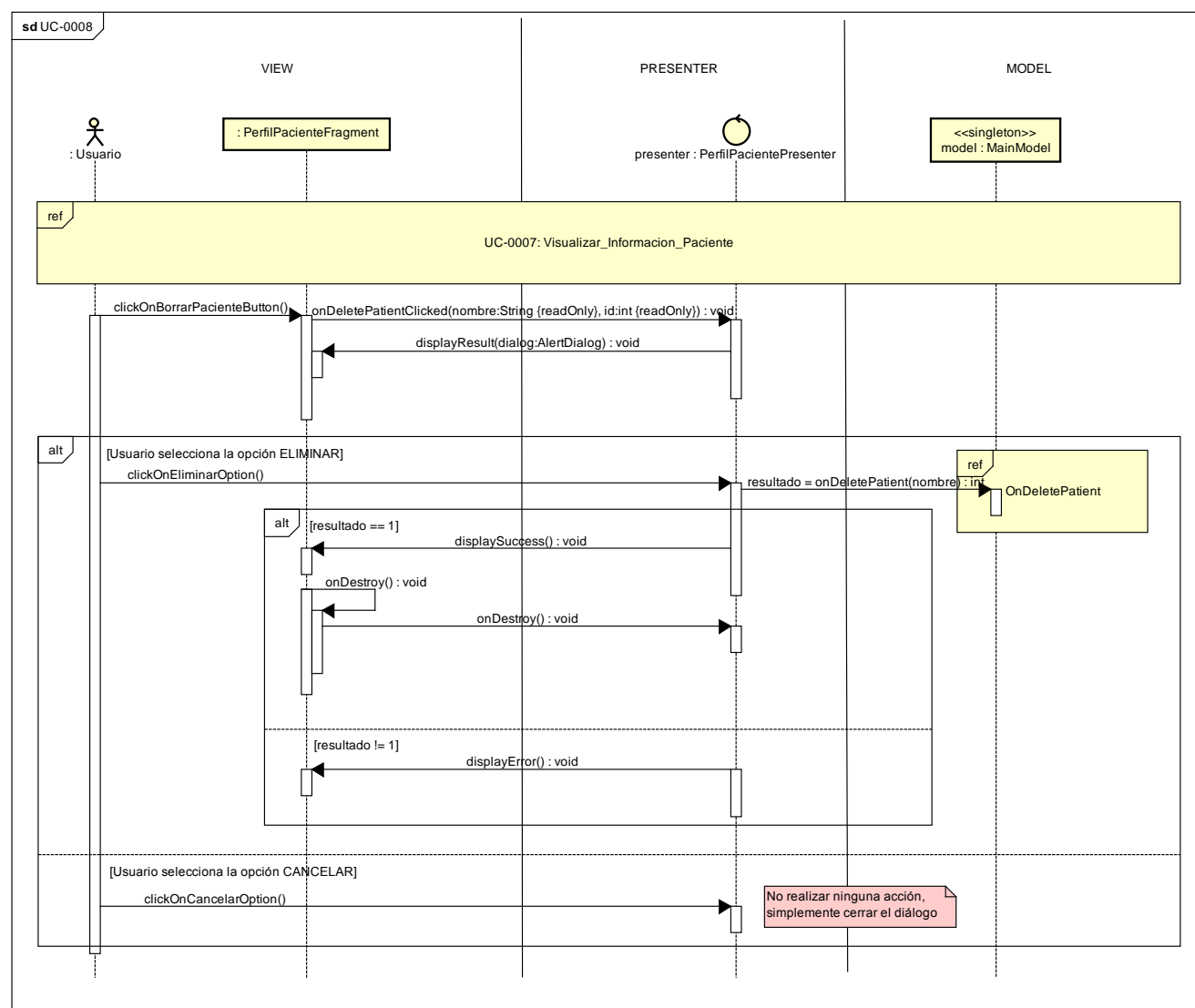


Figura 62 - Diagrama de secuencia UC-0008

6.3.22. Detalle de OnDeletePatient

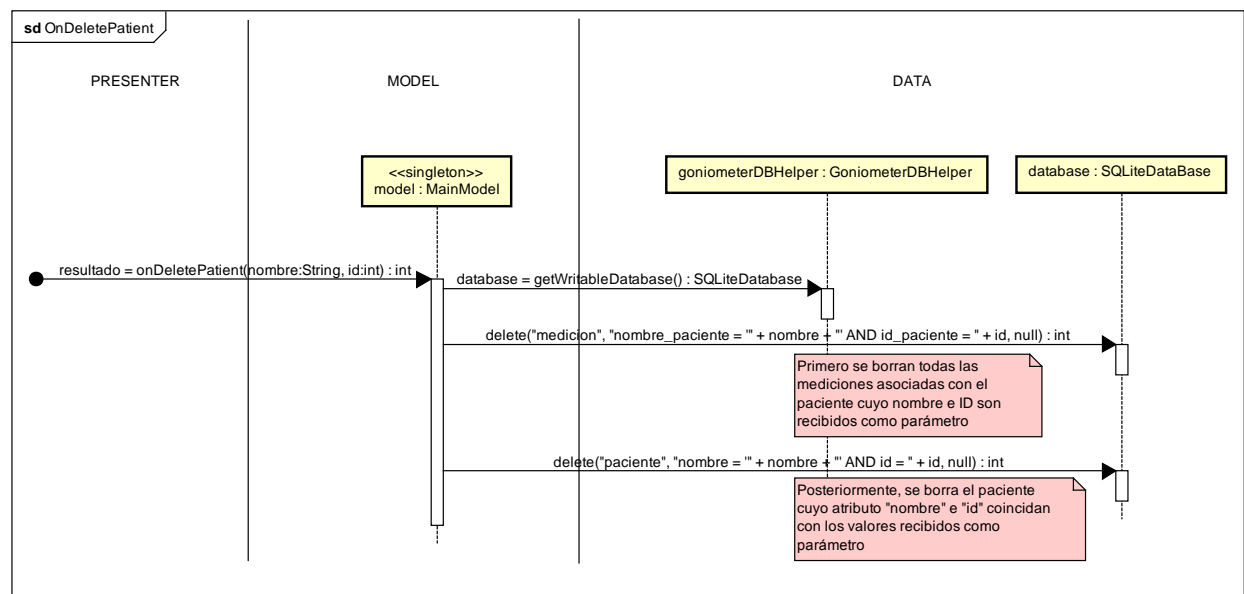


Figura 63 - Diagrama de secuencia OnDeletePatient

Capítulo 7

Implementación y pruebas

7.1. Implementación

En los sucesivos apartados, se explicarán algunas cuestiones relativas a la implementación de *TFG-Goniometer*. En concreto, se va a centrar en el uso de bibliotecas externas que se han incluido en el desarrollo de esta aplicación, así como de las distintas fases en las que se ha desarrollado.

7.1.1. Uso de bibliotecas externas

En este apartado, como se acaba de comentar, se van a explicar qué bibliotecas se han incluido en el desarrollo de *TFG-Goniometer*, haciendo una breve explicación de cada una de ellas.

7.1.1.1. Dagger2

Como ya se ha comentado previamente en el apartado 6.1.2.3 (Patrón Inyección de Dependencias), se ha utilizado *Dagger2* para implementar este patrón de diseño.

Dagger es un framework completamente estático de inyección de dependencias en tiempo de compilación tanto para Java como para Android. Es una adaptación de una versión anterior creada por Square y que ahora es mantenida por Google. A continuación, se describirá muy brevemente de qué se compone Dagger.

Dagger se compone principalmente de módulos y componentes:

- Los *módulos* contienen las diferentes dependencias. Mediante una serie de métodos proveedores, se encargan de crear aquellas dependencias que requieran algún tipo de proceso especial a la hora de ser creadas. Para algunas clases no es necesario hacer ningún tipo de procesamiento antes de ser creadas, para esas, basta con usar la etiqueta `@Inject` en el constructor. Un módulo puede estar asociado a uno o más componentes.
- Los componentes son los encargados de inyectar las dependencias y pueden asociarse a distintos ciclos de vida de la aplicación.

7.1.1.2. Enum Format

Enum Format es un módulo que pertenece a la biblioteca *Xdroid*. Esta biblioteca se creó con el fin de proveer de distintos módulos (cada uno realiza una tarea distinta) que se pudieran incorporar de manera individual en aplicaciones Android en función de las necesidades de los desarrolladores.

En concreto, *Enum Format* se encarga de añadir anotaciones a las distintas claves de una clase de tipo *Enum* y enlazarlas a un recurso de tipo String (dentro de los ficheros `res/values-XX/strings.xml`). De esta manera, lo

que se consigue es que el valor de cada una de las claves de la clase *Enum* sea un valor que cambie en función del idioma del dispositivo en el que se ejecute la aplicación.

Esto permite, de una forma muy sencilla, almacenar la información de estos tipos *Enum* en la Base de Datos en un idioma u otro. A continuación, se adjunta una imagen de un trozo de código de la clase *Articulacion* (de tipo *Enum*) donde se puede apreciar el uso de estas anotaciones que se acaba de comentar.

```
// Principales articulaciones del cuerpo humano
@EnumString(R.string.hombro)
HOMBRO,
@EnumString(R.string.codo)
CODO,
@EnumString(R.string.muñeca)
MUÑECA,
@EnumString(R.string.cadera)
CADERA,
@EnumString(R.string.rodilla)
RODILLA,
@EnumString(R.string.tobillo)
TOBILLO,
@EnumString(R.string.otra)
OTRA, // Este valor se utiliza para introducir una articulación que no esté en esta lista
@EnumString(R.string.cervical)
CERVICAL,
@EnumString(R.string.toracica)
TORACICA,
@EnumString(R.string.lumbar)
LUMBAR;
```

Figura 64 - Ejemplo de uso biblioteca *EnumFormat*

Como se puede apreciar en esta imagen, las claves de cada una de las articulaciones están en español, pero el valor de cada una de éstas es un recurso de tipo *String* que, como ya se ha comentado, tendrá un valor u otro en función del idioma del dispositivo, lo que permite, de una forma muy sencilla internacionalizar la aplicación; ya que, éste fue el principal problema que se ha encontrado a la hora de implementar la traducción de la aplicación a otro idioma que no fuera el español.

De esta forma, además, si se quisiera traducir la aplicación a cualquier otro idioma, bastaría con definir un nuevo fichero *strings.xml* con los recursos de tipo *String* en ese nuevo idioma y ya se almacenaría la información de los tipos *Enum* en ese nuevo idioma sin necesidad de hacer ningún cambio extra.

Por último, *Enum Format* también dispone, como cabría esperar, de un método para devolver el valor asociado a la clave del *Enum* (*format(<E extends Enum<E>> e)*) dada la clave de éste.

7.1.2. Fases de implementación

En el Product Roadmap mostrado en el apartado 4.4.1.1 ya se mostró, de una manera muy básica, las tres distintas fases diferenciadas en las que se desarrollaría *TFG-Goniometer*. En concreto estas fases han sido:

- **Versión 0.1 (04/05/2017):** En esta primera fase, la aplicación aún se encontraba en plena fase de desarrollo; sin embargo, ya disponía de una interfaz de usuario muy básica con la que se podían realizar mediciones (ya se había implementado toda la lógica detrás para poder calcular el ángulo de desplazamiento del dispositivo con respecto a dos posiciones (inicial y final) del mismo), aunque todavía no se disponía de una Base de Datos que pudiera almacenarlas.

- **Versión 0.2 (04/06/2017):** En esta segunda fase, ya se disponía de una Base de Datos de tipo SQLite en la que se podían almacenar los perfiles de los pacientes, recuperar la información sobre los mismos y almacenar mediciones. Por supuesto, previamente al almacenamiento de los perfiles de pacientes en el sistema, ya se disponía de una interfaz de usuario que, valga la redundancia, el usuario podía rellenar para, finalmente, ser almacenada esa información en la Base de Datos.
- **Versión 1.0 o Release (04/07/2017):** En esta última fase, ya se había incorporado el resto de funcionalidad de la aplicación (acceso a las últimas 10 mediciones almacenadas, menú de información sobre la aplicación, borrado de perfiles de pacientes y mediciones del sistema y traducción de la misma al inglés); sin embargo, dada la rápida velocidad con la que se pudieron implementar estas tareas, esto permitió que se dispusiera de algo más de tiempo para poder realizar tareas de modificación de la estructura de la Base de Datos, así como de corrección de errores (tanto de interfaz de usuario como de funcionamiento de la aplicación).

Para la fecha de release de la aplicación, ésta ya era 100% funcional y estaba libre de errores tanto en español como en inglés.

7.2. Pruebas

En este segundo apartado, se va a comentar todo lo relativo a la importancia que tiene el realizar una correcta y exhaustiva fase de pruebas, con el fin de garantizar el buen funcionamiento de cualquier sistema; así como de la especificación y resultado de todas y cada una de las pruebas que se han realizado sobre *TFG-Goniometer*.

7.2.1. Introducción

Un conjunto de pruebas (test suite) suele orientarse a comprobar determinados aspectos de un sistema software (o de una parte del mismo). A continuación, se explicará, brevemente, los tipos de pruebas software que existen atendiendo a las directrices establecidas por ISTQB (International Software Testing Qualifications Board, de sus siglas en inglés).

- **Pruebas de software funcionales:** Este tipo de pruebas se centran en el comportamiento del sistema, subsistema o componente software descrito en la especificación de requisitos o casos de uso. Son definidas a partir de funciones o características y son consideradas, generalmente, **pruebas de caja negra** (*black box testing*), ya que, evalúan el comportamiento externo del sistema.
- **Pruebas de software no funcionales:** Este tipo de pruebas incluyen, entre otras, pruebas de rendimiento, carga, estrés, usabilidad, mantenibilidad... es decir, se centran en la forma en que trabaja el sistema. Se pueden ejecutar en todos los niveles de prueba (que se explicarán a continuación) y, dado que también evalúan el comportamiento externo del sistema, son consideradas **pruebas de caja negra**.
- **Pruebas de software estructurales:** Este tipo de pruebas es el término empleado por ISTQB para las pruebas en las que se evalúa el comportamiento interno del sistema, es decir, **pruebas de caja blanca** (*white box testing*). Se realizan aplicando técnicas estructurales y técnicas estáticas, en lugar de técnicas

basadas en especificación, utilizando el concepto de **cobertura** (*coverage*) para definir la extensión de la estructura que ha sido cubierta por el conjunto de pruebas (en porcentaje).

- **Pruebas de regresión y re-pruebas:** Las re-pruebas son aplicadas después de que un **error es detectado** y corregido, con la finalidad de verificar que el error ya no se está produciendo. Las pruebas de regresión se realizan sobre un componente ya probado, para verificar que no presenta nuevos errores cuando se realiza una modificación después de dichas pruebas. Se deben buscar nuevos errores tanto en el componente que se está probando, como en otros componentes afectados por el cambio. Este tipo de pruebas incluyen los tres tipos anteriores.

7.2.2. Niveles de prueba de software

En este apartado, se van a explicar los distintos niveles que existen en las pruebas de software, especificando en cada uno de ellos cómo se han llevado a cabo las pruebas para el caso concreto de *TFG-Goniometer*.

7.2.2.1. Pruebas de unidad

Las pruebas de unidad son la primera fase de las pruebas dinámicas y se realizan sobre cada módulo de software de manera independiente. El objetivo es comprobar que el módulo, entendido como una unidad funcional, está correctamente codificado.

Se enfocan en un programa o componente que desempeña una función específica que puede ser probada y que se asegura que funcione tal y como lo define la especificación del programa. Además, se realizan en fase de desarrollo, solamente después de que el programador considera que está libre de errores.

En el caso de *TFG-Goniometer*, estas pruebas se han aplicado a cada uno de los métodos y funciones definidas dentro de una clase. Es decir, cuando un método estaba terminado y libre de errores, se comprobaba, primero, que el flujo de ejecución del mismo era el adecuado para un determinado valor de entrada (prueba de caja blanca) y, posteriormente, se probaba el valor obtenido por éste para una sucesión de distintos valores de entrada (prueba de caja negra).

7.2.2.2. Pruebas de integración

El objetivo de las pruebas de integración es identificar errores introducidos por la combinación de programas o componentes probados unitariamente, con el fin de asegurar que la comunicación, enlaces y datos compartidos ocurren apropiadamente.

En este nivel se asegura que las distintas partes del programa trabajan correctamente. Antes de este tipo de pruebas, cada uno de los componentes ha tenido que pasar por pruebas de unidad, por lo que, el enfoque se centra en el flujo de control entre los distintos módulos y los datos intercambiados por éstos.

En el caso de *TFG-Goniometer*, teniendo en cuenta que se ha utilizado una arquitectura MVP, este tipo de pruebas se han centrado en varios puntos.

1. Inyección de dependencias de los Presentadores en los elementos de la Vista.

2. Correcta comunicación entre un elemento de la Vista (típicamente un Fragment) con su Presentador asociado.
3. Pruebas entre los Presentadores y el Modelo.
4. Pruebas entre el Modelo y el módulo *data* (que contiene toda la información relativa a la Base de Datos).

De esta manera, se ha podido verificar la correcta comunicación entre todos los componentes del sistema.

Mencionar, además, que todas las pruebas realizadas de este tipo fueron de caja negra.

7.2.2.3. Pruebas de sistema

Estas pruebas tienen como objetivo verificar que se han integrado adecuadamente todos los elementos del sistema y que realizan las operaciones apropiadas funcionando como un conjunto. Son similares a las pruebas de integración, pero con un alcance mucho más amplio.

En el caso de *TFG-Goniometer*, estas pruebas se han realizado con el fin de verificar el correcto funcionamiento de toda la aplicación dividiéndolas según los casos de uso presentados en este mismo documento.

Es decir, se ha comprobado exhaustivamente que, primero, el sistema ejecuta correctamente todos y cada uno de los casos de uso especificados (tanto la secuencia normal, como flujos alternativos y excepciones) centrándose en la obtención o no de la funcionalidad esperada (pruebas de caja negra); y, posteriormente, se ha hecho un seguimiento del flujo de control entre todos los componentes involucrados en un caso de uso, con el fin de verificar que este flujo es el esperado (pruebas de caja blanca).

7.2.2.4. Pruebas de aceptación

Estas pruebas son realizadas principalmente por los usuarios con el apoyo del equipo del proyecto. El propósito es confirmar que el sistema está terminado, que desarrolla puntualmente las necesidades de la organización y que es aceptado por los usuarios finales.

Las pruebas de aceptación pueden tener una duración de semanas o meses, descubriendo de esta manera errores que pueden ir degradando el sistema. Es recomendable que sean realizadas en el entorno en el que se va a explotar el sistema, incluyendo el personal que lo va a manejar.

