



Universidad de Valladolid

Escuela de Ingeniería Informática

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Mención en Computación

**Aplicación para visualizar la musculatura
del suelo pélvico**

Autor:

Pablo Martín Gómez



Universidad de Valladolid

Escuela de Ingeniería Informática

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Mención en Computación

**Aplicación para visualizar la musculatura
del suelo pélvico**

Autor:

Pablo Martín Gómez

Tutores:

Mario Corrales Astorgano

Alfonso Bahillo Martínez

A todos los que me han acompañado durante este camino.

Agradecimientos

Quiero agradecer su apoyo constante a mis tutores, Mario Corrales Astorgano y Alfonso Bahillo Martínez, su orientación y correcciones han sido imprescindibles durante todo el proceso. También quiero dar las gracias a mi familia, amigos y compañeros por haber hecho este camino más llevadero.

Resumen

La salud del suelo pélvico es fundamental para la funcionalidad del cuerpo humano, sobre todo para las mujeres. Sin embargo, algunos factores como el embarazo, la edad o ejercicios de impacto, pueden causar debilidades en este grupo muscular. Existen ciertos ejercicios conocidos como ejercicios de Kegel, consistentes en una serie de contracciones y relajaciones, para ayudar a resolver este problema. Dispositivos como una sonda vaginal que recoja los datos de las presiones ejercidas sobre los músculos durante el entrenamiento, pueden ser de gran utilidad para el correcto desarrollo de los ejercicios. El objetivo de este TFG es proporcionar una herramienta visual e interactiva que permita interpretar los datos recogidos por una sonda vaginal para facilitar la realización de estos ejercicios.

Palabras clave: Suelo pélvico, sonda vaginal, modelo tridimensional, Kegel, Unity, salud.

Abstract

The health of the pelvic floor is fundamental to the functionality of the human body, especially for women. However, some factors such as pregnancy, age or impact exercises, can cause weaknesses in this muscle group. There are certain exercises known as Kegel exercises, consisting of a series of contractions and relaxations, to help solve this problem. Devices such as a vaginal probe that collects data on the pressures exerted on the muscles during training can be very useful for the correct development of the exercises. The objective of this TFG is to provide a visual and interactive tool that allows the interpretation of the data collected by a vaginal probe to facilitate the performance of these exercises.

Key words: Pelvic floor, vaginal probe, three-dimensional model, Kegel, Unity, health.

Índice general

1. Introducción	1
1.1. Contexto	1
1.2. Motivación	3
1.3. Estado de la cuestión	3
1.3.1. Ejercicios de Kegel	3
1.3.2. Emy	3
1.3.3. Easy Kegel	6
1.3.4. Ejercicios Kegel - Fortalecer el suelo pélvico	7
1.4. Tecnología empleada	7
1.4.1. Unity	7
1.4.2. Blender	8
1.4.3. Android	8
1.4.4. Java	9
1.4.5. Android Studio	11
1.4.6. Sonda vaginal	12
1.5. Objetivos	14
1.6. Metodología	14
1.7. Estructura de la memoria	14
2. Planificación	17
2.1. Planificación inicial	17
2.2. Análisis de riesgos	18
2.3. Entorno de trabajo	19
2.4. Estimación de costes	21
2.4.1. Costes humanos	21
2.4.2. Costes de hardware	22
2.4.3. Costes de software	23
2.4.4. Costes del espacio de trabajo	23
2.4.5. Costes totales	23
2.5. Planificación final	24
3. Descripción de las iteraciones	25
3.1. Iteración 1	25
3.2. Iteración 2	33

3.3. Iteración 3	34
3.4. Iteración 4	34
4. Estado final de la aplicación	37
4.1. Análisis	37
4.1.1. Análisis de requisitos	37
4.1.2. Modelo de dominio	38
4.1.3. Historias de usuario	39
4.1.4. Diagrama de actividad	41
4.2. Diseño	42
4.2.1. Diagrama de paquetes	42
4.2.2. Diagrama de despliegue	43
4.2.3. Estructura del proyecto	43
4.2.4. Patrones de diseño	44
4.2.5. Interfaces	45
5. Conclusiones	47
5.1. Trabajo futuro	48
Apéndices	51
A. Manual de integración	51
A.1. Exportar aplicación Unity a Android	51
A.2. Integrar aplicación Unity en otra aplicación Android Nativo	51
A.3. Comunicación entre Android y Unity	62
B. Manual de despliegue	67
C. Manual de uso	69
Bibliografía	73

