



Universidad de Valladolid

Escuela Técnica Superior de Ingeniería Informática

Trabajo de fin de grado

Grado en Ingeniería Informática. Mención en Ingeniería de Software.

Aplicación Android para rehabilitación de articulaciones

Autor:

Jairo Castrillejo Alonso

Tutor:

Miguel Ángel Laguna Serrano

RESUMEN

Este Trabajo de Fin de Grado consiste en el desarrollo de una aplicación móvil que servirá de ayuda para pacientes en estado de rehabilitación de cualquier lesión en la articulación de la muñeca. Ésta permitirá a los usuarios de la misma realizar varios ejercicios de rehabilitación con el teléfono móvil en su mano. De esta forma, la aplicación recogerá los ángulos máximos de cada uno de los ejercicios y los irá almacenando en una base de datos. Así, se podrá comprobar la progresión de la movilidad de la articulación durante todo el proceso de rehabilitación para, en un momento dado, sacar conclusiones respecto a si se está produciendo, o no, una mejora en el paciente.

Para llevar a cabo dicha tarea, utilizaremos varios de los sensores presentes en la gran mayoría de los dispositivos móviles. Dichos sensores serán: el Acelerómetro, el Magnetómetro y el Giroscopio.

ABSTRACT

This Degree Final Project consists on the development of a mobile application that helps patients in rehabilitation of any wrist injury. This application allows their users to run several rehabilitation exercises with their mobile phone in their hand. In this way, the application will collect the maximum angles of each of the exercises and store them in a database. Therefore, progression of joint mobility throughout the rehabilitation process can be checked to draw conclusions as to whether or not an improvement is occurring in the patient at a given time.

To accomplish this task, we will use several of the sensors on most mobile devices. These sensors will be: the Accelerometer, the Magnetometer and the Gyroscope.

ÍNDICE

Capítulo 1	Introducción.....	9
1.1	Objetivos.....	9
1.2	Contexto de desarrollo.....	10
1.2.1	Plataforma.....	10
1.2.2	Entorno de desarrollo.....	10
1.3	Ejercicios incluidos en la aplicación.....	10
Capítulo 2	Tecnología utilizada.....	11
2.1	Descripción de la tecnología utilizada.....	11
Capítulo 3	Gestión del proyecto.....	14
3.1	Organización del proyecto.....	14
3.2	Roles y responsabilidades	14
3.3	Estimación de tiempos.....	15
3.4	Plan del proyecto.....	15
3.5	Calendario del proyecto.....	16
3.6	Recursos del proyecto.....	17
3.7	Análisis de riesgos.....	18
3.8	Planificación de fases.....	20
3.8.1	Planificación temporal de la fase de inicio.....	20
3.8.2	Planificación temporal de la fase de Elaboración-Iteración 1.....	20
3.8.3	Planificación temporal de la fase de Elaboración-Iteración 2.....	21
3.8.4	Planificación temporal de la fase de Construcción-Iteración 1.....	22
3.8.5	Planificación temporal de la fase de Construcción-Iteración 2.....	22
3.8.6	Planificación temporal de la fase de Transición-Iteración 1.....	23
3.9	Estimación de costes.....	24
3.9.1	Costes hardware.....	24
3.9.2	Costes software.....	24
3.9.3	Costes recursos humanos.....	24
3.9.4	Coste total.....	23
Capítulo 4	Análisis de la aplicación.....	26
4.1	Participantes en el proyecto.....	26
4.2	Objetivos del sistema.....	27
4.3	Requisitos del sistema.....	28
4.3.1	Requisitos funcionales.....	28
4.3.2	Requisitos no funcionales.....	31
4.4	Casos de uso.....	32
4.4.1	Definición de actores.....	32
4.4.2	Diagramas de casos de uso.....	33

4.4.3	Casos de uso del sistema.....	34
4.5	Modelo de dominio.....	42
4.6	Diagramas de secuencia.....	43
4.6.1	Crear programa de rehabilitación.....	43
4.6.2	Realizar ejercicio de Flexión-Extensión.....	44
4.6.3	Realizar ejercicio de Abducción-Aducción.....	45
4.6.4	Realizar ejercicio de Supinación-Pronación.....	46
4.6.5	Consultar programa de rehabilitación.....	47
4.6.6	Finalizar programa de rehabilitación.....	48
4.6.7	Eliminar programa de rehabilitación.....	49
4.6.8	Realizar calibración.....	50
Capítulo 5	Diseño de la aplicación.....	51
5.1	Descripción de la arquitectura lógica y física del sistema.....	51
5.1.1	Vista lógica del sistema. El patrón MVP.....	51
5.2	Diagrama de clases de diseño.....	54
5.2.1	Paquete Vista.....	54
5.2.2	Paquete Presentador.....	57
5.2.3	Paquete Modelo.....	59
5.2.4	Paquete Utilidades.....	60
5.2.5	Visión general de las clases del proyecto.....	61
5.3	Diagrama de casos de uso de diseño.....	62
5.4	Diseño de la base de datos.....	63
5.5	Diseño de la interfaz.....	63
5.5.1	Pantalla de inicio.....	64
5.5.2	Pantalla de creación de programa de rehabilitación.....	66
5.5.3	Pantalla de detalle de programa.....	67
5.5.4	Pantalla realización de ejercicio.....	73
Capítulo 6	Implementación de la aplicación.....	79
6.1	Implementación del patrón MVP.....	79
6.2	Algoritmo Sensor Fusion.....	82
6.2.1	Puesta a punto e inicialización.....	83
6.2.2	Recogida y procesamiento de datos de sensores.....	84
6.2.3	El filtro complementario.....	89
6.2.4	Uso de la clase SensorFusion.....	90
6.3	Decisiones tomadas para el desarrollo de la aplicación.....	91
6.3.1	Señales sonoras para la realización de los ejercicios.....	91
6.3.2	Posicionamiento del dispositivo para los distintos ejercicios.....	92
6.4	Estructura del proyecto Android.....	93
Capítulo 7	Pruebas realizadas.....	96
7.1	Pruebas unitarias.....	96

7.2	Pruebas del sistema.....	96
7.3	Pruebas experimentales.....	99
7.3.1	Flexión-Extensión (azimuth).....	99
7.3.2	Abducción-Aducción (pitch).....	101
7.3.3	Supinación-Pronación (roll).....	103
7.3.4	Conclusiones de las pruebas experimentales.....	104
Capítulo 8	Conclusiones.....	105
8.1	Objetivos alcanzados.....	105
8.2	Líneas de trabajo futuras.....	105
BIBLIOGRAFÍA.....		106
Anexo A	Contenido del CD.....	108

CAPÍTULO 1

INTRODUCCIÓN

Este Trabajo de Fin de Grado forma parte de una serie de trabajos orientados a la rehabilitación de otras articulaciones como el hombro y el codo. La motivación es la de poder ayudar a terapeutas a controlar el proceso de rehabilitación de pacientes con lesiones físicas pudiendo observar y controlar su evolución con el tiempo.

El papel de las tecnologías de la información y las comunicaciones (TIC) como mecanismo de integración social de las personas discapacitadas o dependientes en general se está imponiendo rápidamente, prestando servicios personalizados de bajo coste y con mayor facilidad. Monitorizar la realización de ejercicios de rehabilitación es el mejor ejemplo de estas aplicaciones.

1.1 Objetivos

El objetivo de este proyecto es crear una aplicación móvil que sea capaz de ayudar a pacientes con lesión de muñeca que se encuentren en proceso de rehabilitación. Para ello, la aplicación será capaz de medir los ángulos máximos de giro de diferentes movimientos de la articulación proponiendo al usuario varios ejercicios que habrá de realizar diariamente. Además, la aplicación mostrará la evolución de dichos ejercicios a través de una tabla que contendrá los datos recogidos durante todo el programa.

Para cumplir con este objetivo, se seguirán una serie de etapas que se completarán más adelante:

1. Planificación temporal.
2. Análisis de los objetivos y requisitos del sistema.
3. Diseño de la aplicación atendiendo a los requisitos analizados.
4. Implementación de la aplicación.
5. Pruebas, depuración y correcciones.
6. Documentación.

1.2 Contexto de desarrollo

1.2.1 Plataforma

Se ha elegido Android como plataforma de desarrollo para el proyecto por diversas razones:

- Es un sistema operativo de código abierto, lo que implica que los costes de su uso son nulos.
- Existen gran cantidad de ejemplos y documentación.
- Los dispositivos Android son más asequibles que otras alternativas como iOS.
- Todo lo referente a interfaz de usuario se desarrollará sólo para Smartphone ya que el uso de la aplicación no será útil para tabletas.
- La versión mínima requerida que se ha elegido para ofrecer la máxima compatibilidad es la 4.0.3 (Ice Cream Sandwich) que cubre el 97,4% de los dispositivos Android.

1.2.2 Entorno de desarrollo

Los elementos necesarios para realizar el proyecto han sido los siguientes:

- Windows 10 como sistema operativo donde se ha construido el proyecto completo.
- Microsoft Project 2010 para realizar la planificación temporal.
- REM 1.2.2 para el estudio de requisitos del sistema.
- Astah Professional 6.7.0 para la creación de los diagramas propios de la fase de Diseño.
- Android Studio 2.1.2 para el desarrollo completo de la aplicación.
- Microsoft Word para la elaboración de la memoria del proyecto.

1.3 Ejercicios incluidos en la aplicación

Podrán realizarse 3 tipos de ejercicios:

1. Flexión-Extensión.
2. Abducción-Aducción.
3. Supinación-Pronación.

CAPÍTULO 2

TECNOLOGÍA UTILIZADA

2.1 Descripción de la tecnología utilizada

Para este TFG se plantearon diferentes opciones para medir los ángulos con los sensores del Smartphone.

Teniendo en cuenta que los tres tipos de sensores para calcular la orientación del dispositivo son:

- **Acelerómetro:** Un componente electrónico de tamaño reducido, fabricado en silicio y cuyo funcionamiento se basa simplemente en la física y en la fuerza de la gravedad.
- **Magnetómetro:** Se trata de un componente electrónico capaz de medir y cuantificar la cantidad de fuerza magnética de un objeto. O para lo que muchos dispositivos lo usan, como brújula, detectando el polo norte magnético.
- **Giroscopio:** Es un dispositivo mecánico que nos ayudará a medir, mantener y cambiar la orientación de algún dispositivo. Provee de más y mejor información en cuanto a la posición del dispositivo.

La forma más común de obtener el comportamiento de un dispositivo Android es usando el método *SensorManager.getOrientation()* para obtener los tres ángulos de orientación. Éstos están basados en la salida del acelerómetro y del magnetómetro. El acelerómetro provee del vector de gravedad (el vector que apunta hacia el centro de la Tierra) y el magnetómetro actúa como una brújula. La información de ambos sensores es suficiente para calcular la orientación del dispositivo. No obstante, la salida de dichos sensores es poco precisa, especialmente la salida del sensor de campo magnético que incluye mucho ruido.

El giroscopio es el dispositivo más preciso con diferencia y tiene un muy corto tiempo de respuesta. Su desventaja es el temido “gyro drift” o desviación. El giroscopio provee las

velocidades de rotación angular para los tres ejes. Para calcular la orientación actual, estas velocidades se deben integrar en el tiempo. Esto se hace multiplicando las velocidades angulares por el intervalo de tiempo entre la última salida del sensor y la actual. Esto produce un incremento de rotación. La suma de todos los incrementos de rotación nos dice la orientación absoluta del dispositivo. Durante este proceso, se han ido introduciendo pequeños errores en cada iteración. Estos pequeños errores se van añadiendo en el tiempo resultando en un constante retardo en la rotación de la orientación calculada, el *gyro drift*.

Teniendo todo esto en cuenta, Paul Lawitzki, en su artículo *Android Sensor Fusion Tutorial* nos explica cómo solucionar estos problemas.

Utiliza la técnica denominada *Sensor Fusion* que no es más que combinar los tres sensores para corregir los errores o imprecisiones de cada uno. Para ello, pasa la salida combinada del acelerómetro y magnetómetro por un *low-pass filter* de modo que se elimina el ruido provocado por el segundo.

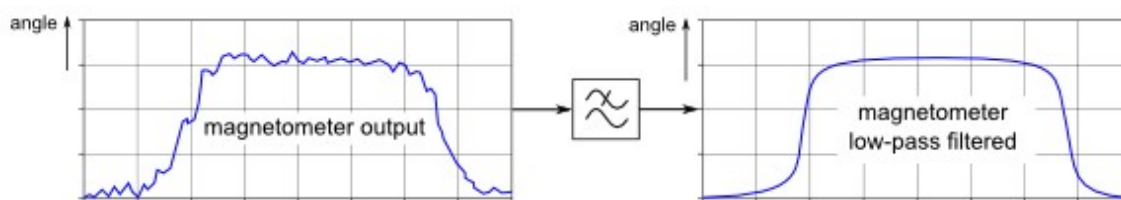


Figura 1: Low-pass filter magnetómetro (según Paul Lawitzki, en "Android Sensor Fusion Tutorial")

A su vez, para evitar el *gyro drift* la salida del giroscopio solo se aplica para cambios de orientación en intervalos cortos de tiempo. Pasando un *high-pass filter* a la salida del giroscopio y, combinándola con la salida filtrada del acelerómetro y el magnetómetro obtiene una medida de la orientación del dispositivo mucho más precisa.

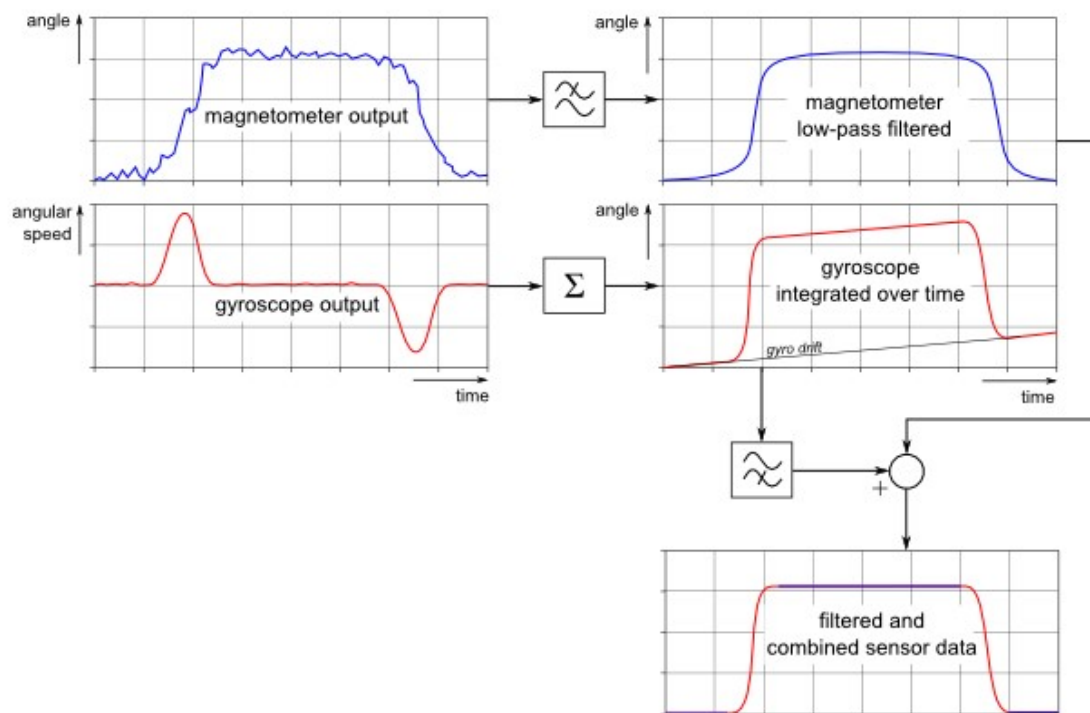


Figura 2: Sensor Fusion gráficas (según Paul Lawitzki, en "Android Sensor Fusion Tutorial")

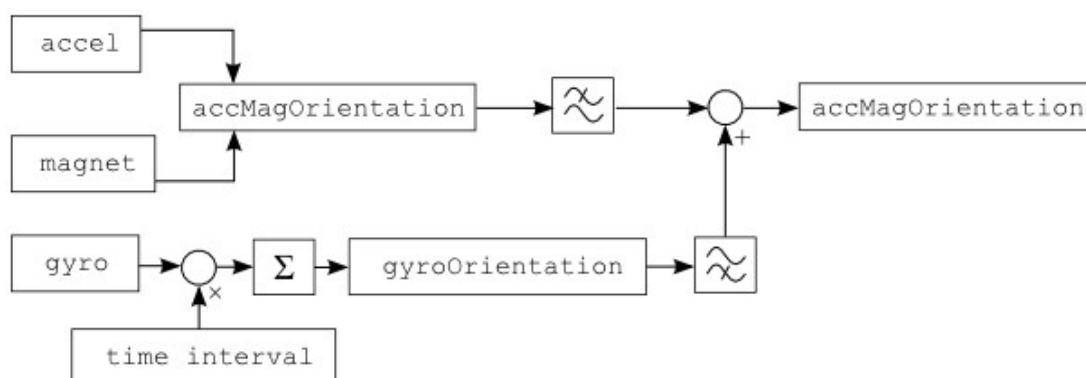


Figura 3: Sensor Fusion esquema (según Paul Lawitzki, en "Android Sensor Fusion Tutorial")

CAPÍTULO 3

GESTIÓN DEL PROYECTO

El propósito de este capítulo consiste en exponer la metodología aplicada durante el desarrollo del proyecto. A continuación, se explicarán los principios usados en términos de organización del proyecto.

3.1 Organización del proyecto

El proyecto será realizado únicamente por una sola persona, Jairo Castrillejo Alonso, por lo que será el encargado de encarnar todos los roles existentes en cada fase del trabajo. No obstante, cuenta con el apoyo del tutor Don Miguel Ángel Laguna Serrano, profesor del Departamento de Informática.

3.2 Roles y responsabilidades

Rol	Responsabilidades	Persona Encargada
Jefe de Proyecto	Planificar, dirigir y controlar el proyecto.	Jairo Castrillejo Alonso
Analista	Estudiar e identificar los requisitos del sistema.	Jairo Castrillejo Alonso
Diseñador	Diseñar la estructura sobre la que se desarrollará la aplicación.	Jairo Castrillejo Alonso
Desarrollador	Implementar la aplicación y corregir errores.	Jairo Castrillejo Alonso
Probador	Realizar las pruebas necesarias sobre la aplicación final.	Jairo Castrillejo Alonso

Tabla 1: Roles y responsabilidades

3.3 Estimaciones de tiempos

La estimación de tiempos se realizará basándose en experiencias anteriores de prácticas y trabajos realizados durante el Grado y en la información obtenida a partir de las memorias de otros proyectos anteriores.

3.4 Plan del proyecto

La planificación del proyecto se va a realizar siguiendo las pautas que ofrece el Proceso Unificado y consta de las siguientes fases:

Fase	Iteraciones	Duración
Inicio	1	2 semanas
Elaboración	2	4 semanas
Construcción	2	8 semanas
Transición	1	2 semanas

Tabla 2: Fases del proyecto

Definiremos cuatro hitos que marcarán el final de cada fase:

Fase	Hito
Inicio	Se ha buscado la documentación necesaria para enfrentarse al problema y se han definido unos requisitos iniciales. Se ha obtenido un plan de fases y se han identificado los riesgos.
Elaboración	Se han definido todos los requisitos del sistema e identificado todos los casos de uso. Se ha realizado un prototipo inicial de la aplicación con funcionalidad limitada.
Construcción	Se han implementado todos los casos de uso. Se han realizado pruebas de caja negra y caja blanca y corregido los fallos. Se ha obtenido la versión final de la aplicación.
Transición	Se ha probado la versión final de la aplicación en diferentes dispositivos para comprobar su correcto funcionamiento. Se ha finalizado la documentación y se ha entregado al usuario final junto con la aplicación.

Tabla 3: Fases e Hitos del Proyecto

3.5 Calendario del proyecto

Nombre de tarea	Duración	Comienzo	Fin	Pr	Nombres de los recursos
Comienzo del Proyecto	0 días	lun 02/01/17	lun 02/01/17		Jairo Castrillejo Alonso
[-] Fase de Inicio	14 días	lun 02/01/17	jue 19/01/17	1	
+ Iteración 1	14 días	lun 02/01/17	jue 19/01/17	1	
Fin Fase de Inicio	0 días	jue 19/01/17	jue 19/01/17	2	Jairo Castrillejo Alonso
[-] Fase de Elaboración	28 días	vie 20/01/17	mar 28/02/17	12	
+ Iteración 1	16 días	vie 20/01/17	vie 10/02/17	12	
+ Iteración 2	12 días	lun 13/02/17	mar 28/02/17	14	
Fin Fase de Elaboración	0 días	mar 28/02/17	mar 28/02/17	13	Jairo Castrillejo Alonso
[-] Fase de Construcción	56 días	mié 01/03/17	mié 17/05/17	24	
+ Iteración 1	32 días	mié 01/03/17	jue 13/04/17	24	
+ Iteración 2	24 días	vie 14/04/17	mié 17/05/17	26	
Fin Fase de Construcción	0 días	mié 17/05/17	mié 17/05/17	25	Jairo Castrillejo Alonso
[-] Fase de Transición	14 días	jue 18/05/17	mar 06/06/17	39	
+ Iteración 1	14 días	jue 18/05/17	mar 06/06/17	39	
Fin Fase Transición	0 días	mar 06/06/17	mar 06/06/17	40	Jairo Castrillejo Alonso
Fin del Proyecto	0 días	mar 06/06/17	mar 06/06/17	45	Jairo Castrillejo Alonso

Figura 4: Calendario del proyecto

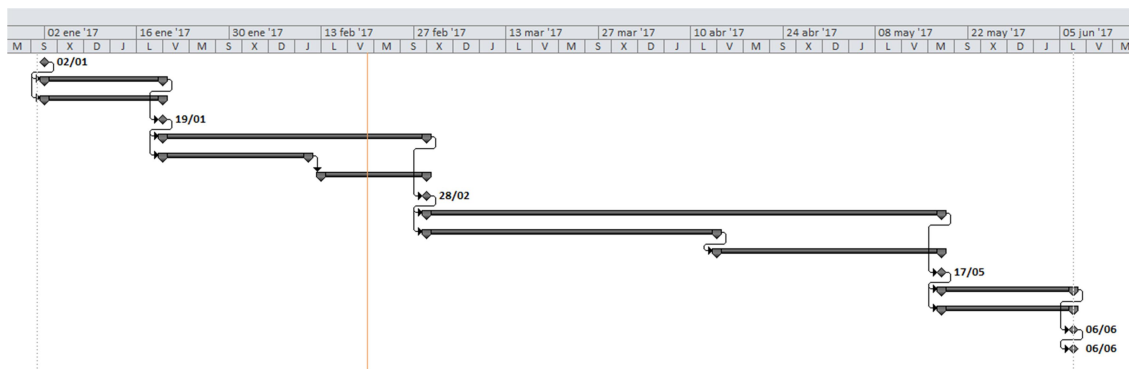


Figura 5: Diagrama de Gantt del proyecto

3.6 Recursos del proyecto

Fase	Recursos Humanos		Recursos Hardware	Recursos software
Inicio	Jairo Alonso	Castrillejo	Ordenador de desarrollo	Microsoft Office 2010, REM 1.2.2
Elaboración	Jairo Alonso	Castrillejo	Ordenador de desarrollo	Microsoft Office 2010, Astah Professional 7.1.0
Construcción	Jairo Alonso	Castrillejo	Ordenador de desarrollo, Xiaomi Redmi Note 2	Microsoft Office 2010, Astah Professional 7.1.0, Android Studio 2.1.2
Transición	Jairo Alonso	Castrillejo	Ordenador de desarrollo, Xiaomi Redmi Note 2, Bq Edison 3, Xiaomi Mi4c	Microsoft Office 2010, REM 1.2.2, Android Studio 2.1.2

Tabla 4: Recursos del Proyecto

	Ordenador de desarrollo
Procesador	Intel i5-4570 3.2GHz
Tarjeta Gráfica	Nvidia Geforce GTX 660 2GB
Memoria RAM	Kingston Hyper-X DDR3 8GB
Disco duro	Seagate Barracuda 1TB
SSD	Samsung 850 evo 120GB
Placa base	MSI Gaming 5
Sistema Operativo	Windows 10 Pro

Tabla 5: Especificaciones del ordenador de desarrollo

	Dispositivo de pruebas
Modelo	Xiaomi Redmi Note 2
CPU	ARM Cortex-A53 2GHz Octa-core
GPU	PowerVR G6200 700MHz
RAM	2GB
HDD	16GB
OS	Android 5.0.2 Lollipop
Pantalla	5,5" 1080x1920 px

Tabla 6: Especificaciones del dispositivo de pruebas

3.7 Análisis de riesgos

Riesgo-001	Aprendizaje de la tecnología empleada
Descripción	Aprender la tecnología con la que se desarrollará la aplicación.
Efecto	Retraso en las fases del proyecto.
Frecuencia	Alta.
Gravedad	Alta.
Detección	Alta.
Acción correctora	Realizar la planificación con un margen suficiente acorde al desconocimiento de la tecnología necesaria.
Plan de contingencia	Aumentar el horario de trabajo para cumplir la planificación

Tabla 7: Riesgo-001

Riesgo-002	Planificación errónea
Descripción	La planificación realizada no es viable para cumplir con los tiempos de entrega.
Efecto	Retraso en el proyecto.
Frecuencia	Alta.
Gravedad	Alta.
Detección	Alta.
Acción correctora	Cada tarea será planificada con amplios márgenes.
Plan de contingencia	Trabajar días que se presuponían libres para cumplir los plazos finales.

Tabla 8: Riesgo-002

Riesgo-003	Pérdida de información
Descripción	Pérdida de algún recurso necesario para la aplicación
Efecto	Retraso en las tareas del proyecto.
Frecuencia	Baja.
Gravedad	Alta.
Detección	Alta.
Acción correctora	Realizar copias de seguridad cada cierto tiempo y en diferentes dispositivos de almacenamiento.
Plan de contingencia	Trabajar días que se presuponían libres para para realizar el trabajo perdido.

Tabla 9: Riesgo-003

Riesgo-004	Enfermedad o asunto familiar
Descripción	No es posible trabajar en el proyecto debido a causas de fuerza mayor.
Efecto	Retraso en el proyecto.
Frecuencia	Media.
Gravedad	Media.
Detección	Alta.
Acción correctora	Se planificarán las tareas con mayor holgura para prevenir retrasos en las fechas del proyecto debido a estos problemas.
Plan de contingencia	Trabajar días que se presuponían libres para cumplir los plazos finales o aumentar las horas de trabajo para recuperar el retraso ocasionado.

Tabla 10: Riesgo-004

Riesgo-005	Pérdida o avería en los dispositivos de trabajo
Descripción	Tanto el dispositivo de desarrollo como el de pruebas se pierden o dejan de funcionar.
Efecto	Retraso en el proyecto.
Frecuencia	Baja.
Gravedad	Baja.
Detección	Alta.
Acción correctora	Disponer de dispositivos alternativos adecuados para continuar con el trabajo sin retrasos.
Plan de contingencia	Obtener otro dispositivo para desarrollar el proyecto. En caso de ser el dispositivo de pruebas, se puede utilizar un emulador virtual.