En el caso de *TFG-Goniometer*, este tipo de pruebas son, precisamente, las que se han llevado a cabo al realizar la comparación de la precisión de la aplicación frente al dispositivo Kinect.

7.2.3. Pruebas realizadas

En este apartado, se van a detallar las pruebas que se han realizado (divididas por casos de uso), mostrando, para cada una de ellas, una breve descripción, la acción realizada por el usuario, el resultado esperado y el obtenido y la solución en caso de que en alguna prueba se obtenga un valor incorrecto.

Dado que el objetivo de estas pruebas era comprobar el correcto cumplimiento de los requisitos establecidos al inicio del desarrollo de esta aplicación, todas ellas son de **caja negra** (las pruebas de caja blanca ya se realizaron durante el desarrollo de la misma).

7.2.3.1. UC-0001: Añadir_Medicion

Prueba-0001	Adecuación del tamaño IU goniómetro a la pantalla del dispositivo
Descripción	Cuando el Usuario entra en la pantalla que se muestra el goniómetro, tanto éste como el botón de reiniciar ajustan su tamaño al de la pantalla del dispositivo.
Acción	En la vista principal de la aplicación, pulsar el botón <i>Nueva medición</i> .
Resultado esperado	El tamaño del goniómetro y el del botón reiniciar es ajustado para que se visualicen por completo independientemente del tamaño de la pantalla del dispositivo.
Resultado obtenido	Incorrecto
Solución implementada	Error: El tamaño de los elementos View que forman el goniómetro y el botón reiniciar tienen un tamaño fijo en dp. Solución: Adaptar el tamaño de éstos en función del tamaño de la pantalla del dispositivo cuando se cree el Fragment correspondiente.

Tabla 21 - Prueba-0001

Tras hacer una re-prueba incorporando esta solución, el resultado es **Correcto**.

Prueba-0002	Fijación de la orientación de la pantalla en posición <i>Landscape</i>
Descripción	Cuando el Usuario entra en la pantalla que se muestra el goniómetro, la orientación de la pantalla se queda fijada en <i>Landscape</i> sin permitir que ésta pueda cambiar hasta que el Usuario no abandone esta vista.
Acción	En la vista principal de la aplicación, pulsar el botón <i>Nueva medición</i> .
Resultado esperado	La posición de la pantalla es fijada en <i>Landscape</i> y, aunque el Usuario intente girar el dispositivo para cambiar la orientación de la misma, la posición no variará.
Resultado obtenido	Correcto
Solución implementada	N/A

Tabla 22 - Prueba-0002

Prueba-0003	Inicio de una medición
Descripción	El Usuario pulsa sobre el goniómetro cuando éste marca 0.0 para comenzar una medición.
Acción	En la vista en la que aparece la interfaz gráfica del goniómetro, pulsar sobre el mismo.
Resultado esperado	El valor numérico mostrado por el goniómetro junto con la barra que representa el desplazamiento del mismo comienzan a mostrar la variación del ángulo de desplazamiento en función del movimiento realizado por el Usuario.
Resultado obtenido	Correcto
Solución implementada	N/A

Tabla 23 - Prueba-0003

Prueba-0004	Pulsación del botón <i>Reiniciar</i> antes de iniciar una medición
Descripción	El Usuario pulsa el botón <i>Reiniciar</i> cuando el goniómetro todavía marca 0.0.
Acción	En la vista en la que aparece la interfaz gráfica del goniómetro, pulsar sobre el botón <i>Reiniciar</i> .
Resultado esperado	La interfaz gráfica no muestra variación alguna
Resultado obtenido	Correcto
Solución implementada	N/A

Tabla 24 - Prueba-0004

Prueba-0005	Pulsación del botón <i>ALMACENAR MEDICIÓN</i> sin haber realizado una medición
Descripción	El Usuario pulsa el botón <i>ALMACENAR MEDICIÓN</i> antes de haber realizado una medición completamente.
Acción	En la vista en la que aparece la interfaz gráfica del goniómetro, pulsar sobre el goniómetro (opcionalmente) y pulsar sobre el botón <i>ALMACENAR MEDICIÓN</i> .
Resultado esperado	Se muestra un diálogo, con el que el usuario tiene que interactuar, indicando este problema
Resultado obtenido	Correcto
Solución implementada	N/A

Tabla 25 - Prueba-0005

Prueba-0006	Pulsación del botón <i>Reiniciar</i> mientras se está realizando una medición
Descripción	El Usuario pulsa el botón <i>Reiniciar</i> antes de haber terminado de realizar una medición.
Acción	En la vista en la que aparece la interfaz gráfica del goniómetro, pulsar sobre el goniómetro, pulsar de nuevo sobre éste (opcionalmente) y pulsar sobre el botón <i>Reiniciar</i> .
Resultado esperado	La interfaz gráfica vuelve al mismo estado que en el momento de su creación (ángulo en 0.0 y barra de desplazamiento en el origen).
Resultado obtenido	Correcto
Solución implementada	N/A

Tabla 26 - Prueba-0006

Prueba-0007	Realización de una medición completamente y avance hasta la siguiente vista
Descripción	El Usuario ha realizado una medición de manera completa y decide almacenarla.
Acción	En la vista en la que aparece la interfaz gráfica del goniómetro, pulsar sobre el goniómetro, pulsar de nuevo sobre éste para indicar el fin de la medición, pulsar sobre el botón <i>ALMACENAR MEDICIÓN</i> y seleccionar la opción <i>Seleccionar paciente existente</i> .
Resultado esperado	El sistema no muestra ningún mensaje indicando problemas durante la medición y se avanza hasta la siguiente vista.

Resultado obtenido	Correcto
Solución implementada	N/A

Tabla 27 - Prueba-0007

Prueba-0008	Realización de una medición completamente y cancelación del proceso de almacenarla
Descripción	El Usuario ha realizado una medición de manera completa y decide almacenarla, pero en el último momento decide cancelar la acción.
Acción	En la vista en la que aparece la interfaz gráfica del goniómetro, pulsar sobre el goniómetro, pulsar de nuevo sobre éste para indicar el fin de la medición, pulsar sobre el botón <i>ALMACENAR MEDICIÓN</i> y seleccionar la opción <i>Cancelar</i> .
Resultado esperado	El sistema cierra el diálogo con el usuario, mostrando el goniómetro en el estado final después de haber realizado la medición.
Resultado obtenido	Correcto
Solución implementada	N/A

Tabla 28 - Prueba-0008

Prueba-0009	Adición de una medición realizada a un paciente
Descripción	El Usuario decide almacenar la medición realizada en el perfil de un paciente en concreto.
Acción	En la vista en la que aparece el perfil completo (datos personales y mediciones realizadas) de un paciente, pulsar el botón <i>AÑADIR NUEVA ENTRADA</i> .
Resultado esperado	Se muestra la vista con los datos personales del paciente junto con una serie de campos extra a rellenar por el Usuario con respecto a la medición.
Resultado obtenido	Incorrecto
Solución implementada	Error: La información personal del paciente no es mostrada tras haber realizado unos cambios en la tabla <i>Paciente</i> de la Base de Datos. El Presentador no enviaba la información del campo <i>id</i> a la hora de crear el <i>Fragment</i> que muestra la información del paciente junto con los campos a rellenar de la medición. Solución: Hacer que el Presentador envíe el <i>id</i> en el objeto <i>Bundle</i> a la hora de crear este <i>Fragment</i> .

Tabla 29 - Prueba-0009

Prueba-0010	Carga de las posibles articulaciones en función del lado seleccionado
Descripción	El Usuario ha seleccionado un lado para la articulación (izquierdo, derecho o columna) y, en función de éste, se muestran unas articulaciones para elegir u otras.
Acción	En la vista en la que aparecen los datos personales del paciente junto con los campos a rellenar sobre la medición, seleccionar un lado para la articulación y, posteriormente, intentar seleccionar una articulación.

Resultado esperado	La lista con las posibles articulaciones a mostrar es la adecuada, tal y como está especificado en el Modelo de Dominio.
Resultado obtenido	Correcto
Solución implementada	N/A

Tras hacer una re-prueba incorporando esta solución, el resultado es **Correcto**.

Tabla 30 - Prueba-0010

Prueba-0011	Carga de los posibles movimientos en función de la articulación seleccionada
Descripción	El Usuario ha seleccionado una articulación en concreto y, en función de ésta, se muestra un listado de movimientos para elegir u otros.
Acción	En la vista en la que aparecen los datos personales del paciente junto con los campos a rellenar sobre la medición, seleccionar un lado para la articulación y una articulación en concreto, y, posteriormente, intentar seleccionar un movimiento para ésta.
Resultado esperado	La lista con los posibles movimientos a mostrar es la adecuada, tal y como está especificado en el Modelo de Dominio.
Resultado obtenido	Correcto
Solución implementada	N/A

Tabla 31 - Prueba-0011

Prueba-0012	Selección del valor <i>Otra</i> para una articulación
Descripción	El Usuario ha seleccionado la articulación <i>Otra</i> en el listado de posibles articulaciones.
Acción	En la vista en la que aparecen los datos personales del paciente junto con los campos a rellenar sobre la medición, seleccionar el lado izquierdo o derecho, y en el campo articulación seleccionar <i>Otra</i> .
Resultado esperado	El sistema muestra un cuadro de texto a rellenar obligatoriamente por el usuario, además de cargar el campo movimiento con todos los posibles movimientos que se disponen de forma predefinida en la aplicación (incluyendo el valor <i>Otro</i>).
Resultado obtenido	Correcto
Solución implementada	N/A

Tabla 32 - Prueba-0012

Prueba-0013	Selección del valor <i>Otro</i> para un movimiento
Descripción	El Usuario ha seleccionado el movimiento <i>Otro</i> en el listado de posibles movimientos.
Acción	En la vista en la que aparecen los datos personales del paciente junto con los campos a rellenar sobre la medición, seleccionar un lado y una articulación en concreto y en el campo movimiento seleccionar <i>Otro</i> .
Resultado esperado	El sistema muestra un cuadro de texto a rellenar obligatoriamente por el usuario.

Resultado obtenido	Correcto
Solución implementada	N/A

Tabla 33 - Prueba-0013

Para el caso de esta prueba, comentar, además, que no cualquier combinación de lado y articulación dejan elegir el valor *Otro* en el campo movimiento, tal y como está especificado en el Modelo de Dominio de este mismo documento.

Prueba-0014	Selección del valor <i>Otro</i> para un tipo de movimiento
Descripción	El Usuario ha seleccionado el tipo de movimiento <i>Otro</i> en el listado de posibles tipos de movimiento.
Acción	En la vista en la que aparecen los datos personales del paciente junto con los campos a rellenar sobre la medición, seleccionar un lado, una articulación y un movimiento en concreto y en el campo tipo de movimiento seleccionar <i>Otro</i> .
Resultado esperado	El sistema muestra un cuadro de texto a rellenar obligatoriamente por el usuario.
Resultado obtenido	Correcto
Solución implementada	N/A

Tabla 34 - Prueba-0014

Prueba-0015	Almacenamiento de una medición sin haber rellenado todos los campos obligatorios
Descripción	El Usuario ha pulsado el botón <i>HECHO</i> (indicando que ha terminado de introducir la información sobre la medición) sin haber rellenado todos los campos obligatorios (marcados con un *).
Acción	En la vista en la que aparecen los datos personales del paciente junto con los campos a rellenar sobre la medición, seleccionar (o no) un lado para la articulación, seleccionar (o no) una articulación en concreto, seleccionar (o no) un movimiento para ésta, seleccionar (o no) un tipo de movimiento para ésta y pulsar el botón <i>HECHO</i> .
Resultado esperado	El sistema muestra un mensaje al Usuario indicando el error de que no ha rellenado todos los campos obligatorios.
Resultado obtenido	Correcto
Solución implementada	N/A

Tabla 35 - Prueba-0015

Prueba-0016	Almacenamiento de una medición produciéndose un error al guardarla
Descripción	El Usuario ha pulsado el botón <i>HECHO</i> (indicando que ha terminado de introducir la información sobre la medición), pero se ha producido un error al intentar almacenarla en la Base de Datos.

Acción	En la vista en la que aparecen los datos personales del paciente junto con los campos a rellenar sobre la medición, rellenar toda la información sobre la misma y pulsar el botón <i>HECHO</i> .
Resultado esperado	El sistema muestra un mensaje al Usuario indicando que se ha producido un error al intentar almacenar la medición en la Base de Datos.
Resultado obtenido	Correcto
Solución implementada	N/A

Tabla 36 - Prueba-0016

Prueba-0017	Almacenamiento de una medición
Descripción	El Usuario ha pulsado el botón <i>HECHO</i> (indicando que ha terminado de introducir la información sobre la medición).
Acción	En la vista en la que aparecen los datos personales del paciente junto con los campos a rellenar sobre la medición, rellenar toda la información sobre la misma y pulsar el botón <i>HECHO</i> .
Resultado esperado	El sistema muestra un mensaje al Usuario indicando que se ha almacenado correctamente la medición y muestra el perfil del paciente actualizado con la información de esa nueva medición.
Resultado obtenido	Correcto
Solución implementada	N/A

Tabla 37 - Prueba-0017

Prueba-0018	Bloqueo del menú lateral durante la medición
Descripción	El menú lateral que permite navegar por las partes principales de la aplicación es ocultado y bloqueado mientras se pasa por el proceso para almacenar una medición.
Acción	Desde el momento en el que se muestra el listado de pacientes al que asignar la medición, el botón para abrir el menú lateral (así como su apertura mediante un desplazamiento horizontal desde un lateral de la pantalla) desaparece, evitando así que el Usuario vaya a otras partes de la aplicación sin haber terminado de almacenar la medición. De esta manera se evita que la aplicación entre en un estado inconsistente y que puede provocar fallos de funcionamiento.
Resultado esperado	El botón del menú desaparece y, ni pulsando donde se encontraba antes el menú, ni deslizando desde el lateral izquierdo de la pantalla, este menú se puede abrir. La única forma de salir de la medición es pulsando el botón <i>Atrás</i> .
Resultado obtenido	Incorrecto
Solución implementada	Error: Inicialmente, sólo se había ocultado el botón para abrir el menú lateral; sin embargo, éste se podía abrir deslizando desde el lateral izquierdo de la pantalla. Solución: Se bloqueó esta segunda opción para abrir el menú desde que se crea la <i>Activity</i> principal de la aplicación, permitiendo, de esta manera, que el menú se pueda abrir únicamente pulsando sobre el botón habilitado para ello.

Tabla 38 - Prueba-0018

Tras hacer una re-prueba incorporando esta solución, el resultado es **Correcto**.

7.2.3.2. UC-0002: Visualizar_10_Ultimas_Mediciones

Prueba-0019	Visualización de las 10 últimas mediciones almacenadas
Descripción	El Usuario desea ver cuáles son las 10 últimas mediciones que ha almacenado en el sistema junto con el paciente al que se refieren.
Acción	En la vista principal de la aplicación, pulsar el botón <i>Últimas 10 mediciones</i> .
Resultado esperado	El sistema muestra la vista que incluye un listado (en forma de tablas) de las últimas 10 mediciones almacenadas, ordenadas de más a menos reciente.
Resultado obtenido	Correcto
Solución implementada	N/A

Tabla 39 - Prueba-0019

7.2.3.3. UC-0003: Borrar_Medicion

Prueba-0020	Borrado de una medición y cancelación del proceso de borrado
Descripción	El Usuario desea borrar una de las últimas 10 mediciones que ha realizado, pero al momento de hacerlo, desea cancelar la operación.
Acción	En la vista de las 10 últimas mediciones almacenadas, pulsar el botón para borrarla (representado por el icono de una papelera) y en el diálogo que aparece, pulsar el botón <i>CANCELAR</i> , o pulsar fuera del diálogo.
Resultado esperado	El sistema muestra de nuevo el listado de las últimas 10 mediciones sin haber eliminado esa medición del sistema.
Resultado obtenido	Correcto
Solución implementada	N/A

Tabla 40 - Prueba-0020

Prueba-0021	Borrado de una medición y confirmación del proceso de borrado
Descripción	El Usuario desea borrar una de las últimas 10 mediciones que ha realizado.
Acción	En la vista de las 10 últimas mediciones almacenadas, pulsar el botón para borrarla (representado por el icono de una papelera) y en el diálogo que aparece, pulsar el botón <i>ELIMINAR</i> .
Resultado esperado	El sistema muestra un mensaje indicando al Usuario que la medición se ha eliminado con éxito, y muestra de nuevo el listado de las últimas 10 mediciones tras haber eliminado esa medición del sistema.
Resultado obtenido	Correcto
Solución implementada	N/A

Tabla 41 - Prueba-0021

Prueba-0022	Borrado de una medición produciéndose un error al borrarla
Descripción	El Usuario desea borrar una de las últimas 10 mediciones que ha realizado.
Acción	En la vista de las 10 últimas mediciones almacenadas, pulsar el botón para borrarla (representado por el icono de una papelera) y en el diálogo que aparece, pulsar el botón <i>ELIMINAR</i> .
Resultado esperado	El sistema muestra un mensaje al Usuario indicando que se ha producido un error al intentar eliminar la medición de la Base de Datos.
Resultado obtenido	Correcto
Solución implementada	N/A

Tabla 42 - Prueba-0022

7.2.3.4. UC-0004: Visualizar_Perfiles_Pacientes

Prueba-0023	Visualización del listado de pacientes almacenados en el sistema
Descripción	El Usuario desea visualizar todo el listado de pacientes almacenados.
Acción	En la vista principal de la aplicación, pulsar el botón <i>Perfiles de Pacientes</i> .
Resultado esperado	El sistema muestra la vista que incluye el listado con todos los nombres de los pacientes almacenados, además de un cuadro de búsqueda para poder filtrar los pacientes que se muestran, y un botón para añadir un nuevo paciente al sistema.
Resultado obtenido	Correcto
Solución implementada	N/A

Tabla 43 - Prueba-0023

7.2.3.5. UC-0005: Filtrar_Pacientes

Prueba-0024	Filtrado de pacientes
Descripción	El Usuario desea filtrar el listado de pacientes que aparecen.
Acción	En la vista que muestra el listado de pacientes en el sistema, elegir un criterio de filtro de entre los que hay disponibles (Nombre, ID o Tags) y escribir algún carácter o palabra en el cuadro de búsqueda.
Resultado esperado	El sistema muestra el listado de pacientes actualizado en función de aquellos que cumplan con los criterios de búsqueda establecidos.
Resultado obtenido	Incorrecto
Solución implementada	<p>Error: Cuando ningún paciente coincide con los criterios de búsqueda, se muestran todos los almacenados en el sistema. El problema radica en que se devuelve <i>null</i> cuando no coincide ningún paciente, en vez de una lista de tamaño 0; y cuando se recibe <i>null</i> se muestran todos los pacientes almacenados.</p> <p>Solución: No realizar la comprobación del tamaño de la lista a devolver (devolviendo <i>null</i> en caso de ser de tamaño 0) para devolverla siempre (aunque tenga tamaño 0).</p>

Tabla 44 - Prueba-0024

Tras hacer una re-prueba incorporando esta solución, el resultado es **Correcto**.