Índice de figuras

1.1. Suelo pélvico. Vista externa[3]	2
1.2. Suelo pélvico. Vista interna[4]	2
1.3. Ejercicios de Kegel	4
1.4. Emy	5
1.5. Easy Kegel	6
1.6. Ejercicios Kegel - Fortalecer el suelo pélvico	7
1.7. Logo de Unity[8]	8
1.8. Logo de Blender[9]	8
1.9. Capas de Android[10]	10
1.10. Logo de Android[10]	10
1.11. Logo de Java[11]	11
1.12. Logo de Android Studio[12]	11
1.13. Sonda vaginal	12
1.14. Datos recogidos por la sonda vaginal en tiempo real	13
1.15. Metodología iterativa e incremental[13]	15
2.1. Horas estimadas iniciales por iteración	18
2.2. Comparación entre planificación inicial y final	24
3.1. Modelo original	26
3.2. Modelo importado	26
3.3. Modelo con texturas reconstruidas	27
3.4. Esquema anatómico del suelo pélvico	27
3.5. Separación de músculos por cota de malla en Blender	28
3.6. Sensor 22	30
3.7. Sensor 23	30
3.8. Sensor 24	31
3.9. Sensor 25	31
3.10. Sensor 31	32
3.11. Sensor 32	32
3.12. Fichero de datos de simulación	35
3.13. Simulación de ejercicios (1)	36
3.14. Simulación de ejercicios (2)	36
4.1. Modelo de dominio	39

4.2. Diagrama de actividad	41
4.3. Diagrama de paquetes Android	42
4.4. Diagrama de paquetes Unity	42
4.5. Diagrama de despliegue	43
4.6. Estructura del proyecto en Android	45
4.7. Estructura del proyecto en Unity	45
4.8. Interfaz Android	46
4.9. Interfaz Unity	46
A.1. Menú Build Settings en Unity	52
A.2. Menú Player Settings en Unity	53
A.3. Menú Configuration en Player Settings	54
A.4. New Project en Android Studio	55
A.5. Script settings.gradle	55
A.6. Script build.gradle(Module:app)	56
A.7. Librería Unity exportada	57
A.8. Script gradle.properties(Project properties)	58
A.9. Script build.gradle(module: unityLibrary)	58
A.10.Script strings.xml	59
A.11.Script build.gradle(Module: unityLibrary)	59
A.12.Pantalla aplicación Android	61
A.13.Pantalla aplicación Unity	61

Índice de tablas

1.1. Propiedades de los sensores de presión	12
2.1. Descripciones y probabilidades de los riesgos del proyecto	19
2.2. Medidas de mitigación y reducción de los riesgos del proyecto	20
2.3. Especificaciones del portátil de trabajo	21
2.4. Especificaciones del smartphone	21
2.5. Costes humanos	22
2.6. Costes de hardware	22
2.7. Costes del espacio de trabajo	23
2.8. Coste total	23
3.1. Tabla de identificadores y músculos	28
3.2. Relación de músculos y sensores	29
4.1. Requisitos funcionales	37
4.2. Requisitos no funcionales	38
4.3. Requisitos de información	38
4.4. Historia de usuario HU01	40
4.5. Historia de usuario HU02	40
4.6. Historia de usuario HU03	40
4.7. Historia de usuario HU04	40

Capítulo 1

Introducción

1.1. Contexto

El suelo pélvico es un conjunto de músculos y ligamentos que cierra la parte inferior de la pelvis (Figuras 1.1 y 1.2). Es un soporte fundamental para nuestra salud, pudiendo así acarrear problemas de estabilidad, incontinencia o disfunción sexual si se experimenta debilidad en este área. Algunas causas por las que se puede perder fuerza en esta zona son el embarazo, el parto, la menopausia o la obesidad, por eso es de especial interés en las mujeres.