Tabla 11: Riesgo-005

3.8 Planificación de fases

3.8.1 Planificación temporal de la fase de inicio:

Nombre de tarea	Duración	Comienzo	Fin	Pr	Nombres de los recursos
Comienzo del Proyecto	0 días	lun 02/01/17	lun 02/01/17		Jairo Castrillejo Alonso
<input checked="" type="checkbox"/> Fase de Inicio	14 días	lun 02/01/17	jue 19/01/17	1	Jairo Castrillejo Alonso
<input checked="" type="checkbox"/> Iteración 1	14 días	lun 02/01/17	jue 19/01/17	1	Jairo Castrillejo Alonso
Planteamiento y toma de decisiones	1 día	lun 02/01/17	lun 02/01/17		Jairo Castrillejo Alonso
Creación del Calendario del proyecto	2 días	mar 03/01/17	mié 04/01/17	4	Jairo Castrillejo Alonso
Planificación Fase de Inicio - Iteración 1	1 día	jue 05/01/17	jue 05/01/17	5	Jairo Castrillejo Alonso
Creación del Plan del proyecto	2 días	vie 06/01/17	lun 09/01/17	6	Jairo Castrillejo Alonso
Gestión de Riesgos	1 día	mar 10/01/17	mar 10/01/17	7	Jairo Castrillejo Alonso
Identificación de requisitos	2 días	mié 11/01/17	jue 12/01/17	8	Jairo Castrillejo Alonso
Identificación de casos de uso	3 días	vie 13/01/17	mar 17/01/17	9	Jairo Castrillejo Alonso
Planificación Fase de Elaboración - Iteración 1	2 días	mié 18/01/17	jue 19/01/17	10	Jairo Castrillejo Alonso

Figura 6: Calendario fase de Inicio

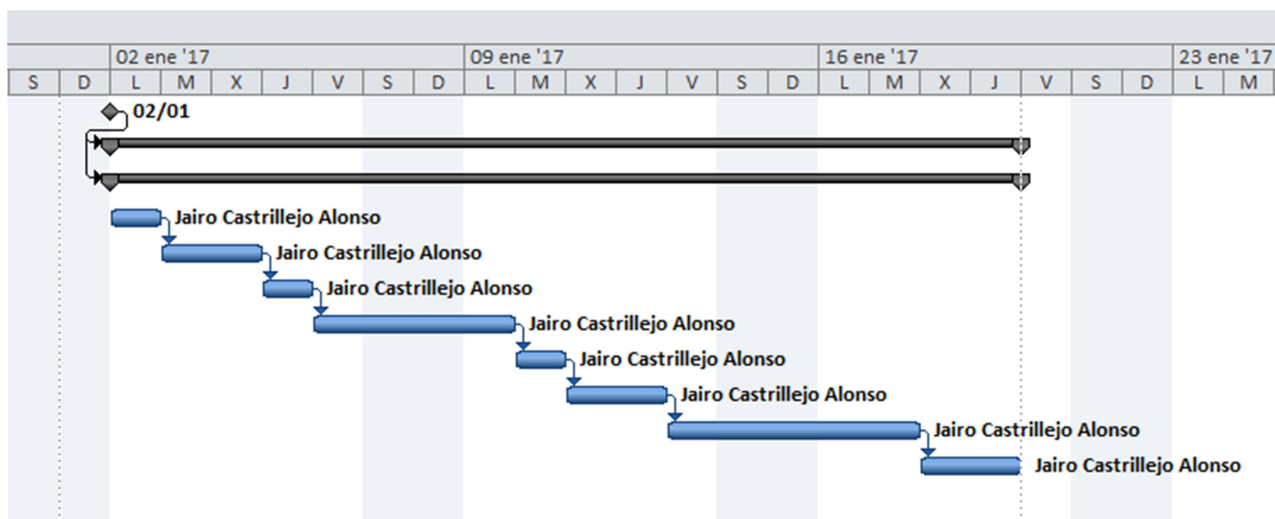


Figura 7: Diagrama de Gantt para la fase de Inicio

3.8.2 Planificación temporal de la fase de Elaboración – Iteración 1:

Nombre de tarea	Duración	Comienzo	Fin	Pr	Nombres de los recursos
<input checked="" type="checkbox"/> Fase de Elaboración	28 días	vie 20/01/17	mar 28/02/17	12	
<input checked="" type="checkbox"/> Iteración 1	16 días	vie 20/01/17	vie 10/02/17	12	
Creación de documento de análisis de requisitos	4 días	vie 20/01/17	mié 25/01/17		Jairo Castrillejo Alonso
Creación de los casos de uso	6 días	jue 26/01/17	jue 02/02/17	15	Jairo Castrillejo Alonso
Descripción del Hardware y del Software	3 días	vie 03/02/17	mar 07/02/17	16	Jairo Castrillejo Alonso
Planificación Fase de Elaboración - Iteración 2	3 días	mié 08/02/17	vie 10/02/17	17	Jairo Castrillejo Alonso

Figura 8: Calendario fase de Elaboración – Iteración 1

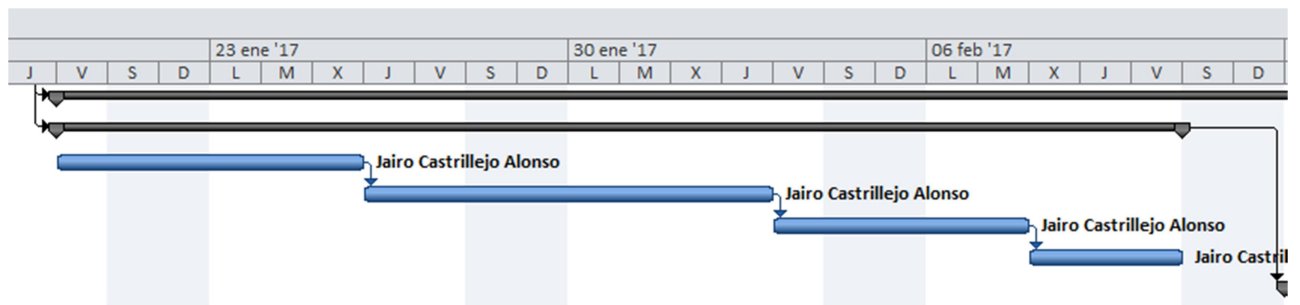


Figura 9: Diagrama de Gantt fase de Elaboración – Iteración 1

3.8.3 Planificación temporal de la fase de Elaboración – Iteración 2:

Nombre de tarea	Duración	Comienzo	Fin	Pr	Nombres de los recursos
[-] Fase de Elaboración	28 días	vie 20/01/17	mar 28/02/17	12	
[+] Iteración 1	16 días	vie 20/01/17	vie 10/02/17	12	
[-] Iteración 2	12 días	lun 13/02/17	mar 28/02/17	14	
Revisión del documento de análisis de requisitos	3 días	lun 13/02/17	mié 15/02/17		Jairo Castrillejo Alonso
Revisión de los casos de uso	3 días	jue 16/02/17	lun 20/02/17	20	Jairo Castrillejo Alonso
Revisión de la descripción del Hardware y del Software	2 días	mar 21/02/17	mié 22/02/17	21	Jairo Castrillejo Alonso
Planificación Fase de Construcción - Iteración 1	4 días	jue 23/02/17	mar 28/02/17	22	Jairo Castrillejo Alonso
Fin Fase de Elaboración	0 días	mar 28/02/17	mar 28/02/17	13	Jairo Castrillejo Alonso

Figura 10: Calendario fase de Elaboración – Iteración 2

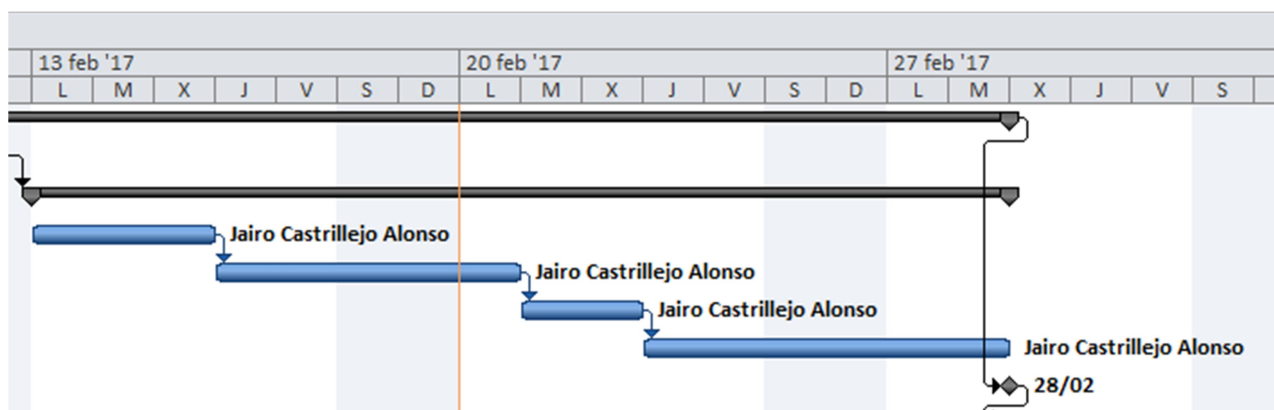


Figura 11: Diagrama de Gantt fase de Elaboración – Iteración 2

3.8.4 Planificación temporal de la fase de Construcción – Iteración 1:

Fase de Construcción	56 días	mié 01/03/17	mié 17/05/17	24	
Iteración 1	32 días	mié 01/03/17	jue 13/04/17	24	
Creación de los diagramas de clases	4 días	mié 01/03/17	lun 06/03/17		Jairo Castrillejo Alonso
Creación de los diagramas de secuencia	4 días	mar 07/03/17	vie 10/03/17	27	Jairo Castrillejo Alonso
Creación del diagrama de arquitectura	2 días	lun 13/03/17	mar 14/03/17	28	Jairo Castrillejo Alonso
Implementación de la aplicación	20 días	mié 15/03/17	mar 11/04/17	29	Jairo Castrillejo Alonso
Planificación Fase de Construcción - Iteración 2	2 días	mié 12/04/17	jue 13/04/17	30	Jairo Castrillejo Alonso

Figura 12: Calendario fase de Construcción – Iteración 1

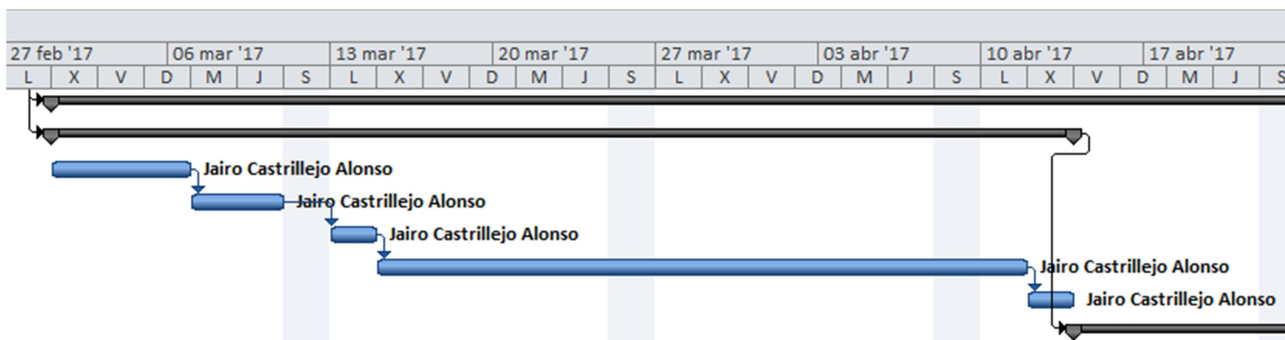


Figura 13: Diagrama de Gantt fase de Construcción – Iteración 1

3.8.5 Planificación temporal de la fase de Construcción – Iteración 2:

Nombre de tarea	Duración	Comienzo	Fin	Pr	Nombres de los recursos
Fase de Construcción	56 días	mié 01/03/17	mié 17/05/17	24	
Iteración 1	32 días	mié 01/03/17	jue 13/04/17	24	
Iteración 2	24 días	vie 14/04/17	mié 17/05/17	26	
Implementación de la aplicación	14 días	vie 14/04/17	mié 03/05/17		Jairo Castrillejo Alonso
Creación del manual de usuario	3 días	jue 04/05/17	lun 08/05/17	33	Jairo Castrillejo Alonso
Revisión de los diagramas de clases	2 días	mar 09/05/17	mié 10/05/17	34	Jairo Castrillejo Alonso
Revisión de los diagramas de secuencia	2 días	jue 11/05/17	vie 12/05/17	35	Jairo Castrillejo Alonso
Revisión del diagrama de arquitectura	1 día	lun 15/05/17	lun 15/05/17	36	Jairo Castrillejo Alonso
Planificación Fase de Transición - Iteración 1	2 días	mar 16/05/17	mié 17/05/17	37	Jairo Castrillejo Alonso
Fin Fase de Construcción	0 días	mié 17/05/17	mié 17/05/17	25	Jairo Castrillejo Alonso

Figura 14: Calendario fase de Construcción – Iteración 2

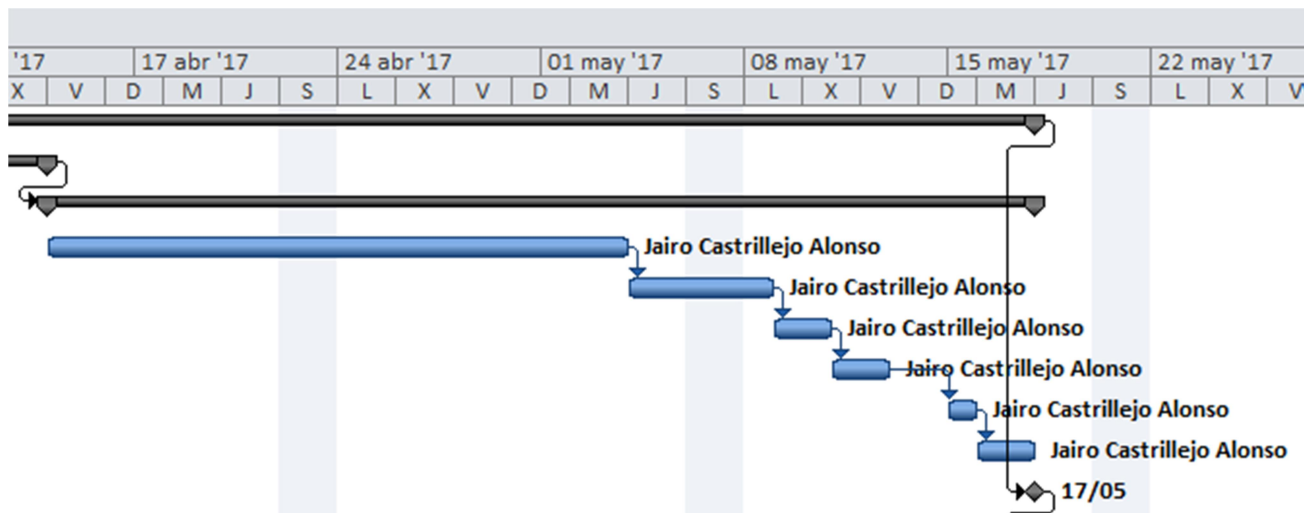


Figura 15: Diagrama de Gantt fase de Construcción – Iteración 2

3.8.6 Planificación temporal de la fase de Transición – Iteración 1:

Nombre de tarea	Duración	Comienzo	Fin	Pr	Nombres de los recursos
Fase de Transición	14 días	jue 18/05/17	mar 06/06/17	39	
Iteración 1	14 días	jue 18/05/17	mar 06/06/17	39	
Depuración y pruebas	6 días	jue 18/05/17	jue 25/05/17		Jairo Castrillejo Alonso
Corrección de errores	4 días	vie 26/05/17	mié 31/05/17	42	Jairo Castrillejo Alonso
Revisión de la documentación	4 días	jue 01/06/17	mar 06/06/17	43	Jairo Castrillejo Alonso
Fin Fase Transición	0 días	mar 06/06/17	mar 06/06/17	40	Jairo Castrillejo Alonso
Fin del Proyecto	0 días	mar 06/06/17	mar 06/06/17	45	Jairo Castrillejo Alonso

Figura 16: Calendario fase de Transición

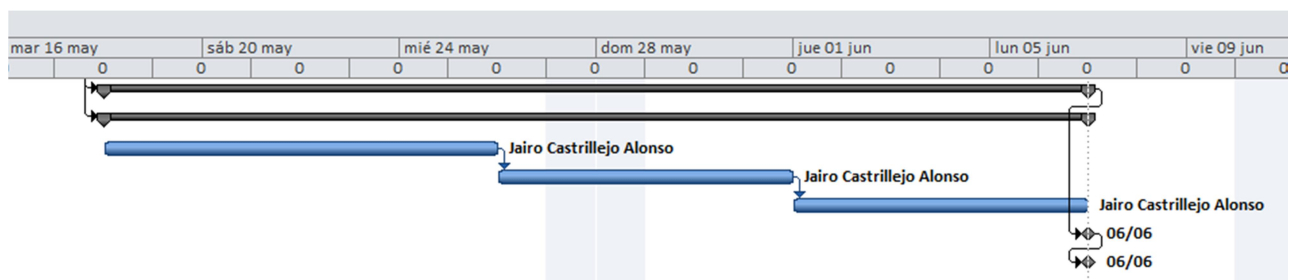


Figura 17: Diagrama de Gantt fase de Transición

3.9 Estimación de costes

A continuación, se especificarán los costes del proyecto incluyendo costes hardware, software y de personal.

3.9.1 Costes hardware

Para el desarrollo de esta aplicación es necesario un PC y un dispositivo móvil de prueba que cuente con giroscopio, acelerómetro y magnetómetro.

El ordenador utilizado para el proyecto es el indicado en el punto 2.6 *Recursos del proyecto* y tuvo un coste de unos 700€ aprox. La vida media de un ordenador de sobremesa es muy difícil de calcular ya que depende del uso que se haga del mismo y de la vida media de cada uno de sus componentes. Pero esta se calcula entre 3 y 8 años. Se tomará, para el cálculo del coste, una vida media de 5,5 años. En el caso del Smartphone utilizado (valorado en 150€) sucede algo parecido. La vida media depende del uso y esta se calcula en una media de 4.7 años. Ambos dispositivos han sido usados durante un total de 4 meses a lo largo del proyecto.

Concepto	Coste
PC desarrollo	$(700\text{€}/(5,5 \times 12) \text{ meses}) \times 4 \text{ meses} = 42,42\text{€}$
Smartphone	$(150\text{€}/(4,7 \times 12) \text{ meses}) \times 4 \text{ meses} = 10,64\text{€}$
TOTAL	53,06€

Tabla 11: Resumen costes hardware

3.9.2 Costes software

Todas las librerías, frameworks y entornos de desarrollo utilizados para el proyecto son gratuitos. En el caso del sistema operativo, se ha utilizado uno proporcionado por el programa Microsoft Dreamspark por ser alumno de la Universidad de Valladolid.

3.9.3 Costes recursos humanos

Suponiendo el sueldo medio de un analista-programador en 16€/hora aproximadamente

Concepto	Coste
Analista-Programador	$16\text{€/hora} \times 4 \text{ horas/día} \times 112 \text{ días} = 7.168\text{€}$
TOTAL	7.168€

Tabla 12: Resumen costes de recursos humanos

3.9.4 Coste total

Finalmente obtenemos el coste total como la suma de todo lo anterior.

Concepto	Coste
Costes hardware	53,06€
Costes software	0€
Costes humanos	7.168€
Coste total	7.221,06€

Tabla 13: Coste total

CAPÍTULO 4

ANÁLISIS DE LA APLICACIÓN

En este capítulo se presentan los requisitos software específicos obtenidos de la descripción del problema obtenida en las reuniones con el cliente.

4.1 Participantes en el proyecto

Organización-001	Departamento de informática
Dirección	E.T.S. Ingeniería Informática, Campus Miguel Delibes, Paseo de Belén, número 15, 47011 VALLADOLID.
Teléfono	No consta.
Fax	No consta.
Comentarios	Organización que desarrolla el sistema.

Tabla 14: Organización-001

Participante-001	Jairo Castrillejo Alonso
Organización	Departamento de informática.
Rol	Jefe de proyecto, analista, diseñador, desarrollador, probador.
Es desarrollador	Sí.
Es cliente	No.
Es usuario	No.
Comentarios	Estudiante en la E.I.I. del Grado de Ingeniería Informática con Mención en Ingeniería de Software.

Tabla 15: Participante-001

Participante-002	Miguel Ángel Laguna Serrano
Organización	Departamento de informática.
Rol	Tutor del proyecto.
Es desarrollador	No.
Es cliente	Sí.
Es usuario	Sí.
Comentarios	Profesor de la Universidad de Valladolid en la E.I.I.

Tabla 16: Participante-002

4.2 Objetivos del sistema

Objetivo-001	Realizar ejercicios de rehabilitación de muñeca
Autores	Jairo Castrillejo Alonso
Fuentes	Miguel Ángel Laguna Serrano
Descripción	El sistema deberá permitir al usuario realizar tres tipos diferentes de ejercicios de rehabilitación de muñeca.
Importancia	Vital.
Urgencia	Inmediatamente.
Estabilidad	Baja

Tabla 17: Objetivo-001

Objetivo-002	Gestionar planes de rehabilitación
Autores	Jairo Castrillejo Alonso
Fuentes	Miguel Ángel Laguna Serrano
Descripción	El sistema deberá permitir al usuario crear un programa de rehabilitación para ir almacenando los resultados de los ejercicios que vaya realizando y consultar dicho programa para comprobar la evolución de la lesión. Además debería permitir al usuario dar por finalizado el programa así como eliminarlo por completo. El Usuario podrá también consultar todos los datos almacenados de los programas.
Importancia	Vital.
Urgencia	Inmediatamente.
Estabilidad	Baja

Tabla 18: Objetivo-002

Objetivo-003	Consultar planes de rehabilitación
Autores	Jairo Castrillejo Alonso
Fuentes	Miguel Ángel Laguna Serrano
Descripción	El sistema deberá permitir al usuario revisar y obtener datos de las sesiones realizadas a lo largo de la duración del programa de rehabilitación. Incluso cuando el programa ya haya finalizado.
Importancia	Vital.
Urgencia	Inmediatamente.
Estabilidad	Baja

Tabla 19: Objetivo-003

4.3 Requisitos del sistema

4.3.1 Requisitos funcionales

FRQ-001	Crear nuevo programa de rehabilitación
Autores	Jairo Castrillejo Alonso
Fuentes	Miguel Ángel Laguna Serrano
Descripción	El sistema deberá permitir al usuario crear un nuevo programa de rehabilitación donde se irán guardando los datos de las sesiones de ejercicios.
Importancia	Vital.
Urgencia	Inmediatamente.
Estabilidad	Baja

Tabla 20: Requisito Funcional-001

FRQ-002	Realizar ejercicios de rehabilitación
Autores	Jairo Castrillejo Alonso
Fuentes	Miguel Ángel Laguna Serrano
Descripción	El sistema deberá permitir al usuario realizar los diferentes ejercicios para proveer de datos al programa de rehabilitación mediante la medición de los ángulos máximos realizados en los diferentes movimientos de la muñeca.
Importancia	Vital.
Urgencia	Inmediatamente.
Estabilidad	Baja

Tabla 21: Requisito Funcional-002

FRQ-003	Consultar programa de rehabilitación
Autores	Jairo Castrillejo Alonso
Fuentes	Miguel Ángel Laguna Serrano
Descripción	El sistema deberá permitir al usuario seleccionar un plan de rehabilitación para observar los datos recogidos durante el periodo del mismo y ver la evolución de éstos.
Importancia	Vital.
Urgencia	Inmediatamente.
Estabilidad	Baja

Tabla 22: Requisito Funcional-003

FRQ-004	Proporcionar tipos de ejercicios
Autores	Jairo Castrillejo Alonso
Fuentes	Miguel Ángel Laguna Serrano
Descripción	El sistema deberá proporcionar diversos tipos de ejercicios especificados en el glosario.
Importancia	Vital.
Urgencia	Inmediatamente.
Estabilidad	Baja

Tabla 23: Requisito Funcional-004

FRQ-005	Medir ángulos
Autores	Jairo Castrillejo Alonso
Fuentes	Miguel Ángel Laguna Serrano
Descripción	El sistema deberá medir el ángulo que forma la muñeca con la mano.
Importancia	Vital.
Urgencia	Inmediatamente.
Estabilidad	Baja

Tabla 24: Requisito Funcional-005

FRQ-006	Visualizar resultados en forma de gráfica
Autores	Jairo Castrillejo Alonso
Fuentes	Miguel Ángel Laguna Serrano
Descripción	El sistema deberá permitir al usuario visualizar los resultados de los distintos ejercicios realizados por un paciente.
Importancia	Vital.
Urgencia	Inmediatamente.
Estabilidad	Baja

Tabla 25: Requisito Funcional-006

FRQ-007	Eliminar programa de rehabilitación
Autores	Jairo Castrillejo Alonso
Fuentes	Miguel Ángel Laguna Serrano
Descripción	El sistema deberá permitir al usuario eliminar un programa de rehabilitación cuando lo desee.
Importancia	Vital.
Urgencia	Inmediatamente.
Estabilidad	Baja

Tabla 26: Requisito Funcional-007

FRQ-008	Finalizar programa de rehabilitación
Autores	Jairo Castrillejo Alonso
Fuentes	Miguel Ángel Laguna Serrano
Descripción	El sistema deberá permitir al usuario finalizar un programa de rehabilitación. De esta manera ya no podrán realizarse más sesiones de ejercicios, pero si consultar todos los datos referentes al programa.
Importancia	Vital.
Urgencia	Inmediatamente.
Estabilidad	Baja

Tabla 27: Requisito Funcional-008

FRQ-009	Implementar señales sonoras
Autores	Jairo Castrillejo Alonso
Fuentes	Miguel Ángel Laguna Serrano
Descripción	El sistema deberá informar al usuario mediante pitidos para informarle de cuando ha de comenzar a realizar los movimientos de los ejercicios.
Importancia	Vital.
Urgencia	Inmediatamente.
Estabilidad	Baja

Tabla 28: Requisito Funcional-009

4.3.2 Requisitos no funcionales

NFR-001	Compatibilidad con diversos dispositivos
Autores	Jairo Castrillejo Alonso
Fuentes	Miguel Ángel Laguna Serrano
Descripción	El sistema deberá ser compatible con dispositivos de diversas características hardware y software siempre que cumplan con las características mínimas exigidas por la aplicación. Concretamente desde la versión 4.0.3 Ice Cream Sándwich en adelante.
Importancia	Vital.
Urgencia	Inmediatamente.
Estabilidad	Baja

Tabla 29: Requisito No Funcional-001

NFR-002	Facilidad de uso
Autores	Jairo Castrillejo Alonso
Fuentes	Miguel Ángel Laguna Serrano
Descripción	El sistema deberá proporcionar una interfaz sencilla e intuitiva que facilite el uso.
Importancia	Vital.
Urgencia	Inmediatamente.
Estabilidad	Baja

Tabla 30: Requisito No Funcional-002

NFR-003	Aprendizaje de la interfaz
Autores	Jairo Castrillejo Alonso
Fuentes	Miguel Ángel Laguna Serrano
Descripción	El sistema deberá contar con una interfaz con un tiempo de aprendizaje por parte del usuario de menos de un día.
Importancia	Vital.
Urgencia	Inmediatamente.
Estabilidad	Baja

Tabla 31: Requisito No Funcional-003

NFR-004	Fiabilidad de la medida
Autores	Jairo Castrillejo Alonso
Fuentes	Miguel Ángel Laguna Serrano
Descripción	El sistema deberá recoger medidas precisas de los ángulos para cada ejercicio y en ningún caso deberán superar el 10% de error.
Importancia	Vital.
Urgencia	Inmediatamente.
Estabilidad	Baja

Tabla 32: Requisito No Funcional-004

4.4 Casos de uso

4.4.1 Definición de actores

ACT-001	Usuario
Autores	Jairo Castrillejo Alonso
Descripción	Este actor representa al usuario final de la aplicación.
Comentarios	La aplicación está destinada a pacientes que se encuentren en proceso de rehabilitación de la muñeca y quieran monitorizar el progreso de la misma mediante los ángulos de giro de la articulación a través de varios ejercicios propuestos.