7.2.3.6. UC-0006: Crear_Perfil_Paciente

Prueba-0025	Creación del perfil de un paciente
Descripción	El Usuario desea crear un nuevo perfil de paciente para almacenarlo en el sistema.
Acción	En la vista que muestra el listado de pacientes almacenados en el sistema, pulsar el botón para crear un nuevo perfil.
Resultado esperado	El sistema muestra la vista que incluye todos los campos a rellenar por el usuario sobre el nuevo paciente, marcando con un * aquellos que son obligatorios.
Resultado obtenido	Correcto
Solución implementada	N/A

Tabla 45 - Prueba-0025

Prueba-0026	Almacenamiento del perfil de un paciente sin haber rellenado todos los campos obligatorios
Descripción	El Usuario desea almacenar el perfil del paciente que acaba de rellenar, pero uno o varios de los campos marcados con un * están vacíos.
Acción	En la vista que muestra los campos a rellenar sobre el nuevo paciente, rellenar la información parcialmente y pulsar el botón GUARDAR.
Resultado esperado	El sistema muestra un mensaje al Usuario indicando el error de que no ha rellenado todos los campos obligatorios.
Resultado obtenido	Correcto
Solución implementada	N/A

Tabla 46 - Prueba-0026

Prueba-0027	Cancelación del proceso de creación del perfil de un paciente
Descripción	El Usuario decide volver atrás en la aplicación sin haber terminado de almacenar el perfil del paciente.
Acción	En la vista que muestra los campos a rellenar sobre el nuevo paciente, rellenar (o no) la información del mismo y pulsar el botón Atrás del dispositivo.
Resultado esperado	El sistema muestra el listado de pacientes almacenados en el mismo sin haber incluido el paciente que estaba creando el Usuario.
Resultado obtenido	Correcto
Solución implementada	N/A

Tabla 47 - Prueba-0027

Prueba-0028	Almacenamiento del perfil de un paciente produciéndose un error al guardarlo
Descripción	El Usuario ha pulsado el botón GUARDAR (indicando que ha terminado de introducir la información sobre el paciente), pero se ha producido un error al intentar almacenarlo en la Base de Datos.
Acción	En la vista que muestra los campos a rellenar sobre el nuevo paciente, rellenar toda la información del mismo (al menos, los campos obligatorios) y pulsar el botón GUARDAR.

Resultado esperado	El sistema muestra un mensaje al Usuario indicando que se ha producido un error al intentar almacenar el paciente en la Base de Datos.
Resultado obtenido	Correcto
Solución implementada	N/A

Tabla 48 - Prueba-0028

Prueba-0029	Almacenamiento del perfil de un paciente
Descripción	El Usuario ha pulsado el botón <i>GUARDAR</i> (indicando que ha terminado de introducir la información sobre el paciente).
Acción	En la vista que muestra los campos a rellenar sobre el nuevo paciente, rellenar toda la información del mismo (al menos, los campos obligatorios) y pulsar el botón <i>GUARDAR</i> .
Resultado esperado	El sistema muestra un mensaje al Usuario indicando que se ha almacenado correctamente el paciente y muestra el listado de pacientes almacenados en el sistema, actualizado con el nuevo paciente introducido por el Usuario.
Resultado obtenido	Correcto
Solución implementada	N/A

Tabla 49 - Prueba-0029

Prueba-0030	Almacenamiento del perfil de un paciente que ya existe en el sistema
Descripción	El Usuario intenta almacenar el perfil de un paciente cuya combinación de nombre e ID ya existe en la Base de Datos.
Acción	En la vista que muestra los campos a rellenar sobre el nuevo paciente, rellenar toda la información del mismo (al menos, los campos obligatorios) y pulsar el botón <i>GUARDAR</i> .
Resultado esperado	El sistema muestra un mensaje al Usuario indicando que ya existe un paciente en el sistema con esa combinación de nombre e ID.
Resultado obtenido	Incorrecto
Solución implementada	Error: No se había incluido ningún método que gestionara este comportamiento de la aplicación, por lo que, se producía una excepción y la aplicación se detenía. Solución: Capturar la excepción lanzada y enviar un mensaje al Usuario indicando esta situación de error.

Tabla 50 - Prueba-0030

Tras hacer una re-prueba incorporando esta solución, el resultado es **Correcto**.

7.2.3.7. UC-0007: Visualizar_Informacion_Paciente

Prueba-0031	Visualización de la información de un paciente
Descripción	El Usuario ha seleccionado un paciente para ver toda la información relacionada con él (tanto datos personales como mediciones realizadas sobre él).
Acción	En la vista que muestra el listado de pacientes almacenados en el sistema, pulsar sobre uno de ellos.

Resultado esperado	El sistema muestra una nueva vista con toda la información que se dispone de ese paciente (tanto datos personales como mediciones que se le han realizado), además de un botón para borrar el perfil del mismo.
Resultado obtenido	Correcto
Solución implementada	N/A

Tabla 51 - Prueba-0031

Prueba-0032	Visualización de información del paciente cuya longitud excede el ancho del cuadro de texto
Descripción	La información introducida por el Usuario con respecto a un paciente puede exceder el ancho del cuadro de texto, provocando que ésta no pueda ser leído al completo.
Acción	Entrar en la vista que muestra el perfil de un paciente.
Resultado esperado	La información contenida en el cuadro de texto se puede desplazar horizontalmente para permitir su lectura completa.
Resultado obtenido	Incorrecto
Solución implementada	Error: Cuando la información del cuadro de texto superaba el ancho del mismo, ésta se cortaba y se hacía imposible ver el texto al completo. Solución: Cambiar los elementos <i>TextView</i> empleados para mostrar la información por elementos <i>EditText</i> que sí pueden ser desplazados horizontalmente para ver toda la información.

Tabla 52 - Prueba-0032

Tras hacer una re-prueba incorporando esta solución, el resultado es **Correcto**.

7.2.3.8. UC-0008: Borrar_Perfil_Paciente

Prueba-0033	Borrado de un perfil de paciente y cancelación del proceso de borrado
Descripción	El Usuario desea borrar el perfil de un paciente (incluyendo tanto datos personales como mediciones que se le han realizado), pero al momento de hacerlo, desea cancelar la operación.
Acción	En la vista que muestra toda la información relativa a un paciente en concreto, pulsar el icono para borrarlo (representado por el icono de una papelera) y en el diálogo que aparece, pulsar el botón <i>CANCELAR</i> , o pulsar fuera del diálogo.
Resultado esperado	El sistema muestra de nuevo la vista con toda la información relativa al paciente (incluyendo tanto datos personales como mediciones que se le han realizado).
Resultado obtenido	Correcto
Solución implementada	N/A

Tabla 53 - Prueba-0033

Prueba-0034	Borrado de un perfil de paciente y confirmación del proceso de borrado
Descripción	El Usuario desea borrar el perfil de un paciente (incluyendo tanto datos personales como mediciones que se le han realizado).
Acción	En la vista que muestra toda la información relativa a un paciente en concreto, pulsar el icono para borrarlo (representado por el icono de una papelera) y en el diálogo que aparece, pulsar el botón <i>ELIMINAR</i> .
Resultado esperado	El sistema muestra un mensaje al Usuario indicando que el paciente y sus mediciones han sido eliminados con éxito, y muestra el listado con todos los pacientes almacenados en el sistema, actualizado tras haber eliminado el paciente.
Resultado obtenido	Correcto
Solución implementada	N/A

Tabla 54 - Prueba-0034

Prueba-0035	Borrado de un perfil de paciente produciéndose un error al borrarlo
Descripción	El Usuario desea borrar el perfil de un paciente (incluyendo tanto datos personales como mediciones que se le han realizado).
Acción	En la vista que muestra toda la información relativa a un paciente en concreto, pulsar el icono para borrarlo (representado por el icono de una papelera) y en el diálogo que aparece, pulsar el botón <i>ELIMINAR</i> .
Resultado esperado	El sistema muestra un mensaje al Usuario indicando que se ha producido un error al intentar eliminar el paciente de la Base de Datos.
Resultado obtenido	Correcto
Solución implementada	N/A

Tabla 55 - Prueba-0035

Las siguientes pruebas que se muestran, no son relativas a un único caso de uso en particular, sino que, se han realizado sobre la aplicación de manera global.

Prueba-0036	Configuración del menú lateral
Descripción	La opción que aparece marcada en el menú lateral cambia en función de la parte de la aplicación en la que se encuentre el Usuario, independientemente de que se haya accedido a esa parte utilizando el menú, o mediante la navegación a través de los distintos botones de la aplicación.
Acción	Navegar por las distintas partes de la aplicación.
Resultado esperado	Cuando el Usuario va a una determinada parte de la aplicación, o vuelve a otra distinta, este cambio se ve reflejado en la opción que aparece seleccionada en el menú lateral.
Resultado obtenido	Incorrecto
Solución implementada	Error: Cuando el Usuario pulsa el botón <i>Atrás</i> para volver a otra parte de la aplicación, el menú lateral no refleja ese cambio y aparece como marcada una opción distinta a la real.

	Solución: Configurar los <i>Fragment</i> de tal manera, que cuando sean creados/restaurados, éstos modifiquen la posición que aparece seleccionada en el menú lateral.
--	---

Tabla 56 - Prueba-0036

Tras hacer una re-prueba incorporando esta solución, el resultado es **Correcto**.

Prueba-0037	Configuración de la orientación de la pantalla
Descripción	La orientación de la pantalla queda fijada en una posición u otra en función de dónde se encuentre el Usuario dentro de la aplicación.
Acción	Navegar por las distintas partes de la aplicación.
Resultado esperado	Cuando el Usuario navega por cualquier parte de la aplicación que no sea la vista en la que se muestra el goniómetro, la orientación de la pantalla queda fijada en <i>Portrait</i> y no puede ser modificada por el Usuario, aunque gire el dispositivo.
Resultado obtenido	Correcto
Solución implementada	N/A

Tabla 57 - Prueba-0037

Prueba-0038	Adecuación del tamaño de la tabla que muestra las mediciones
Descripción	La tabla en la que se muestran las mediciones de un paciente puede no verse de manera completa en dispositivos con un tamaño de pantalla menor a 5.2".
Acción	Entrar en el perfil de un paciente al que se hayan realizado mediciones, o entrar en el listado de las 10 últimas mediciones almacenadas.
Resultado esperado	Cuando el tamaño de la tabla excede el ancho de la pantalla del dispositivo, ésta se puede desplazar horizontalmente, de tal manera que siempre se pueden visualizar los datos que contiene, independientemente del tamaño de la pantalla que tenga el dispositivo.
Resultado obtenido	Incorrecto
Solución implementada	Error: Aunque la tabla que almacena las mediciones estaba configurada para adaptarse al ancho de la pantalla del dispositivo, cuando el texto que ésta contenía era demasiado largo, se hacía inevitable que no se pudiera mostrar la tabla entera. Solución: Utilizar un elemento <i>HorizontalScrollView</i> para contener al <i>TableLayout</i> que engloba las mediciones y, así, poder desplazar horizontalmente la tabla en caso de que no se pudiera mostrar entera.

Tabla 58 - Prueba-0038

Tras hacer una re-prueba incorporando esta solución, el resultado es **Correcto**.

Todas estas pruebas se han realizado tanto en dispositivos con el idioma en español, como en dispositivos con el idioma en inglés (Estados Unidos). La inmensa mayoría de las pruebas han obtenido los mismos resultados, pero en algunas que no han presentado problemas en español, sí los han presentado en inglés.

Estas pruebas son:

- **Pruebas 0012, 0013 y 0014:** Estas tres pruebas presentaban todas el mismo error. En un principio se realizaba la comprobación de si se había introducido o no el valor *Otro/a* con una cadena de texto “hardcoded”, con lo cual, en el momento en que se seleccionaba el valor *Other*, ambas cadenas no coincidían y, por tanto, no se mostraba el cuadro de texto a rellenar por el Usuario.

La solución fue cambiar esa cadena de texto literal por un recurso de tipo *String* en *res/values-XX/strings.xml* (que varía con el idioma del dispositivo) como una expresión regular, y utilizar el método *matches* de la clase *String* para comprobar si se había seleccionado o no el valor *Otro/a*.

- **Prueba 0024:** El error en esta prueba era exactamente el mismo que en las tres que se acaban de nombrar. La comprobación del filtro seleccionado por el Usuario se hacía con una cadena de texto “hardcoded” y, por tanto, cuando hacía la comparación entre el nombre del filtro en inglés y en español, ambas cadenas no coincidían y, por tanto, esta funcionalidad no se podía utilizar.

La solución, muy similar al caso anterior, fue cambiar esa cadena de texto literal por un recurso de tipo *String* a la hora de comprobar qué filtro había seleccionado el Usuario.

Por último, es preciso comentar que, dado que *TFG-Goniometer* no está pensada para funcionar en dos idiomas al mismo tiempo (es decir, almacenar mediciones/pacientes en un idioma y recuperar esa información en otro) el resultado obtenido de varias de las pruebas presentadas es *Incorrecto* cuando se utiliza de esta manera.

Esto es debido, sobre todo, a la forma en que se almacenan la fecha y la hora según el idioma, por lo que, su formato es adaptado a esa condición, pudiendo provocar errores en el comportamiento de la aplicación si, como se ha mencionado, se almacena información en un idioma y se recupera en otro distinto.

Capítulo 8

Estudio comparativo Kinect - App

8.1. Introducción

En este apartado, se van a mostrar las pruebas y los resultados que se han obtenido tanto con Kinect como la aplicación desarrollada en este proyecto (*TFG-Goniometer*) con el fin de comparar la precisión de ambos sistemas.

En este punto, es importante destacar que la obtención del ángulo de desplazamiento con Kinect se realiza proyectando el movimiento de la articulación sobre un plano en **2 dimensiones**, y es en este plano donde se mide el ángulo formado por la misma desde la posición de inicio hasta la posición final.

Sin embargo, con *TFG-Goniometer*, la obtención del ángulo de desplazamiento se realiza en **3 dimensiones** mediante el uso de cuaterniones (tal y como se ha explicado en el apartado 3.2), es decir, si se quiere realizar un movimiento de una articulación en un plano formado por 2 ejes (X e Y, por ejemplo), un movimiento (bien sea voluntario o no) en el tercer eje (Z, siguiendo con el ejemplo) va a afectar a la lectura final obtenida por el goniómetro.

Debido a esto, sumando a las posibles imprecisiones que puedan tener ambos sistemas (ya que, no han podido ser validados fuera de las condiciones de laboratorio, ni con un muestreo suficiente para obtener estadísticas de precisión fiables), más las imprecisiones de las personas encargadas de realizar estas pruebas (ninguna de ellas es fisioterapeuta o profesional de la salud con experiencia en medir rangos articulares), es posible que, en algunos movimientos, las lecturas obtenidas por ambos sistemas disten más de lo que cabría esperar.

8.2. Mediciones realizadas

Todas las mediciones que se han realizado para comparar ambos sistemas se han centrado en las articulaciones del hombro y del codo, ya que, éstas son las que mejor detecta el sistema Kinect (además de que, en un principio, el desarrollo del sistema Kinect sólo contemplaba las articulaciones del torso). En concreto, se han medido los movimientos de abducción, flexión, rotación externa e interna y flexión horizontal del hombro, y la flexión del codo.

Para todos estos movimientos, se han medido los valores obtenidos cada 30º desde la posición de inicio del movimiento hasta donde la articulación lo permitiera, haciendo hincapié en los valores 60º y 90º, para los cuales se ha decidido realizar varias mediciones.

Articulación	Hombro izquierdo		
Movimiento	Abducción		
Resultados			
TFG-Goniometer	Kinect		
30º	27º		
60º	56º	52º	53º
90º	85º	85º	87º
120º	116º		
150º	147º		
170º	180º		

Tabla 59 – Comparación Kinect y App - Abducción hombro

En el caso de este movimiento, el último ángulo medido es de 170º y no 180º como cabría esperar debido a las limitaciones de movilidad articular del sujeto de pruebas con el que se han realizado.

Articulación	Hombro izquierdo		
Movimiento	Flexión		
Resultados			
TFG-Goniometer	Kinect		
30º	36º		
60º	65º	65º	71º
90º	110º	112º	112º
120º	146º		
150º	180º		

Tabla 60 - Comparación Kinect y App - Flexión hombro

Articulación	Codo izquierdo		
Movimiento	Flexión		
Resultados			
TFG-Goniometer	Kinect		
30º	24º		
60º	51º	59º	49º
90º	100º	104º	85º
120º	129º		

Tabla 61 - Comparación Kinect y App - Flexión codo

Articulación	Hombro derecho		
Movimiento	Rotación		
Resultados			
TFG-Goniometer	Kinect		
30º	34º		
60º	54º	56º	58º
90º	80º	76º	80º
-30º	-28º		

Tabla 62 - Comparación Kinect y App - Rotación externa/interna hombro

Como la posición de inicio de este movimiento se da cuando el hombro forma un ángulo de 90° con el torso, las mediciones con signo positivo hacen referencia al movimiento de rotación externa del hombro, y la medición con signo negativo hace referencia al movimiento de rotación interna.

Es preciso destacar, además, que la lectura obtenida por *TFG-Goniometer* para el caso de rotación interna es de 30° (mide el ángulo de desplazamiento en términos absolutos); sin embargo, se ha decidido incluir el signo negativo para equipararlo a la lectura obtenida por el sistema Kinect (éste sí distingue entre ángulos de desplazamiento positivos y negativos).

Articulación	Hombro derecho		
Movimiento	Flexión horizontal		
Resultados			
TFG-Goniometer	Kinect		
30º	23º		
60º	52º	57º	58º
90º	83º	84º	82º
120º	110º		
150º	124º		

Tabla 63 - Comparación Kinect y App - Flexión horizontal hombro

8.3. Conclusiones

La diferencia que existe entre las mediciones realizadas con uno y otro sistema se puede considerar aceptable considerando que las personas que se han encargado de realizarlas no eran expertas en este ámbito, y la diferencia entre ambos sistemas a la hora de calcular las rotaciones (tal y como se ha comentado en el apartado 8.1).