Históricamente ha sido un tema tabú, pero hasta el 30 % de las mujeres sufren las consecuencias de un suelo pélvico débil en algún momento de su vida, especialmente en la etapa adulta. En respuesta a este problema, se han desarrollado una variedad de ejercicios y técnicas conocidos como ejercicios de Kegel [1] que permiten fortalecer y cuidar esta musculatura. Estos ejercicios consisten en distintas contracciones y relajaciones de los músculos de esta zona, pero la propiocepción del suelo pélvico, es decir, la percepción inconsciente de estos músculos independiente de su visión [2], es fundamental para un tratamiento efectivo y esto puede resultar complicado ya que dicho grupo muscular no es fácilmente identificable ni visible.

Existe tecnología avanzada, utilizada por fisioterapeutas que permiten guiar en el proceso de fortalecer esta musculatura. Un ejemplo de esta, es una sonda vaginal con distintos sensores de presión que recoge información de la fuerza o presión ejercida por los músculos y ligamentos durante la realización del ejercicio.

En este Trabajo de Fin de Grado se pretende construir una herramienta de apoyo para la aplicación existente en el Hospital Universitario Príncipe de Asturias (Alcalá de Henares, Madrid). Actualmente, se dispone de una aplicación Android que recoge la presión en tiempo real que se ejerce a través de los sensores de una sonda vaginal, mediante conexión Bluetooth. El objetivo de este trabajo es desarrollar un modelo interactivo en 3 dimensiones que permita, en tiempo real, ver qué músculos se están ejercitando y cuanta presión se está ejerciendo con ellos. Se trata de una función de gran utilidad, pues como se mencionaba anteriormente, la identificación de los músculos puede no ser sencilla, y el uso de un modelo táctil que muestre las zonas que se están activando ayudará a visualizarlos, así como

a mejorar la realización de los ejercicios.

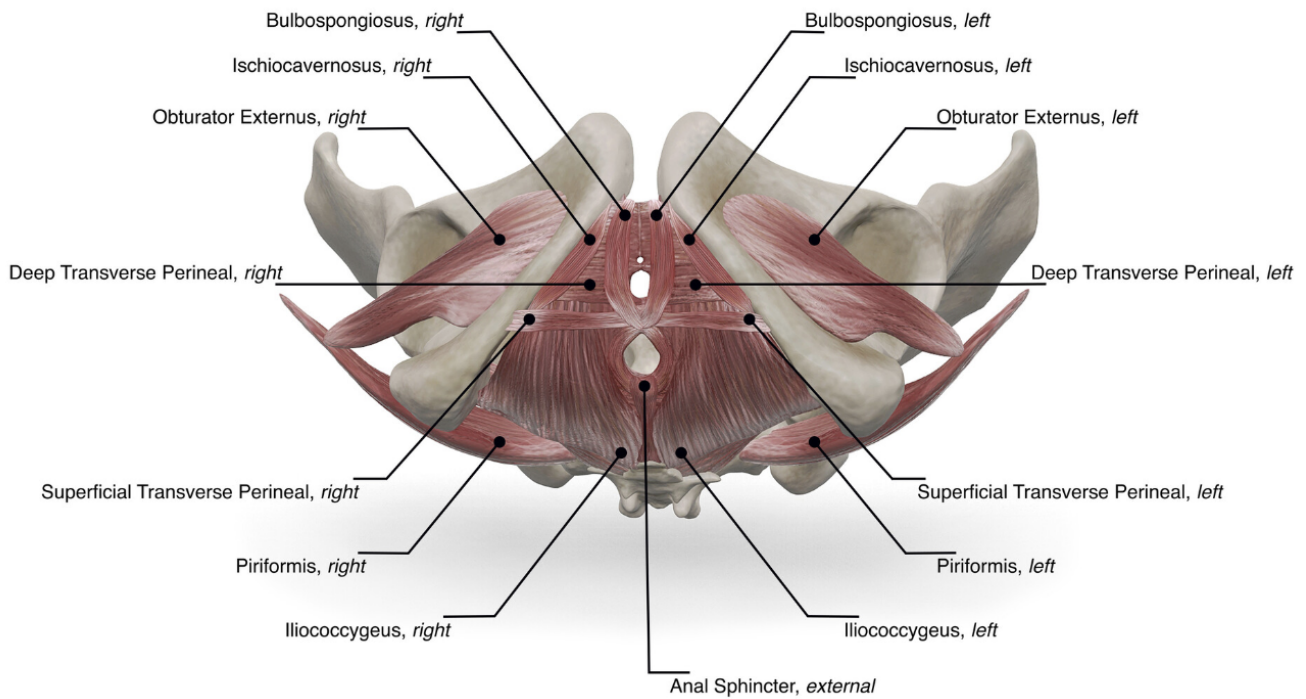


Figura 1.1: Suelo pélvico. Vista externa[3]