Tabla 33: Actor-001

4.4.2 Diagramas de casos de uso

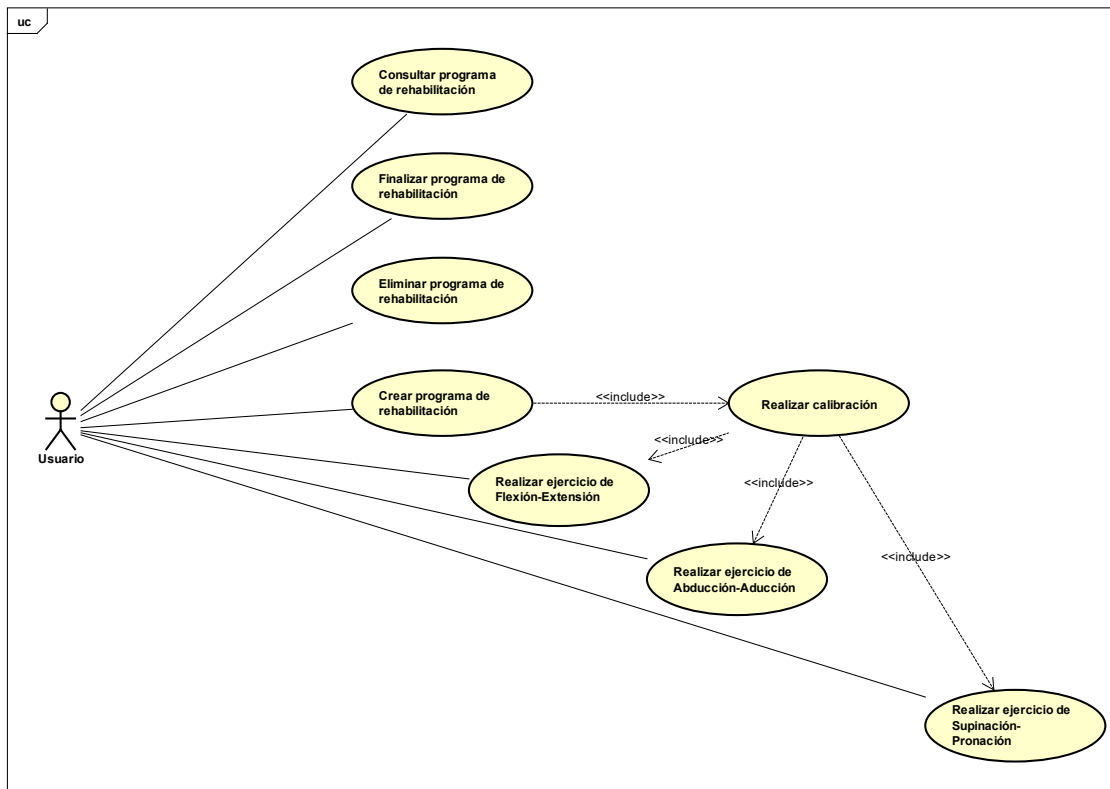


Figura 18: Diagrama de Casos de Uso

4.4.3 Casos de uso del sistema

UC-001	Crear programa de rehabilitación	
Autores	Jairo Castrillejo Alonso	
Fuentes	Miguel Ángel Laguna Serrano	
Dependencias	Ninguno.	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el Usuario desee crear un nuevo programa de rehabilitación.	
Precondición	No existe un programa en el sistema con el mismo nombre.	
Secuencia normal	Paso	Acción
	1	El actor Usuario (ACT-0001) selecciona Crear programa en el menú principal.
	2	El sistema muestra el formulario de creación del programa.
	3	El actor Usuario (ACT-0001) introduce un nombre identificativo.
	4	El sistema registra el nombre del programa.
	5	Se realiza el caso de uso Realizar calibración (UC-0008).
Postcondición	Se ha creado un nuevo programa de rehabilitación.	
Excepciones	Paso	Acción
	4	Si el nombre del programa no es único, el sistema muestra un mensaje de error, a continuación este caso de uso continúa (en el paso 2).
	4	Si el campo de Nombre está vacío, el sistema informa al Usuario, a continuación este caso de uso continúa (en el paso 2).
Importancia	Vital.	
Urgencia	Inmediata.	
Estabilidad	Baja.	

Tabla 34: Caso de Uso-001

UC-002	Realizar ejercicio de Flexión-Extensión	
Autores	Jairo Castrillejo Alonso	
Fuentes	Miguel Ángel Laguna Serrano	
Dependencias	Ninguno.	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el Usuario quiere realizar el ejercicio de Flexión-Extensión. o durante la realización de los siguientes casos de uso: [UC-0008] Realizar calibración	
Precondición	Existe al menos un programa de rehabilitación.	
Secuencia normal	Paso	Acción
	1	Si el Usuario está realizando una sesión rutinaria, el actor Usuario (ACT-0001) selecciona realizar un ejercicio de Flexión-Extensión
	2	El sistema ofrece la realización del ejercicio de Flexión
	3	El actor Usuario (ACT-0001) comienza el contador para colocar el dispositivo de forma correcta
	4	El sistema recoge el ángulo inicial y el final después de que transcurra el tiempo marcado y muestra el resultado por pantalla.
	5	El actor Usuario (ACT-0001) elige continuar con la siguiente parte del ejercicio
	6	El sistema ofrece la realización del ejercicio de Extensión.
	7	El actor Usuario (ACT-0001) comienza el contador para colocar el dispositivo de forma correcta.
	8	El sistema recoge el ángulo inicial y el final después de que transcurra el tiempo marcado y muestra el resultado por pantalla.
	9	El actor Usuario (ACT-0001) selecciona la opción de Guardar
	10	El sistema guarda los ángulos registrados
Postcondición	Quedan almacenados los datos del ejercicio en el programa de rehabilitación.	
Excepciones	Paso	Acción
	4	Si el Usuario decide parar el ejercicio antes de terminar la medición, el sistema reinicia la actividad, a continuación este caso de uso continúa (en el paso 2)
	5	Si el Usuario decide reiniciar el ejercicio después de finalizar la medición, el sistema reinicia la actividad, a continuación este caso de uso continúa (en el paso 2)
	8	Si el Usuario decide parar el ejercicio antes de terminar la medición, el sistema reinicia la actividad, a continuación este caso de uso continúa (en el paso 6)
	9	Si el Usuario decide reiniciar el ejercicio después de finalizar la medición, el sistema reinicia la actividad, a continuación este caso de uso continúa (en el paso 6)
	10	Si el Usuario ha accedido al ejercicio desde la calibración inicial, el sistema guarda los ángulos registrados, a continuación este caso de uso queda sin efecto
Importancia	Vital.	
Urgencia	Inmediata.	
Estabilidad	Baja	

Tabla 35: Caso de Uso-002

UC-003	Realizar ejercicio de Abducción-Aducción	
Autores	Jairo Castrillejo Alonso	
Fuentes	Miguel Ángel Laguna Serrano	
Dependencias	Ninguno.	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el Usuario quiere realizar el ejercicio de Abducción-Aducción. o durante la realización de los siguientes casos de uso: [UC-0008] Realizar calibración	
Precondición	Existe al menos un programa de rehabilitación.	
Secuencia normal	Paso	Acción
	1	Si el Usuario está realizando una sesión rutinaria, el actor Usuario (ACT-0001) selecciona realizar un ejercicio de Abducción-Aducción
	2	El sistema ofrece la realización del ejercicio de Abducción
	3	El actor Usuario (ACT-0001) comienza el contador para colocar el dispositivo de forma correcta
	4	El sistema recoge el ángulo inicial y el final después de que transcurra el tiempo marcado y muestra el resultado por pantalla.
	5	El actor Usuario (ACT-0001) elige continuar con la siguiente parte del ejercicio
	6	El sistema ofrece la realización del ejercicio de Aducción.
	7	El actor Usuario (ACT-0001) comienza el contador para colocar el dispositivo de forma correcta.
	8	El sistema recoge el ángulo inicial y el final después de que transcurra el tiempo marcado y muestra el resultado por pantalla.
	9	El actor Usuario (ACT-0001) selecciona la opción de Guardar
	10	El sistema guarda los ángulos registrados
Postcondición	Quedan almacenados los datos del ejercicio en el programa de rehabilitación.	
Excepciones	Paso	Acción
	4	Si el Usuario decide parar el ejercicio antes de terminar la medición, el sistema reinicia la actividad, a continuación este caso de uso continúa (en el paso 2)
	5	Si el Usuario decide repetir el ejercicio después de finalizar la medición, el sistema reinicia la actividad, a continuación este caso de uso continúa (en el paso 2)
	8	Si el Usuario decide parar el ejercicio antes de terminar la medición, el sistema reinicia la actividad, a continuación este caso de uso continúa (en el paso 6)
	9	Si el Usuario elige repetir el ejercicio después de finalizar la medición, el sistema reinicia la actividad, a continuación este caso de uso continúa (en el paso 6)
	10	Si el Usuario ha accedido al ejercicio desde la calibración inicial, el sistema guarda los ángulos registrados, a continuación este caso de uso queda sin efecto
Importancia	Vital.	
Urgencia	Inmediata.	
Estabilidad	Baja	

Tabla 36: Caso de Uso-003

UC-004	Realizar ejercicio de Supinación-Pronación	
Autores	Jairo Castrillejo Alonso	
Fuentes	Miguel Ángel Laguna Serrano	
Dependencias	Ninguno.	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el Usuario quiere realizar el ejercicio de Supinación-Pronación o durante la realización de los siguientes casos de uso: [UC-0008] Realizar calibración	
Precondición	Existe al menos un programa de rehabilitación.	
Secuencia normal	Paso	Acción
	1	Si el Usuario está realizando una sesión rutinaria, el actor Usuario (ACT-0001) selecciona realizar un ejercicio de Supinación-Pronación.
	2	El sistema ofrece la realización del ejercicio de Supinación.
	3	El actor Usuario (ACT-0001) comienza el contador para colocar el dispositivo de forma correcta.
	4	El sistema recoge el ángulo inicial y el final después de que transcurra el tiempo marcado y muestra el resultado por pantalla.
	5	El actor Usuario (ACT-0001) elige continuar con la siguiente parte del ejercicio.
	6	El sistema ofrece la realización del ejercicio de Pronación.
	7	El actor Usuario (ACT-0001) comienza el contador para colocar el dispositivo de forma correcta.
	8	El sistema recoge el ángulo inicial y el final después de que transcurra el tiempo marcado y muestra el resultado por pantalla.
	9	El actor Usuario (ACT-0001) selecciona la opción de Guardar.
	10	El sistema guarda los ángulos registrados.
Postcondición	Quedan almacenados los datos del ejercicio en el programa de rehabilitación.	
Excepciones	Paso	Acción
	4	Si el Usuario decide parar el ejercicio antes de terminar la medición, el sistema reinicia la actividad, a continuación este caso de uso continúa (en el paso 2)
	5	Si el Usuario decide repetir el ejercicio después de finalizar la medición, el sistema reinicia la actividad, a continuación este caso de uso continúa (en el paso 2)
	8	Si el Usuario decide parar el ejercicio antes de terminar la medición, el sistema reinicia la actividad, a continuación este caso de uso continúa (en el paso 6)
	10	Si el Usuario ha accedido al ejercicio desde la calibración inicial, el sistema guarda los ángulos registrados, a continuación este caso de uso queda sin efecto
Importancia	Vital.	
Urgencia	Inmediata.	
Estabilidad	Baja	

Tabla 37: Caso de Uso-004

UC-005	Consultar programa de rehabilitación	
Autores	Jairo Castrillejo Alonso	
Fuentes	Miguel Ángel Laguna Serrano	
Dependencias	Ninguno.	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el Usuario quiera consultar un programa de rehabilitación.	
Precondición	Existe al menos un programa de rehabilitación.	
Secuencia normal	Paso	Acción
	1	El actor Usuario (ACT-0001) selecciona un programa de rehabilitación de la lista
	2	El sistema muestra un resumen de los datos del programa de rehabilitación
	3	Si el Usuario selecciona la opción Flexión-Extensión, el sistema muestra las gráficas correspondientes al ejercicio de Flexión-Extensión
	4	Si el Usuario selecciona la opción Abducción-Aducción, el sistema muestra las gráficas correspondientes al ejercicio de Abducción-Aducción
	5	Si el Usuario selecciona la opción Supinación-Pronación, el sistema muestra las gráficas correspondientes al ejercicio de Supinación-Pronación.
Postcondición	Ninguna.	
Excepciones	Paso	Acción
	3	Si no existen datos relativos a este ejercicio, el sistema muestra una imagen informándole de cómo realizar un nuevo ejercicio de Flexión-Extensión, a continuación este caso de uso queda sin efecto
	4	Si no existen datos relativos a este ejercicio, el sistema muestra una imagen informándole de cómo realizar un nuevo ejercicio de Abducción-Aducción, a continuación este caso de uso queda sin efecto
	5	Si no existen datos relativos a este ejercicio, el sistema muestra una imagen informándole de cómo realizar un nuevo ejercicio de Supinación-Pronación, a continuación este caso de uso queda sin efecto
Importancia	Vital.	
Urgencia	Inmediata.	
Estabilidad	Baja	

Tabla 38: Caso de Uso-005

UC-006	Finalizar programa de rehabilitación	
Autores	Jairo Castrillejo Alonso	
Fuentes	Miguel Ángel Laguna Serrano	
Dependencias	Ninguno.	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el Usuario quiere dar por finalizado un programa de rehabilitación	
Precondición	Existe al menos un programa de rehabilitación	
Secuencia normal	Paso	Acción
	1	El actor Usuario (ACT-0001) selecciona un programa de rehabilitación de la lista
	2	El sistema muestra los datos de resumen del programa seleccionado.
	3	El actor Usuario (ACT-0001) selecciona en el menú la opción de finalizar programa.
	4	El sistema muestra al Usuario un diálogo avisándole de que la acción será irreversible y que ya no se podrán realizar nuevas sesiones de ejercicios.
	5	El actor Usuario (ACT-0001) elige la opción de Confirmar en el diálogo
	6	El sistema cambia el estado del programa.
Postcondición	El programa queda en estado finalizado.	
Excepciones	Paso	Acción
	5	Si elige la opción Cancelar, el sistema cierra el diálogo, a continuación este caso de uso queda sin efecto
Importancia	Vital.	
Urgencia	Inmediata.	
Estabilidad	Baja	

Tabla 39: Caso de Uso-006

UC-007	Eliminar programa de rehabilitación	
Autores	Jairo Castrillejo Alonso	
Fuentes	Miguel Ángel Laguna Serrano	
Dependencias	Ninguno.	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el Usuario desea eliminar por completo un programa de rehabilitación del sistema	
Precondición	Hay al menos un programa de rehabilitación ya creado.	
Secuencia normal	Paso	Acción
	1	El actor Usuario (ACT-0001) selecciona un programa de rehabilitación de la lista
	2	El sistema muestra el resumen con los detalles del programa.
	3	El actor Usuario (ACT-0001) selecciona la opción de Eliminar programa
	4	El sistema muestra un diálogo al usuario para preguntarle si está seguro de que desea eliminar el programa y que esa acción será irreversible.
	5	El actor Usuario (ACT-0001) selecciona la opción Eliminar en el diálogo
	6	El sistema elimina el programa de rehabilitación de la base del sistema
Postcondición	El programa queda eliminado de la aplicación	
Excepciones	Paso	Acción
	5	Si selecciona la opción Cancelar, el sistema cierra el diálogo, a continuación este caso de uso queda sin efecto
Importancia	Vital.	
Urgencia	Inmediata.	
Estabilidad	Baja	

Tabla 40: Caso de Uso-007

UC-008	Realizar calibración	
Autores	Jairo Castrillejo Alonso	
Fuentes	Miguel Ángel Laguna Serrano	
Dependencias	Ninguno.	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el Usuario quiere realizar la calibración inicial o durante la realización de los siguientes casos de uso: [UC-0001] Crear programa de rehabilitación	
Precondición	Se ha iniciado la creación del Programa	
Secuencia normal	Paso	Acción
	1	Se realiza el caso de uso Realizar ejercicio de Flexión-Extensión (UC-0003)
	2	Se realiza el caso de uso Realizar ejercicio de Abducción-Aducción (UC-0010)
	3	Se realiza el caso de uso Realizar ejercicio de Supinación-Pronación (UC-0011)
	4	El actor Usuario (ACT-0001) selecciona la opción de finalizar calibración.
	5	El sistema guarda los datos y crea un nuevo programa de rehabilitación
Postcondición	Se ha creado un nuevo programa de rehabilitación	
Excepciones	Paso	Acción
	-	-
Importancia	Vital.	
Urgencia	Inmediata.	
Estabilidad	Baja	

Tabla 41: Caso de Uso-008

4.5 Modelo de dominio

En la siguiente figura se muestra el modelo de dominio de la aplicación.