El movimiento en el que, sin duda, se han logrado obtener lecturas más próximas entre ambos sistemas es la abducción de hombro. Se trata de un movimiento que Kinect detecta con bastante precisión, además de haber prestado especial atención con la aplicación para evitar movimientos en el eje no deseado (por las rotaciones en 3 dimensiones), por lo que, se han logrado obtener valores bastante cercanos entre sí.

Sin embargo, en movimientos como la flexión y flexión horizontal de hombro, el error ha sido bastante superior debido a que, Kinect tiene algunos problemas para fijar bien la posición de las articulaciones durante la medición, además de que, con la aplicación, es más fácil realizar algún movimiento no deseado durante la misma, lo que se traduce en que las lecturas obtenidas por ambos sistemas distan más entre sí.

Teniendo en cuenta que el sistema Kinect no ha podido ser validado fuera de un ambiente de laboratorio, con un muestreo lo suficientemente grande, todas las conclusiones obtenidas en relación con la comparación de *TFG-Goniometer* y este sistema lo que indican es cómo de próximos, en cuanto a precisión se refiere, son ambos, pero no se pueden utilizar para concluir cuál de los dos es más preciso.

Por último, es necesario destacar que *TFG-Goniometer* ya ha sido enviada a un terapeuta ocupacional que se encargará de evaluar la aplicación, por lo que, en un futuro se podrá tener más información sobre su precisión.

Capítulo 9

Conclusiones

9.1. Objetivos alcanzados

Tras la finalización de este proyecto, todos los objetivos establecidos al comienzo de éste se han podido completar con éxito en el lapso que se había planteado. En concreto:

- Se dispone de una aplicación Android con la que, fácilmente, se pueden realizar mediciones de los rangos articulares de prácticamente cualquier combinación de articulación/movimiento (no sólo las medidas con Kinect, sino también todas las articulaciones del tren inferior del cuerpo humano, como pueden ser, la cadera, rodillas, tobillos).
- Se puede llevar un registro de los pacientes, almacenando, para cada uno de ellos, sus datos personales, diagnóstico, así como un listado de todas las mediciones que se le han realizado (incluyendo la fecha y la hora de las mismas).
- Se ha realizado un estudio comparativo de la aplicación con un sistema Kinect, en el que se puede apreciar que, salvando las distancias por las imprecisiones propias de haber realizado las mediciones sin tener conocimientos específicos de fisioterapia o medicina, las lecturas obtenidas con ambos sistemas son bastante similares.
- Se ha logrado implementar la aplicación en dos idiomas (español e inglés), resultando bastante más compleja de lo esperado la implementación en este segundo idioma, debido al almacenamiento de los tipos Enumerados en la Base de Datos.
- Se ha podido incluir el patrón arquitectónico MVP, así como determinados patrones de diseño, que nunca se habían utilizado, dando como resultado un código bastante más legible y sencillo de mantener de cara al futuro.
- Se ha depurado la aplicación todo lo posible con el fin de proporcionar al usuario una experiencia agradable a la hora de usarla y, sobre todo, que de verdad resulte de utilidad a las personas que quieran utilizarla.

9.2. Líneas de trabajo futuras

Teniendo en cuenta las limitaciones, sobre todo temporales, que se tenían desde el inicio del desarrollo de este Trabajo de Fin de Grado, se ha procurado implementar todo lo posible, aun así, existen ciertos aspectos muy interesantes en los que se podría trabajar en un futuro. A saber:

- Dada la gran sensibilidad de la aplicación al obtener el ángulo de desplazamiento, el sólo hecho de tocar la pantalla ya afecta, mínimamente, a la lectura obtenida. Por ello, podría resultar muy interesante incorporar el reconocimiento de voz a la aplicación, de tal manera, que con un simple comando se pudiera inicializar la medición y con otro terminarla. Esto, además, permitiría a un paciente, sin ayuda de un fisioterapeuta o médico, realizarse a sí mismo una medición.
- La aplicación únicamente muestra la información obtenida, dejando a cuenta del usuario el cómo interpretarla. En este sentido, podría resultar muy interesante, poder mostrar al usuario algún tipo de gráfico de evolución de un paciente, mostrando las lecturas realizadas a lo largo del tiempo, para una articulación en particular.
- La aplicación se podría intentar mejorar, procurando que las lecturas del goniómetro se realicen en 2 dimensiones y no en 3 como se hacen en estos momentos, con el fin de evitar que los movimientos involuntarios del dispositivo afecten a la medición final.
- Actualmente, cada vez que realizas una medición, tienes que pasar por un proceso de seleccionar un paciente, introducir los datos de la nueva medición, etc. que puede resultar algo tedioso. Un punto donde se podría mejorar es, que antes de realizar la medición, se pueda elegir qué movimiento se va a intentar medir y, a partir de ahí, agilizar el proceso del almacenamiento de la misma, o incluso poder realizar varias mediciones de forma seguida y luego almacenarlas todas a la vez.
- Finalmente, resultaría de gran interés, poder realizar un estudio estadístico para validar la precisión de la aplicación frente a un goniómetro tradicional, ya que, este es precisamente el objetivo que se tenía en mente a la hora de empezar a desarrollar *TFG-Goniometer*.

9.3. Valoración personal

En lo referente al ámbito personal, no se pueden cuantificar los conocimientos y experiencia obtenidos durante el desarrollo de este Trabajo de Fin de Grado.

Desde el primer momento, al decidir utilizar una metodología de desarrollo ágil en la que no se tenía demasiada experiencia, ya ha resultado todo un reto, pero, según avanzaba el proyecto e iban surgiendo complicaciones, éstas han supuesto un reto aún mayor.

No sólo ha supuesto un reto en cuanto a los conocimientos adquiridos para poder desarrollarlo, sino, también, en cuanto al tiempo disponible para su completitud.

A lo largo de todo este proyecto, se ha procurado hacer las cosas de la mejor manera posible, primando, sobre otras muchas cosas, que el trabajo obtenido al final fuera algo de lo que poder sentirse orgulloso.

Bibliografía

Álvarez Castro, Marco Antonio. *Biomecánica de la muñeca* (2014). Disponible en <https://es.slideshare.net/marcoantonioalvarezc/biomecanica-de-mueca-38796478> (Último acceso: 28 de abril de 2017)

Android Developers. *Motion Sensors – Using the Rotation Vector Sensor*. Disponible en https://developer.android.com/guide/topics/sensors/sensors_motion.html#sensors-motion-rotate (Último acceso: 5 de julio de 2017)

Android Developers. *SensorEvent*. Disponible en <https://developer.android.com/reference/android/hardware/SensorEvent.html> (Último acceso: 27 de junio de 2017)

Barbic, Jernej. *Quaternions and Rotations*. Disponible en <http://run.usc.edu/cs520-s12/quaternions/quaternions-cs520.pdf> (Último acceso: 4 de julio de 2017)

Belk, Jim. *Quaternion distance* (2011). Disponible en <https://math.stackexchange.com/a/90098> (Último acceso: 6 de julio de 2017)

Carr, Richard. *Builder Design Pattern* (2008). Disponible en <http://www.blackwasp.co.uk/builder.aspx> (Último acceso: 11 de mayo de 2017)

Castellanos, Carolina. *Anatomía y biomecánica de la columna vertebral*. Disponible en <http://www.monografias.com/trabajos63/anatomia-columna-vertebral/anatomia-columna-vertebral2.shtml> (Último acceso: 28 de abril de 2017)

Cataldi, Z. Lage, F. Pessacq, R. García Martínez, R. *INGENIERÍA DE SOFTWARE EDUCATIVO*. Disponible en <http://www.iidia.com.ar/rgm/comunicaciones/c-icie99-ingenieriasoftwareeducativo.pdf> (Último acceso: 27 de abril de 2017)

Chandel, Sumit. *Testing methodologies using GWT* (2009). Disponible en http://www.gwtproject.org/articles/testing_methodologies_using_gwt.html (Último acceso: 10 de mayo de 2017)

Chen, An. *Evaluación del sensor Kinect v2 para rehabilitación funcional* (2016). Disponible en <https://uvadoc.uva.es/handle/10324/20946> (Último acceso: 7 de julio de 2017)

CodePath. Android guides. *Dependency Injection with Dagger 2*. Disponible en https://github.com/codepath/android_guides/wiki/Dependency-Injection-with-Dagger-2 (Último acceso: 11 de mayo de 2017)

Díaz Suarez, Daniel. *Inyección de dependencias en Android con Dagger 2* (2016). Disponible en <https://www.adictosaltrabajo.com/tutoriales/inyeccion-de-dependencias-en-android-con-dagger-2/> (Último acceso: 15 de mayo de 2017)

EcuRed. *Niveles de prueba de software* (2009). Disponible en https://www.ecured.cu/Niveles_de_prueba_de_software (Último acceso: 18 de mayo de 2017)

Fowler, Martin. *Inversion of Control Containers and the Dependency Injection pattern* (2004). Disponible en <https://martinfowler.com/articles/injection.html> (Último acceso: 11 de mayo de 2017)

Geary, David. *Simply Singleton* (2003). Disponible en <http://www.javaworld.com/article/2073352/core-java/simply-singleton.html> (Último acceso: 11 de mayo de 2017)

Google. *Dagger*. Disponible en <https://google.github.io/dagger/> (Último acceso: 15 de mayo de 2017)

GoogleTechTalks. *Sensor Fusion on Android Devices: A Revolution in Motion Processing*. Disponible en <https://www.youtube.com/watch?v=C7JQ7Rpwn2k> (Último acceso 7 de julio de 2017)

I. Kapandji, A. *Fisiología articular – Miembro Inferior – 5ª edición*

Jenkov, Jakob. *Dependency Injection Benefits* (2014). Disponible en <http://tutorials.jenkov.com/dependency-injection/dependency-injection-benefits.html> (Último acceso: 11 de mayo de 2017)

Kouwenberg, Sandra. *Articulación del codo: movimientos* (2015). Disponible en <https://anatomyayfisio.wordpress.com/2015/11/29/articulacion-del-codo-movimientos/> (Último acceso: 28 de abril de 2017)

Kropachov, Oleksii. *Xdroid*. Disponible en <https://github.com/shamanland/xdroid/blob/master/readme.md> (Último acceso: 15 de mayo de 2017)

Lopategui Corsino, Edgar. *La rodilla*. Disponible en <http://www.saludmed.com/CsEjerci/Cinesiolo/Rodilla.html> (Último acceso: 28 de abril de 2017)

Luedke, Matt. *Common Design Patterns for Android* (2015). Disponible en <https://www.raywenderlich.com/109843/common-design-patterns-for-android> (Último acceso: 11 de mayo de 2017)

Makabee, Hayim. *Separation of Concerns* (2012). Disponible en <https://effectivesoftwaredesign.com/2012/02/05/separation-of-concerns/> (Último acceso: 10 de mayo de 2017)

Megali, Tin. *An Introduction to Model View Presenter on Android* (2016). Disponible en <https://code.tutsplus.com/tutorials/an-introduction-to-model-view-presenter-on-android--cms-26162> (Último acceso: 10 de mayo de 2017)

Microsoft Developer Network. *The Model-View-Presenter (MVP) Pattern*. Disponible en <https://msdn.microsoft.com/en-us/library/ff649571.aspx> (Último acceso: 10 de mayo de 2017)

National Center for Biotechnology Information. *A Study on the Measurement of Wrist Motion Range Using the iPhone 4 Gyroscope Application* (2014). Disponible (abstract) en <https://www.ncbi.nlm.nih.gov/pubmed/24322647> (Último acceso: 28 de abril de 2017)

Palais, Bob. Palais, Richard. *Euler's fixed point theorem: The axis of a rotation* (2007). Disponible en <http://vmm.math.uci.edu/PalaisPapers/EulerFPT.pdf> (Último acceso: 6 de julio de 2017)

Panel Testing – Centro de Excelencia. *Software QA - ¿Cuáles son los tipos de pruebas software?* Disponible en <http://www.panel.es/blog/software-qa-cuales-son-los-tipos-de-pruebas-software/> (Último acceso: 18 de mayo de 2017)

- PMOinformatica. *Tipos de pruebas de software definidos por el ISTQB* (2014). Disponible en <http://www.pmoinformatica.com/2014/01/tipos-de-pruebas-de-software-istqb.html> (Último acceso: 18 de mayo de 2017)
- PortalesMedicos, *Goniometro*. Disponible en http://www.portalesmedicos.com/diccionario_medico/index.php/Goniometro (Último acceso: 22 de abril de 2017)
- Q. Huynh, Du. *Metrics for 3D Rotations: Comparison and Analysis* (2009). Disponible (abstract) en <https://link.springer.com/article/10.1007%2Fs10851-009-0161-2> (Último acceso: 7 de julio de 2017)
- Ruiz Pacheco, Juan Carlos. *C# - Inyección de Dependencias*. Disponible en <https://msdn.microsoft.com/es-es/communitydocs/net-dev/csharp/inyeccion-de-dependencias> (Último acceso: 11 de mayo de 2017)
- S. Pressman, Roger. *Ingeniería del software. Un enfoque práctico*, 7ª edición (2010).
- Schünke, Michael. Schulte, Erik. Schumacher, Udo. *PROMETHEUS – Texto y Atlas de anatomía (Tomo 1 – Anatomía General y Aparato Locomotor)* – 2ª edición (2010)
- Scrum Manager. Guía de formación. Versión 2.6 – Julio 2016. Disponible en http://www.scrummanager.net/files/sm_proyecto.pdf (Último acceso: 27 de abril de 2017)
- Tejedor Varillas, Alejandro. Miraflores Carpio, José L. *Exploración del hombro doloroso* (2008). Disponible en <http://www.jano.es/ficheros/sumarios/1/0/1705/43/00430047-LR.pdf> (Último acceso: 28 de abril de 2017)
- Viladot Voegeli, A. *Anatomía funcional y biomecánica del tobillo y el pie* (2003). Disponible en <http://www.elsevier.es/es-revista-revista-espanola-reumatologia-29-articulo-anatomia-funcional-biomecanica-del-tobillo-13055077> (Último acceso: 28 de abril de 2017)
- Waters, Kelly. *Step 2: How To Estimate Your Product Backlog* (2007). Disponible en <http://www.101ways.com/how-to-implement-scrum-in-10-easy-steps-step-2-how-to-estimate-your-product-backlog/> (Último acceso: 12 de abril de 2017)
- Wei Lee, Wang et al. *A Smartphone-Centric System for the Range of Motion Assessment in Stroke Patients* (2014). IEEE Journal of Biomedical and Health Informatics, Vol. 18, NO. 6.

Anexo I

Manual de usuario

I.1. Introducción

En este apéndice, se detalla cómo utilizar *TFG-Goniometer* por parte del usuario. Para ello, se va a ilustrar paso por paso las principales funcionalidades de la aplicación, con el fin de despejar todas las posibles dudas que se puedan tener sobre cómo utilizar la misma.

I.2. Principales funcionalidades

I.2.1. Añadir una medición

I.2.1.1. Paso 1. Iniciar una medición



(1) Pulsar el botón *Nueva medición*

Figura 65 - Instrucciones para iniciar una medición

I.2.1.1.1. Alternativa al Paso 1. Iniciar una medición desde el menú lateral



(1) Pulsar en el icono del menú lateral



(2) Pulsar en *Nueva medición*

Figura 66 - Instrucciones para iniciar una medición desde el menú lateral

I.2.1.2. Paso 2. Realizar la medición



(1) Pulsar sobre el goniómetro para iniciar la medición



(2) Pulsar de nuevo para finalizar la medición

Figura 67 - Instrucciones para realizar la medición

En este paso, dado que uno de los sensores empleados es el magnetómetro, que, de por sí es más sensible a las interferencias con otros objetos, y que funciona mejor en exteriores, es posible que la primera vez que se vaya a realizar una medición, sea necesario pulsar el botón de reiniciar hasta que los sensores se estabilicen para dar lecturas fiables.

I.2.1.2.1. Alternativa al Paso 2. Reiniciar la medición



(1) Pulsar el botón *Reiniciar*



(2) Resultado tras pulsar el botón

Figura 68 - Instrucciones para reiniciar la medición

I.2.1.3. Paso 3. Almacenar la medición



(1) Pulsar el botón *Almacenar medición*



(2) Pulsar en *Seleccionar paciente existente*

Figura 69 - Instrucciones para almacenar la medición

I.2.1.3.1. Alternativa al Paso 3. Cancelar el almacenamiento de la medición



(1) Pulsar en *Cancelar*



(2) Resultado tras cerrarse el diálogo

I.2.1.4. Paso 4. Seleccionar paciente para almacenar medición



(1) Pulsar sobre un paciente



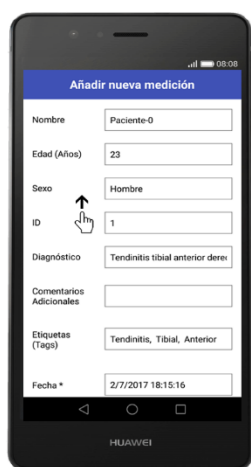
(2) Desplazar la vista hasta abajo del todo



(3) Pulsar el botón *Añadir nueva entrada*

Figura 71 - Instrucciones para seleccionar paciente para almacenar la medición

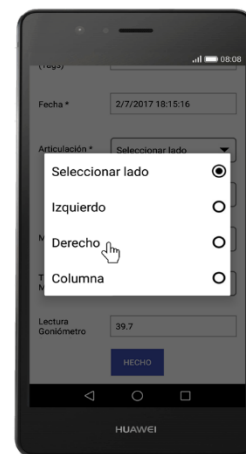
I.2.1.5. Paso 5. Rellenar la información sobre la nueva medición



(1) Desplazar la vista hasta que aparezcan las opciones a rellenar



(2) Pulsar en *Seleccionar lado*



(3) Seleccionar una opción

Figura 70 - Instrucciones para cancelar el almacenamiento de la medición



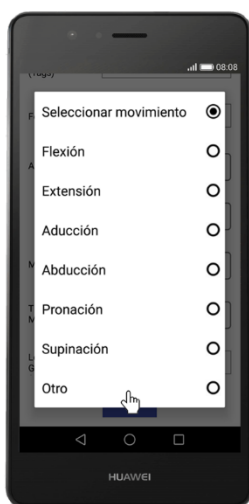
(4) Pulsar en *Seleccionar articulación*



(5) Seleccionar una opción



(6) Pulsar en *Seleccionar movimiento*



(7) Seleccionar una opción



(8) Al seleccionar la opción *Otro/a* se muestra un cuadro de texto a rellenar



(9) Escribir la información en el cuadro de texto



(10) Pulsar en *Seleccionar tipo de movimiento*



(11) Seleccionar una opción



(12) Pulsar el botón *Hecho*



(13) Resultado tras haber almacenado la medición

Figura 72 - Instrucciones para rellenar la información sobre la nueva medición

I.2.2. Visualizar 10 últimas mediciones almacenadas

I.2.2.1. Paso 1. Ir a *Últimas 10 mediciones*



(1) Pulsar el botón *Últimas 10 mediciones*

Figura 73 - Instrucciones para ir a *Últimas 10 mediciones*

I.2.2.1.1. Alternativa al Paso 1. Ir a *Últimas 10 mediciones* a través del menú lateral



(1) Pulsar en el icono del menú lateral



(2) Pulsar en *Últimas 10 mediciones*

Figura 74 - Instrucciones para ir a *Últimas 10 mediciones* a través del menú lateral

Tras realizar, bien el paso 1, o bien, la alternativa al mismo, el resultado mostrado es el siguiente:



Figura 75 - Visualizar 10 últimas mediciones almacenadas

I.2.3. Borrar una medición

Esta funcionalidad sólo se puede realizar después de la acción *Visualizar 10 últimas mediciones almacenadas*.