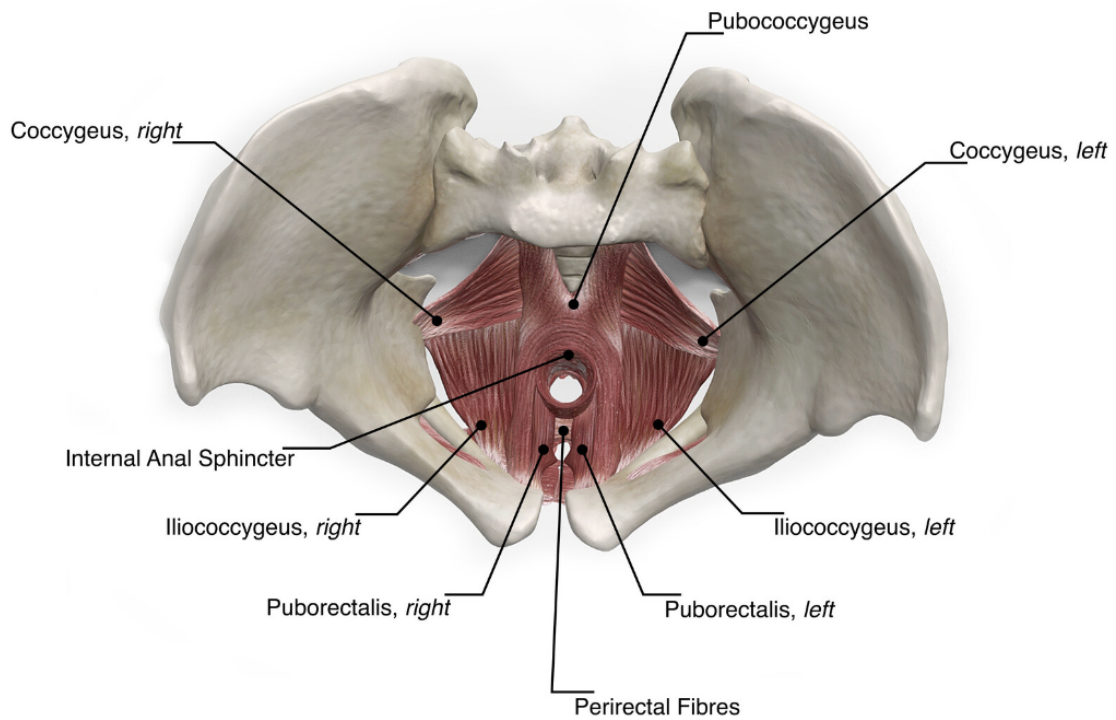


Figura 1.2: Suelo pélvico. Vista interna[4]

1.2. Motivación

Como se mencionaba en la sección anterior, la falta de retroalimentación visual o táctil puede hacer que la realización de los ejercicios no sea tarea sencilla y limite su eficacia a largo plazo.

Por eso, en este trabajo, se desarrolla una herramienta que proporcione retroalimentación sobre la actividad muscular en tiempo real mientras se realicen los ejercicios de Kegel. Los datos extraídos por la sonda vaginal con sensores de presión y su posterior representación en un modelo tridimensional ofrecen una solución prometedora. Se prevé que este sistema además de mejorar la precisión y la efectividad del entrenamiento, aumente la adherencia al tratamiento de los pacientes.

A mayores, los profesionales de la salud que trabajan en este campo pueden encontrar de interés esta aplicación, pues les proporcionará una forma objetiva de evaluar la función del suelo pélvico y el progreso del tratamiento, mejorando así, los resultados de sus pacientes.

1.3. Estado de la cuestión

En la siguiente sección se describirán brevemente las principales aplicaciones que hay actualmente en el mercado para fortalecer el suelo pélvico y una comparativa con lo que se plantea conseguir en este trabajo.