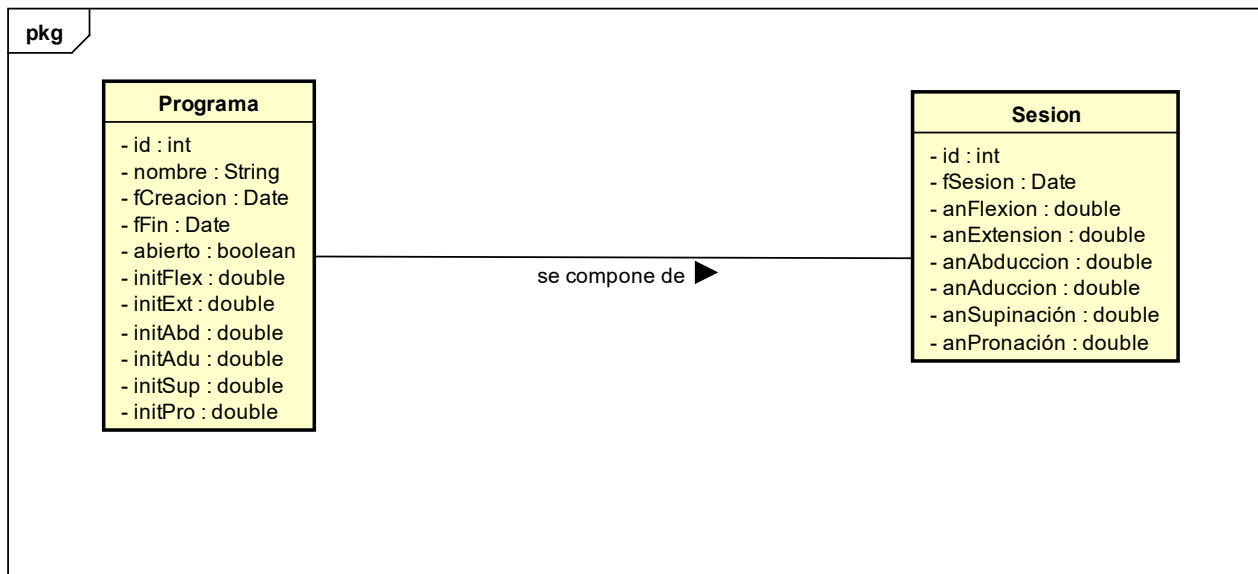


Figura 19: Modelo de Dominio

4.6 Diagramas de secuencia

4.6.1 Crear programa de rehabilitación

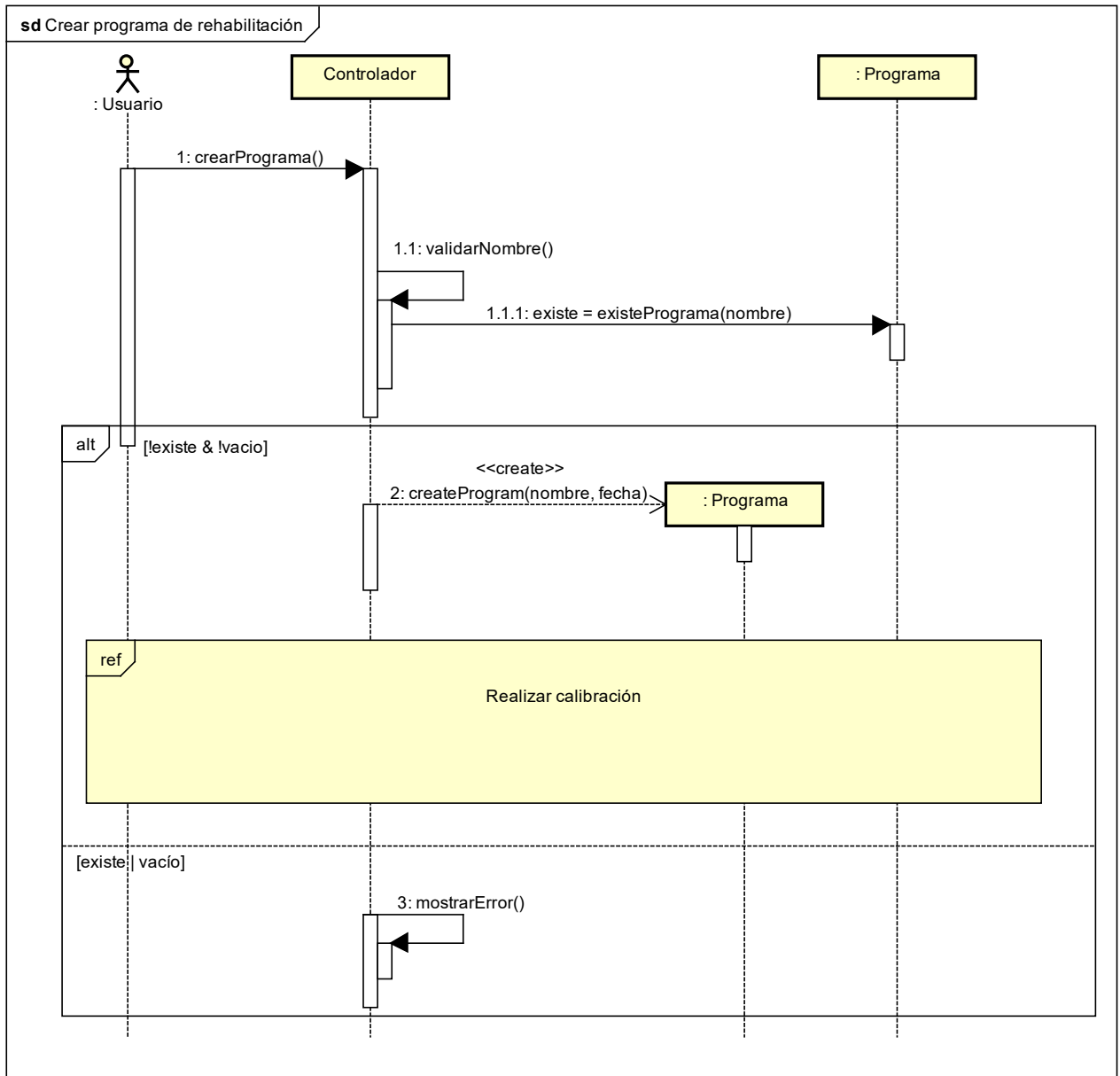


Figura 20: Diagrama de secuencia Crear programa de rehabilitación

4.6.2 Realizar ejercicio de Flexión-Extensión

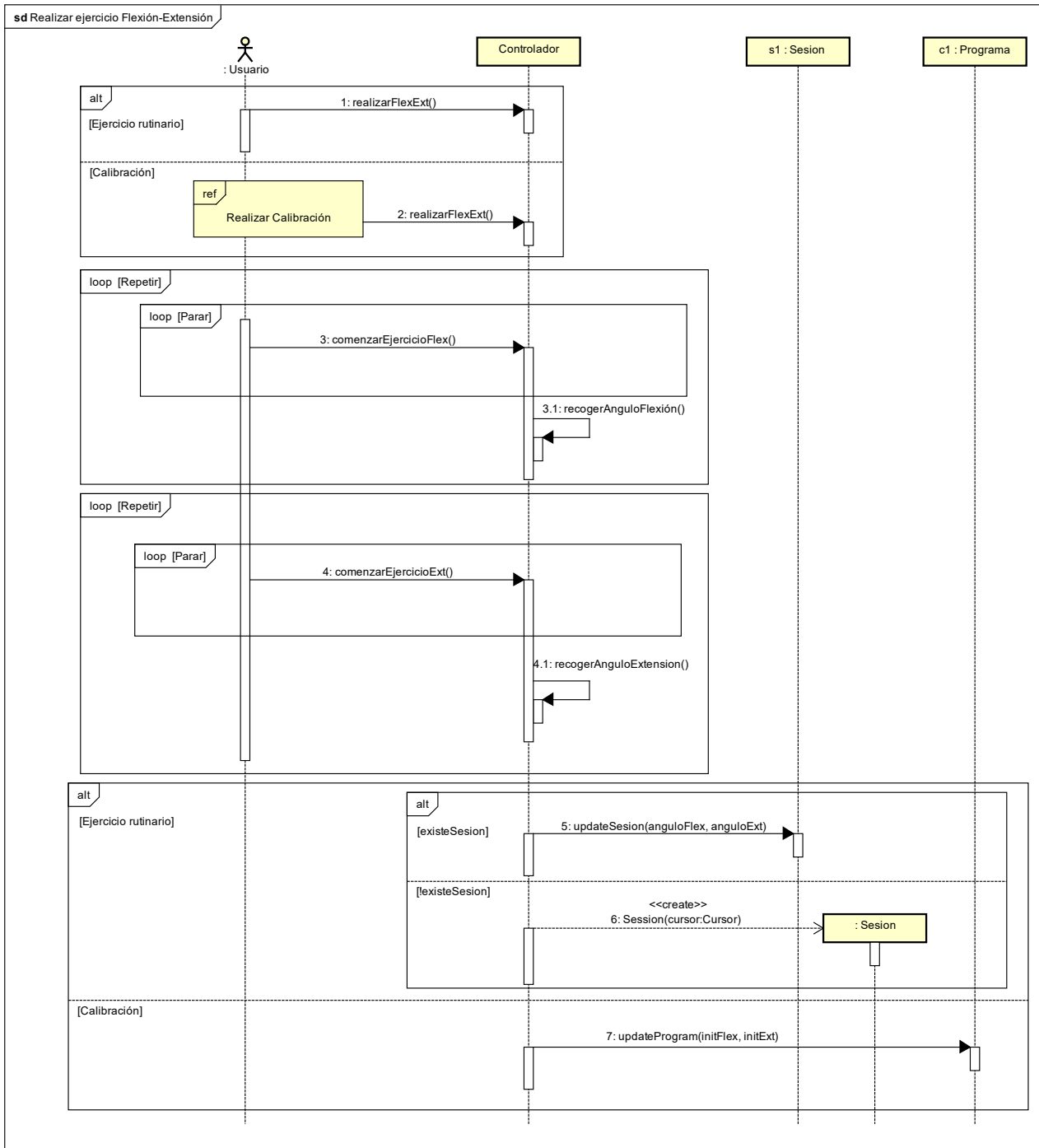


Figura 21: Diagrama de secuencia Realizar ejercicio de Flexión-Extensión

4.6.3 Realizar ejercicio de Abducción-Aducción

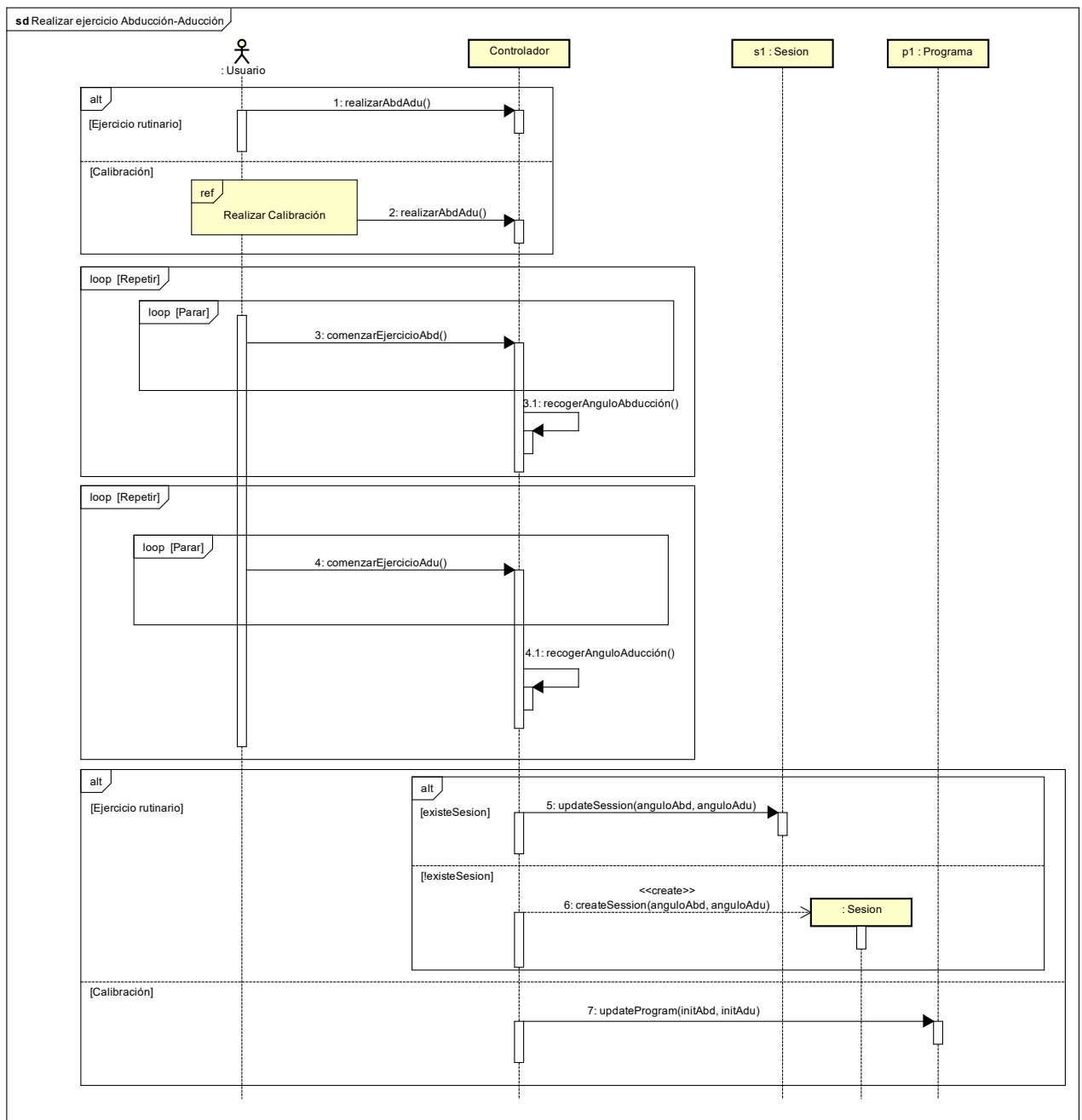


Figura 22: Diagrama de secuencia Realizar ejercicio de Abducción-Aducción

4.6.4 Realizar ejercicio de Supinación-Pronación

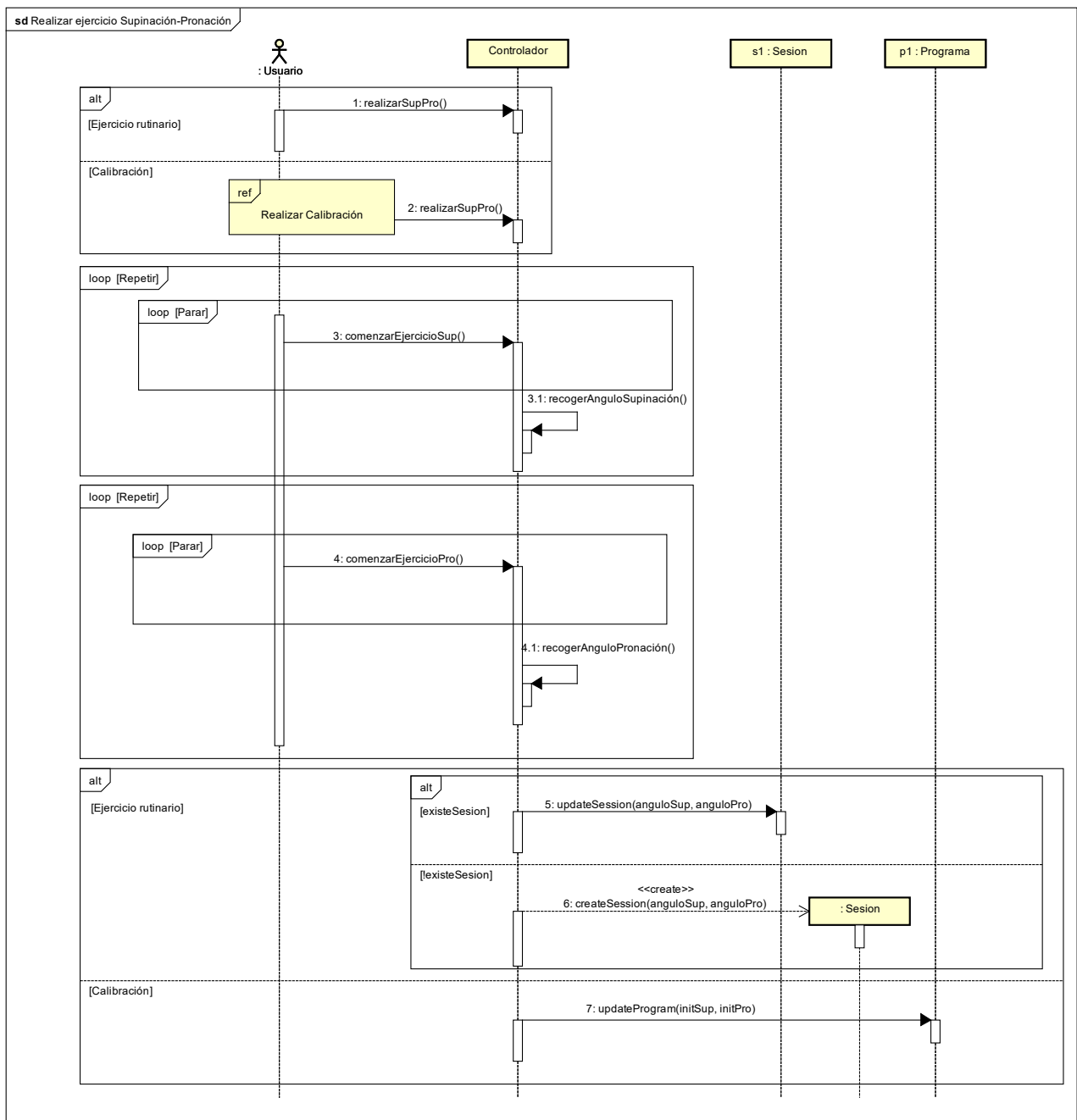


Figura 23: Diagrama de secuencia Realizar ejercicio de Supinación-Pronación

4.6.5 Consultar programa de rehabilitación

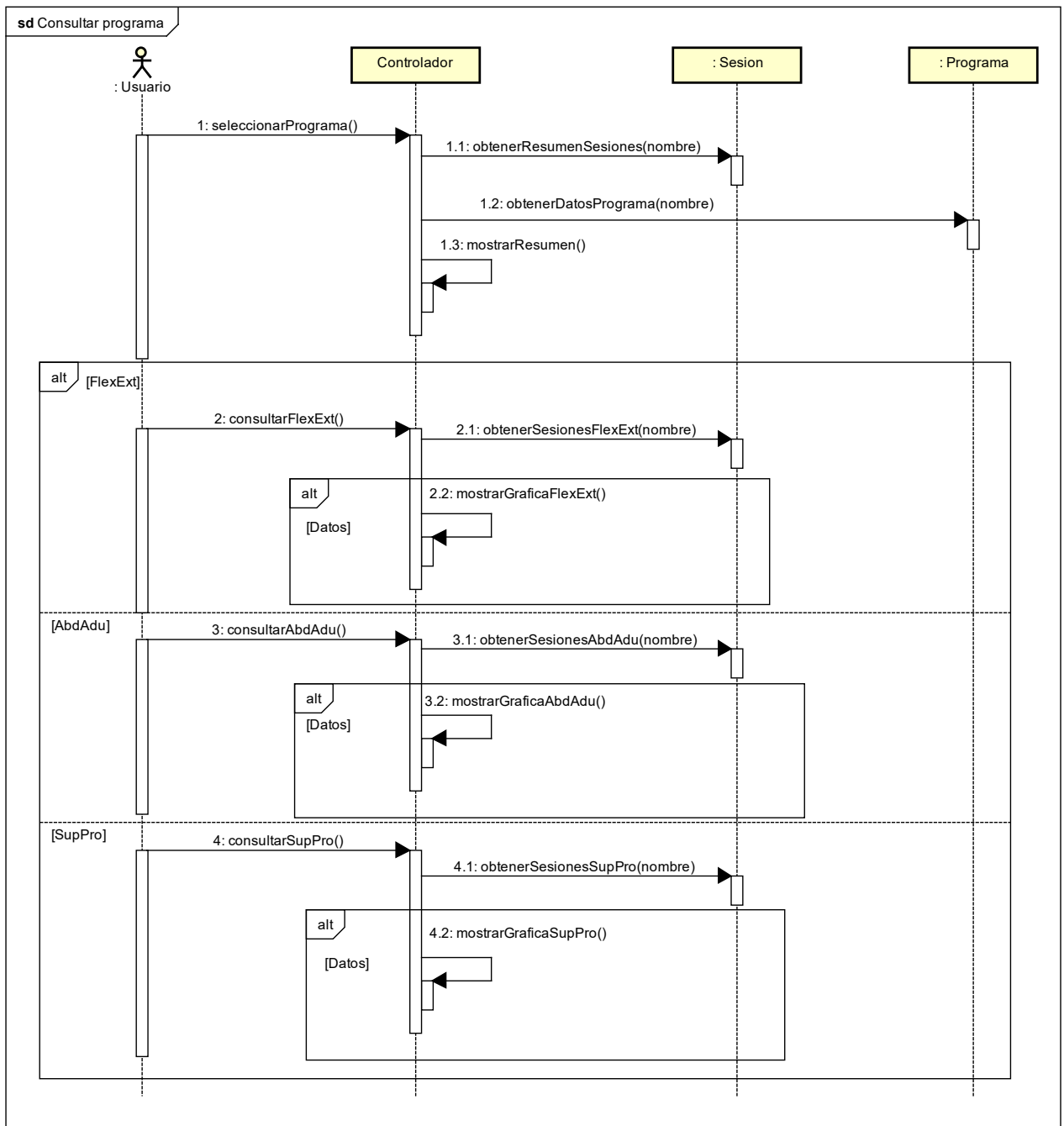


Figura 24: Diagrama de secuencia Consultar programa de rehabilitación

4.6.6 Finalizar programa de rehabilitación

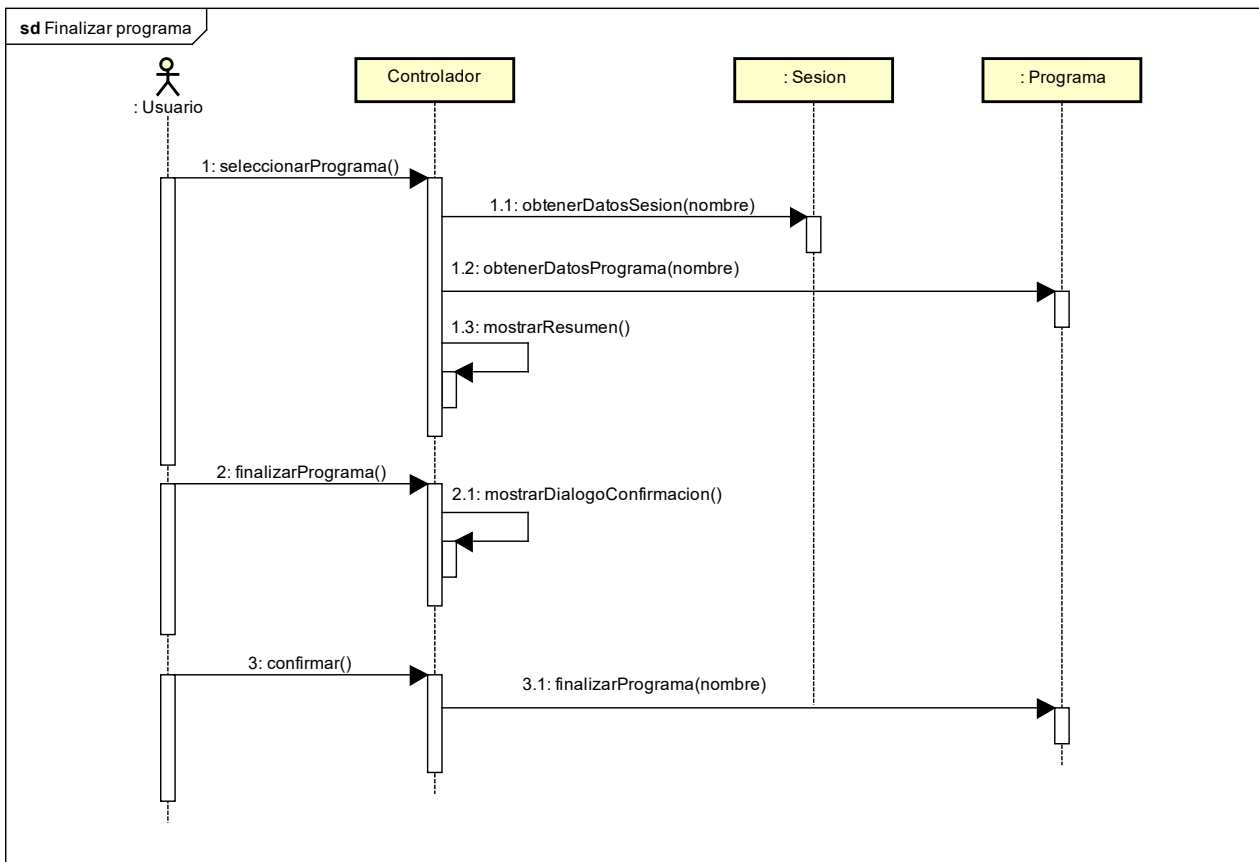


Figura 25: Diagrama de secuencia Finalizar programa de rehabilitación

4.6.7 Eliminar programa de rehabilitación

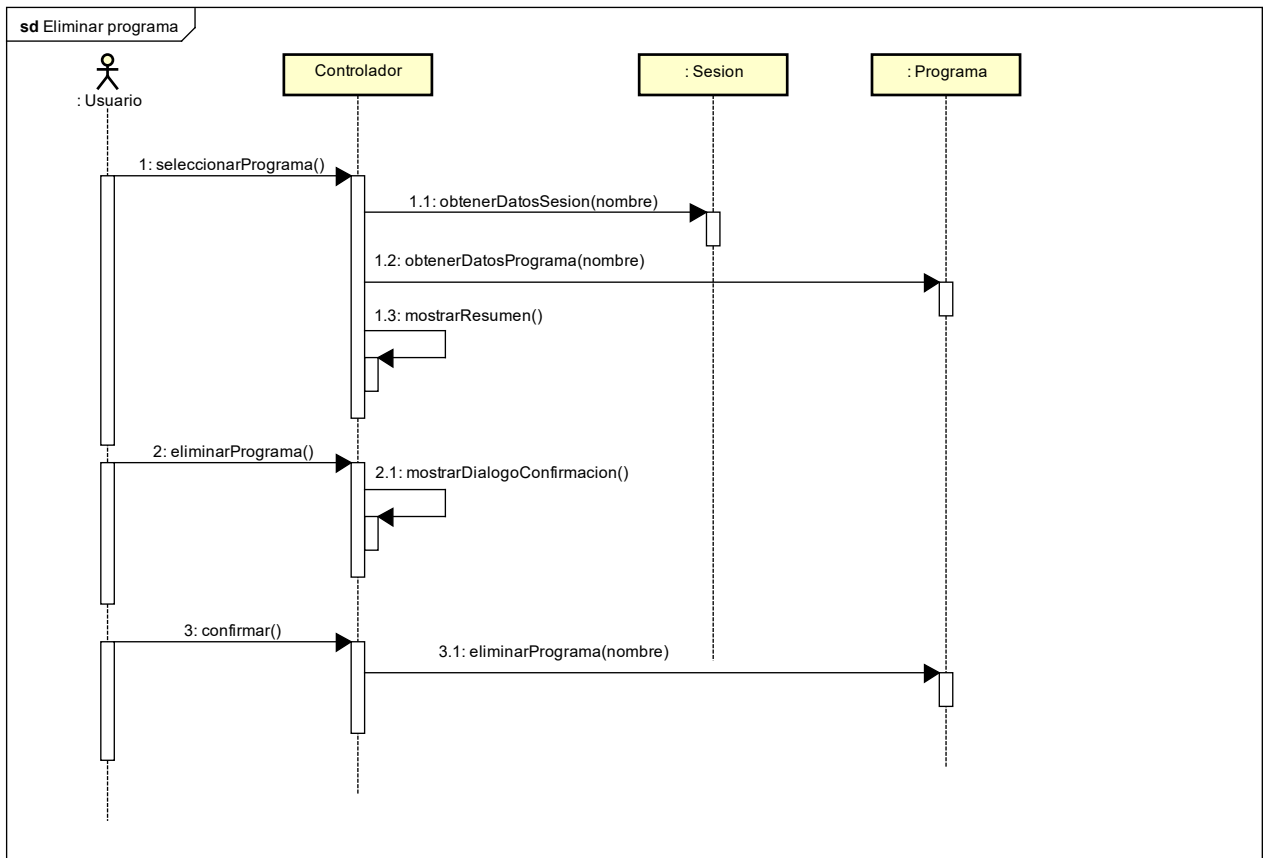


Figura 26: Diagrama de secuencia Eliminar programa de rehabilitación

4.6.8 Realizar calibración

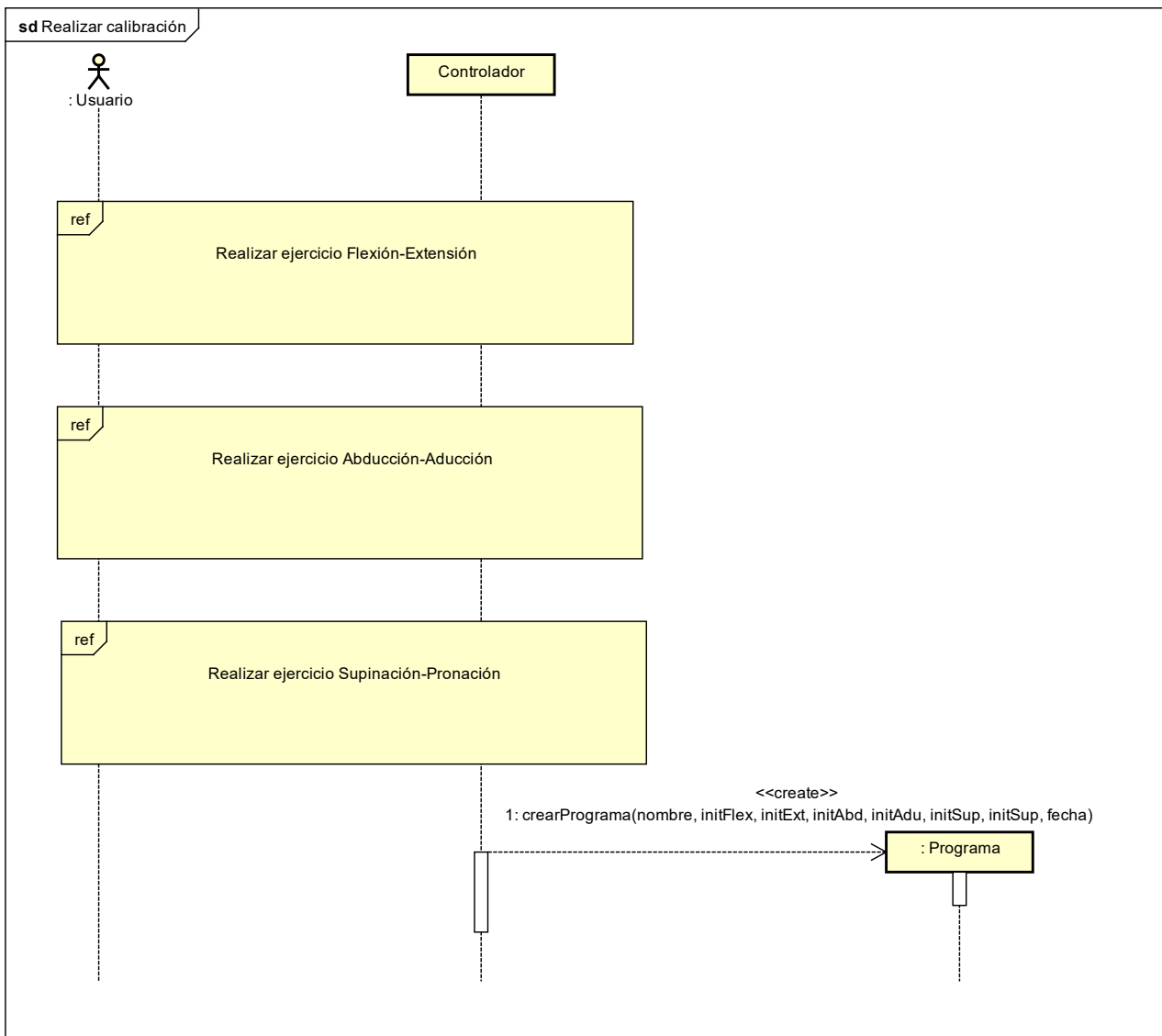


Figura 27: Diagrama de secuencia Realizar calibración

CAPÍTULO 5

DISEÑO DE LA APLICACIÓN

En este capítulo se detalla la etapa de diseño de este proyecto. Su objetivo es mostrar cómo se deberá implementar el sistema para cumplir con los requisitos descritos en el capítulo anterior buscando la mayor eficiencia, robustez y fiabilidad.

5.1 Descripción de la arquitectura lógica y física del sistema

A continuación, se detallará el diseño arquitectónico, tanto lógico como físico del sistema desarrollado. Se ha decidido desarrollar la aplicación para la plataforma Android. La base de datos será de tipo local ya que el cliente no necesita una gestión de datos en la nube puesto que, en principio, el uso de esta aplicación será privado. Como patrón de diseño se ha decidido utilizar MVP (Modelo-Vista-Presentador) puesto que es perfecto para aislar el lenguaje Android en la capa de la Vista. Se proporcionarán más detalles sobre este tema en a continuación.

5.1.1 Vista lógica del sistema. El patrón MVP.

Se ha elegido este patrón de diseño ya que abstrae por completo el lenguaje Android para las capas que estén por debajo de la capa de presentación. De este modo, nos aislamos del lenguaje Android, con todos los beneficios que ello conlleva.

Los componentes de este patrón son:

- **Vista** es una capa que muestra datos y reacciona a la interacción del usuario. En Android, esto puede ser una Activity, un Fragment, una `android.view.View` o un Dialog.
- **Modelo** es una capa de acceso a datos como puede ser una base de datos, una API o una API de servidor remoto.
- **Presentador** es una capa que proporciona datos a la Vista desde el Modelo. El Presentador también se encarga de las tareas en segundo plano.

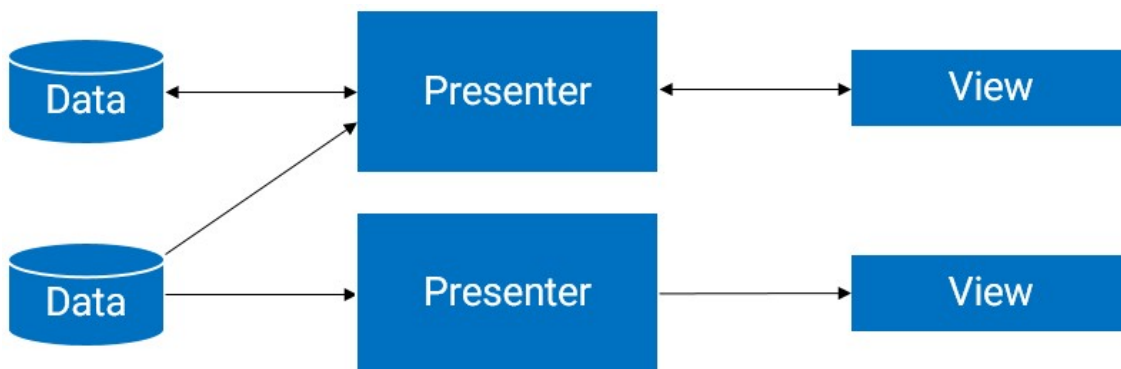


Figura 28: Esquema patrón Modelo-Vista-Presentador

A continuación, se muestra el diagrama de la estructura lógica del sistema:

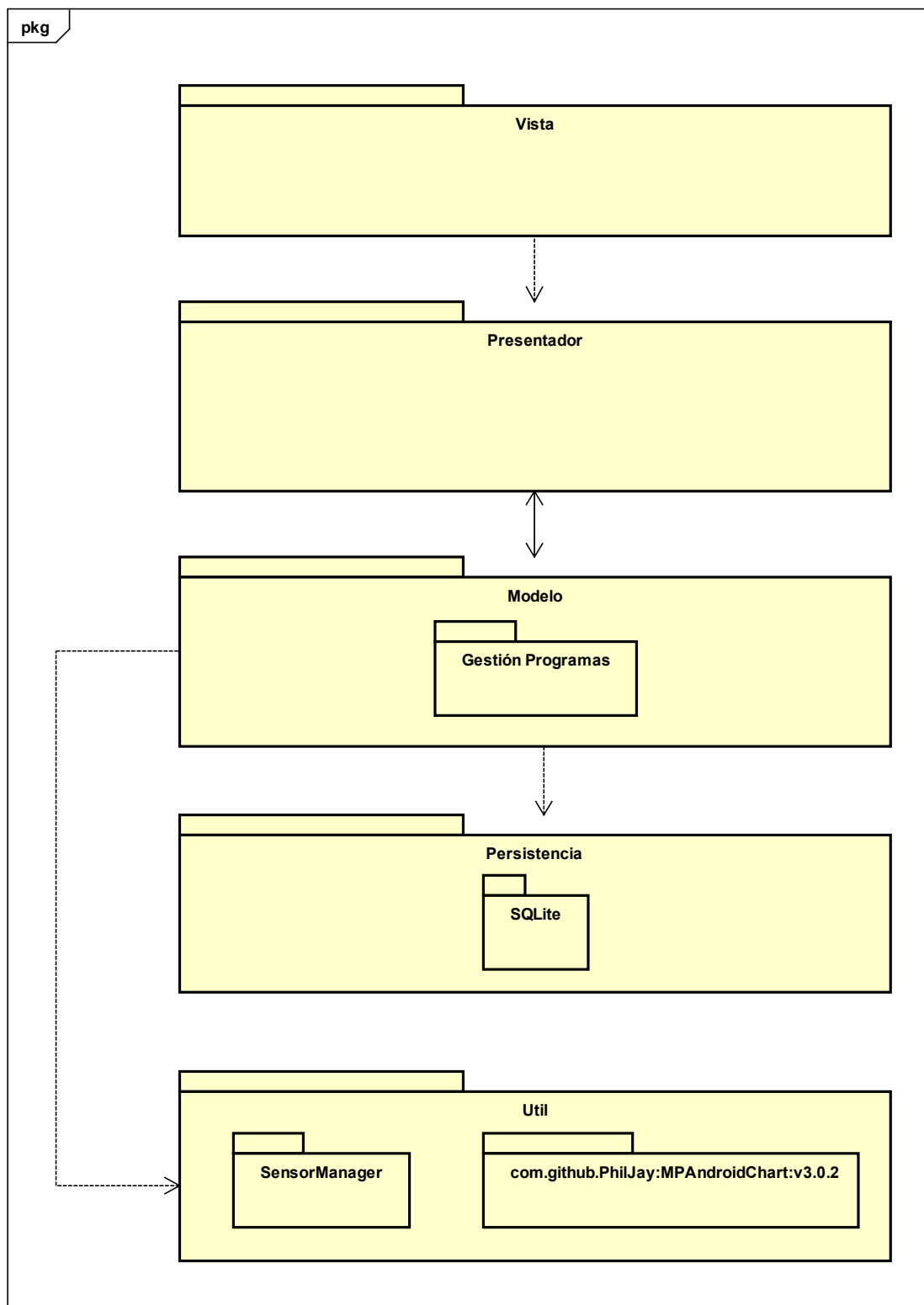


Figura 29: Vista lógica del sistema

5.2 Diagrama de clases de diseño

Se procederá a mostrar el diseño final de las clases de la aplicación. Primero se pondrán los diagramas correspondientes a cada paquete con todas las clases que pertenecen al mismo. Finalmente se mostrará una visión general del sistema.

5.2.1 Paquete Vista

Contiene las vistas de la aplicación por las que el usuario navegará. La elección del patrón MVP hará que estas clases contengan el 100% del código Android de toda la aplicación.

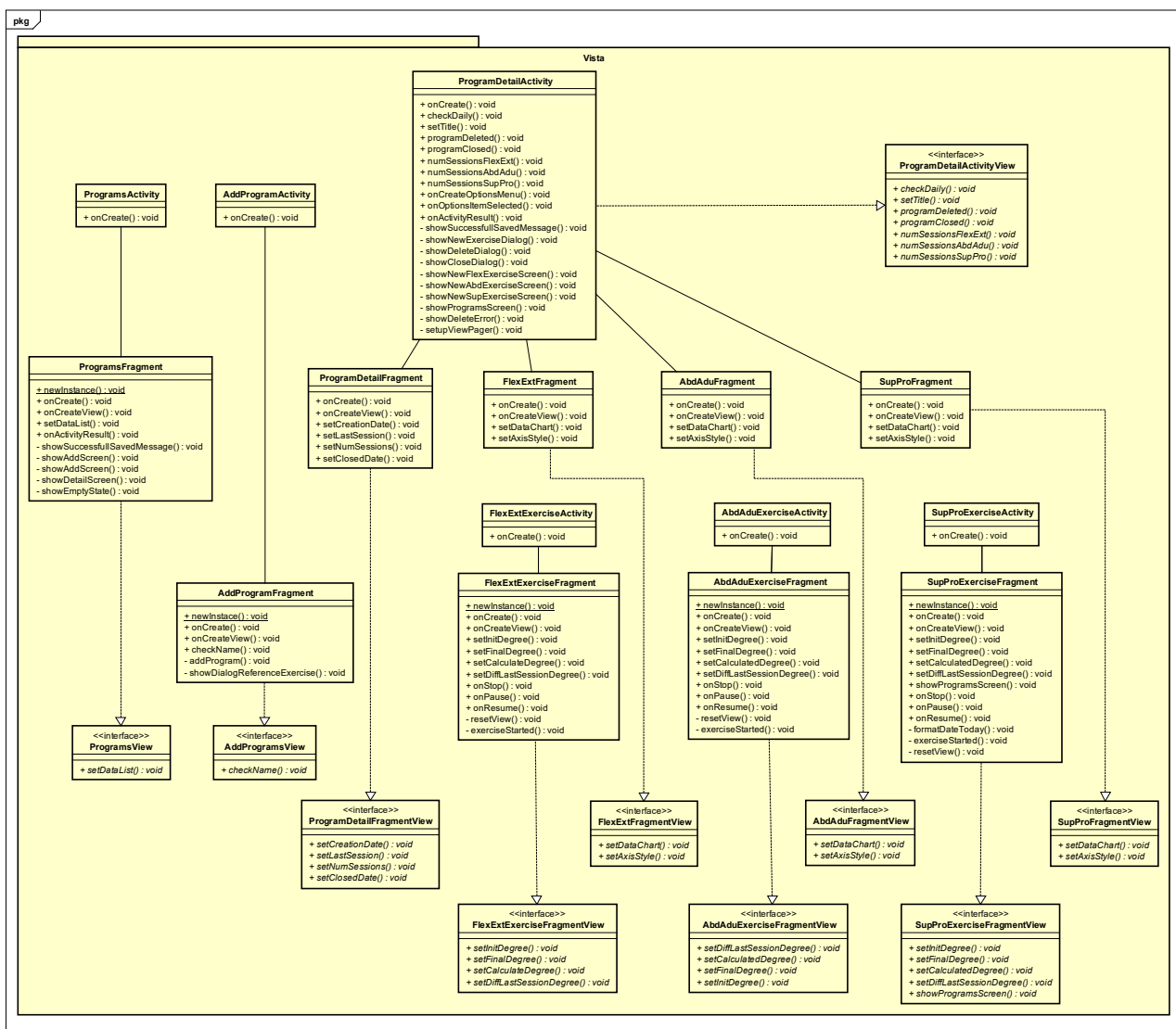


Figura 30: Paquete Vista

A continuación, se explica la función de cada una de estas clases en la aplicación.