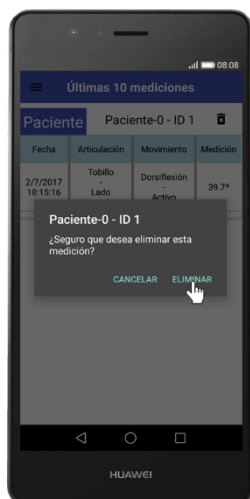
I.2.3.1. Paso 1. Borrar la medición seleccionada



(1) Pulsar el botón con el icono de la papelera

Figura 76 - Instrucciones para borrar la medición seleccionada

I.2.3.2. Paso 2. Confirmar el borrado de la medición



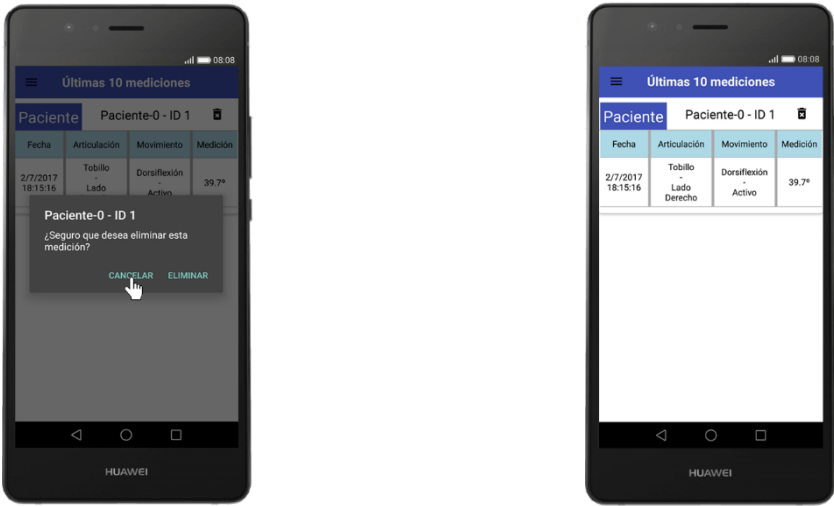
(1) Pulsar en *Eliminar*



(2) Resultado tras cerrarse el diálogo

Figura 77 - Instrucciones para confirmar el borrado de la medición

I.2.3.2.1. Alternativa al Paso 2. Cancelar el borrado de la medición



(1) Pulsar en *Cancelar*

(2) Resultado tras cerrarse el diálogo

Figura 78 - Instrucciones para cancelar el borrado de la medición

I.2.4. Visualizar los perfiles de pacientes

I.2.4.1. Paso 1. Ir a *Perfiles de pacientes*



(1) Pulsar el botón Perfiles de pacientes

Figura 79 - Instrucciones para ir a *Perfiles de pacientes*

I.2.4.1.1. Alternativa al Paso 1. Ir a *Perfiles de pacientes* a través del menú lateral



(1) Pulsar en el icono del menú lateral



(2) Pulsar en *Perfiles de pacientes*

Figura 80 - Instrucciones para ir a *Perfiles de pacientes* a través del menú lateral

Tras realizar, bien el paso 1, o bien, la alternativa al mismo, el resultado mostrado es el siguiente:

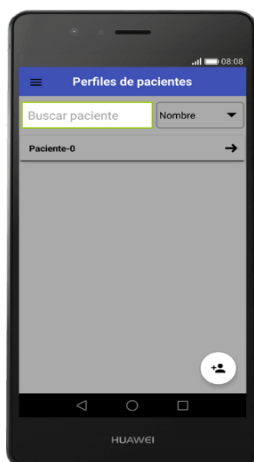
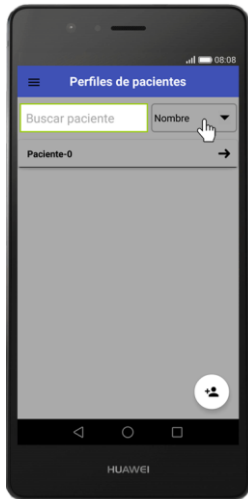


Figura 81 – Visualizar los perfiles de pacientes

I.2.5. Filtrar el listado de pacientes

Esta funcionalidad sólo se puede realizar después de la acción *Visualizar los perfiles de pacientes*.

I.2.5.1. Paso 1 (Opcional). Seleccionar un criterio de filtro de pacientes



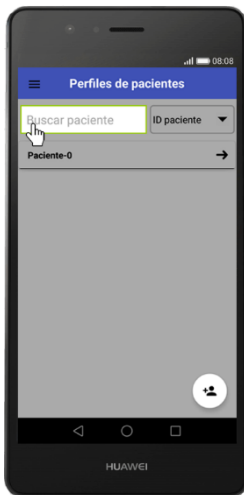
(1) Pulsar en el menú para que se despliegue



(2) Seleccionar una opción

Figura 82 - Instrucciones para seleccionar un criterio de filtro de pacientes

I.2.5.2. Paso 2. Escribir información en el cuadro de búsqueda



(1) Pulsar en el cuadro de búsqueda



(2) Introducir un carácter (o varios)



(3) Resultado tras el filtrado

Figura 83 - Instrucciones para escribir información en el cuadro de búsqueda

I.2.6. Crear perfil de un paciente

Esta funcionalidad sólo se puede realizar después de la acción *Visualizar los perfiles de pacientes*.

I.2.6.1. Paso 1. Ir a la pantalla con los campos a rellenar sobre el nuevo paciente



(1) Pulsar el botón de la esquina inferior derecha

Figura 84 - Instrucciones para ir a la pantalla con los campos a rellenar sobre el nuevo paciente

I.2.6.2. Paso 2. Rellenar la información sobre el nuevo paciente



(1) Rellenar los primeros campos visibles sobre el paciente



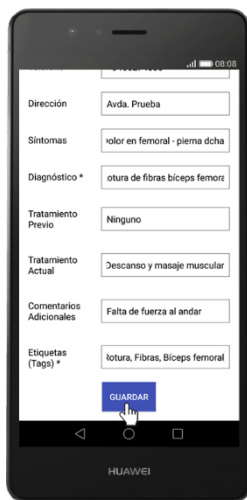
(2) Desplazar la vista hasta abajo para ver los nuevos campos a rellenar



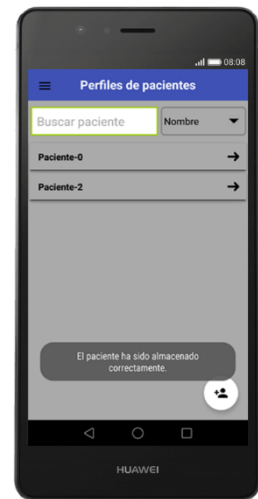
(3) Rellenar los nuevos campos

Figura 85 - Instrucciones para rellenar la información sobre el nuevo paciente

I.2.6.3. Paso 3. Almacenar el perfil del paciente



(1) Pulsar el botón *Guardar*

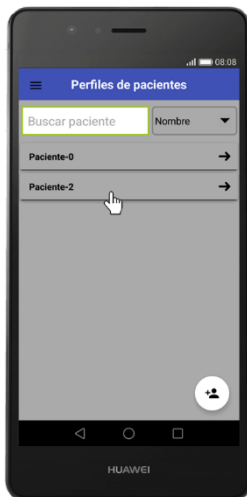


(2) Resultado tras almacenar el nuevo paciente

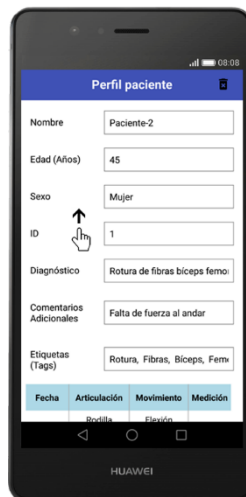
Figura 86 - Instrucciones para almacenar el perfil del paciente

I.2.7. Visualizar la información de un paciente

Esta funcionalidad sólo se puede realizar después de la acción *Visualizar los perfiles de pacientes*.



(1) Pulsar sobre un paciente



(2) Desplazar la vista hacia abajo para ver sus mediciones



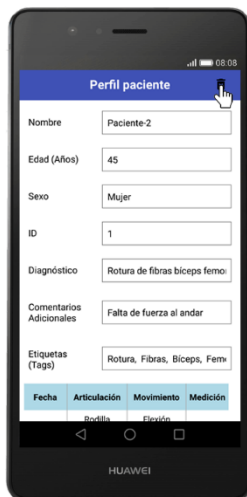
(3) Resultado tras la pulsación

Figura 87 - Instrucciones para visualizar la información de un paciente

I.2.8. Borrar el perfil de un paciente

Esta funcionalidad sólo se puede realizar después de la acción *Visualizar la información de un paciente*.

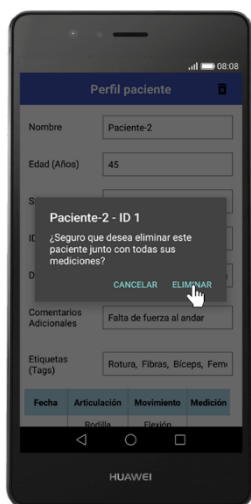
I.2.8.1. Paso 1. Borrar el perfil del paciente seleccionado



(1) Pulsar el botón con el icono de una papelera

Figura 88 - Instrucciones para borrar el perfil del paciente seleccionado

I.2.8.2. Paso 2. Confirmar el borrado del perfil del paciente



(1) Pulsar en *Eliminar*



(2) Resultado tras cerrarse el diálogo

Figura 89 - Instrucciones para confirmar el borrado del perfil del paciente

I.2.8.2.1. Alternativa al Paso 2. Cancelar el borrado del perfil del paciente



(1) Pulsar en *Cancelar*

(2) Resultado tras cerrarse el diálogo

Figura 90 - Instrucciones para cancelar el borrado del perfil del paciente

Anexo II

Plan de seguimiento

II.1. Introducción

En este apéndice, se va a mostrar toda la información relativa al desarrollo de este proyecto a lo largo de los sucesivos sprints; sin embargo, antes de eso, es preciso profundizar un poco más en la metodología que se ha empleado para el desarrollo del mismo, que es Scrum.

Scrum se puede adoptar de forma técnica, aplicando reglas definidas, o pragmática, adoptando los valores originales de Scrum con reglas personalizadas.

En este caso, a pesar de haber empleado Scrum previamente para el desarrollo de otro proyecto en la asignatura *Planificación y Gestión de Proyectos*, se ha decidido aplicar el enfoque más técnico a Scrum (basado en la aplicación de reglas concretas en un marco de roles, eventos y artefactos definidos), ya que, no se tiene la experiencia suficiente para “romper las reglas” y aplicar el enfoque más pragmático.

La siguiente ilustración, obtenida de *Scrum Manager*, resume perfectamente el proceso de desarrollo en Scrum:

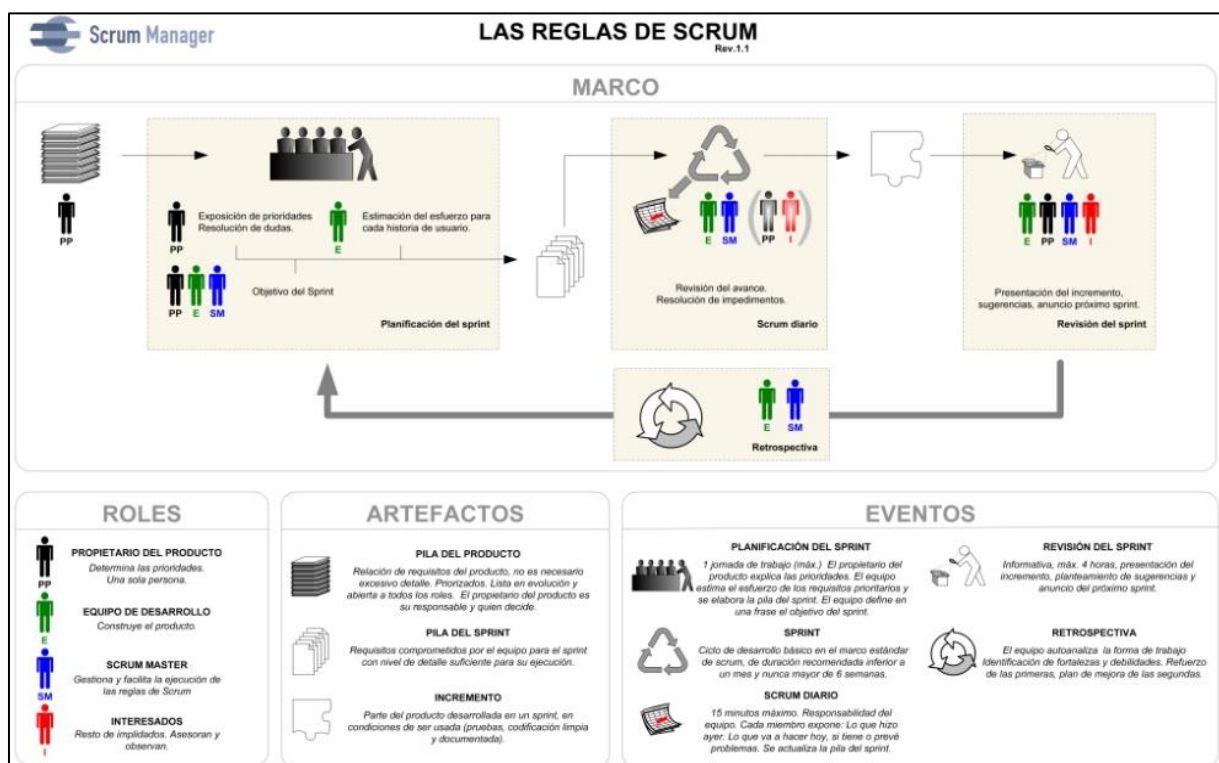


Figura 91 - Proceso de desarrollo - Scrum

La pieza clave al abordar un enfoque más técnico de Scrum es el **sprint**, el cual se podría denominar como un ciclo o iteración de trabajo que produce una parte del producto terminada y funcionalmente operativa (**incremento**).

Al emplear este enfoque en el que se trabaja con sprints, el incremento se obtiene de manera iterativa, es decir, el incremento no se va obteniendo de manera continua a lo largo de todo el proceso de desarrollo, sino que se obtiene tras un pulso de tiempo prefijado (sprint).

En la siguiente ilustración se puede apreciar la diferencia en la obtención del incremento aplicando el enfoque técnico (izquierda) o pragmático (derecha).

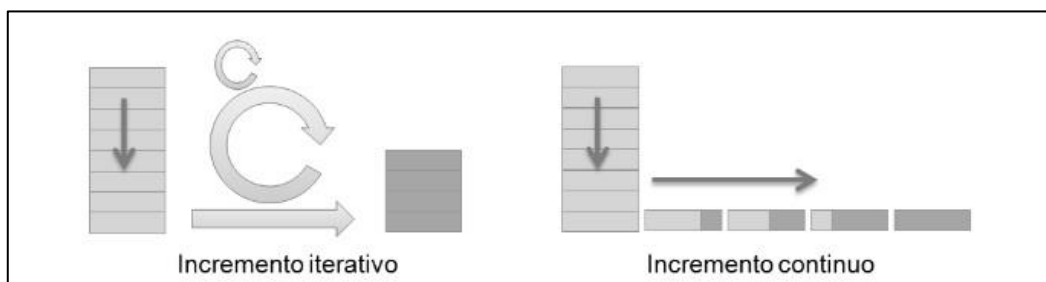


Figura 92 - Incremento iterativo vs incremento continuo - Scrum

Por último, se muestra en la siguiente ilustración, de una forma más detallada, en qué consiste el ciclo iterativo de Scrum.

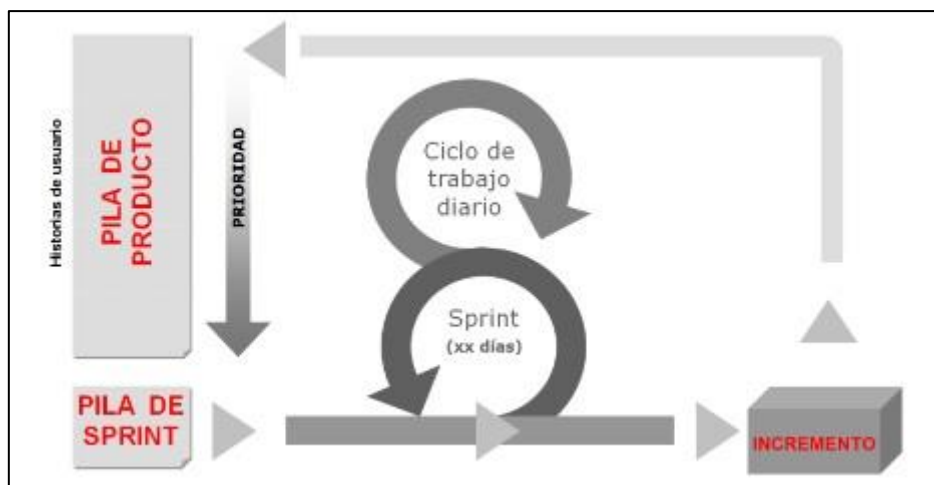


Figura 93 - Ciclo iterativo - Scrum

De forma resumida, este ciclo está compuesto por los siguientes pasos:

1. Se parte de una Pila de Producto (*Product Backlog*) formada por historias de usuario, ordenada de mayor a menor prioridad, y se hace una estimación sobre el tamaño de cada una de éstas (este punto es explicado en mayor profundidad en el apartado II.1.4).
2. En base al Product Backlog, durante la planificación del sprint, se establece una Pila de Sprint (*Sprint Backlog*) dividiendo una parte de las historias de usuario en tareas más concretas, que se llevarán a cabo durante el sprint.
3. A partir de ahí, comienza el *sprint* como tal, que durará un número determinado de días (fijado previamente) entre 2 y 4 semanas, aproximadamente. Durante éste, se irán trabajando en las sucesivas tareas del Sprint Backlog, y se hará diariamente una reunión (*scrum daily*) donde se comenta brevemente el trabajo realizado el día anterior y el que se va a realizar en ese día, así como si se prevé algún problema en la realización del mismo.
4. Al final del sprint se obtiene un incremento, que es una parte del producto (en este caso) perfectamente operativa y que ha pasado un control de pruebas para asegurar su correcto funcionamiento.
5. Una vez obtenido el incremento, si todo es correcto, se actualiza el Product Backlog, y el proceso vuelve a comenzar desde el paso 1.