1.3.1. Ejercicios de Kegel

Ejercicios de Kegel [1] (Figura 1.3) es una aplicación desarrollada por Olson Applications Ltd que cuenta con 10 sesiones diferentes de ejercicios según el nivel que tengas. Mediante indicaciones sonoras o por vibración, indica cuando tienes que apretar o relajar los músculos durante el ejercicio. A mayores permite establecer recordatorios para realizar las sesiones.

Esta aplicación no cuenta con una respuesta en tiempo real de cómo estás realizando el ejercicio, aunque puede ser una buena alternativa gratuita para practicar en casa por tu cuenta una vez que ya esté dominada la técnica.

1.3.2. Emy

Emy [5] (Figura 1.4) es una de las aplicaciones más potentes del mercado. Desarrollada por Fizimed, esta app está pensada para ser utilizada junto con una sonda vaginal que se vende por separado. También se puede utilizar sin esta y realizar los ejercicios sin recibir respuestas en tiempo real. En su versión con sonda, permite entrenar con una serie de juegos que te muestran tu progreso de forma intuitiva, pero no cuenta con ninguna representación de los músculos que están siendo ejercitados.

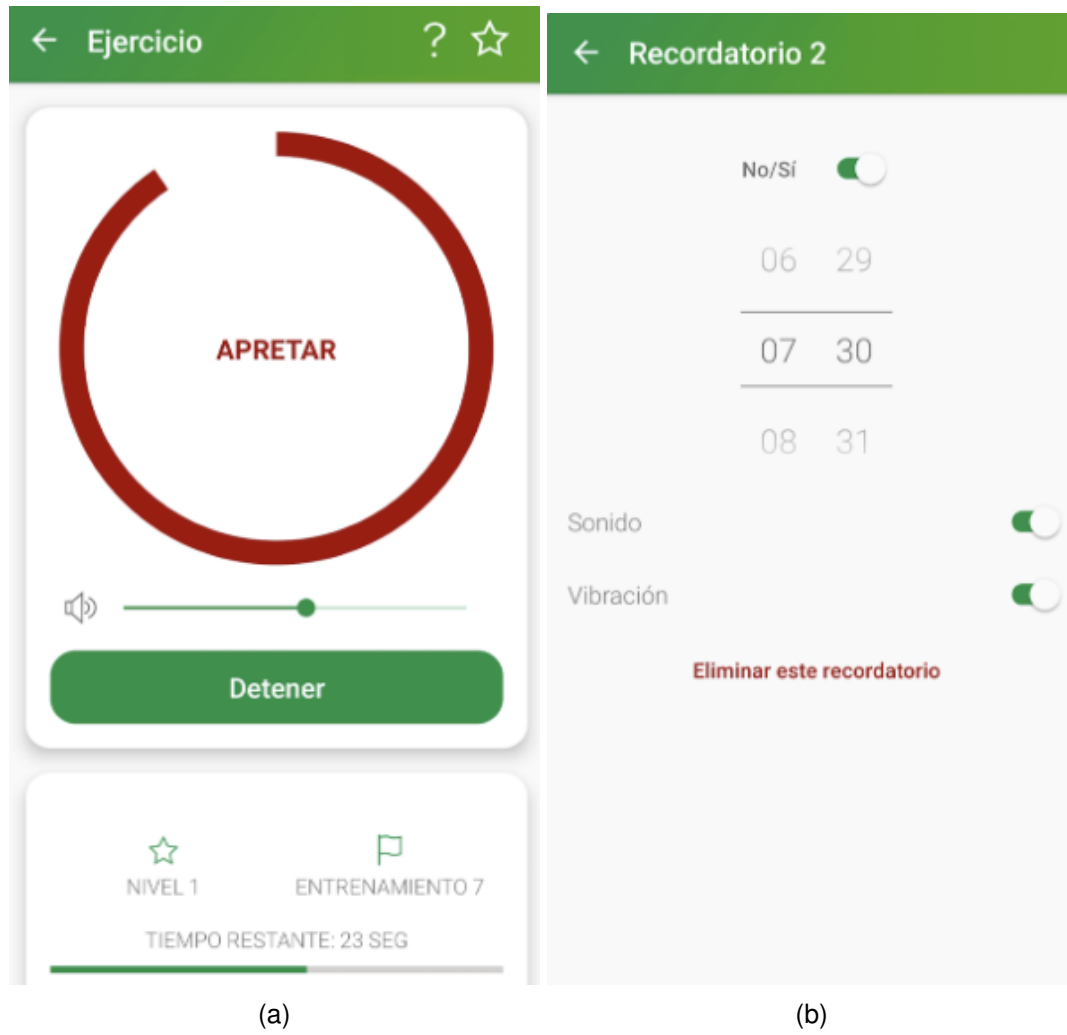


Figura 1.3: Ejercicios de Kegel

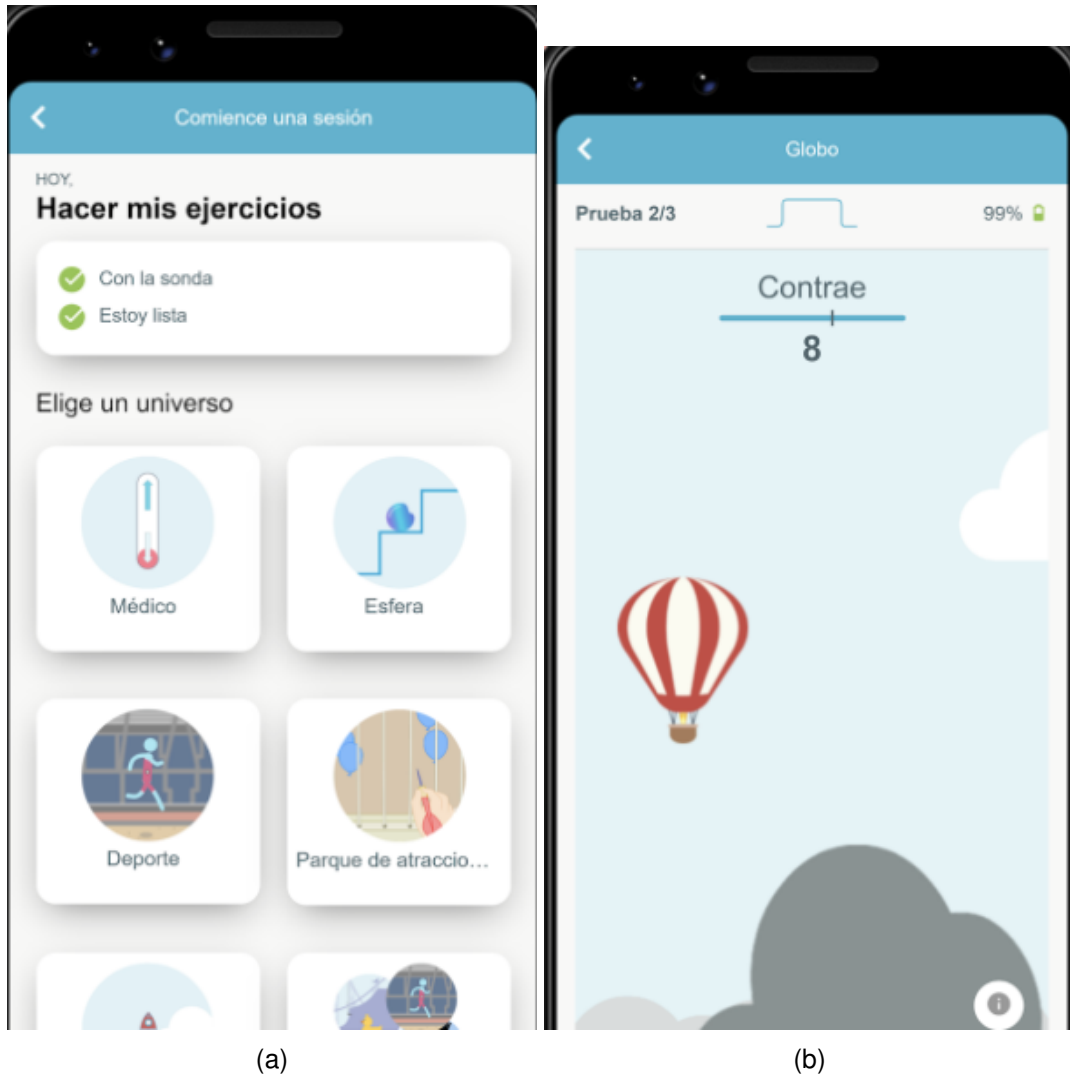


Figura 1.4: Emy

1.3.3. Easy Kegel

Easy Kegel [6] (Figura 1.5) es una aplicación desarrollada por Dream Bear Ltd y permite optimizar tus sesiones cambiando según tus necesidades todos los parámetros de tu entrenamiento (el número de repeticiones, el tiempo de contracción o de descanso...). Es una aplicación más destinada a usuarios