- **ProgramsActivity:** Clase que hereda de AppCompatActivity y se encarga de lanzar el fragmento ProgramsFragment.
- **ProgramsFragment:** Clase que hereda de Fragment e implementa la interfaz ProgramsView. Aloja el contenido de la pantalla principal y se encarga de mostrar la lista de programas creados y un botón para crear un nuevo programa.
- **ProgramsView:** Interfaz encargada de proveer el método para cargar la lista de programas en la pantalla principal.
- **AddProgramActivity:** Clase que hereda de AppCompatActivity y se encarga de lanzar el fragmento AddProgramFragment.
- **AddProgramFragment:** Clase que hereda de Fragment e implementa la interfaz AddProgramsView. Se encarga de realizar la creación de un nuevo programa de rehabilitación.
- **AddProgramsView:** Interfaz encargada de proveer el método para comprobar si existe el nombre introducido por el usuario.
- **ProgramDetailActivity:** Clase que hereda de AppCompatActivity e implementa la interfaz ProgramDetailActivityView. Se encarga de crear los fragmentos que contendrán los detalles del programa así como cada una de las gráficas de los diferentes ejercicios. Además gestionará el borrado y finalización de los programas y las comprobaciones previas a la creación de un nuevo ejercicio.
- **ProgramDetailActivityView:** Interfaz encargada de proveer los métodos que se encargan de modelar la interfaz y la Toolbar según los datos del programa.
- **ProgramDetailFragment:** Clase que hereda de Fragment e implementa la interfaz ProgramDetailFragmentView
- **ProgramDetailFragmentView:** Interfaz encargada de proveer los métodos que rellenarán los datos referentes al resumen del programa.
- **FlexExtFragment:** Clase que hereda de Fragment y se encarga de mostrar la gráfica que contiene los datos del ejercicio de Flexión-Extensión.
- **FlexExtFragmentView:** Interfaz que provee los métodos para rellenar la gráfica y dar formato a sus ejes.
- **AbdAduFragment:** Clase que hereda de Fragment y se encarga de mostrar la gráfica que contiene los datos del ejercicio de Abducción-Aducción.
- **AbdAduFragmentView:** Interfaz que provee los métodos para rellenar la gráfica y dar formato a sus ejes.
- **SupProFragment:**
- **SupProFragmentView:** Interfaz que provee los métodos para rellenar la gráfica y dar formato a sus ejes.
- **FlexExtExerciseActivity:** Clase que hereda de AppCompatActivity y se encarga de lanzar el fragmento FlexExtExerciseFragment.
- **FlexExtExerciseFragment:** Clase que hereda de Fragment y se encarga de ofrecer el ejercicio de Flexión-Extensión al usuario y de recoger los resultados.

- **FlexExtExerciseFragmentView:** Interfaz encargada de proveer los métodos para mostrar los ángulos de los ejercicios realizados por pantalla.
- **AbdAduExerciseActivity:** Clase que hereda de AppCompatActivity y se encarga de lanzar el fragmento AbdAduExerciseFragment.
- **AbdAduExerciseFragment:** Clase que hereda de Fragment y se encarga de ofrecer el ejercicio de Abducción-Aducción al usuario de y recoger los resultados.
- **AbdAduExerciseFragmentView:** Interfaz encargada de proveer los métodos para mostrar los ángulos de los ejercicios realizados por pantalla.
- **SupProExerciseActivity:** Clase que hereda de AppCompatActivity y se encarga de lanzar el fragmento SupProExerciseFragment.
- **SupProExerciseFragment:** Clase que hereda de Fragment y se encarga de ofrecer el ejercicio de Supinación-Pronación al usuario de y recoger los resultados.
- **SupProExerciseFragmentView:** Interfaz encargada de proveer los métodos para mostrar los ángulos de los ejercicios realizados por pantalla.

5.2.2 Paquete Presentador

Contiene las clases que se encargan de la lógica puramente Java del sistema y es el encargado de contactar con la capa del Modelo y del paquete de Utilidades.

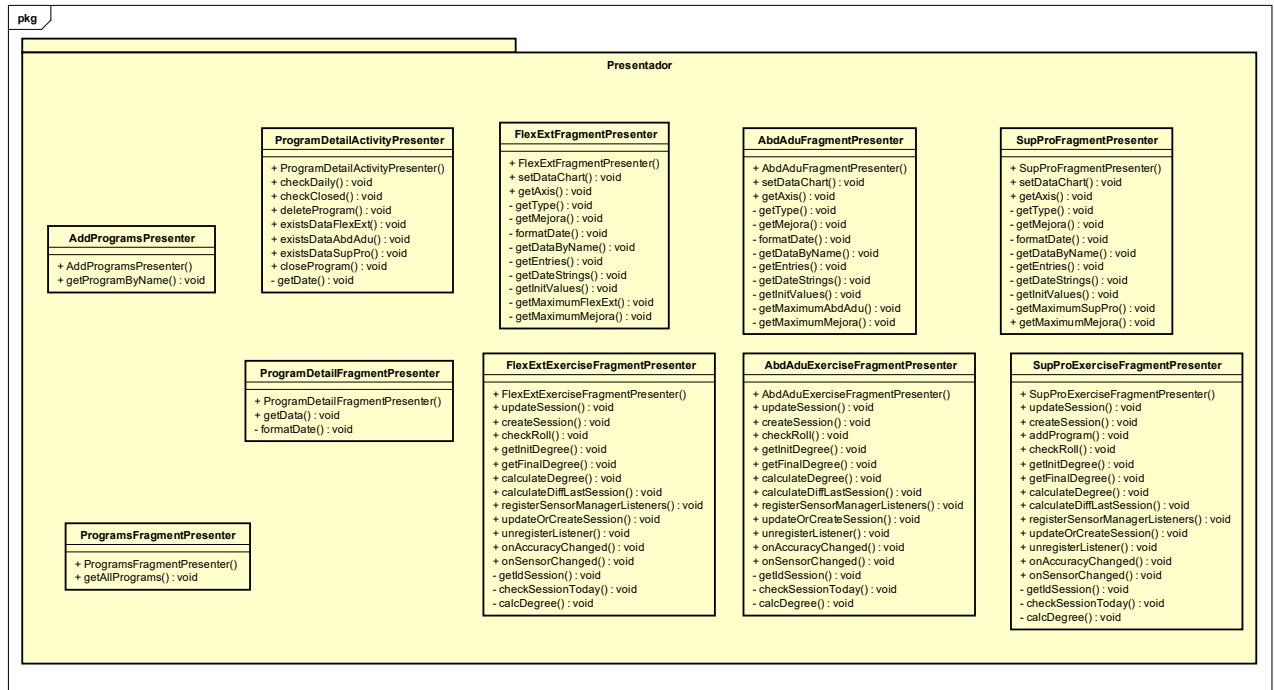


Figura 31: Paquete Presentador

A continuación, se explica la función de cada una de estas clases en la aplicación.

- **ProgramsFragmentPresenter:** Clase que se encarga de recoger la lista de programas.
- **AddProgramsPresenter:** Clase que se encarga de recoger el programa del nombre introducido por el usuario para comprobaciones.
- **ProgramDetailActivityPresenter:** Clase que se encarga de comprobar si el programa está finalizado y de si existe algún ejercicio realizado. Además, se encarga de finalizar y eliminar el programa.
- **ProgramDetailFragmentPresenter:** Clase que se encarga de recoger los datos necesarios para mostrar los detalles del programa.
- **FlexExtFragmentPresenter:** Clase que se encarga de recoger los datos necesarios para representar en la gráfica del ejercicio de Flexión-Extensión.
- **FlexExtExerciseFragmentPresenter:** Clase que se encarga de calcular los ángulos recogidos en los ejercicios de Flexión-Extensión así como calcular los datos comparando con ejercicios anteriores.
- **AbdAduFragmentPresenter:** Clase que se encarga de recoger los datos necesarios para representar en la gráfica del ejercicio de Abducción-Aducción.

- **AbdAduExerciseFragmentPresenter:** Clase que se encarga de calcular los ángulos recogidos en los ejercicios de Abducción-Aducción así como calcular los datos comparando con ejercicios anteriores.
- **SupProFragmentPresenter:** Clase que se encarga de recoger los datos necesarios para representar en la gráfica del ejercicio de Supinación-Pronación.
- **SupProExerciseFragmentPresenter:** Clase que se encarga de calcular los ángulos recogidos en los ejercicios de Supinación-Pronación así como calcular los datos comparando con ejercicios anteriores.

5.2.3 Paquete Modelo

Contiene las clases que se encargan de acceder a la base de datos y realizar consultas en la misma. Además contiene las clases básicas de la aplicación.

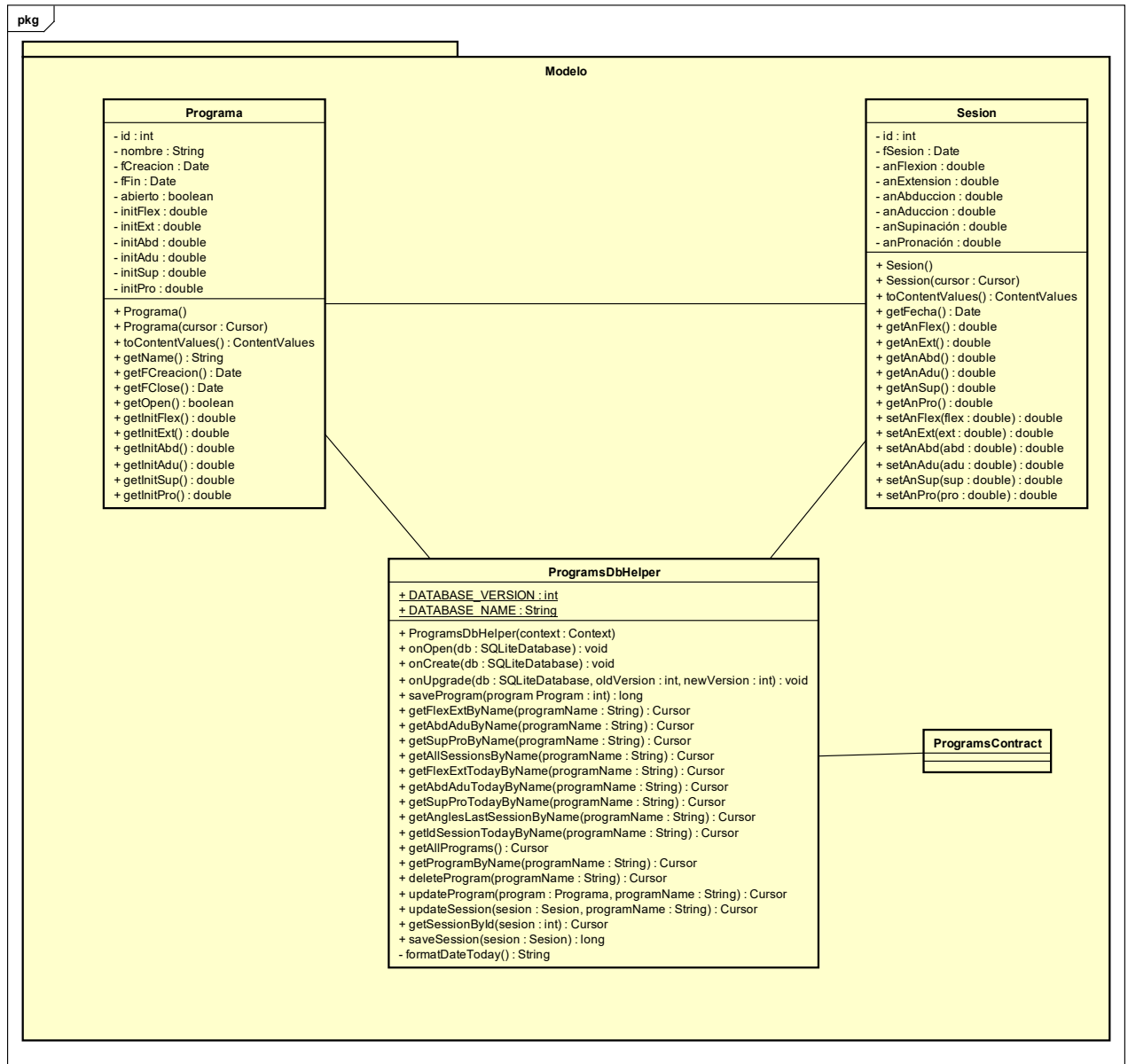


Figura 32: Paquete Modelo

A continuación, se explica la función de cada una de estas clases en la aplicación.

- **Programa:** Clase que guardará los datos del programa de rehabilitación.
- **Sesión:** Clase que guardará los datos de las sesiones realizadas por el usuario.
- **ProgramsDbHelper:** Clase que realiza todas las consultas necesarias para el sistema en la base de datos.
- **ProgramsContract:** Contiene el tipo de datos Programa y Sesión y define las columnas y los tipos para crear la base de datos.

5.2.4 Paquete Utilidades

Contiene las librerías y clases a las que se accederá en el Modelo para la obtención de información de los sensores así como de la librería donde se encuentra la información de las gráficas.

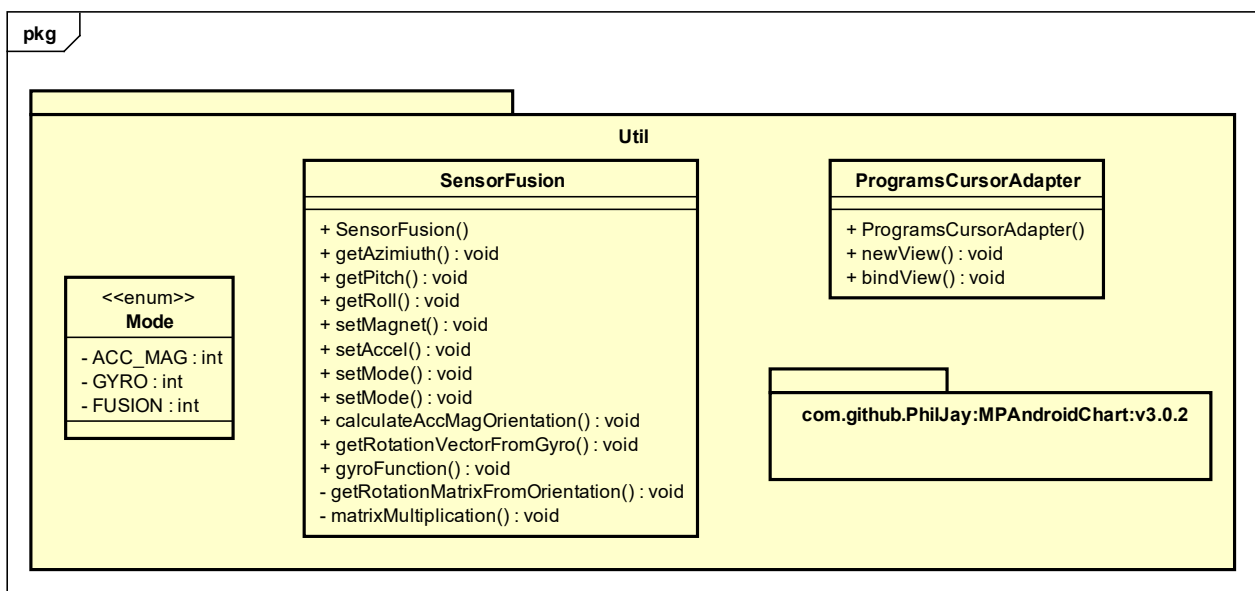


Figura 33: Paquete Util.

A continuación, se explica la función de cada una de estas clases en la aplicación.

- **SensorFusion:** Clase que contiene todos los cálculos necesarios para obtener los ángulos y la posición del Smartphone en tiempo real.
- **ProgramsCursorAdapter:** Clase que se encarga de enlazar un Cursor a un elemento de tipo lista.
- **Com.github.PhilJay.MPAndroidChart.v3.0.2:** Paquete que contiene todas las clases necesarias para gestionar y diseñar las gráficas de los resultados.

Visión general de las clases del proyecto

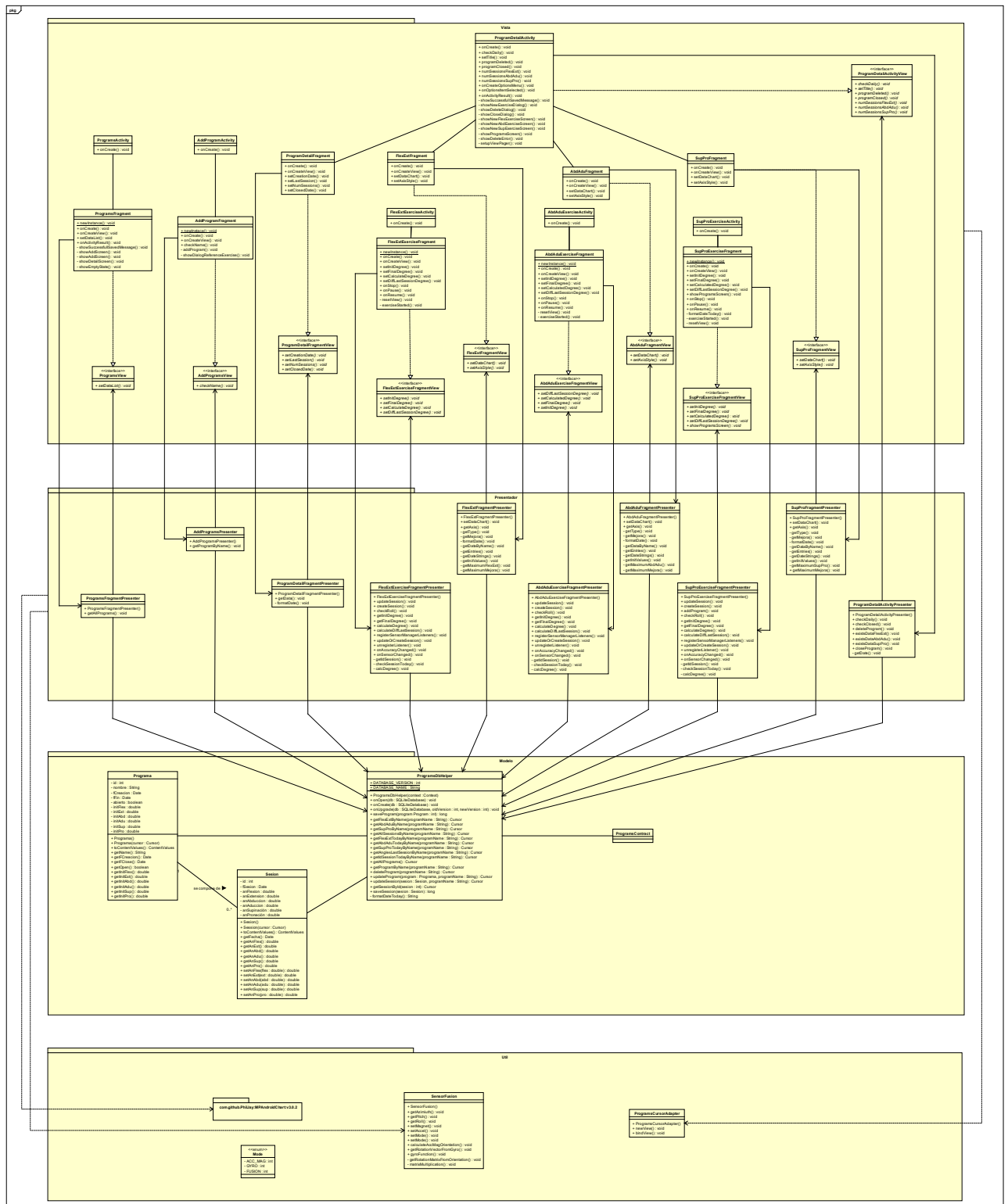


Figura 34: Visión general Diagrama de Clases.

5.3 Diagrama de casos de uso de diseño

En esta sección veremos el diagrama de secuencia de un caso de uso significativo del sistema para observar el paso de mensajes típico.

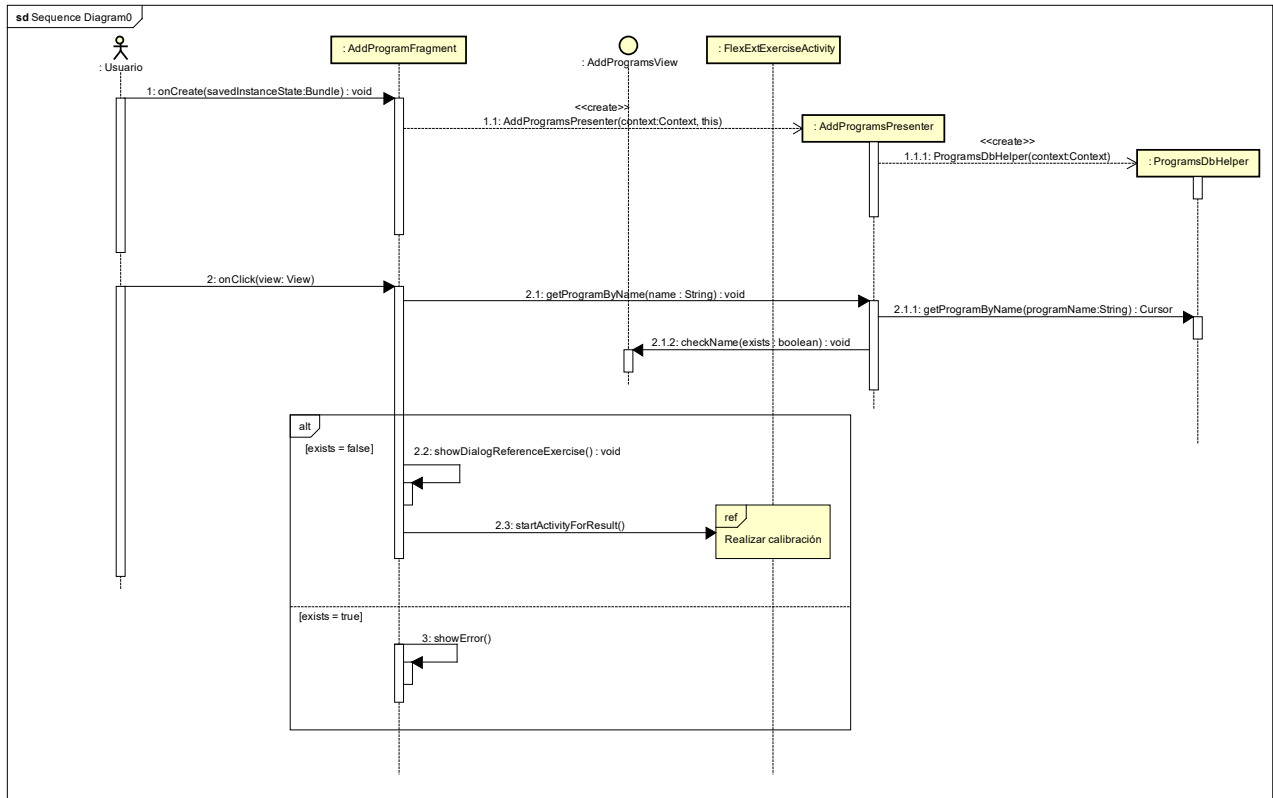


Figura 35: Diagrama de casos de uso de diseño significativo.

Como puede observarse, al crear la vista, se crea una instancia del Presentador correspondiente y éste, a su vez, crea una instancia de `ProgramsDbHelper` de donde obtendrá los resultados de las consultas a base de datos. De esta forma, la vista sólo espera interacciones del usuario como el `onClick()`, en este caso. Una vez se produce esta llamada, la vista cede la implementación de la lógica necesaria al Presenter, el cual obtiene todo lo necesario de la base de datos para dar una respuesta sobre si el nombre del programa existe o no llamando al método de la interfaz correspondiente de forma que la vista, que implementa dicho método, decida qué hacer con los elementos del Fragment.

5.4 Diseño de la base de datos

A continuación, se mostrará el diagrama Entidad-Relación.

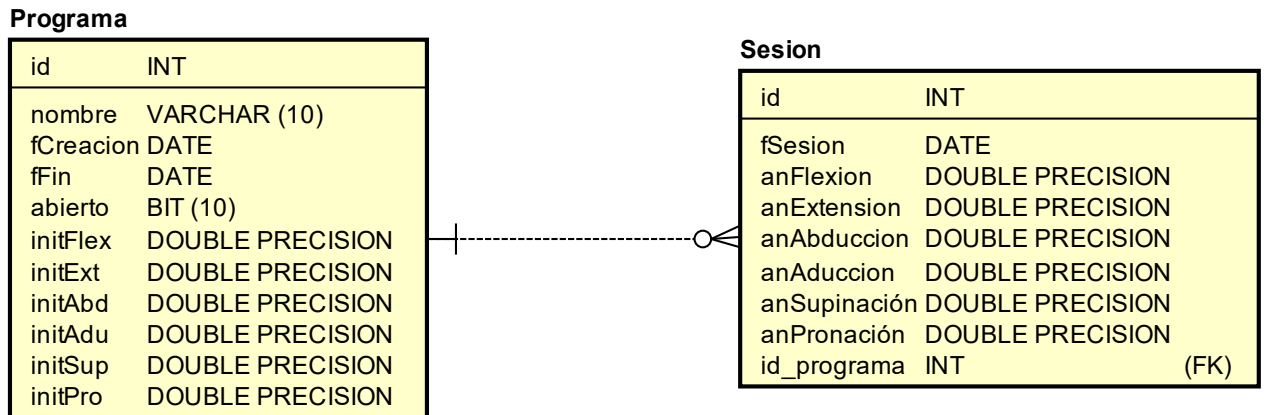


Figura 36: Diagrama Entidad-Relación

Las entidades que podemos ver en el modelo Entidad-Relación son:

- **Programa:** Contiene la información básica sobre el paciente. Así como los ángulos obtenidos de la calibración inicial.
- **Sesión:** Contiene la información de los ángulos obtenidos en los ejercicios rutinarios realizados en cada sesión.

5.5 Diseño de la interfaz

Se han puesto como principales objetivos del diseño obtener una interfaz que cumpla las siguientes condiciones:

- Centrada en la facilidad de aprendizaje.
- Simple.
- Visualmente atractiva.

Se ha decidido utilizar el lenguaje visual *Material Design* ya que es capaz de aunar estas tres características. No solo se han utilizado los componentes característicos de dicho lenguaje visual sino también su filosofía. Desde la elección de colores, hasta la gestión de *empty-state*. Que no es más que utilizar imágenes de fondo cuando la pantalla se encuentra vacía.

La interfaz cuenta con cuatro pantallas. La pantalla de inicio, la pantalla de creación de programa, la pantalla de detalles de programas y la de realización de ejercicios.