II.1.1. Propósito

Como se ha comentado brevemente al principio de este anexo, la intención en este apartado es mostrar un seguimiento del trabajo realizado a lo largo de los sucesivos sprints.

Ya en el apartado 4.4.1.2 (Planificación del proyecto – División del trabajo), se ha comentado que a lo largo del desarrollo de este proyecto se iban a realizar un total de 5 sprints con una duración de 13 días de trabajo cada uno (15 días naturales).

Para cada sprint, se mostrará la siguiente información relativa al mismo:

- El *product backlog* actualizado antes de comenzar cada sprint.
- Las historias de usuario que se van a realizar.
- La sub-división de esas historias de usuario en tareas más concretas (*sprint backlog*), para ser llevadas a cabo, junto con su estimación en puntos de usuario.
- Valor obtenido para la métrica *velocidad* para ese sprint en concreto.
- Comentarios en referencia a cuestiones de interés sobre el sprint (si ha habido retrasos o problemas durante el mismo).
- Una revisión del sprint explicando en detalle el incremento que se ha obtenido, así como los riesgos que se puedan haber disparado durante el mismo.

II.1.2. Alcance

El seguimiento del avance del proyecto es una tarea muy importante, sea cual sea la metodología de desarrollo empleada; sin embargo, Scrum hace hincapié en este aspecto, y, por este motivo, se ha decidido mostrar en este Anexo todo lo relacionado con cada uno de los sprints.

De esta forma, se podrá observar de forma clara cómo avanza el proyecto, si se prevén retrasos o adelantos en algún momento del desarrollo, o si es necesario hacer cambios durante el mismo.

II.1.3. Códigos de colores



Historias de usuario que se van a realizar por completo en el sprint actual



Historias de usuario que se van a realizar parcialmente en el sprint actual

Como ya se ha comentado en la **Nota importante** del apartado 4.4.1.1 (Product roadmap), algunas historias de usuario van a ser implementadas parcialmente a lo largo de los sucesivos sprints, dada la metodología de trabajo de Scrum (desarrollo iterativo e incremental), como la especificación de requisitos, especificación de casos de uso, etc. La versión final no se obtendrá hasta fechas cercanas a la entrega final del proyecto.

II.1.4. Estimación del esfuerzo

El primer aspecto por comentar con respecto a este punto es que no se va a realizar una estimación de las historias de usuario (presentadas en el *product backlog*), sino que lo que se va a estimar son las tareas asociadas a las historias de usuario que se van a realizar en cada uno de los sprints (presentadas en el *sprint backlog*).

Esto es así porque, dada la escasa experiencia que se tiene utilizando Scrum, es muy probable que cualquier estimación que se haga de las historias de usuario resulte en ser errónea; mientras que, es más probable que las estimaciones que se hagan de las tareas asociadas a éstas en cada uno de los sprints (que, por definición, son tareas más modulares) resulten más acertadas.

Dicho esto, a la hora de estimar el esfuerzo de trabajo que supone cada una de estas tareas, se pueden emplear dos enfoques principales.

El primero de ellos consiste en hacer una estimación de tipo temporal, en la que, a cada tarea se le asigna un valor numérico que representa la estimación en horas-hombre para completarla. El principal problema de este enfoque (se recuerda que se tiene poca experiencia en el empleo de Scrum como metodología de desarrollo), es que las estimaciones muy probablemente se alejen de la realidad (generalmente se harán estimaciones bastante inferiores con respecto al tiempo real de completitud de la tarea en concreto).

Como se puede intuir, subestimar una tarea (en función de cuánto se desvíe la estimación de la realidad) va a traer consigo retrasos con respecto a la funcionalidad que se prevé obtener en ese sprint, y eso es algo que, salvo causa mayor, se debe evitar a toda costa.

En este contexto, aparece otro enfoque para la estimación del esfuerzo que no trata de cuantificar el tiempo que se va a tardar en realizar una tarea, sino cómo de grande es ésta. En este caso, el valor asignado a cada tarea es lo que se conoce como **puntos de usuario**.

Existen varias escalas de asignación de puntos de usuario. En este caso, se ha decidido optar por la sucesión de Fibonacci. Más concretamente, se van a utilizar únicamente los siguientes valores: 1, 2, 3, 5, 8, 13 y 21. Se tiene la opinión de que con este rango de valores es suficiente para poder estimar todas las tareas del *sprint backlog* de cada uno de los sprints.

La forma en la que se quiere utilizar esta estimación para tareas es la siguiente:

- En primera instancia, se escogerá la tarea que se considera de mayor tamaño del *sprint backlog* y se le asignará a ésta el valor 21.
- Después, se escogerá la tarea que se considera de menor tamaño del *sprint backlog* y se le asignará a ésta el valor 1.
- En última instancia, al resto de tareas, se les asignarán los valores entre el 1 y el 21 en función de su tamaño en comparación con las tareas mayor y menor del *sprint backlog*.

Por último, es preciso comentar que este procedimiento que se acaba de describir se empleará para cada uno de los sprints, obteniendo, al final de los mismos, la media de puntos de usuario de todas las tareas que los componen. De esta manera se puede comprobar, si esta media varía mucho de un sprint a otro, si sería necesario realizar algún sprint más o menos.

II.2. Pasos previos antes del primer sprint – 4 de abril a 19 de abril

Antes de dar comienzo al primer sprint, se han tenido que realizar unas cuantas tareas previas. En este apartado, se quiere dar a conocer, de un modo resumido, cuál ha sido el trabajo realizado antes de poder comenzar a trabajar realmente en un entregable del producto (bien sea documentación o software).

II.2.1. Comprensión inicial del problema

El primer paso ha sido trabajar en la comprensión del proyecto que se iba a realizar. Para ello se disponía de una serie de artículos (facilitados por el tutor del proyecto) en los que se hablaba de la utilización de dispositivos móviles (mediante una aplicación) como goniómetros tradicionales y se validaba su precisión para ser utilizados con fines médicos.

La conclusión obtenida es que, la diferencia entre la precisión de estas aplicaciones y las mediciones realizadas con un goniómetro tradicional es menor que 1 grado (una diferencia que no es estadísticamente significativa).

II.2.2. Elección de la metodología de desarrollo

Una vez se había comprendido de forma aproximada el objetivo y alcance de este proyecto, el siguiente paso fue decidir qué metodología de desarrollo emplear para la realización del mismo.

Inicialmente, se pensó en emplear un modelo de desarrollo que se viene utilizando a lo largo de casi todo el grado, que es el *modelo en cascada*.

La principal ventaja de esta metodología de desarrollo es que se tienen varios años de experiencia empleándola, por lo que, se puede empezar a utilizar de inmediato sin necesitar de tiempo para revisar documentación relativa a esta metodología o posibles dudas que se pudieran tener con respecto a su uso.

Sin embargo, la principal desventaja que se encuentra en el uso de esta metodología (y que ha sido la razón de rechazarla frente a una metodología ágil), es que, transcurre demasiado tiempo desde el momento en que se tiene claro lo que se quiere fabricar hasta que finalmente se dispone de un producto funcional.

En este caso, y con la restricción temporal que se ha citado en el apartado 4.2.2, no es demasiado aconsejable dejar toda la fase de implementación a falta de 4-5 semanas para la entrega del proyecto al completo (es fácil que se produzcan retrasos con respecto a lo planificado en la fase de implementación), por lo que, finalmente se ha decidido rechazar el empleo de este tipo de metodología.

La primera opción una vez se había descartado seguir el modelo en cascada fue elegir algún tipo de metodología que permitiera seguir un **desarrollo iterativo e incremental**. En este contexto, se pensó inmediatamente en alguna metodología ágil y, en concreto, en *Scrum*, ya que, se había empleado antes esta metodología, en la asignatura *Planificación y Gestión de Proyectos*.

Tras haber revisado las transparencias que aún se tenían de esta asignatura y de información bastante completa que se encontró en el documento *Scrum Manager*, se decidió emplear este tipo de metodología en este proyecto.

A continuación, se detalla brevemente en qué consisten estas dos metodologías de las que se ha hablado en este apartado.

II.2.2.1. Modelo en cascada

El modelo en cascada es el modelo tradicional de desarrollo de software propuesto por Winston W. Royce en 1970 y posteriormente revisado por Barry Boehm en 1980 e Ian Sommerville en 1985.

Este modelo sugiere un enfoque sistemático y secuencial para el desarrollo de software, que comienza con la especificación de los requisitos por parte del cliente y avanza a través de las distintas fases de análisis, diseño, implementación y pruebas hasta la obtención del software terminado.

Al final de cada etapa, el modelo está diseñado para llevar a cabo una revisión final, que se encarga de determinar si el proyecto está listo para avanzar a la siguiente fase. De esta forma, cualquier error de diseño detectado en la etapa de prueba conduce necesariamente al rediseño y nueva programación del código afectado, aumentando los costes del desarrollo.

Hoy en día, sigue siendo una de las metodologías más empleadas en el ámbito académico y empresarial.

II.2.2.2. Scrum

Scrum es un modelo de desarrollo ágil caracterizado por:

- Adoptar una estrategia de desarrollo incremental, en lugar de la planificación y ejecución completa del producto.
- Basar la calidad del resultado más en el conocimiento tácito de las personas en equipos auto organizados, que en la calidad de los procesos empleados.
- Solapamiento de las diferentes fases del desarrollo, en lugar de realizar una tras otra en un ciclo secuencial o de cascada.

Este modelo fue identificado y definido por Ikujiro Nonaka e Hirotaka Takeuchi a principios de los 80, al analizar cómo desarrollaban los nuevos productos las principales empresas de manufactura tecnológica.

Aunque esta forma de trabajo surgió en empresas de productos tecnológicos, es apropiada para proyectos con requisitos inestables y para los que requieren rapidez y flexibilidad, situaciones frecuentes en el desarrollo de determinados sistemas de software.

II.2.3. Primeros artefactos de Scrum

Una vez ya se había decidido emplear Scrum como metodología de desarrollo, se empezó a trabajar en los artefactos propios de ésta.

Inicialmente se estableció la **visión del producto**, la cual, se encuentra resumida, más o menos, en el apartado 3.2. (Visión general del proyecto).

A continuación, se desarrolló el **product backlog**, el **product roadmap** y el **plan de release**. De estos tres artefactos, el primero de ellos se muestra en el apartado siguiente, donde se explica lo relativo al primer sprint

del proyecto, y los otros dos ya se han mostrado previamente en los apartados 4.4.1.1 y 4.4.1.3, respectivamente.

Todos estos artefactos, aparte de ser necesaria su obtención para el desarrollo con Scrum, han servido para comprender mucho mejor los objetivos y el alcance de este proyecto, así como para obtener una visión de más “bajo nivel” sobre las tareas en concreto que se necesitaban realizar.

Por último, se realizó una planificación del primer sprint, en el que se obtuvo el **sprint backlog** con las tareas asociadas a las historias de usuario del *product backlog* que se iban a realizar en el primer sprint.

Como esta información ya es más específica del primer sprint, se explicará en el siguiente apartado.

II.3. Sprint 1 – 20 de abril a 4 de mayo

II.3.1. Product backlog

En este apartado, se muestra la *pila de producto* o *product backlog* actualizada a fecha 20 de abril, el día de comienzo del sprint 1.

ID	Prioridad	Descripción
1	Muy alta	Como Product Owner quiero disponer de un Plan de Desarrollo completo para poder tener una visión global del proyecto.
2	Muy alta	Como Product Owner quiero disponer de una especificación completa de los requisitos del sistema.
3	Muy alta	Como Product Owner quiero disponer de un listado completo, junto con la especificación y realización en análisis de cada uno de los casos de uso del sistema.
4	Muy alta	Como Product Owner quiero disponer de un modelo de dominio del sistema, así como una descripción de las clases que lo forman.
5	Muy alta	Como Usuario quiero poder hacer mediciones de los movimientos de las articulaciones mediante el uso del sistema.
6	Alta	Como Product Owner quiero disponer de un documento completo sobre el diseño de la arquitectura del sistema, así como una descripción de los patrones utilizados.
7	Alta	Como Usuario quiero poder decidir si almacenar o no cada una de las mediciones que realizo.
8	Alta	Como Usuario quiero poder crear un perfil para cada paciente donde pueda introducir sus datos personales, así como un diagnóstico, y que esa información sea almacenada en una Base de Datos.
9	Media	Como Usuario quiero poder acceder al perfil de cada uno de los pacientes que he almacenado previamente.
10	Media	Como Usuario quiero poder visualizar las 10 últimas mediciones que he guardado.
11	Media	Como Product Owner quiero disponer de un documento detallado sobre la implementación del sistema y las pruebas que se han realizado sobre el mismo.
12	Media	Como Product Owner quiero disponer de un documento detallado sobre los resultados obtenidos tanto con Kinect como con la aplicación, así como de las conclusiones obtenidas a raíz de la comparación de ambos sistemas.

13	Baja	Como Usuario quiero disponer de un menú de información sobre la aplicación.
14	Baja	Como Usuario quiero disponer de un manual de usuario de la aplicación.

Tabla 64 - Product backlog - Sprint 1

II.3.2. Descripción del sprint

En este primer sprint, se va a realizar, en primer lugar, el Plan de Desarrollo del proyecto al completo (en el que se incluye el Plan de Gestión de Riesgos), ya que, sobre todo éste último, es totalmente imprescindible antes de empezar a desarrollar un proyecto.

Una vez se tenga el Plan de Desarrollo, se realizará una especificación inicial de los requisitos asociados a la funcionalidad a implementar en este primer sprint.

A continuación, se implementará una primera parte de la aplicación Android, con el único objetivo de que se puedan realizar mediciones articulares con el dispositivo móvil. De momento, éstas no se podrán almacenar, ni crear perfiles de pacientes, ni nada por el estilo. Los esfuerzos se centrarán en diseñar la interfaz gráfica, así como la parte lógica de la aplicación para que se pueda hacer lo básico con ella.

Y, por último, y dado que lo exige Scrum, se empezará a realizar un documento en el que se detallen las pruebas que se han realizado sobre esta primera versión de la aplicación y los resultados que se han obtenido.

II.3.3. Sprint backlog

Backlog ID	Tarea	Tipo	Estimación
1	Elaborar Introducción	Documentación	2
1	Elaborar Visión general y Organización del proyecto	Documentación	8
1	Elaborar Planificación	Documentación	13
1	Elaborar Seguimiento	Documentación	8
1	Elaborar Plan de Gestión de Riesgos completo	Documentación	5
2	Elaborar especificación inicial de requisitos	Análisis	1
5	IU - Crear la pantalla principal de la aplicación	Implementación	3
5	IU - Crear el círculo principal con el indicador del ángulo	Implementación	13
5	IU - Crear el botón de reiniciar y el de guardar la medición	Implementación	5
5	IU - Crear el menú	Implementación	3
5	Obtener datos de los sensores del dispositivo	Implementación	13
5	Obtener el ángulo de giro en función de los datos de los sensores	Implementación	21
5	Implementar el comportamiento del botón de reset	Implementación	3
5	IU - Mostrar la variación del ángulo gráficamente	Implementación	8
5	Realizar pruebas de la funcionalidad del primer sprint	Pruebas	5
11	Elaborar documento inicial sobre la implementación y pruebas realizadas	Documentación	5

Tabla 65 - Sprint backlog - Sprint 1

II.3.4. Comentarios

La obtención de los datos de los sensores, así como el ángulo de desplazamiento en función de éstos ha resultado bastante más complejo de lo que se había pensado en un principio. El problema no ha radicado en que fuera una tarea difícil de realizar, sino que se ha tenido que invertir mucho tiempo en tareas de investigación, lectura de documentos, trabajos, etc. sobre, primero, qué sensores eran los que se necesitaban emplear, y, segundo, de qué manera se podía obtener el ángulo de desplazamiento en función de los datos proporcionados por éstos.

Finalmente se ha logrado obtener una primera versión que mide correctamente el ángulo de desplazamiento del dispositivo en dos de los tres ejes tridimensionales (x, y) mediante la información proporcionada por el acelerómetro del dispositivo. Con estos datos se logra medir la gran mayoría de movimientos articulares, salvo aquellos de rotación. Por ahora se ha decidido conservar esta versión para poder cumplir con el resto de tareas correspondientes a este sprint. Más adelante se barajará la posibilidad de utilizar algún sensor más con el fin de hacer el goniómetro más completo.

Con respecto al resto de tareas, la única que ha presentado algunos problemas más de lo que se pensaba ha sido la correspondiente a realizar el diseño de la interfaz gráfica del goniómetro, debido a la escasa experiencia que se tiene diseñando interfaces de usuario.

II.3.5. Revisión de Sprint

II.3.5.1. Incremento

- Puntos de usuario planificados: **116**
- Puntos de usuario realizados: **116**
- Media de puntos de usuario: **7.25 puntos/sprint**
- Número de tareas realizadas: **16 tareas/sprint**

Toda la funcionalidad que se había planificado realizar al comienzo del sprint se ha podido cumplir.

En concreto, ya se dispone del Plan de Desarrollo del proyecto al completo (que incluye el Plan de Gestión de Riesgos), de una especificación inicial de requisitos, de una primera versión del documento en el que se tratan las cuestiones relativas a la implementación y pruebas de la aplicación, y de una primera versión, también, de la misma, que permite realizar y visualizar en tiempo real el ángulo de desplazamiento del dispositivo con respecto a dos de los tres ejes tridimensionales mediante el uso del acelerómetro.

Además, con esta funcionalidad, se ha obtenido la **versión 0.1** de la aplicación.

Por último, para comprobar el correcto funcionamiento del incremento software en este sprint, se han realizado las pruebas: **Prueba-0001, Prueba-0002, Prueba-0003, Prueba-0004 y Prueba-0006**.