5.5.1 Pantalla de inicio

La pantalla de inicio cuenta únicamente con una lista que muestra todos los programas creados (finalizados o no) y un Floating Action Button (FAB en adelante) para crear un nuevo programa. Si no existiera ningún programa se mostrará una imagen de *empty-state* siguiendo la filosofía de Material Design. Una imagen informativa sin mucho contraste con el resto de la interfaz y un texto orientativo. En el caso del uso de la aplicación por parte de un particular, se espera que la lista de programas no contenga más de 1 ó 2 programas de rehabilitación. Por ello, he utilizado una imagen de fondo para llenar el espacio.

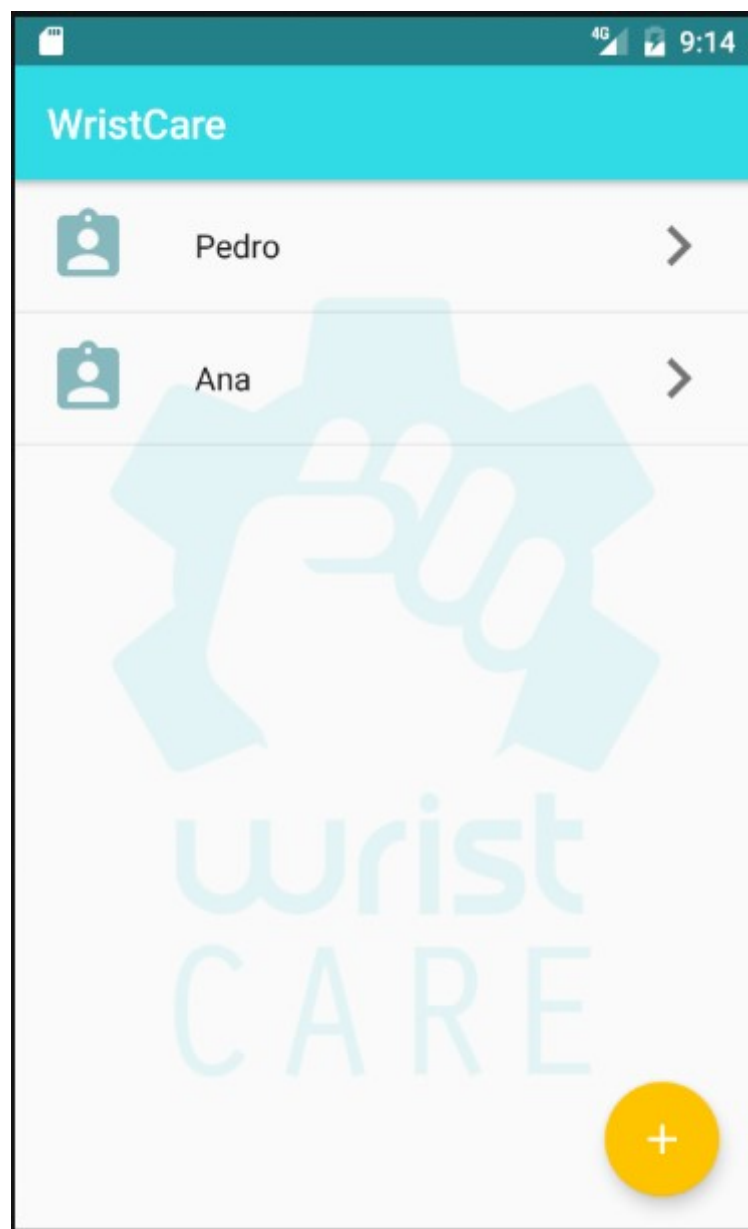


Figura 37: Pantalla principal

Aquí se muestra el caso de que no exista ningún programa de rehabilitación ya creado.

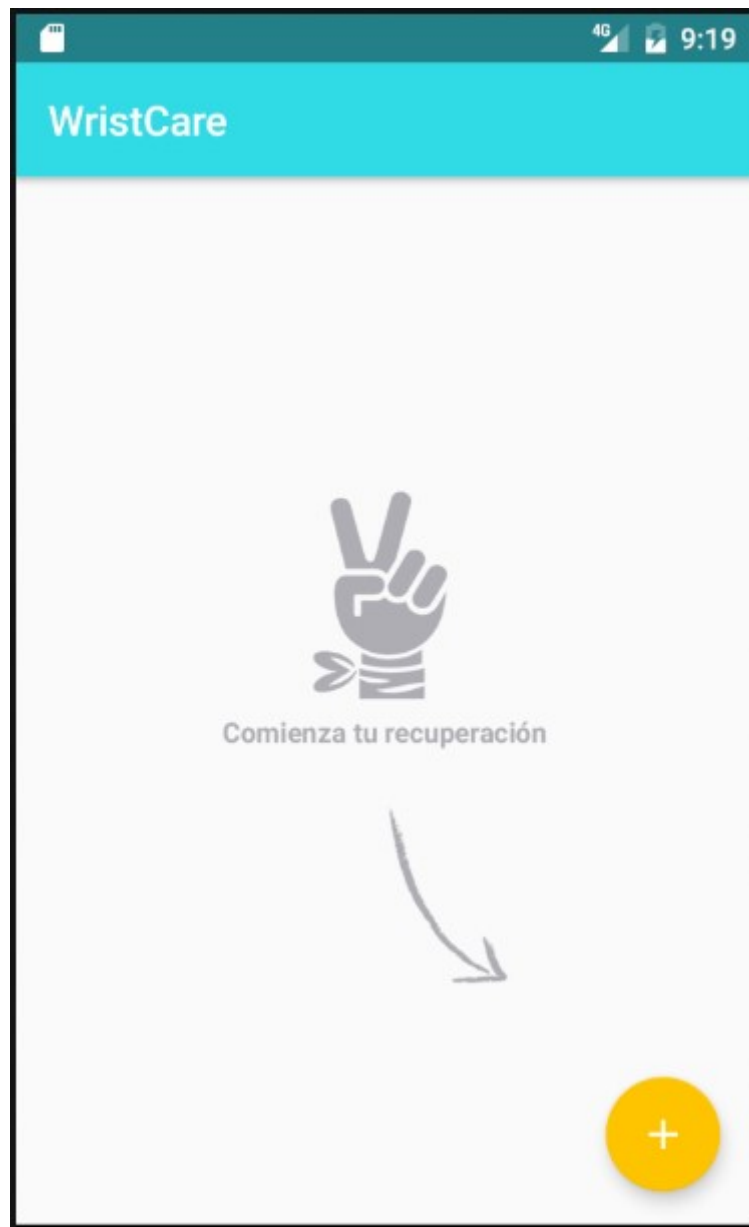


Figura 38: Pantalla principal (empty-state)

5.5.2 Pantalla de creación de programa de rehabilitación

Puesto que no se guardan apenas datos del usuario, solo será necesario indicar un nombre único para el programa. Se comprobará si el campo está vacío o si el nombre ya existe. Una vez se pulse el botón de añadir y el nombre sea único, se realizará la calibración inicial para los tres ejercicios.

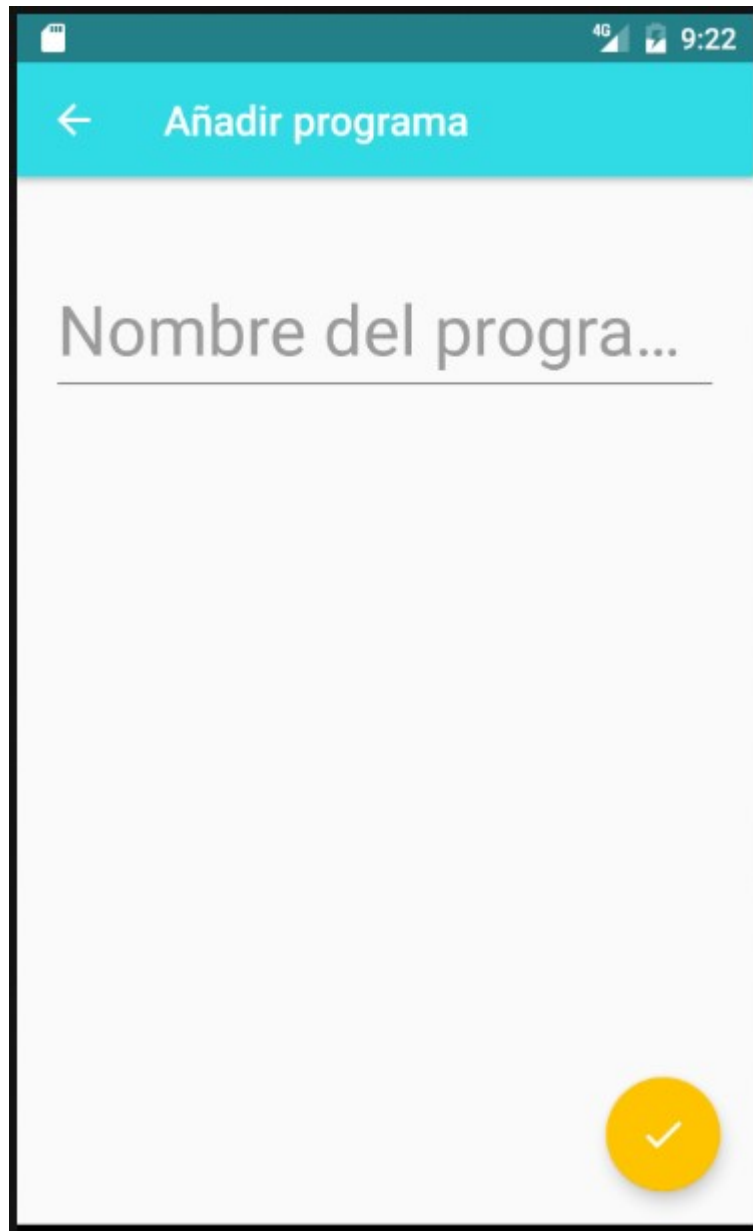
The image shows a mobile application interface for creating a rehabilitation program. At the top, there is a teal-colored header bar containing a white back arrow on the left and the text 'Añadir programa' in white. Below the header, the main area is light gray and contains a large text input field with the placeholder text 'Nombre del programa...' in a light gray font. At the bottom right of the screen, there is a prominent yellow circular button with a white checkmark inside, indicating a confirmation or 'add' action.

Figura 39: Pantalla creación programa

5.5.3 Pantalla detalles de programa

La pantalla de detalles se compone de un *Tab Layout* que contiene cuatro pestañas, una para cada tipo de ejercicio y otra que muestra un resumen del programa. El usuario podrá navegar entre cada tipo de ejercicio para observar el progreso del programa de rehabilitación.



Figura 40: Pantalla detalles de programa

Cada tipo de ejercicio contendrá cuatro tipos de gráficas, que se seleccionarán con el *Spinner*, o *Dropdown Menu*, situado arriba a la izquierda. En el capítulo de implementación se explicarán los cálculos realizados para poblar estas gráficas. Las cuatro opciones son:

Comparativa de ambas componentes del ejercicio:

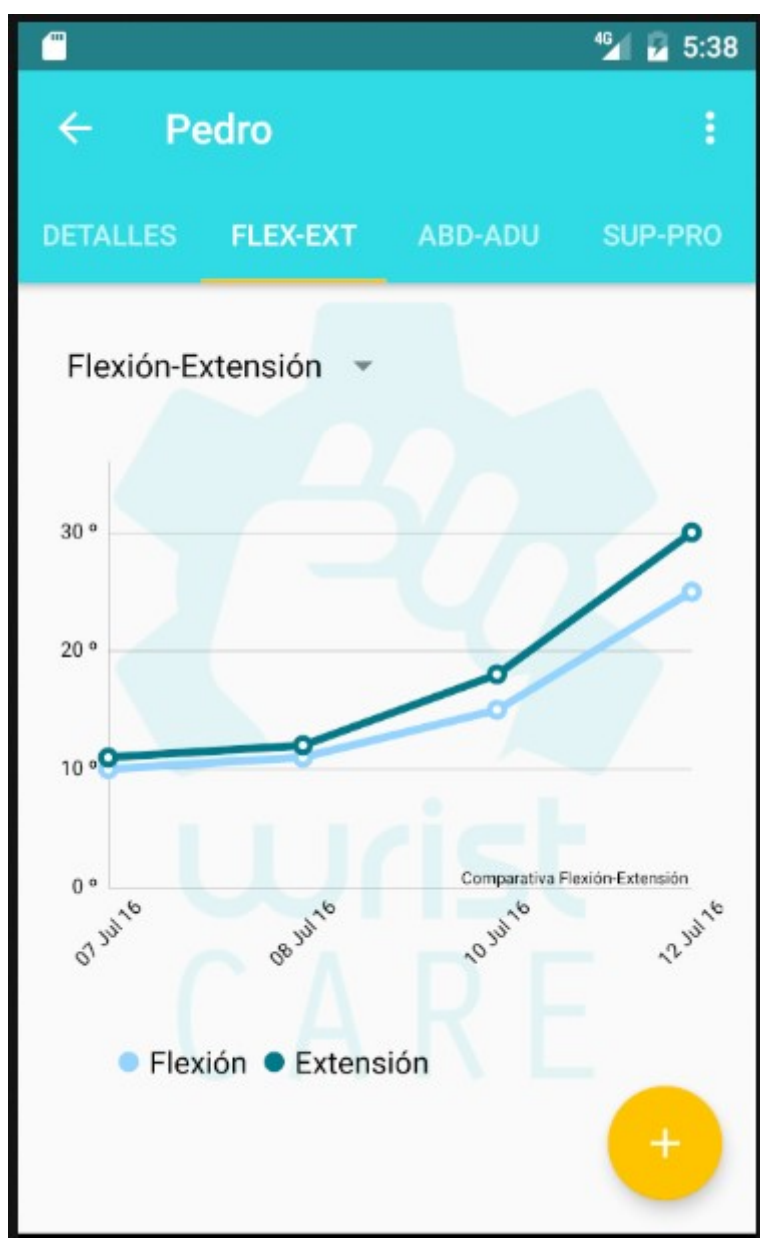


Figura 41: Gráfica Flexión-Extensión

Componente #1 del ejercicio junto a su mejora media y la referencia del componente recogida en la creación del programa:

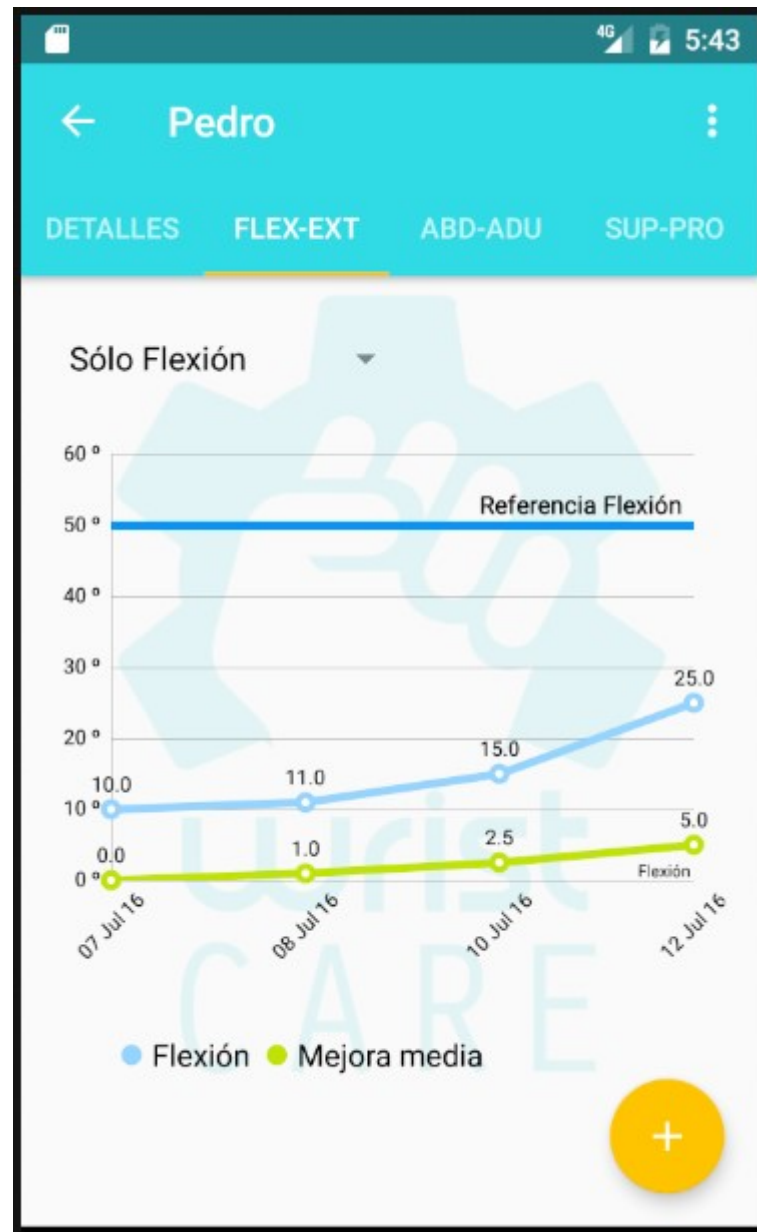


Figura 42: Gráfica sólo Flexión

Componente #1 del ejercicio junto a su mejora media y la referencia del componente recogida en la creación del programa:

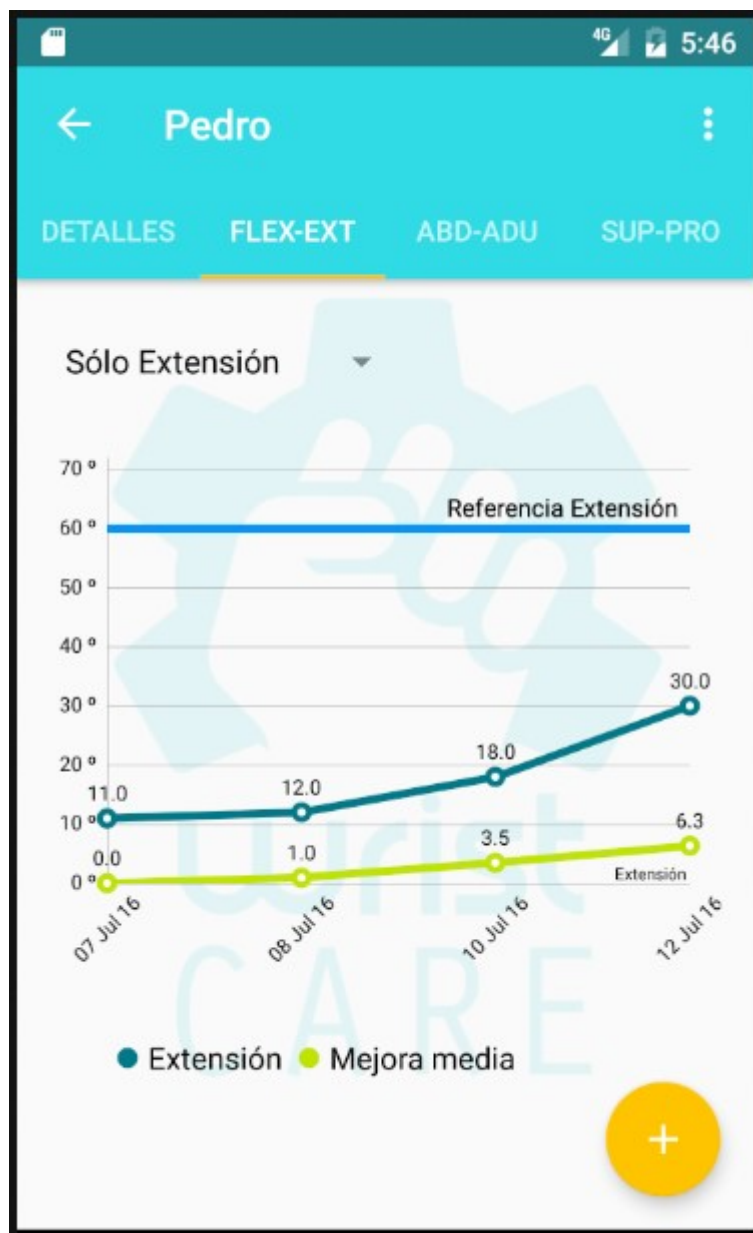


Figura 43: Gráfica sólo Extensión

Comparativa de las mejoras medias de cada componente del ejercicio:

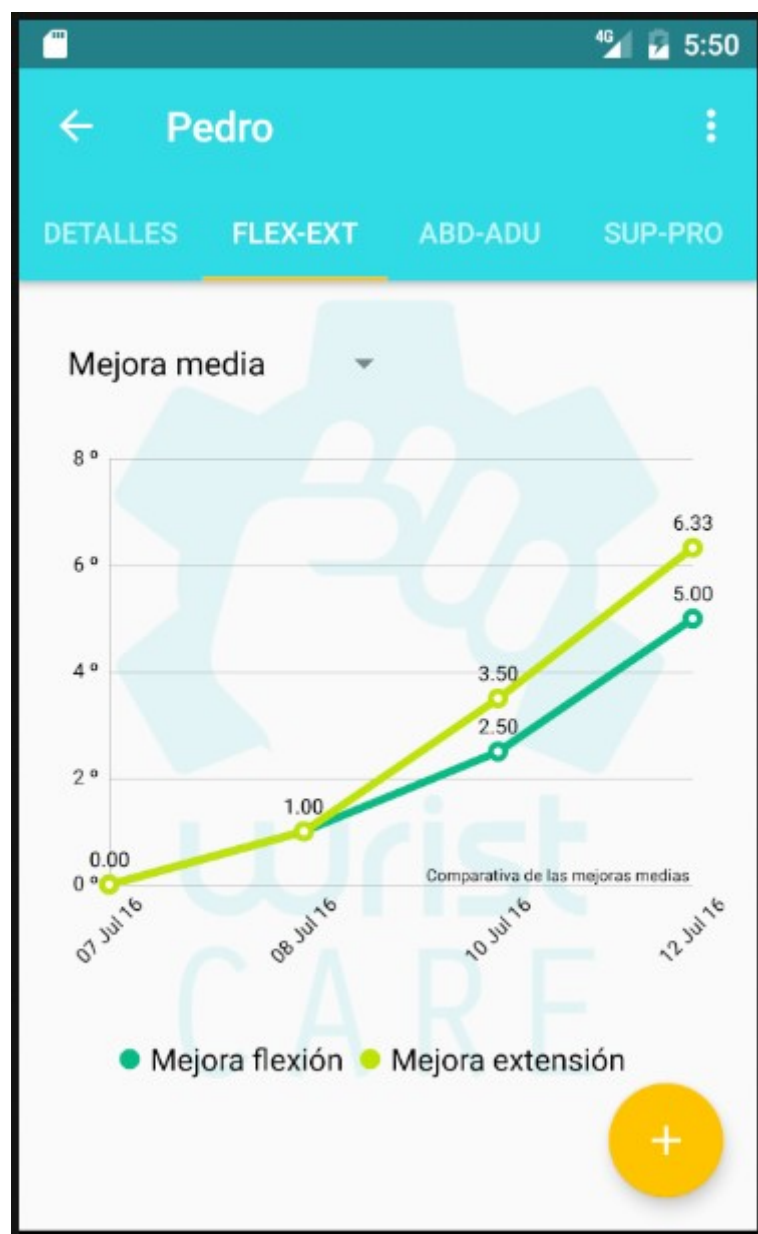


Figura 44: Gráfica comparativa mejoras medias

En el caso de no haber ningún ejercicio registrado, se mostrará de nuevo un *empty-state* con una imagen orientativa.

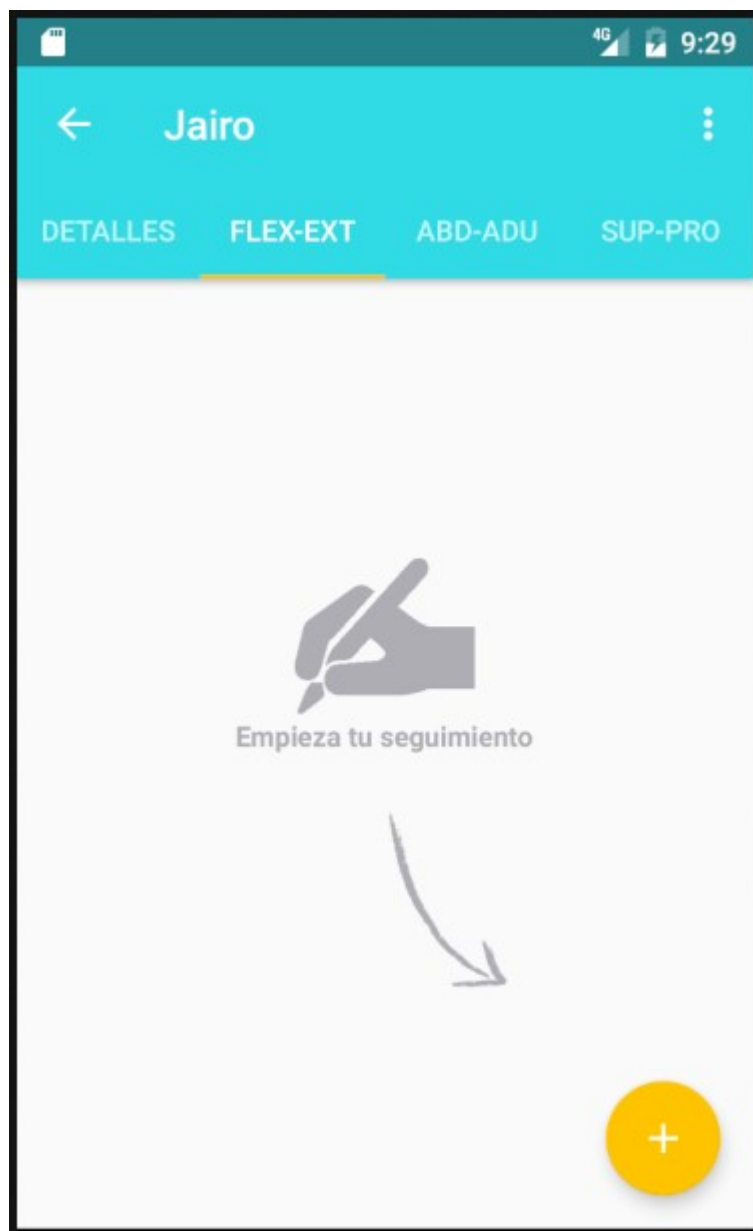


Figura 45: Gráfica ejercicio (*empty-state*)

5.5.4 Pantalla realización de ejercicio

Esta pantalla es accedida en dos puntos de la aplicación. La primera, cuando se va a realizar una nueva sesión de un ejercicio para guardar los registros en la base de datos y actualizar las gráficas. La segunda, cuando se realiza la calibración inicial para guardar en el programa los valores orientativos que se mostrarán como líneas límite en las gráficas.

Cada una de estas pantallas se compone de varias fases. Una vez el usuario pulsa el botón verde, se iniciará un temporizador de cinco segundos para dar margen al usuario a colocarse el dispositivo como se indica. Cuando termina el temporizador, se mide el ángulo actual, comienza otro temporizador de otros cinco segundos para dar tiempo al usuario a realizar el movimiento más amplio posible y el botón cambiará a rojo con un símbolo de Stop. Éste servirá para parar el ejercicio si el usuario lo desea por haberse colocado tarde el dispositivo o por cualquier otro motivo. Una vez termine el temporizador se recogerá el ángulo final, se calculará la diferencia y se mostrará el resultado al usuario junto con la diferencia del ángulo recogido en la última sesión realizada del ejercicio.

Ahora el botón cambiará a uno azul que servirá para repetir el ejercicio si el usuario no queda conforme con el resultado obtenido o si algo salió mal. Además, aparece un nuevo botón amarillo que nos llevará a la siguiente parte del ejercicio.

Esta parte será igual al anterior con la excepción de que el botón que aparece, al final, servirá para guardar el registro en la base de datos en el caso de ser un ejercicio rutinario o para pasar al siguiente ejercicio en el caso de estar realizándose los ejercicios de calibración.

A continuación se mostrarán cada una de estas fases en la aplicación:

Fase 1: Estado inicial.