II.3.5.2. Riesgos

En este sprint se ha disparado un único riesgo, **R02** (Mala estimación de la complejidad de alguna tarea del *sprint backlog*), que, ha resultado en que, desde el tercer día de comienzo de este primer sprint, se han tenido que incrementar el número de horas de trabajo diarias de 6 a 8 (algunos días hasta incluso 9-10), así como el

número de días de trabajo semanales de 6 a 7 para poder cumplir con la funcionalidad que se tenía que entregar al final de este sprint.

Este riesgo se ha disparado, como ya se ha dejado entrever en los comentarios del sprint, en las tareas *Obtener datos de los sensores del dispositivo*, *Obtener el ángulo de giro en función de los sensores* y *Crear el círculo principal con el indicador del ángulo*.

II.4. Sprint 2 – 5 de mayo a 19 de mayo

II.4.1. Product backlog

En este apartado, se muestra la *pila de producto* o *product backlog* actualizada a fecha 5 de mayo, el día de comienzo del sprint 2.

ID	Prioridad	Descripción
1	Muy alta	Como Product Owner quiero disponer de una especificación completa de los requisitos del sistema.
2	Muy alta	Como Product Owner quiero disponer de un listado completo, junto con la especificación y realización en análisis de cada uno de los casos de uso del sistema.
3	Muy alta	Como Product Owner quiero disponer de un modelo de dominio del sistema, así como una descripción de las clases que lo forman.
4	Muy alta	Como Product Owner quiero disponer de un documento completo sobre el diseño de la arquitectura del sistema, así como una descripción de los patrones utilizados.
5	Muy alta	Como Usuario quiero poder crear un perfil para cada paciente donde pueda introducir sus datos personales, así como un diagnóstico, y que esa información sea almacenada en una Base de Datos.
6	Muy alta	Como Usuario quiero poder acceder al perfil de cada uno de los pacientes que he almacenado previamente.
7	Muy alta	Como Product Owner quiero que la aplicación siga el patrón arquitectónico Modelo-Vista- Presentador, para facilitar el diseño de la misma, así como su posterior mantenimiento.
8	Muy alta	Como Product Owner quiero que la aplicación utilice el patrón de diseño <i>Inyección de dependencias</i> para que la Vista obtenga la instancia de su Presentador sin que sea ésta la encargada de crearlo, con el fin de implementar mejor el patrón MVP.
9	Alta	Como Usuario quiero poder decidir si almacenar o no cada una de las mediciones que realizo.
10	Media	Como Usuario quiero poder visualizar las 10 últimas mediciones que he guardado.
11	Media	Como Product Owner quiero disponer de un documento detallado sobre la implementación del sistema y las pruebas que se han realizado sobre el mismo.
12	Media	Como Product Owner quiero disponer de un documento detallado sobre los resultados obtenidos tanto con Kinect como con la aplicación, así como de las conclusiones obtenidas a raíz de la comparación de ambos sistemas.
13	Baja	Como Usuario quiero disponer de un menú de información sobre la aplicación.
14	Baja	Como Usuario quiero disponer de un manual de usuario de la aplicación.

Tabla 66 - Product backlog - Sprint 2

II.4.2. Descripción del sprint

En un principio, tal y como se puede apreciar en el *product backlog* del primer sprint, se pensaba implementar la funcionalidad en relación al almacenamiento de mediciones en la Base de Datos, pero, en la planificación de este segundo sprint, se observó que antes de poder almacenar mediciones, era necesario disponer de perfiles de pacientes a los cuales se asocian, por lo que, las historias de usuario en relación al almacenamiento y recuperación de la información de los pacientes adquirieron más prioridad, y son las que finalmente se van a implementar en este sprint.

Teniendo esto en cuenta, se procederá a realizar la especificación de requisitos y de casos de uso (así como la realización en análisis de éstos últimos) y modelo de dominio inicial, junto con la descripción de las clases que lo forman.

Posteriormente, se procederá a explicar el diseño de la arquitectura que se va a emplear, así como los patrones de diseño, descomposición modular, estructura de la Base de Datos y realización de los casos de uso en diseño.

Estas últimas historias de usuario que se acaban de mencionar, como ya se ha comentado previamente, se van a realizar de manera parcial; es decir, sólo se va a obtener esta información relativa a la funcionalidad software que se va a implementar en este sprint (en relación con el almacenamiento y recuperación de pacientes en el sistema).

Una vez que se disponga de toda esta información en relación con el análisis y diseño de la aplicación, se procederá a implementar la creación de perfiles de pacientes junto con su almacenamiento en una Base de Datos, así como su posterior recuperación de la misma.

Y, por último, y dado que lo exige Scrum, se continuará con el documento en el que se detallan las pruebas que se han realizado sobre esta nueva funcionalidad de la aplicación y los resultados que se han obtenido.

II.4.3. Sprint backlog

Backlog ID	Tarea	Tipo	Estimación
1	Elaborar especificación inicial de requisitos	Análisis	3
2	Elaborar diagrama inicial de casos de uso	Análisis	5
2	Elaborar listado y especificación inicial de casos de uso	Análisis	5
2	Elaborar el diagrama de actividad de cada caso de uso	Análisis	8
3	Elaborar el modelo de dominio inicial	Análisis	5
3	Elaborar descripción de las clases del modelo de dominio inicial	Análisis	3
4	Elaborar descripción arquitectura MVP a implementar	Documentación	5
4	Elaborar descripción patrones diseño iniciales a utilizar	Documentación	8
4	Elaborar diagrama inicial arquitectura del proyecto	Diseño	3
4	Elaborar descripción inicial arquitectura del proyecto	Diseño	1
4	Elaborar descomposición modular inicial del proyecto	Diseño	3
4	Elaborar diagramas iniciales módulos del proyecto	Diseño	8
4	Elaborar diagramas de secuencia de cada caso de uso	Diseño	13
5	IU - Crear vista que muestra listado de pacientes en el sistema	Implementación	8

5	IU - Crear vista con los campos a rellenar de un nuevo paciente	Implementación	21
5	Crear Base de Datos	Implementación	8
5	Añadir lógica para almacenar paciente en la Base de Datos	Implementación	13
6	Obtener el listado de pacientes almacenados en la Base de Datos	Implementación	13
6	IU - Crear vista que muestra la información de un paciente	Implementación	8
6	Recuperar la información de un paciente de la Base de Datos	Implementación	5
7	Implantación arquitectura Modelo-Vista-Presentador	Implementación	13
8	Implantación patrón de diseño <i>Inyección de dependencias</i>	Implementación	13
11	Actualizar documento sobre la implementación y pruebas realizadas	Documentación	5

Tabla 67 - Sprint backlog - Sprint 2

II.4.4. Comentarios

A la hora de implementar el patrón arquitectónico MVP (en el anterior sprint no se implementó debido a que la aplicación sólo consistía en una clase y un layout que mostraba el goniómetro), ha resultado ser más complejo de lo que se había estimado.

Se trataba de la primera vez que se utilizaba este patrón (en la asignatura *Interacción Persona-Computadora* se implantó el patrón Modelo-Vista-Controlador, pero en aquel entonces lo se realizó entre dos personas y desde entonces no se ha vuelto a utilizar) y, al no existir un consenso, sino que sólo existen unas pautas generales de cómo llevarlo a cabo, fue necesario revisar mucha documentación para intentar implantarlo de la mejor forma posible.

Además de ese asunto, en el aspecto en el que también se tuvieron ciertos problemas fue a la hora de utilizar el patrón *Inyección de dependencias*. De nuevo se trataba de algo que nunca se había implantado y no se sabía muy bien cómo hacerlo, por lo que, se tuvo que revisar bastante documentación, artículos y opiniones en foros especializados para ver cuál era la estrategia que más suelen utilizar otras personas del sector.

Finalmente, esta última tarea se pudo llevar a cabo mediante la utilización del framework Dagger 2, el cual es explicado en el apartado 7.1.1.1 (Implementación – Uso de bibliotecas externas – Dagger 2).

II.4.5. Revisión de Sprint

II.4.5.1. Incremento

- Puntos de usuario planificados: **177**
- Puntos de usuario realizados: **177**
- Media de puntos de usuario: **7.7 puntos/sprint**
- Número de tareas realizadas: **23 tareas/sprint**

Como se puede apreciar, el número de puntos de usuario se ha incrementado con respecto al anterior sprint en casi un 53% (177 frente a 116), así como el número de tareas en casi un 48% (23 frente a 16) y la media de puntos de usuario en un **6.2%** (7.7 frente a 7.25), lo que indica que el esfuerzo en ambos sprints ha sido muy similar (aunque algo superior en éste), algo que es una buena señal, ya que, por ahora, se están planificando correctamente los sprints.

Con respecto a la funcionalidad, se ha podido cumplir con toda la que se había planificado realizar al comienzo del sprint.

En concreto, ya se dispone de unos datos completos de análisis (requisitos, casos de uso y modelo de dominio) y de diseño (arquitectura empleada, patrones de diseño, descomposición modular y realización en diseño de los casos de uso) sobre la funcionalidad implementada hasta ahora de la aplicación.

Y, en relación con la aplicación, ya se dispone del listado de pacientes almacenados en el sistema, se pueden almacenar nuevos pacientes en el mismo y recuperar su información (de momento sin mediciones asociadas a éstos).

También se ha actualizado el documento en relación con la implementación y las pruebas para incluir todo lo relativo al framework Dagger 2, así como todas las pruebas que se han realizado para verificar el correcto funcionamiento de la aplicación hasta el momento.

Por último, para comprobar el correcto funcionamiento del incremento software en este sprint, se han realizado las pruebas: **Prueba-0023, Prueba-0025, Prueba-0026, Prueba-0027, Prueba-0028, Prueba-0029, Prueba-0030, Prueba-0031, Prueba-0032, Prueba-0036 y Prueba-0037.**

II.4.5.2. Riesgos

En este sprint se ha disparado, de nuevo, un único riesgo, **R02** (Mala estimación de la complejidad de alguna tarea del *sprint backlog*), que, ha resultado en que, desde el octavo día de comienzo del sprint, se han tenido que incrementar el número de horas de trabajo diarias de 6 a 10, así como el número de días de trabajo semanales de 6 a 7 para poder cumplir con la funcionalidad que se tenía que entregar al final de este sprint.

Este riesgo se ha disparado, como ya se ha dejado entrever en los comentarios del sprint, en las tareas *Implantación arquitectura Modelo-Vista-Presentador* e *Implantación patrón de diseño Inyección de dependencias*.

II.5. Sprint 3 – 20 de mayo a 3 de junio

II.5.1. Product backlog

En este apartado, se muestra la *pila de producto* o *product backlog* actualizada a fecha 20 de mayo, el día de comienzo del sprint 3.

ID	Prioridad	Descripción
1	Muy alta	Como Product Owner quiero disponer de una especificación completa de los requisitos del sistema.
2	Muy alta	Como Product Owner quiero disponer de un listado completo, junto con la especificación y realización en análisis de cada uno de los casos de uso del sistema.
3	Muy alta	Como Product Owner quiero disponer de un modelo de dominio del sistema, así como una descripción de las clases que lo forman.
4	Muy alta	Como Product Owner quiero disponer de un documento completo sobre el diseño de la arquitectura del sistema, así como una descripción de los patrones utilizados.

5	Muy alta	Como Usuario quiero poder decidir si almacenar o no cada una de las mediciones que realizo.
6	Alta	Como Usuario quiero poder visualizar las 10 últimas mediciones que he guardado.
7	Alta	Como Product Owner quiero disponer de un documento detallado sobre la implementación del sistema y las pruebas que se han realizado sobre el mismo.
8	Alta	Como Product Owner quiero disponer de un documento detallado sobre los resultados obtenidos tanto con Kinect como con la aplicación, así como de las conclusiones obtenidas a raíz de la comparación de ambos sistemas.
9	Media	Como Usuario quiero disponer de un menú de información sobre la aplicación.
10	Media	Como Usuario quiero poder utilizar la aplicación en inglés.
11	Media	Como Usuario quiero disponer de un manual de usuario de la aplicación.

Tabla 68 - Product backlog - Sprint 3

II.5.2. Descripción del sprint

En este tercer sprint, se va a actualizar toda la información en relación con la especificación de requisitos, modelado de casos de uso y del dominio y la arquitectura y diseño de la aplicación.

Se prevé que esta primera parte va a ser bastante larga, debido a que, con la incorporación de las mediciones en la aplicación, ésta estará completada prácticamente en su totalidad. Por consecuencia, se actualizará toda la documentación en cuanto a análisis y diseño con la información de las mediciones, pero también se revisará lo que se tiene hasta el momento, con el fin de verificar que, efectivamente, todo cuadra.

Una vez se disponga de toda la documentación actualizada, se incorporará la posibilidad de almacenar mediciones en el sistema, así como se actualizarán los perfiles de los pacientes para que se puedan visualizar las mediciones que se han realizado sobre éstos.

Y, por último, y dado que lo exige Scrum, se continuará con el documento en el que se detallan las pruebas que se han realizado sobre esta nueva funcionalidad de la aplicación y los resultados que se han obtenido.

II.5.3. Sprint backlog

Backlog ID	Tarea	Tipo	Estimación
1	Actualizar especificación de requisitos	Análisis	1
2	Actualizar diagrama de casos de uso	Análisis	3
2	Actualizar listado y especificación de casos de uso	Análisis	8
2	Elaborar los nuevos diagramas de actividad de cada caso de uso	Análisis	8
3	Actualizar el modelo de dominio	Análisis	5
3	Actualizar descripción de las clases del modelo de dominio	Análisis	3
4	Elaborar descripción nuevos patrones diseño a utilizar	Documentación	3
4	Actualizar diagrama arquitectura del proyecto	Diseño	2
4	Actualizar descomposición modular del proyecto	Diseño	3
4	Actualizar diagramas módulos del proyecto	Diseño	5
4	Elaborar los nuevos diagramas de secuencia de cada caso de uso	Diseño	13

5	Añadir comportamiento al botón para almacenar medición	Implementación	5
5	IU - Crear botón para añadir una medición a un paciente	Implementación	5
5	IU - Crear vista que muestra las opciones sobre nueva medición	Implementación	21
5	Actualizar la Base de Datos para incluir mediciones	Implementación	8
5	Añadir lógica para almacenar mediciones en la Base de Datos	Implementación	13
5	IU - Actualizar el perfil del paciente para mostrar sus mediciones	Implementación	8
7	Actualizar documento sobre la implementación y pruebas realizadas	Documentación	5

Tabla 69 - Sprint backlog - Sprint 3

II.5.4. Comentarios

La creación de la interfaz gráfica asociada con la adición de una nueva medición a un paciente ha resultado ser de mayor complejidad de la esperada, ya que, en función de los valores que vaya seleccionando el usuario para un lado, articulación... se cargan unos valores u otros (en función de los movimientos que puede realizar cada articulación).

A parte de esto, se han encontrado problemas al mostrar la información de las mediciones de cada uno de los pacientes, ya que, muchas veces esta información no se podía visualizar al completo (su tamaño excedía el ancho de la pantalla del dispositivo), dependiendo del nombre de la articulación y el movimiento seleccionado. Finalmente se pudo solucionar permitiendo que la tabla de las mediciones se pudiera desplazar horizontalmente.

II.5.5. Revisión de Sprint

II.5.5.1. Incremento

- Puntos de usuario planificados: **119**
- Puntos de usuario realizados: **119**
- Media de puntos de usuario: **6.6 puntos/sprint**
- Número de tareas realizadas: **18 tareas/sprint**

Como se puede, apreciar, el número de puntos de usuario se ha reducido con respecto a los sprints anteriores, así como la media de puntos de usuario, lo que indica que el esfuerzo en este sprint ha sido menor que anteriores sprints, algo que se ha notado en varios aspectos. El primero de ellos (como se va a comentar en el siguiente apartado) es que no se ha tenido que incrementar el número de días de trabajo semanales, y el segundo de ellos, es que no se ha tenido que incrementar las horas de trabajo diarias excesivamente.

El esfuerzo de trabajo en este sprint ha sido bastante más asequible y sostenible que el que se ha estado manteniendo hasta ahora en anteriores sprints, por lo que, se intentará, en la medida de lo posible, que el esfuerzo en sucesivos sprints se asemeje más a estos valores que a los de los primeros sprints.

Con respecto a la funcionalidad, se ha podido cumplir con toda la que se había planificado realizar al comienzo del sprint.

En concreto, se dispone de los documentos de análisis y diseño en un estado muy avanzado, ya que, la funcionalidad de la aplicación está implementada casi por completo.

A parte de esto, ya se pueden almacenar mediciones sin ningún problema, así como recuperar, para cada paciente, cada una de las mediciones que se han realizado sobre él.

Con toda esta funcionalidad añadida, se ha obtenido la **versión 0.2** de la aplicación.

También se ha actualizado el documento en relación con la implementación y las pruebas para incluir todas las pruebas que se han realizado para verificar el correcto funcionamiento de la aplicación hasta el momento, y, salvo que posteriormente se tenga que realizar algún cambio, también se ha finalizado el modelo de dominio de la aplicación, así como la descripción de las clases que lo forman.

Por último, para comprobar el correcto funcionamiento del incremento software en este sprint, se han realizado las pruebas: **Prueba-0005, Prueba-0007, Prueba-0008, Prueba-0009, Prueba-0010, Prueba-0011, Prueba-0012, Prueba-0013, Prueba-0014, Prueba-0015, Prueba-0016, Prueba-0017, Prueba-0018 y Prueba-0038.**

II.5.5.2. Riesgos

En este sprint se han disparado varios riesgos.

El primero de ellos es el riesgo **R02** (Mala estimación de la complejidad de alguna tarea del *sprint backlog*), aunque, en este sprint, no ha sido necesario tomar medidas demasiado drásticas para subsanarlo. Simplemente se ha tenido que incrementar el número de horas de trabajo diarias de 6 a 8 en 5 de los 13 días de trabajo que tiene un sprint. En este caso, no se ha tenido que incrementar el número de días de trabajo semanales.

Este riesgo se ha disparado, como ya se ha dejado entrever en los comentarios del sprint, en las tareas *IU - Crear vista que muestra las opciones sobre nueva medición* e *IU - Actualizar el perfil del paciente para mostrar sus mediciones*.

Además de este riesgo, se ha disparado el riesgo **R04** (Descubrimiento de un nuevo requisito). En este caso, se han obtenido los nuevos requisitos: *Filtración de perfiles de pacientes, Borrado de perfiles de pacientes y Borrado de mediciones realizadas* (ver apartado 5.1.2 – Requisitos funcionales).