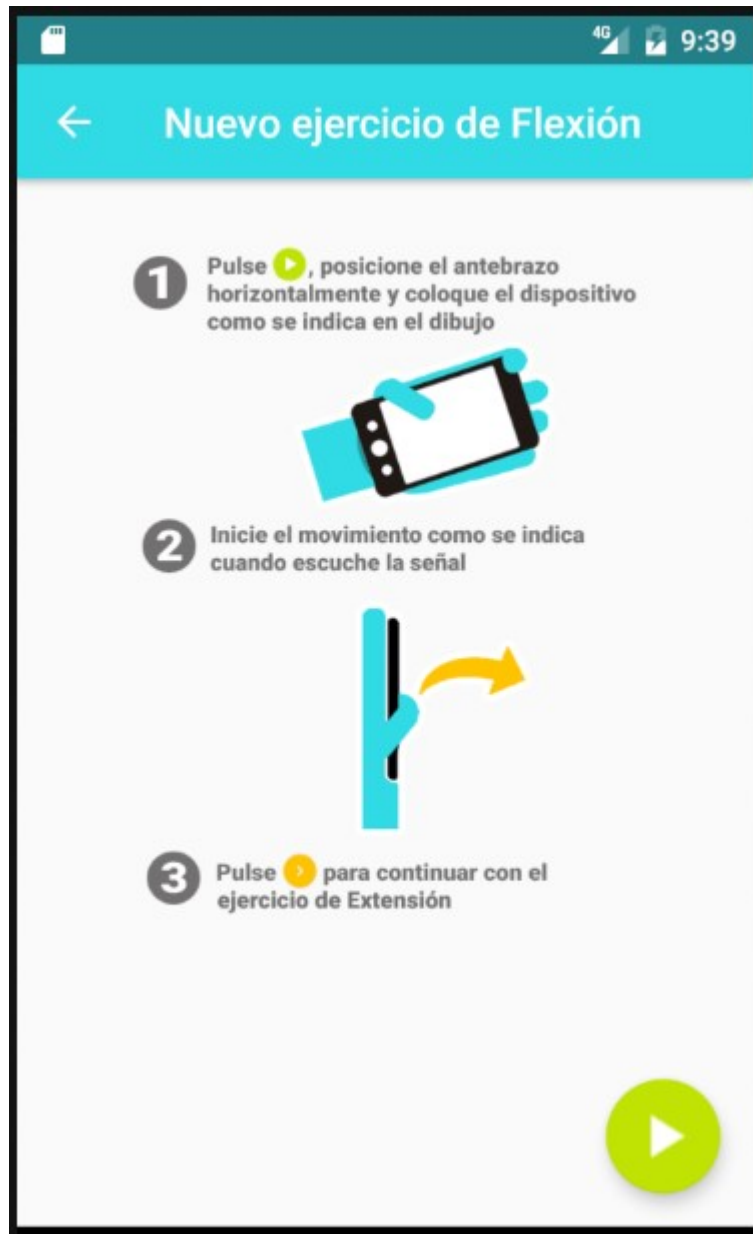


Figura 46: Pantalla de realización de ejercicio (estado inicial)

Fase 2: Comienzo del contador para el posicionamiento del dispositivo.

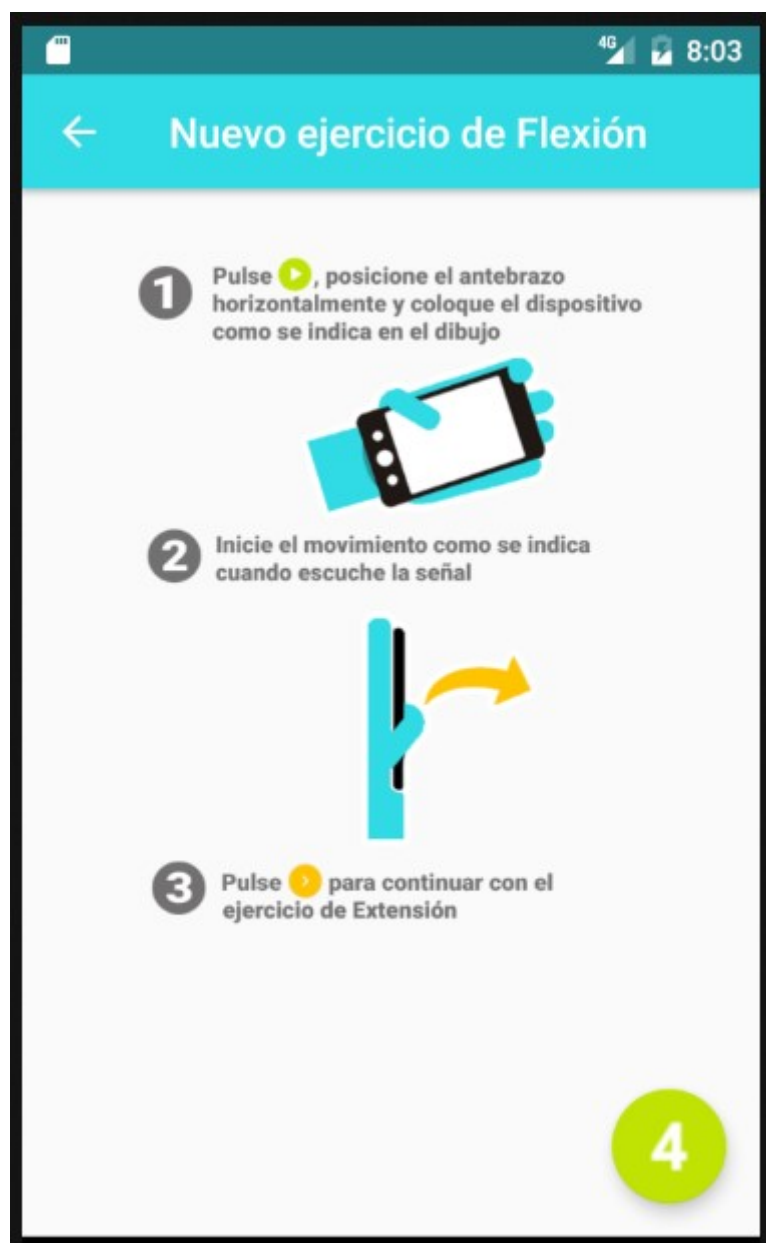


Figura 47: Pantalla de realización de ejercicio (comienzo contador)

Fase 3: Comienzo de la medición del ángulo.

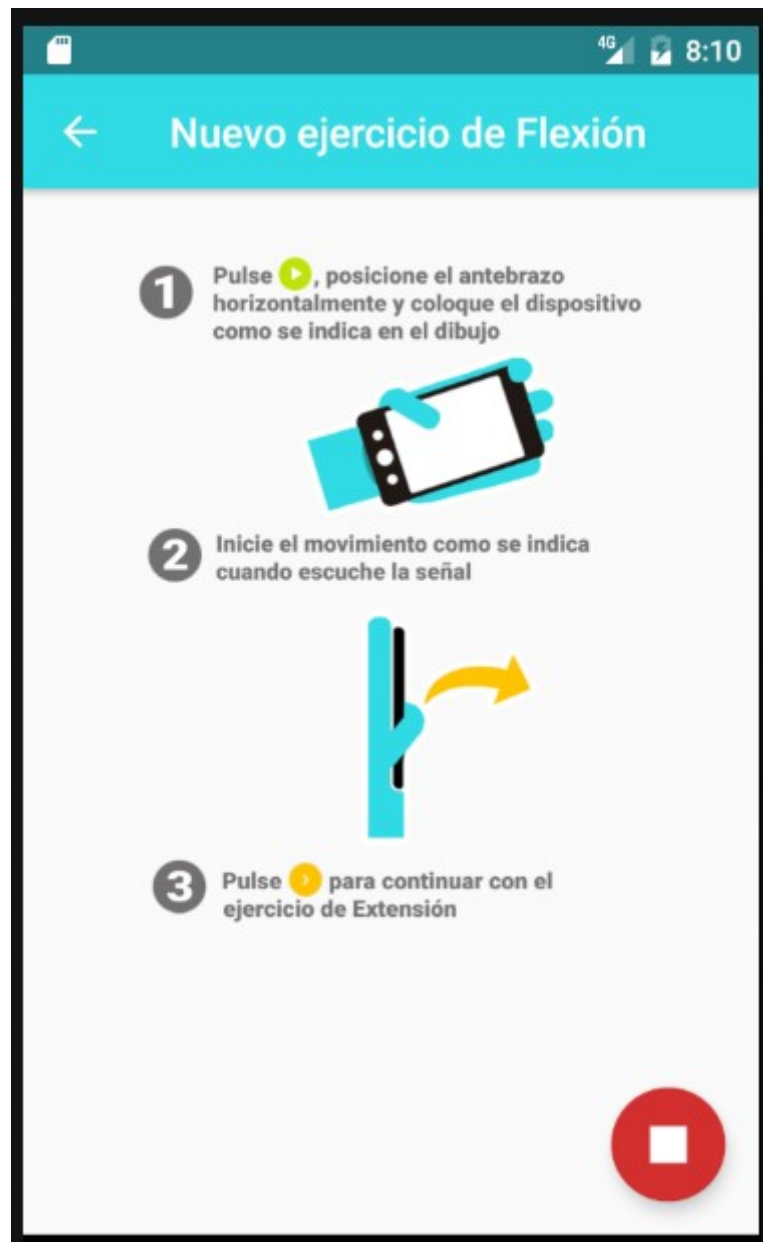


Figura 48: Pantalla de realización de ejercicio (comienzo medición)

Fase 4: Resultado ejercicio.



Figura 49: Pantalla de realización de ejercicio (resultado ejercicio)

Fase 5: Registro de los resultados.



Figura 50: Pantalla de realización de ejercicio (registrar resultados)

CAPÍTULO 6

IMPLEMENTACIÓN DE LA APLICACIÓN

En este capítulo se detallará la implementación del patrón MVP así como los algoritmos utilizados para el cálculo de los ángulos y se justificarán algunas de las decisiones tomadas para el desarrollo de la aplicación.

6.1 Implementación del patrón MVP

Como se ha comentado anteriormente se ha decido usar el patrón MVP. En las siguientes líneas se expondrán los principales conceptos de dicho patrón y cómo se ha utilizado para la implementación.

Ejemplo de los elementos que intervienen en el caso de uso de Añadir programa:

AddProgramsView

```
8  ↓ public interface AddProgramsView {  
9  
10 ↓     void checkName(Cursor c);  
11 }
```

Figura 51: Interfaz AddProgramsView

Esta será la interfaz que cuente con los métodos que usará el Presentador para comunicar los datos obtenidos del Modelo, a la Vista.

AddProgramsFragment

```
26 public class AddProgramFragment extends Fragment implements AddProgramsView {
27     private static final String ARG_PROGRAM_NAME = "arg_program_name";
28     private static final int REQUEST_INITIAL_EXERCISE = 1;
29
30     private FloatingActionButton mSaveButton;
31     private TextInputEditText mNameField;
32     private TextInputLayout mNameLabel;
33
34     private AddProgramsPresenter mAddProgramsPresenter;
35
36     public AddProgramFragment() {
37         // Required empty public constructor
38     }
39
40     public static AddProgramFragment newInstance(String programName) {
41         AddProgramFragment fragment = new AddProgramFragment();
42         Bundle args = new Bundle();
43         args.putString(ARG_PROGRAM_NAME, programName);
44         fragment.setArguments(args);
45         return fragment;
46     }
47
48     @Override
49     public void onCreate(Bundle savedInstanceState) {
50         super.onCreate(savedInstanceState);
51         mAddProgramsPresenter = new AddProgramsPresenter(getActivity(), this);
52     }
```

Figura 51: Clase AddProgramsFragment

En el fragmento creamos un atributo de clase de tipo AddProgramsPresenter que será la clase encargada de realizar toda la lógica Java y se comunicará con el Modelo para realizar consultas a base de datos. Crearemos una nueva instancia de este objeto en el método *onCreate(Bundle savedInstanceState)* clásico de Android para inicializar variables pasándole dos argumentos. El primero es el *Context* de la actividad en la que nos situamos, que será necesario para crear una nueva instancia de la clase del Modelo dentro del constructor del Presentador. En este caso, al encontrarnos en un fragmento, tendremos que pasarle *getActivity()* que pasará el *Context* de la actividad que sostiene y crea una instancia del fragmento en el que nos encontramos. El segundo argumento será *this* que le pasará el fragmento actual para poder inicializar en el Presentador el atributo de clase de tipo *AddProgramsView* como veremos a continuación.

```

public class AddProgramsPresenter {

    private AddProgramsView mAddProgramsView;
    private ProgramsDbHelper mProgramsDbHelper;

    public AddProgramsPresenter(Context context, AddProgramsView view) {
        mProgramsDbHelper = new ProgramsDbHelper(context);
        mAddProgramsView = view;
    }

    public void getProgramByName(String name) {

        mAddProgramsView.checkName(mProgramsDbHelper.getProgramByName(name));
    }

}

```

Figura 52: Clase AddProgramsFragment

Aquí puede verse como el Presentador consta de dos atributos de clase, uno será de tipo *ProgramsDbHelper* y el otro de tipo *AddProgramsView*. En el constructor del Presentador se crea una nueva instancia de la clase *ProgramsDbHelper* pasándole el *Context* que recibimos del fragmento y se asigna el valor del atributo de tipo *AddProgramsView* para poder llamar a los métodos de la interfaz para que la vista los implemente.

Cuando desde la vista se llama al método *getProgramByName(String name)* del Presentador, éste llama al método de la interfaz y le pasa el resultado de la consulta a base de datos realizada desde el Modelo.

Como puede observarse en la *Figura 51*, la clase implementa la interfaz *AddProgramsView*. En concreto, el único método que especifica, *checkName(Cursor c)* donde se tratarán los datos recibidos desde el Presentador.

```

75      @Override
76      public void checkName(Cursor c) {
77          boolean error = false;
78
79          String name = mNameField.getText().toString();
80
81          if (TextUtils.isEmpty(name)) {
82              mNameLabel.setError("Ingresa un valor");
83              error = true;
84          }
85
86          if(c != null && c.moveToNext()){
87              mNameLabel.setError("Ya existe un programa con ese nombre");
88              error = true;
89              c.close();
90          }
91
92
93          if (!error) {
94              showDialogReferenceExercise();
95          }
96      }

```

Figura 53: Clase AddProgramsFragment (método checkName)

El método obtiene el Cursor con la información necesaria pasada por el Presentador desde el Modelo y comprueba si el nombre introducido coincide con uno ya existente.

6.2 Algoritmo Sensor Fusion

A continuación se explicarán paso a paso todos los elementos utilizados para calcular la posición del dispositivo y los ángulos realizados en los ejercicios de rehabilitación

6.2.1 Puesta a punto e inicialización

Primero definiremos e inicializaremos los atributos de la clase y las constantes:

```
42     public class SensorFusion {
43
44         // Velocidades angulares del giroscopio
45         private float[] gyro = new float[3];
46
47         // Matriz de rotacion de los datos del giroscopio
48         private float[] gyroMatrix = new float[9];
49
50         // Ángulos de orientación de la matriz del giroscopio
51         private float[] gyroOrientation = new float[3];
52
53         // Vector del campo magnético
54         private float[] magnet = new float[3];
55
56         // Vector del acelerómetro
57         private float[] accel = new float[3];
58
59         // Ángulos de orientación del acelerómetro y del magnetómetro
60         private float[] accMagOrientation = new float[3];
61
62         // Ángulos de orientación finales de la fusión de sensores
63         private float[] fusedOrientation = new float[3];
64
65         private float[] selectedOrientation = fusedOrientation;
66
67         public enum Mode {
68             ACC_MAG, GYRO, FUSION
69         }
70
71         // Matriz de rotación basada en el acelerómetro y magnetómetro
72         private float[] rotationMatrix = new float[9];
73
74         public static final float EPSILON = 0.000000001f;
75         private static final float NS2S = 1.0f / 1000000000.0f;
76         private long timestamp;
77         private boolean initState = true;
78
79         public static final int TIME_CONSTANT = 30;
80         public static final float FILTER_COEFFICIENT = 0.98f;
81         private Timer fuseTimer = new Timer();
```

Figura 54: Inicialización de variables y definición de constantes

En el constructor inicializamos todos estos atributos y lanzamos el *Filtro complementario* del que hablaremos más adelante

```
83     public SensorFusion() {
84
85         gyroOrientation[0] = 0.0f;
86         gyroOrientation[1] = 0.0f;
87         gyroOrientation[2] = 0.0f;
88
89         // Inicializar la matriz del giroscopio con la matriz identidad
90         gyroMatrix[0] = 1.0f;
91         gyroMatrix[1] = 0.0f;
92         gyroMatrix[2] = 0.0f;
93         gyroMatrix[3] = 0.0f;
94         gyroMatrix[4] = 1.0f;
95         gyroMatrix[5] = 0.0f;
96         gyroMatrix[6] = 0.0f;
97         gyroMatrix[7] = 0.0f;
98         gyroMatrix[8] = 1.0f;
99
100        // Esperar un segundo hasta que los datos del giroscopio y el magnetómetro/acelerómetro
101        // se inicialicen. Después se programará la tarea de filtrado complementario
102
103        fuseTimer.scheduleAtFixedRate(new calculateFusedOrientationTask(),
104                                     1000, TIME_CONSTANT);
105
106    }
```

Figura 55: Constructor SensorFusion

6.2.2 Recogida y procesamiento de datos de sensores

La API de Android nos proporciona funciones muy útiles para obtener la orientación absoluta del acelerómetro y del magnetómetro. Esto es todo lo que necesitamos hacer para obtener la orientación del acelerómetro/magnetómetro:

```
152        // Calcula los ángulos de orientación de las salidas del acelerómetro y del magnetómetro
153        public void calculateAccMagOrientation() {
154            if (SensorManager.getRotationMatrix(rotationMatrix, null, accel, magnet)) {
155                SensorManager.getOrientation(rotationMatrix, accMagOrientation);
156            }
157        }
```

Figura 56: Método cálculo orientación acelerómetro/magnetómetro

Los datos del giroscopio requieren algún procedimiento adicional. La página de referencia de Android muestra cómo obtener un vector de rotación de los datos del giroscopio. Simplemente se ha reutilizado el código propuesto y se le han añadido algunos parámetros de modo que tenemos esto:


```

159 // Esta función se coge de la referencia de Android en:
160 // http://developer.android.com/reference/android/hardware/SensorEvent.html#values
161 // Calcula un vector de rotación a partir de los valores de la velocidad angular del giroscopio
162
163 private void getRotationVectorFromGyro(float[] gyroValues,
164                                       float[] deltaRotationVector,
165                                       float timeFactor) {
166     float[] normValues = new float[3];
167
168     // Calcula la velocidad angular de la muestra
169     float omegaMagnitude =
170         (float) Math.sqrt(gyroValues[0] * gyroValues[0] +
171                          gyroValues[1] * gyroValues[1] +
172                          gyroValues[2] * gyroValues[2]);
173
174     // Normaliza el vector de rotación si es lo suficientemente grande como para llegar al eje
175     if (omegaMagnitude > EPSILON) {
176         normValues[0] = gyroValues[0] / omegaMagnitude;
177         normValues[1] = gyroValues[1] / omegaMagnitude;
178         normValues[2] = gyroValues[2] / omegaMagnitude;
179     }
180
181     // Integra alrededor de este eje con la velocidad angular en el lapso
182     // con el fin de obtener una rotación delta de esta muestra durante el lapso
183     // Se convertirá esta representación ángulo-eje de la rotación delta
184     // en un cuaternión antes de pasarlo a una matriz de rotación.
185     float thetaOverTwo = omegaMagnitude * timeFactor;
186     float sinThetaOverTwo = (float) Math.sin(thetaOverTwo);
187     float cosThetaOverTwo = (float) Math.cos(thetaOverTwo);
188     deltaRotationVector[0] = sinThetaOverTwo * normValues[0];
189     deltaRotationVector[1] = sinThetaOverTwo * normValues[1];
190     deltaRotationVector[2] = sinThetaOverTwo * normValues[2];
191     deltaRotationVector[3] = cosThetaOverTwo;
192 }

```

Figura 57: Método cálculo orientación giroscopio

La función anterior crea un vector de rotación que es similar a un cuaternión. Expresa el intervalo de rotación del dispositivo entre la última medición y la actual del giroscopio. La velocidad de rotación se multiplica con el intervalo de tiempo (aquí es el parámetro *timeFactor*) que se pasó desde la última medición. Esta función es llamada después en el método *gyroFunction* actual para el procesamiento de datos del giroscopio. Aquí es donde se añaden los intervalos de rotación del giroscopio a la orientación basada en el giro absoluto. Pero como se tienen matrices de rotación en lugar de ángulos, esto no se puede hacer simplemente agregando los intervalos de rotación. Se necesitan aplicar los intervalos de rotación mediante multiplicación matricial:

```

194 // Esta función realiza la integración de los datos del giroscopio.
195 // Escribe la orientación basada en el giroscopio en gyroOrientation.
196 public void gyroFunction(SensorEvent event) {
197     // No empezar hasta que la primera orientación se haya conseguido
198     if (accMagOrientation == null)
199         return;
200
201     // inicialización de la matriz de rotación basada en el giroscopio
202     if (initState) {
203         float[] initMatrix;
204         initMatrix = getRotationMatrixFromOrientation(accMagOrientation);
205         float[] test = new float[3];
206         SensorManager.getOrientation(initMatrix, test);
207         gyroMatrix = matrixMultiplication(gyroMatrix, initMatrix);
208         initState = false;
209     }
210
211     // copia de los nuevos valores del giroscopio en el array
212     // convierte los datos en crudo en un vector de rotación
213     float[] deltaVector = new float[4];
214     if (timestamp != 0) {
215         final float dT = (event.timestamp - timestamp) * NS2S;
216         System.arraycopy(event.values, 0, gyro, 0, 3);
217         getRotationVectorFromGyro(gyro, deltaVector, dT / 2.0f);
218     }
219
220     // con la medida hecha, guarda el tiempo actual para el siguiente intervalo
221     timestamp = event.timestamp;
222
223     // convierte el vector de rotación en una matriz de rotación
224     float[] deltaMatrix = new float[9];
225     SensorManager.getRotationMatrixFromVector(deltaMatrix, deltaVector);
226
227     // aplicar el nuevo intervalo de rotación en la matriz de rotación basada en el giroscopio
228     gyroMatrix = matrixMultiplication(gyroMatrix, deltaMatrix);
229
230     // conseguir la orientación basada en el giroscopio de la matriz de rotación
231     SensorManager.getOrientation(gyroMatrix, gyroOrientation);
232 }

```

Figura 58: Método `gyroFunction(SensorEvent event)`

Los datos del giroscopio no se procesan hasta que los ángulos de orientación del acelerómetro y del magnetómetro estén disponibles (en la variable `accMagOrientation`). Estos datos son necesarios como orientación inicial para los datos del giroscopio. De lo contrario, nuestra matriz de orientación contendrá valores indefinidos. La orientación actual del dispositivo y el vector de rotación del giroscopio son transformados en una matriz de rotación.

La variable `gyroMatrix` es la orientación total calculada a partir de todas las mediciones del giroscopio procesadas hasta ahora. La variable `deltaMatrix` contiene el último intervalo de rotación que debe aplicarse a la `gyroMatrix` en el siguiente paso. Esto se hace multiplicando `gyroMatrix` por `deltaMatrix`. Esto es equivalente a la rotación de `gyroMatrix` sobre `deltaMatrix`. El método de multiplicación de la matriz se describe más adelante.

El vector de rotación se puede convertir en una matriz llamando a la función de conversión `getRotationMatrixFromVector` del `SensorManager`. Para convertir los ángulos de orientación en una matriz de rotación se utiliza esta función de conversión:

```
234     private float[] getRotationMatrixFromOrientation(float[] o) {
235         float[] xM = new float[9];
236         float[] yM = new float[9];
237         float[] zM = new float[9];
238
239         float sinX = (float) Math.sin(o[1]);
240         float cosX = (float) Math.cos(o[1]);
241         float sinY = (float) Math.sin(o[2]);
242         float cosY = (float) Math.cos(o[2]);
243         float sinZ = (float) Math.sin(o[0]);
244         float cosZ = (float) Math.cos(o[0]);
245
246         // rotación sobre el eje x (pitch)
247         xM[0] = 1.0f; xM[1] = 0.0f; xM[2] = 0.0f;
248         xM[3] = 0.0f; xM[4] = cosX; xM[5] = sinX;
249         xM[6] = 0.0f; xM[7] = -sinX; xM[8] = cosX;
250
251         // rotación sobre el eje y (roll)
252         yM[0] = cosY; yM[1] = 0.0f; yM[2] = sinY;
253         yM[3] = 0.0f; yM[4] = 1.0f; yM[5] = 0.0f;
254         yM[6] = -sinY; yM[7] = 0.0f; yM[8] = cosY;
255
256         // rotación sobre el eje z (azimuth)
257         zM[0] = cosZ; zM[1] = sinZ; zM[2] = 0.0f;
258         zM[3] = -sinZ; zM[4] = cosZ; zM[5] = 0.0f;
259         zM[6] = 0.0f; zM[7] = 0.0f; zM[8] = 1.0f;
260
261         // El orden de rotación es y, x, z (roll, pitch, azimuth)
262         float[] resultMatrix = matrixMultiplication(xM, yM);
263         resultMatrix = matrixMultiplication(zM, resultMatrix);
264         return resultMatrix;
265     }
```

Figura 59: Método conversión ángulos a matriz

A pesar de que esta función no es óptima y puede ser mejorada en términos de rendimiento, para nuestro caso servirá. Básicamente, crea una matriz de rotación por cada eje y multiplica las matrices en el orden correcto (y, x, z en nuestro caso).

Esta es la función para la multiplicación de matrices:

```
267 @ private float[] matrixMultiplication(float[] A, float[] B) {  
268     float[] result = new float[9];  
269  
270     result[0] = A[0] * B[0] + A[1] * B[3] + A[2] * B[6];  
271     result[1] = A[0] * B[1] + A[1] * B[4] + A[2] * B[7];  
272     result[2] = A[0] * B[2] + A[1] * B[5] + A[2] * B[8];  
273  
274     result[3] = A[3] * B[0] + A[4] * B[3] + A[5] * B[6];  
275     result[4] = A[3] * B[1] + A[4] * B[4] + A[5] * B[7];  
276     result[5] = A[3] * B[2] + A[4] * B[5] + A[5] * B[8];  
277  
278     result[6] = A[6] * B[0] + A[7] * B[3] + A[8] * B[6];  
279     result[7] = A[6] * B[1] + A[7] * B[4] + A[8] * B[7];  
280     result[8] = A[6] * B[2] + A[7] * B[5] + A[8] * B[8];  
281  
282     return result;  
283 }
```

Figura 60: Método multiplicación de matrices

6.2.3 El filtro complementario

Por último, pero no menos importante, se puede implementar el filtro complementario. Para tener más control sobre su salida, se ejecuta el filtrado en un hilo temporizado separado. La calidad de la señal del sensor depende en gran medida de la frecuencia de muestreo, es decir, la frecuencia con la que el método de filtro se llama por segundo. Es por eso que se ponen todos los cálculos en un *TimerTask* y se definen más tarde el intervalo de tiempo entre cada llamada.

```
285 class calculateFusedOrientationTask extends TimerTask {
286     public void run() {
287         float oneMinusCoeff = 1.0f - FILTER_COEFFICIENT;
288
289         /*
290          * Arreglo para el problema de transición para 179° <--> -179°
291          * Comprueba si uno de los dos ángulos de orientación (giroscopio o acelerómetro/magnetómetro) es negativo mientras que el otro es positivo.
292          * Si es así, añadir 360° (2 * math.PI) al valor negativo, realizar la fusión de sensores, y eliminar el 360° del resultado
293          * si esto es mayor que 180°. Esto estabiliza la salida en los casos de transición-de-positivo-a-negativo.
294          */
295
296         // azimuth
297         if (gyroOrientation[0] < -0.5 * Math.PI && accMagOrientation[0] > 0.0) {
298             fusedOrientation[0] = (float) (FILTER_COEFFICIENT * (gyroOrientation[0] + 2.0 * Math.PI) + oneMinusCoeff * accMagOrientation[0]);
299             fusedOrientation[0] -= (fusedOrientation[0] > Math.PI) ? 2.0 * Math.PI : 0;
300         } else if (accMagOrientation[0] < -0.5 * Math.PI && gyroOrientation[0] > 0.0) {
301             fusedOrientation[0] = (float) (FILTER_COEFFICIENT * gyroOrientation[0] + oneMinusCoeff * (accMagOrientation[0] + 2.0 * Math.PI));
302             fusedOrientation[0] -= (fusedOrientation[0] > Math.PI) ? 2.0 * Math.PI : 0;
303         } else {
304             fusedOrientation[0] = FILTER_COEFFICIENT * gyroOrientation[0] + oneMinusCoeff * accMagOrientation[0];
305         }
306
307         // pitch
308         if (gyroOrientation[1] < -0.5 * Math.PI && accMagOrientation[1] > 0.0) {
309             fusedOrientation[1] = (float) (FILTER_COEFFICIENT * (gyroOrientation[1] + 2.0 * Math.PI) + oneMinusCoeff * accMagOrientation[1]);
310             fusedOrientation[1] -= (fusedOrientation[1] > Math.PI) ? 2.0 * Math.PI : 0;
311         } else if (accMagOrientation[1] < -0.5 * Math.PI && gyroOrientation[1] > 0.0) {
312             fusedOrientation[1] = (float) (FILTER_COEFFICIENT * gyroOrientation[1] + oneMinusCoeff * (accMagOrientation[1] + 2.0 * Math.PI));
313             fusedOrientation[1] -= (fusedOrientation[1] > Math.PI) ? 2.0 * Math.PI : 0;
314         } else {
315             fusedOrientation[1] = FILTER_COEFFICIENT * gyroOrientation[1] + oneMinusCoeff * accMagOrientation[1];
316         }
317
318         // roll
319         if (gyroOrientation[2] < -0.5 * Math.PI && accMagOrientation[2] > 0.0) {
320             fusedOrientation[2] = (float) (FILTER_COEFFICIENT * (gyroOrientation[2] + 2.0 * Math.PI) + oneMinusCoeff * accMagOrientation[2]);
321             fusedOrientation[2] -= (fusedOrientation[2] > Math.PI) ? 2.0 * Math.PI : 0;
322         } else if (accMagOrientation[2] < -0.5 * Math.PI && gyroOrientation[2] > 0.0) {
323             fusedOrientation[2] = (float) (FILTER_COEFFICIENT * gyroOrientation[2] + oneMinusCoeff * (accMagOrientation[2] + 2.0 * Math.PI));
324             fusedOrientation[2] -= (fusedOrientation[2] > Math.PI) ? 2.0 * Math.PI : 0;
325         } else {
326             fusedOrientation[2] = FILTER_COEFFICIENT * gyroOrientation[2] + oneMinusCoeff * accMagOrientation[2];
327         }
328
329         // sobrescribir la matriz del giroscopio y de orientación con fusión de orientación para compensar el gyro drift.
330         gyroMatrix = getRotationMatrixFromOrientation(fusedOrientation);
331         System.arraycopy(fusedOrientation, 0, gyroOrientation, 0, 3);
332
333         // update sensor output in GUI
334         //mainHandler.post(updateOrientationDisplayTask);
335     }
336 }
337 }
```

Figura 61: Método filtro complementario

Como puede observarse, sobrescribimos la orientación basada en el giroscopio y la matriz de rotación en cada pasada del filtro. Esto reemplaza la orientación del giroscopio con los datos del sensor “mejorados” y elimina el *gyro drift*.