En este caso, dado que no se tratan de requisitos críticos de la aplicación, se ha podido posponer su incorporación a la misma hasta el próximo sprint, sin que afectara a la funcionalidad a desarrollar en éste.

II.6. Sprint 4 – 5 de junio a 19 de junio

II.6.1. Product backlog

En este apartado, se muestra la *pila de producto* o *product backlog* actualizada a fecha 5 de junio, el día de comienzo del sprint 4.

ID	Prioridad	Descripción
1	Muy alta	Como Product Owner quiero disponer de una especificación completa de los requisitos del sistema.
2	Muy alta	Como Product Owner quiero disponer de un listado completo, junto con la especificación y realización en análisis de cada uno de los casos de uso del sistema.

3	Muy alta	Como Product Owner quiero disponer de un documento completo sobre el diseño de la arquitectura del sistema, así como una descripción de los patrones utilizados.
4	Muy alta	Como Usuario quiero poder visualizar las 10 últimas mediciones que he guardado.
5	Muy alta	Como Usuario quiero poder filtrar el listado de pacientes almacenados en el sistema por su nombre, su ID o las etiquetas que haya utilizado al crear su perfil.
6	Muy alta	Como Usuario quiero poder eliminar una de las últimas 10 mediciones que he almacenado en el sistema.
7	Muy alta	Como Usuario quiero poder eliminar el perfil de un paciente del sistema junto con todas las mediciones que he realizado sobre él.
8	Alta	Como Product Owner quiero disponer de un documento detallado sobre la implementación del sistema y las pruebas que se han realizado sobre el mismo.
9	Alta	Como Product Owner quiero disponer de un documento detallado sobre los resultados obtenidos tanto con Kinect como con la aplicación, así como de las conclusiones obtenidas a raíz de la comparación de ambos sistemas.
10	Media	Como Usuario quiero disponer de un menú de información sobre la aplicación.
11	Media	Como Usuario quiero poder utilizar la aplicación en inglés.
12	Media	Como Usuario quiero disponer de un manual de usuario de la aplicación.

Tabla 70 - Product backlog - Sprint 4

II.6.2. Descripción del sprint

En este cuarto sprint, se va a actualizar toda la información en relación con la especificación de requisitos, modelado de casos de uso y la arquitectura y diseño de la aplicación.

Se prevé que los cambios que se van a realizar en la documentación no serán demasiado grandes, ya que, la nueva funcionalidad a incluir en este sprint, aun siendo importante, es de un tamaño significativamente menor que la que se ha tenido que incluir en anteriores sprints.

Una vez se disponga de toda la documentación actualizada, se procederá a incorporar, en primera instancia, la recuperación de las últimas 10 mediciones almacenadas por el usuario en el sistema. En segunda instancia, se incorporará la posibilidad de filtrar el listado de pacientes almacenados en el sistema atendiendo a los criterios de Nombre, ID o Tags de un paciente. En última instancia, se incorporará la posibilidad de borrar una medición del sistema (siempre que ésta sea una de las 10 últimas almacenadas por el usuario).

Y, por último, y dado que lo exige Scrum, se continuará con el documento en el que se detallan las pruebas que se han realizado sobre esta nueva funcionalidad de la aplicación y los resultados que se han obtenido.

II.6.3. Sprint backlog

Backlog ID	Tarea	Tipo	Estimación
1	Actualizar especificación de requisitos	Análisis	3
2	Actualizar diagrama de casos de uso	Análisis	3
2	Actualizar listado y especificación de casos de uso	Análisis	5
2	Elaborar los nuevos diagramas de actividad de cada caso de uso	Análisis	5

3	Actualizar descomposición modular del proyecto	Diseño	5
3	Actualizar diagramas módulos del proyecto	Diseño	5
3	Elaborar los nuevos diagramas de secuencia de cada caso de uso	Diseño	8
4	IU - Crear vista que muestra el listado de las últimas 10 mediciones almacenadas	Implementación	21
4	Añadir lógica para la recuperación de la Base de Datos de las mediciones	Implementación	8
5	IU - Actualizar vista que muestra el listado de pacientes para incluir un cuadro de texto y un filtro con los campos <i>Nombre, ID y Tags</i>	Implementación	8
5	Implementar interfaz con evento <i>listener</i> para detectar la escritura de un carácter en el cuadro de búsqueda	Implementación	1
5	Añadir lógica para la recuperación de pacientes de la Base de Datos en función de la información introducida por el usuario y el filtro seleccionado	Implementación	13
6	IU - Añadir botón para eliminar la medición a cada una de las mediciones mostradas en la vista de las últimas 10 mediciones almacenadas	Implementación	5
6	Añadir lógica para la eliminación de la medición seleccionada de la Base de Datos	Implementación	8
6	Actualizar el listado de las últimas 10 mediciones almacenadas que se muestra al usuario después de borrar una medición	Implementación	8
8	Actualizar documento sobre la implementación y pruebas realizadas	Documentación	5

Tabla 71 - Sprint backlog - Sprint 4

II.6.4. Comentarios

La creación de la interfaz gráfica que muestra las últimas 10 mediciones almacenadas en el sistema por el usuario ha llevado más tiempo del esperado. Se han presentado algunos problemas a la hora de lograr que toda la información de cada una de las mediciones se pudiera visualizar correctamente y, al igual que en el sprint anterior, se ha presentado el problema de que algunas mediciones no se podían visualizar de manera completa en la pantalla debido a que éstas excedían el ancho de la misma.

La solución a este problema fue la misma que en el anterior caso; sin embargo, se complicó algo más que en el sprint anterior debido a que, al incluir un nuevo elemento *HorizontalScrollView*, el resto de estructura de la medición se desalineaba completamente, por lo que, corregirlo ha llevado más tiempo del esperado.

II.6.5. Revisión de Sprint

II.6.5.1. Incremento

- Puntos de usuario planificados: **111**
- Puntos de usuario realizados: **111**
- Media de puntos de usuario: **6.9 puntos/sprint**
- Número de tareas realizadas: **16 tareas/sprint**

Como se puede apreciar, en este sprint se ha disminuido el número de puntos de usuario totales en un 7.2% (119 frente a 111), así como el número de tareas realizadas en un 12.5% (18 frente a 16); sin embargo, la media de puntos de usuario ha aumentado en un **4.5%** (6.6 frente a 6.9).

En términos generales, el esfuerzo en estos últimos dos sprints ha sido similar. Algo ligeramente superior al anterior, pero ha habido una diferencia notable entre éste y los dos primeros.

Con respecto a la funcionalidad, se ha podido cumplir con toda la que se había planificado realizar al comienzo del sprint.

En concreto, se dispone de los documentos de análisis y diseño prácticamente terminados, ya que, es mínima la funcionalidad de la aplicación que falta por implementar.

A parte de esto, ya se puede obtener el listado con las 10 últimas mediciones almacenadas en el sistema, así como borrarlas del mismo.

También se puede hacer un filtrado del listado de pacientes que se encuentran en el sistema, bien por nombre, ID o Tags (etiquetas) introducidas durante la creación del perfil del mismo.

Además, se ha actualizado el documento en relación con la implementación y las pruebas para incluir todas las pruebas que se han realizado para verificar el correcto funcionamiento de la aplicación hasta el momento.

Por último, para comprobar el correcto funcionamiento del incremento software en este sprint, se han realizado las pruebas: **Prueba-0019, Prueba-0020, Prueba-0021, Prueba-0022 y Prueba-0024.**

II.6.5.2. Riesgos

En este sprint se ha disparado, de nuevo, un único riesgo, **R02** (Mala estimación de la complejidad de alguna tarea del *sprint backlog*), aunque, en este sprint, no ha sido necesario tomar medidas demasiado drásticas para subsanarlo. Simplemente se ha tenido que incrementar el número de horas de trabajo diarias de 6 a 8 en 7 de los 13 días de trabajo que tiene un sprint. En este caso, no se ha tenido que incrementar el número de días de trabajo semanales.

Este riesgo se ha disparado, como ya se ha dejado entrever en los comentarios del sprint, en la tarea *IU - Crear vista que muestra el listado de las últimas 10 mediciones almacenadas*.

II.7. Sprint 5 – 20 de junio a 4 de julio

II.7.1. Product backlog

En este apartado, se muestra la *pila de producto* o *product backlog* actualizada a fecha 20 de junio, el día de comienzo del sprint 5.

ID	Prioridad	Descripción
1	Muy alta	Como Product Owner quiero disponer de una especificación completa de los requisitos del sistema.
2	Muy alta	Como Product Owner quiero disponer de un listado completo, junto con la especificación y realización en análisis de cada uno de los casos de uso del sistema.

3	Muy alta	Como Product Owner quiero disponer de un documento completo sobre el diseño de la arquitectura del sistema, así como una descripción de los patrones utilizados.
4	Muy alta	Como Usuario quiero poder eliminar el perfil de un paciente del sistema junto con todas las mediciones que he realizado sobre él.
5	Muy alta	Como Product Owner quiero disponer de un documento detallado sobre la implementación del sistema y las pruebas que se han realizado sobre el mismo.
6	Muy alta	Como Product Owner quiero disponer de un documento detallado sobre los resultados obtenidos tanto con Kinect como con la aplicación, así como de las conclusiones obtenidas a raíz de la comparación de ambos sistemas.
7	Muy alta	Como Product Owner quiero disponer de la documentación de la aplicación, así como el código de la misma, actualizada y revisada, tras corregir los posibles errores que hayan surgido durante su realización.
8	Muy alta	Como Usuario quiero disponer de un menú de información sobre la aplicación.
9	Muy alta	Como Usuario quiero poder utilizar la aplicación en inglés.
10	Muy alta	Como Usuario quiero disponer de un manual de usuario de la aplicación.

Tabla 72 - Product backlog - Sprint 5

II.7.2. Descripción del sprint

Este sprint es el último de los planificados para la realización de este proyecto, por lo tanto, a la finalización del mismo, todas las historias de usuario restantes deberían estar completadas.

En concreto, se van a completar los documentos de análisis y diseño de la aplicación, antes de incluir la implementación de la funcionalidad software referente a este sprint.

Posteriormente, se incluirá la posibilidad de eliminar pacientes (junto con todas sus mediciones) del sistema, un menú de información acerca de la aplicación y se realizará la traducción de la misma al inglés.

Una vez toda esta funcionalidad haya sido implementada, se procederá a completar el documento de implementación y pruebas con todas las pruebas realizadas tras la implementación de la funcionalidad de este sprint.

Tras la finalización de estas tareas, se procederá a la creación del anexo referente al manual de usuario de la aplicación.

Y, por último, se procederá a realizar la comparación de los resultados obtenidos tanto con Kinect como con la aplicación tras realizar una serie de mediciones, para su posterior inclusión en el estudio comparativo, así como las conclusiones obtenidas tras la realización de este proyecto.

II.7.3. Sprint backlog

Backlog ID	Tarea	Tipo	Estimación
1	Actualizar especificación de requisitos	Análisis	2
2	Actualizar diagrama de casos de uso	Análisis	2
2	Actualizar listado y especificación de casos de uso	Análisis	3
2	Elaborar los nuevos diagramas de actividad de cada caso de uso	Análisis	3

3	Actualizar descomposición modular del proyecto	Diseño	3
3	Actualizar diagramas módulos del proyecto	Diseño	5
3	Elaborar los nuevos diagramas de secuencia de cada caso de uso	Diseño	8
4	IU - Añadir botón para eliminar el perfil del paciente	Implementación	1
4	Añadir lógica para la eliminación del paciente seleccionado (junto con sus mediciones) de la Base de Datos	Implementación	5
8	IU - Crear vista que muestra la información sobre la aplicación	Implementación	8
8	Añadir creación del Fragment asociado a esta vista	Implementación	5
9	Crear nuevo fichero de <i>String</i> con la traducción al inglés de todos los <i>String</i> empleados en la aplicación	Implementación	13
9	Implementar la traducción de las clases <i>Enum</i> para almacenar y recuperar las mediciones y los pacientes en inglés	Implementación	21
5	Actualizar documento sobre la implementación y pruebas realizadas	Documentación	5
10	Realizar el manual de usuario de la aplicación	Documentación	13
6	Realizar mediciones tanto con Kinect como con la aplicación	Documentación	8
6	Realizar el estudio comparativo tras las mediciones obtenidas	Documentación	13
6	Redactar las conclusiones obtenidas tras la realización de este proyecto	Documentación	8
7	Revisar toda la documentación y el código tras la finalización del proyecto	Documentación	8

Tabla 73 - Sprint backlog - Sprint 5

II.7.4. Comentarios

La implementación de traducción de las clases *Enum* ha resultado ser significativamente más complejo de lo esperado. En un principio se optó por incluir nuevos valores a estas clases con las articulaciones, movimientos... en inglés (algo que es un completo disparate para un futuro ingeniero de software, pero servía para probar si se almacenaba y se recuperaba bien la información en este idioma).

Posteriormente a esto, se empleó buena parte del tiempo para investigar sobre cómo implementar la traducción de una clase Enumerada, y se barajó la posibilidad de utilizar una clase *i18n* (internacionalización), para que, dada una clave de la clase *Enum* escogiera un valor u otro en función del idioma; sin embargo, finalmente se encontró una biblioteca externa que había sido diseñada especialmente para este caso en concreto. Esta biblioteca se llama *Enum Format*, y es explicada en detalle en el apartado 7.1.1.2 (Implementación y pruebas - Implementación - Uso de bibliotecas externas - Enum Format).

Tras haber solucionado este problema, hacia la mitad del sprint, se revisó con el Product Owner toda la documentación que se poseía hasta el momento, así como la aplicación. El resultado fue que se encontraron varios errores en ambas partes.

En cuanto a la documentación, existían varios errores tanto en la parte de análisis como en la de diseño (además de errores varios algo menores en algunas partes del resto de elementos de la documentación); y, en cuanto a la aplicación, como ya se conocía desde el primer sprint, ésta no reconocía movimientos de rotación, lo cual, no es un fallo como tal, pero sí que es una gran limitación en la aplicación que, si se puede subsanar, se logrará obtener una aplicación más completa.

En este sentido, sobre todo para implementar el reconocimiento del movimiento del dispositivo en los tres ejes tridimensionales, se ha tenido que dedicar varios días de trabajo completos para recabar información, de nuevo, sobre los sensores que se pueden utilizar para este propósito, así como la manera de obtener el ángulo final de desplazamiento en función de las lecturas obtenidas.

Finalmente, esto se ha logrado implementar, sin embargo, se ha terminado por producir un retraso en la fecha de finalización del sprint, no siendo el 4 de julio sino el 10 de julio.

II.7.5. Revisión de Sprint

II.7.5.1. Incremento

- Puntos de usuario planificados: **134**
- Puntos de usuario realizados: **134**
- Media de puntos de usuario: **7.1 puntos/sprint**
- Número de tareas realizadas en el sprint: **19 tareas/sprint**

Como se puede apreciar, el número total de puntos de usuario se ha incrementado, así como el número de tareas realizadas en el sprint; sin embargo, la media de puntos de usuario en este sprint ha sido prácticamente idéntica con respecto al sprint anterior (7.1 frente a 6.9), por lo que, el esfuerzo en ambos ha sido muy similar, al menos en cuanto a las tareas que se había pensado llevar a cabo en primera instancia.

Con respecto a la funcionalidad, se ha podido cumplir con toda la que se había planificado realizar al comienzo del sprint, además de todas las correcciones que han surgido durante la realización del mismo.

En concreto, se han terminado los documentos de análisis, diseño e implementación y pruebas sobre la aplicación. Ya se pueden borrar pacientes del sistema, visualizar el menú de información sobre la aplicación, y utilizar la misma tanto en español como en inglés.

Con esta funcionalidad añadida, ya se dispone de **la versión 1.0** o Release de la aplicación.

También se dispone de un manual de usuario, presentado en el anexo anterior, así como un documento con el estudio comparativo realizado tras la obtención de una serie de mediciones, tanto con Kinect como con la aplicación, y las conclusiones obtenidas tras la realización de este proyecto.

Además, toda la documentación presentada en este documento, así como el código de la aplicación ha sido revisado exhaustivamente para asegurar su corrección antes de ser entregada.

Por último, para comprobar el correcto funcionamiento del incremento software en este sprint, se han realizado las pruebas: **Prueba-0033**, **Prueba-0034**, **Prueba-0035** y se repitieron todas las pruebas realizadas hasta ahora con el idioma del dispositivo en inglés (Estados Unidos) para verificar el funcionamiento de la aplicación en este idioma.

II.7.5.2. Riesgos

Este ha sido, sin duda, el sprint en el que más riesgos se han disparado, como se ha podido entrever en los comentarios del sprint.

En concreto, se han disparado los riesgos **R02** (Mala estimación de la complejidad de alguna tarea del *sprint backlog*), aunque, en este sprint, no ha sido necesario tomar medidas demasiado drásticas para subsanarlo.

Simplemente se ha tenido que incrementar el número de horas de trabajo diarias de 6 a 8 en 8 de los 13 días de trabajo que tiene un sprint. En este caso, no se ha tenido que incrementar el número de días de trabajo semanales.

Este riesgo se ha disparado en la tarea *Implementar la traducción de las clases Enum para almacenar y recuperar las mediciones y los pacientes en inglés*.

Los principales desencadenantes de que se haya producido un retraso en este último sprint han sido los siguientes riesgos: **R05** (Errores en fase de análisis), **R06** (Errores en fase de diseño), **R07** (Errores en fase de implementación) y **R09** (Documentación escasa o mal elaborada).

Si bien se han producido los errores **R05**, **R06** y **R09**, el impacto que éstos han tenido sobre la duración del sprint ha sido mínima (a lo sumo 1 día de trabajo extra entre todos); sin embargo, **R07** (en lo referente a la obtención de datos de los sensores en el eje Z para la detección de los movimientos de rotación), ha sido el principal culpable de este retraso. Todo el proceso de recolección de nueva información sobre los sensores y los datos que se obtienen de cada uno, así como el cálculo del ángulo de desplazamiento en función de éstos, ha supuesto un esfuerzo de trabajo extra de entre 3 y 4 días más de lo estimado en el sprint, provocando este eventual retraso con respecto a la fecha de finalización del sprint.

Además, la corrección del error del goniómetro era de vital importancia, por lo que, se tuvieron que paralizar el resto de tareas del sprint hasta que se corrigió. Posteriormente se finalizaron el resto de tareas salvo la revisión de la documentación y el código, se procedió a aplicar las correcciones en la parte de análisis, diseño y al resto de la documentación; y, una vez hecho todo esto, se revisó todo lo que se tenía antes de entregarlo.