Por último, destacar que surge un problema de transición para el paso de los ángulos 179° y -179°. Se comprueba si uno de los dos ángulos de orientación (giroscopio o acelerómetro/magnetómetro) es negativo mientras que el otro es positivo. Si es así, hay que añadir 360° ($2 * \text{math.PI}$) al valor negativo, se realiza la fusión de sensores, y se elimina el

360° del resultado si esto es mayor que 180°. Esto estabiliza la salida en los casos de transición-de-positivo-a-negativo.

6.2.4 Uso de la clase SensorFusion

Una vez descrita la clase SensorFusion, veremos donde se usan sus métodos. Puesto que esta clase se encarga de calcular la posición del dispositivo, ésta se requerirá únicamente en las vistas referentes a la realización de ejercicios. Veremos el ejemplo en el caso de la vista del ejercicio de Flexión-Extensión.

```
31     public FlexExtExerciseFragmentPresenter(Context context,  
32                                           FlexExtExerciseFragmentView view,  
33                                           SensorManager sensorM) {  
34         mProgramsDbHelper = new ProgramsDbHelper(context);  
35         mFlexExtExerciseFragmentView = view;  
36         sensorManager = sensorM;  
37  
38         registerSensorManagerListeners();  
39  
40         DecimalFormatSymbols symbols = DecimalFormatSymbols.getInstance();  
41         symbols.setDecimalSeparator('.');  
42         d = new DecimalFormat("#.##", symbols);  
43         d.setMaximumFractionDigits(2);  
44         d.setMinimumFractionDigits(2);  
45  
46         sensorFusion = new SensorFusion();  
47         sensorFusion.setMode(SensorFusion.Mode.FUSION);  
48     }
```

Figura 62: Constructor Presentador ejercicio Flexión-Extensión

Se recibe el contexto y el fragmento como se dijo en el capítulo anterior y además se recibe un objeto `SensorManager` para dar valor al atributo de clase `sensorManager` que utilizaremos para registrar los listeners como se ve a continuación.

```
122 public void registerSensorManagerListeners() {
123     sensorManager.registerListener(this,
124         sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER),
125         SensorManager.SENSOR_DELAY_FASTEST);
126
127     sensorManager.registerListener(this,
128         sensorManager.getDefaultSensor(Sensor.TYPE_GYROSCOPE),
129         SensorManager.SENSOR_DELAY_FASTEST);
130
131     sensorManager.registerListener(this,
132         sensorManager.getDefaultSensor(Sensor.TYPE_MAGNETIC_FIELD),
133         SensorManager.SENSOR_DELAY_FASTEST);
134 }
```

Figura 63: Método `registerSensorManagerListeners()`

Una vez registrados los listeners, la Vista únicamente tendrá que llamar a estos tres métodos. El primero comprobará si el dispositivo está en la posición correcta. El segundo devolverá el valor del ángulo inicial y el último, nos dirá el valor del ángulo final.

```
87 public boolean checkRoll() {
88     return ((Math.abs(sensorFusion.getRoll()) > 80) && (Math.abs(sensorFusion.getRoll()) < 100));
89 }
90
91 public void getInitDegree() {
92     mFlexExtExerciseFragmentView.setInitDegree(sensorFusion.getAzimuth());
93 }
94
95 public void getFinalDegree() {
96     mFlexExtExerciseFragmentView.setFinalDegree(sensorFusion.getAzimuth());
97 }
```

Figura 64: Métodos que acceden a la clase `SensorFusion`

6.3 Decisiones tomadas para el desarrollo de la aplicación

En este apartado explicaremos algunas de las decisiones tomadas a la hora de desarrollar la aplicación.

6.3.1 Señales sonoras para la realización de los ejercicios

En primera instancia se pensó realizar la interacción del usuario con el dispositivo mediante voz para la realización de los ejercicios, pero a la hora de consultar métodos para poder realizar dicha característica en Android se vio que este procedimiento no tenía un tiempo de respuesta adecuado ya que el tiempo que se tardaba en procesar lo que el usuario decía por el micro hacía la realización de los ejercicios muy densa y demasiado larga.

Se planteó otra alternativa como era una sincronización vía web en la que el usuario realizaría los ejercicios con un PC al lado de modo que, teniendo el dispositivo en la mano para realizar los ejercicios, con la otra podría interactuar con la página web para empezar, terminar y repetir los ejercicios. También se descartó esta idea por el aumento de la complejidad del problema, por una parte y de la infraestructura del proyecto, por otra.

Finalmente se decidió optar por utilizar contadores presentes en Android acompañados de señales sonoras para que el usuario obtenga información de la realización del ejercicio sin tener que mirar la pantalla puesto que, en el momento de los ejercicios, el dispositivo solo necesita servir de instrumento de medida. De este modo, dejamos un tiempo (cinco segundos en nuestro caso) para que el usuario coloque el dispositivo de forma correcta avisándole tres segundos antes de comenzar la medición con pitidos tenues seguidos de un pitido más largo que indicará el comienzo. Una vez transcurren otros cinco segundos, el ejercicio termina con un pitido intenso igual que el del comienzo del ejercicio y se muestra el ángulo obtenido por pantalla.

Cabe destacar que si el usuario no tiene el dispositivo en la posición correcta, el ejercicio no comienza y se avisa al usuario con un pitido de error.

6.3.2 Posicionamiento del dispositivo para los distintos ejercicios

Para el posicionamiento del dispositivo se ha optado por ofrecer la realización de los ejercicios con el móvil en vertical debido a que es más sencillo de sostener sin que se resbale o se caiga al realizar las sesiones de ejercicios. Además, al ser la misma posición para cada ejercicio, el usuario se acostumbrará más rápidamente a la utilización de la aplicación.

De esta manera, la comprobación del posicionamiento del dispositivo por software se realizará comprobando siempre el mismo ángulo del espacio.

6.4 Estructura del proyecto Android

En este apartado veremos cómo se estructura la aplicación en el IDE de Android Studio.

Aquí vemos el árbol general de la aplicación, donde el código fuente se encuentra bajo la carpeta *java*. Más adelante veremos las clases que contiene cada paquete.

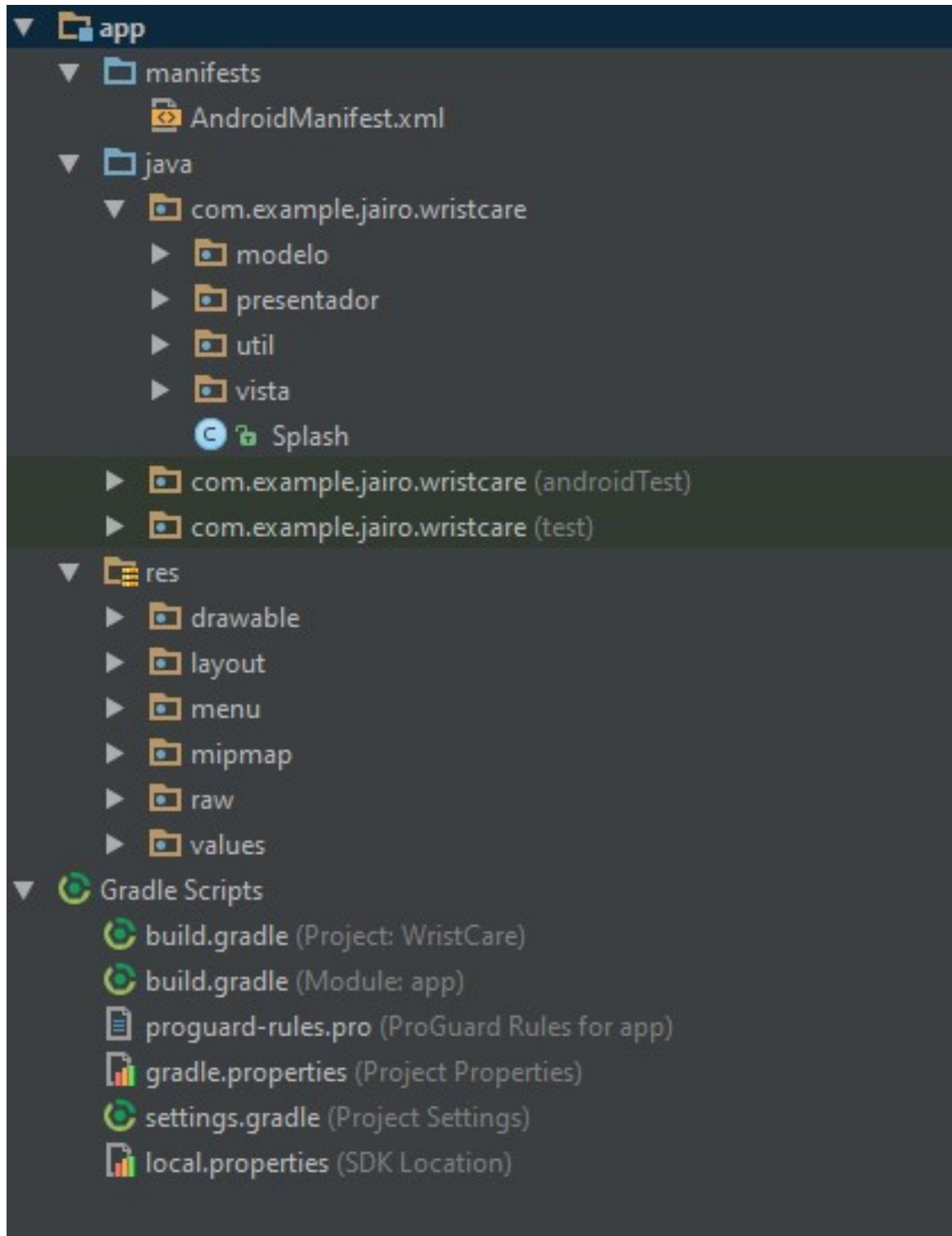


Figura 65: Visión general estructura del proyecto

Paquete modelo:

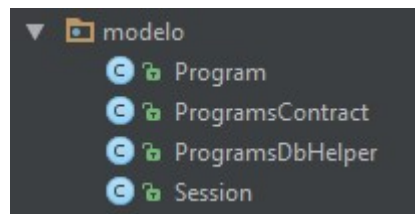


Figura 66: Paquete modelo

Paquete vista:

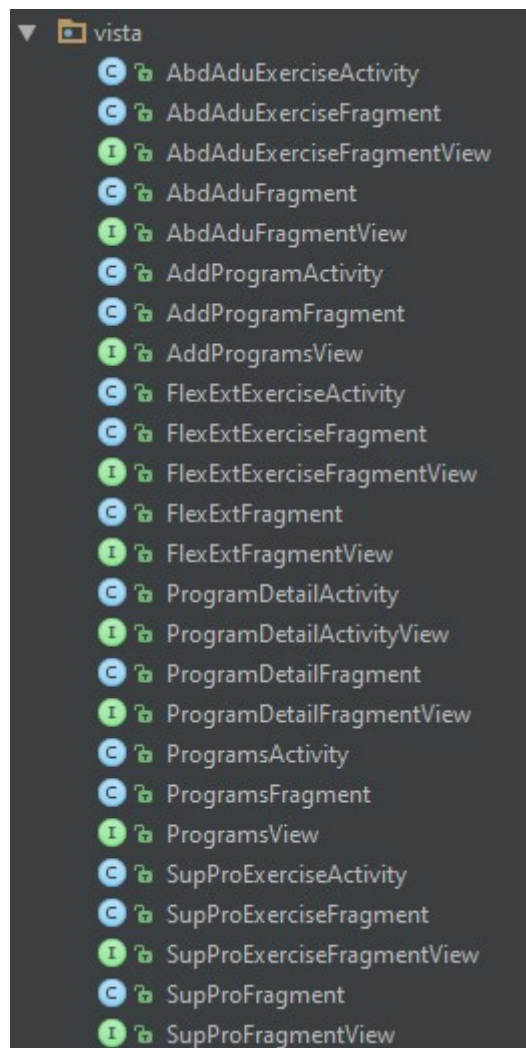


Figura 67: Paquete vista

Paquete presentador:

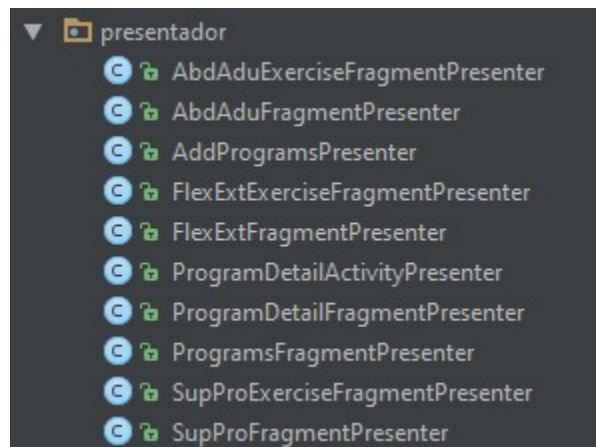


Figura 68: Paquete presentador

Paquete util:

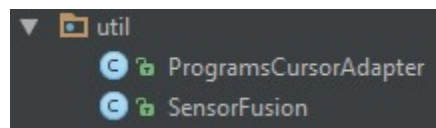


Figura 69: Paquete util

CAPÍTULO 7

PRUEBAS REALIZADAS

7.1 Pruebas unitarias

Estas pruebas se han ido realizando a medida que se desarrollaba el código de la aplicación de forma que cada método ha sido probado durante el periodo de implementación.

A medida que se desarrollaba el código se han tratado de recorrer todos los posibles caminos de ejecución para los métodos y se han probado todas las operaciones aritméticas.

En el caso de esta aplicación, los métodos más importantes son los referidos al cálculo de ángulos. Para cada ejercicio, se ha simulado un movimiento de 90 grados para los tres ángulos del espacio. Se ha comprobado que para cada uno de estos ángulos el resultado obtenido por el método era aproximado al esperado.

7.2 Pruebas del sistema

Las pruebas de caja negra están enfocadas a comprobar que los requisitos funcionales se han cumplido.

A continuación, se presenta un resumen de las diferentes pruebas de caja negra realizadas. Estas pruebas se han diseñado en base a los casos de uso del sistema.

Prueba	Resultado esperado	Resultado de la prueba
Creación de programa	El programa queda creado	OK
Creación de programa con nombre vacío	Se muestra un mensaje de error	OK
Creación de programa con nombre ya existente	Se muestra un mensaje de error	OK
Realizar ejercicio (para los tres tipos)	Los dos ángulos quedan registrados	OK
Realizar ejercicio (solo primera fase)	No se registra ningún ángulo	OK
Mostrar ángulo al realizar ejercicio (para cualquiera)	Se muestra el ángulo obtenido para los ejercicio	OK
Mostrar diferencia ángulo sesión anterior (para cualquier ejercicio)	Se muestra la diferencia calculada correctamente	Diferencia incorrecta
Consultar mejora media (para los tres ejercicios)	Se muestra la mejora media calculada correctamente	OK
Finalizar programa	El programa queda en estado Finalizado	OK
Si programa cerrado no mostrar botón nuevo ejer. (para los tres ejercicios)	No se muestran los botones de realizar nuevo ejercicio	OK
Mostrar detalles programa	Se muestran los datos correctamente	OK
Mostrar fecha finalización al cerrar programa	Se muestra la fecha del día de hoy al cerrar programa	OK
Mostrar imagen empty-state lista de programas	Se muestra la imagen en la pantalla principal	OK
Mostrar imagen empty-state gráficas ejercicios (para los tres ejercicios)	Se muestra la imagen para todos los ejercicios	OK
Eliminar programa	El programa queda eliminado	No elimina prog. de la lista

Tabla 42: Pruebas del sistema

A continuación, se desarrollan los casos de prueba *Mostrar diferencia ángulo sesión anterior* y *Eliminar programa*.

CP-007	Mostrar diferencia ángulo sesión anterior
Descripción	Mostrar la diferencia respecto al ángulo del mismo ejercicio de la sesión anterior.
Acción	Mostrar dicho ángulo al realizar un ejercicio.
Resultado obtenido	Habiendo realizado un ejercicio y al realizar el siguiente, la diferencia del ángulo respecto de la última sesión del segundo no se calcula bien.
Resultado	Error.
Solución	Al realizar un ejercicio, no se tuvo en cuenta que se genera una nueva sesión, por lo tanto, al realizar cualquier otro ejercicio, la diferencia siempre se calculaba sobre 0° puesto que la sesión se crea con el ángulo del ejercicio realizado primero. Por tanto, se han añadido estructuras de control para evitar esto y calcular la diferencia del ángulo respecto de la última sesión correctamente.

Tabla 43: CP-015 Eliminar programa

CP-015	Eliminar programa
Descripción	Eliminar programa del sistema.
Acción	Seleccionar la opción <i>Eliminar programa</i> en el menú de la pantalla de detalles del programa.
Resultado obtenido	Al regresar a la pantalla de inicio con la lista de programas, se sigue mostrando el programa eliminado.
Resultado	Error.
Solución	Se ha observado que el método que recibe el resultado de la actividad que finaliza y cede de nuevo el control a la pantalla principal no llamaba, en todos los casos, al método que se encarga de poblar la lista. De modo que se ha incluido la llamada a dicho método para todos los casos, quedando ahora la lista actualizada al eliminar un programa.

Tabla 44: CP-015 Eliminar programa

7.3 Pruebas experimentales

En esta sección se detallarán las pruebas realizadas para comprobar la fiabilidad de los resultados proporcionados por la aplicación.

Para cada uno de los ángulos del espacio (azimuth, pitch y roll) se han hecho pruebas con un sujeto al que se le ha dicho que realice diez movimientos por cada componente de cada ángulo (positiva y negativa). El proceso que se ha seguido es el siguiente: El sujeto se coloca el dispositivo en la mano (fijado a la palma con una goma o con una de las nuevas carcasas de móvil que incluyen un anillo que se utilizan para hacer *selfies*) y se le dice que realice movimientos en los ángulos indicados de forma aleatoria y anotamos los resultados ofrecidos por la aplicación y los ángulos que marca el goniómetro.

7.3.1 Flexión-Extensión (Azimuth)

Para este ángulo, se han realizado diez movimientos de Flexión y otros tantos de Extensión. Para medir este ángulo, se fija una de las dos partes del goniómetro a un plano horizontal y con la otra parte se acompaña el movimiento de la muñeca del sujeto en la realización del ejercicio. De este modo:



Figura 65: Procedimiento para pruebas del ángulo azimuth

Obteniendo los siguientes resultados:

Goniómetro	Dispositivo	Error absoluto	Error relativo
10	9,76	0,24	0,024
10	9,07	0,93	0,093
12	13,3	1,3	0,108333333
16	16,21	0,21	0,013125
19	20,38	1,38	0,072631579
20	20,66	0,66	0,033
23	24,76	1,76	0,076521739
26	25,81	0,19	0,007307692
28	30,74	2,74	0,097857143
28	29,94	1,94	0,069285714
29	30,26	1,26	0,043448276
32	31,23	0,77	0,0240625
36	37,72	1,72	0,047777778
36	36,2	0,2	0,005555556
40	39,21	0,79	0,01975
41	40,34	0,66	0,016097561
43	41,42	1,58	0,036744186
43	44,56	1,56	0,03627907
45	44,68	0,32	0,007111111
50	50,47	0,47	0,0094
Error relativo medio:			4,21%

Tabla 45: Pruebas experimentales azimuth

7.3.2 Abducción-Aducción (Pitch)

Para este ángulo, se han realizado diez movimientos de Abducción y otros tantos de Aducción. Para medir este ángulo, se fija una de las dos partes del goniómetro a un plano vertical y con la otra parte se acompaña el movimiento de la muñeca del sujeto en la realización del ejercicio. De este modo:



Figura 66: Procedimiento para pruebas del ángulo pitch

Obteniendo los siguientes resultados:

Goniómetro	Dispositivo	Error absoluto	Error relativo
8	7,45	0,55	0,06875
10	10,47	0,47	0,047
12	12,43	0,43	0,035833333
12	11,15	0,85	0,070833333
15	15,51	0,51	0,034
18	16,85	1,15	0,063888889
19	20,02	1,02	0,053684211
19	19,49	0,49	0,025789474
19	18,92	0,08	0,004210526
20	20,67	0,67	0,0335
22	23,14	1,14	0,051818182
25	25,06	0,06	0,0024
26	25,73	0,27	0,010384615
27	28,56	1,56	0,057777778
28	28,94	0,94	0,033571429
30	29,55	0,45	0,015
30	31,19	1,19	0,039666667
30	30,69	0,69	0,023
35	36,62	1,62	0,046285714
35	35,75	0,75	0,021428571
Error relativo medio:			3,7%

Tabla 46: Pruebas experimentales pitch

7.3.3 Supinación-Pronación (Roll)

Para este ángulo, se han realizado diez movimientos de Supinación y otros tantos de Pronación. Para medir este ángulo, se fija una de las dos partes del goniómetro a un plano vertical y con la otra parte se acompaña el movimiento de la muñeca del sujeto en la realización del ejercicio. De este modo:



Figura 67: Procedimiento para pruebas del ángulo roll

Obteniendo los siguientes resultados:

Goniómetro	Dispositivo	Error absoluto	Error relativo
8	7,54	0,46	0,0575
21	20,1	0,9	0,042857143
24	25,91	1,91	0,079583333
26	26,17	0,17	0,006538462
30	30,89	0,89	0,029666667
30	28,69	1,31	0,043666667
32	32,85	0,85	0,0265625
32	33,37	1,37	0,0428125
32	32,41	0,41	0,0128125
35	35,73	0,73	0,020857143
39	38,18	0,82	0,021025641
42	41,37	0,63	0,015
42	41,77	0,23	0,00547619
45	44,17	0,83	0,018444444
46	48,04	2,04	0,044347826
51	51,56	0,56	0,010980392
52	53,18	1,18	0,022692308
58	56,55	1,45	0,025
60	60,88	0,88	0,014666667
60	59,69	0,31	0,005166667
Error relativo medio:			2,73%

Tabla 47: Pruebas experimentales roll

7.3.4 Conclusiones de las pruebas experimentales

En el futuro se planea realizar más pruebas y que las realizadas para este TFG son limitadas y el número de muestras no es suficiente para realizar conclusiones fiables. Sería recomendable que estas pruebas fueran realizadas por un especialista para mayor fiabilidad de las mismas. No obstante, con las pruebas realizadas hemos podido apreciar que el error relativo medio obtenido para cada ángulo no solo no es elevado si no que es mucho menor que el error planteado en la fase de análisis como requisito no funcional que fijaba dicho error en un 10% lo que hace que esta aplicación se pueda utilizar para los fines propuestos.

CAPÍTULO 8

CONCLUSIONES

8.1 Objetivos alcanzados

Con la finalización del TFG se han cumplido todos los objetivos planteados. Los resultados han sido los siguientes:

1. Se ha desarrollado una aplicación que facilita el trabajo a los terapeutas a la hora de llevar un control del proceso de rehabilitación de sus pacientes con lesión de muñeca de forma más rápida y eficiente que la forma tradicional.
2. La aplicación permite que pacientes pueda llevar un control de su lesión sin necesidad de tener que visitar a su terapeuta.
3. Se ha conseguido una interfaz sencilla, moderna y minimalista.
4. Se ha desarrollado un algoritmo preciso para los movimientos básicos de la muñeca.
5. La aplicación estará disponible en Google Play por lo que será de fácil instalación y acceso para los usuarios.

8.2 Líneas de trabajo futuras

La aplicación desarrollada podría ser mejorada en un futuro incluyendo por ejemplo las siguientes funcionalidades:

- Podría implementarse una funcionalidad para mandar los programas de rehabilitación a plataformas del estilo de Google Drive para poder enviar datos a terapeutas, por ejemplo.
- Podrían añadirse nuevos ejercicios más complejos como el de rotación.
- Podría ampliarse el control del proceso de rehabilitación a otras articulaciones como el codo, el hombro o la rodilla.

BIBLIOGRAFÍA

Gutiérrez, Manuel J. “¿Cuáles son y para qué sirven los sensores de nuestros Android?”. Consultado 15 de Febrero de 2017, en

<https://elandroidelibre.elespanol.com/2014/07/cuales-son-y-para-que-sirven-los-sensores-de-nuestros-android.html>

Poland, Ashley. “The Average Lifespan of a Desktop PC”. Consultado 20 de Febrero de 2017, en

<https://www.techwalla.com/articles/the-average-lifespan-of-a-desktop-pc>

Tiwari, Rahul. “What is the average lifespan of a cell phone?”. Consultado 20 de Febrero de 2017, en

<https://www.quora.com/Mobile-Devices-What-is-the-average-lifespan-of-a-cell-phone>

Pascual, Juan Antonio. “El sueldo de los programadores, al descubierto”. Consultado 20 de Febrero de 2017, en

<http://computerhoy.com/noticias/software/sueldo-programadores-descubierto-31147>

Leiva, Antonio. “Usando MVP e inversión de dependencias para abstraernos del framework en Android”. Consultado 2 de Mayo de 2017, en

<https://www.genbetadev.com/paradigmas-de-programacion/usando-mvp-e-inversion-de-dependencias-para-abstraernos-del-framework-en-android>

“Material Design Guidelines”. Consultado 13 de Mayo de 2017 en

<https://material.io/guidelines/>

Revelo, James. “Tutorial De Bases De Datos SQLite En Aplicaciones Android.” Consultado 18 de Mayo en

<http://www.hermosaprogramacion.com/2014/10/android-sqlite-bases-de-datos/>

Lawitzki, Paul. “Android Sensor Fusion Tutorial”. Consultado 20 de Mayo de 2017 en

<https://www.codeproject.com/Articles/729759/Android-Sensor-Fusion-Tutorial>

Tamada, Ravi. "Android Material Design working with Tabs". Consultado 24 de Mayo de 2017 en

<http://www.androidhive.info/2015/09/android-material-design-working-with-tabs/>

Jay, Phil. "Using MPAndroidChart". Consultado 30 de Mayo de 2017 en

<https://github.com/PhilJay/MPAndroidChart>

Kapandji, Adalbert. "Fisiología Articular Tomo 1 6ed." Madrid: Panamericana

ANEXO A

CONTENIDO DEL CD

La distribución de contenidos del soporte digital que acompaña a la memoria entregada es la siguiente:

- **WristCare:** directorio que contiene el código fuente de todo el software desarrollado.
- **wristcare.apk:** archivo ejecutable de la aplicación con el que se podrá instalar en los dispositivos deseados.
- **memoria.pdf:** es el archivo digital en formato pdf que contiene todo lo expuesto en esta memoria.