que ya tienen un conocimiento previo acerca de cómo ejercitar la musculatura y que pueden personalizar sus propios entrenamientos. No cuenta con respuestas en tiempo real ni con representaciones interactivas del suelo pélvico.

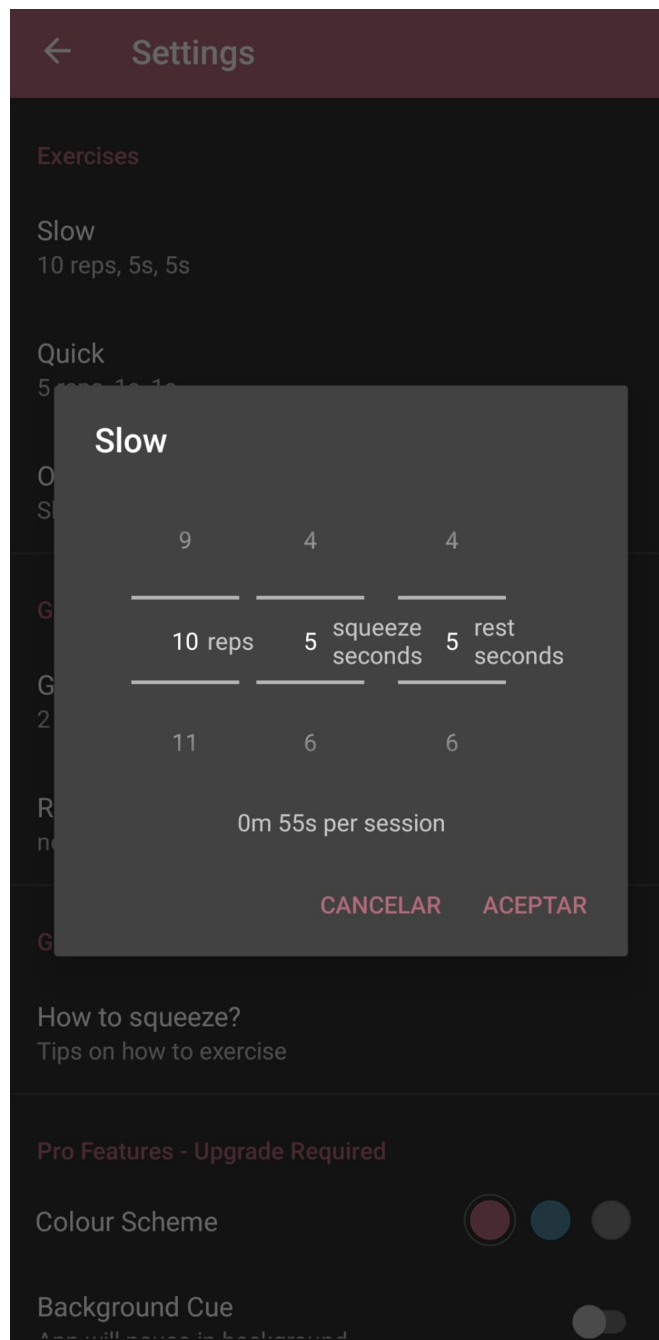


Figura 1.5: Easy Kegel

1.3.4. Ejercicios Kegel - Fortalecer el suelo pélvico

Ejercicios Kegel - Fortalecer el suelo pélvico [7] (Figura 1.6) es una aplicación desarrollada por Steveloper en la que además de encontrar ejercicios para el suelo pélvico tanto para hombres como para mujeres, puedes encontrar otros entrenamientos para fortalecer el abdomen o los glúteos. Al ser una aplicación menos especializada, no cuenta con una respuesta en tiempo real ni con representaciones de ningún tipo de la zona ejercitada.

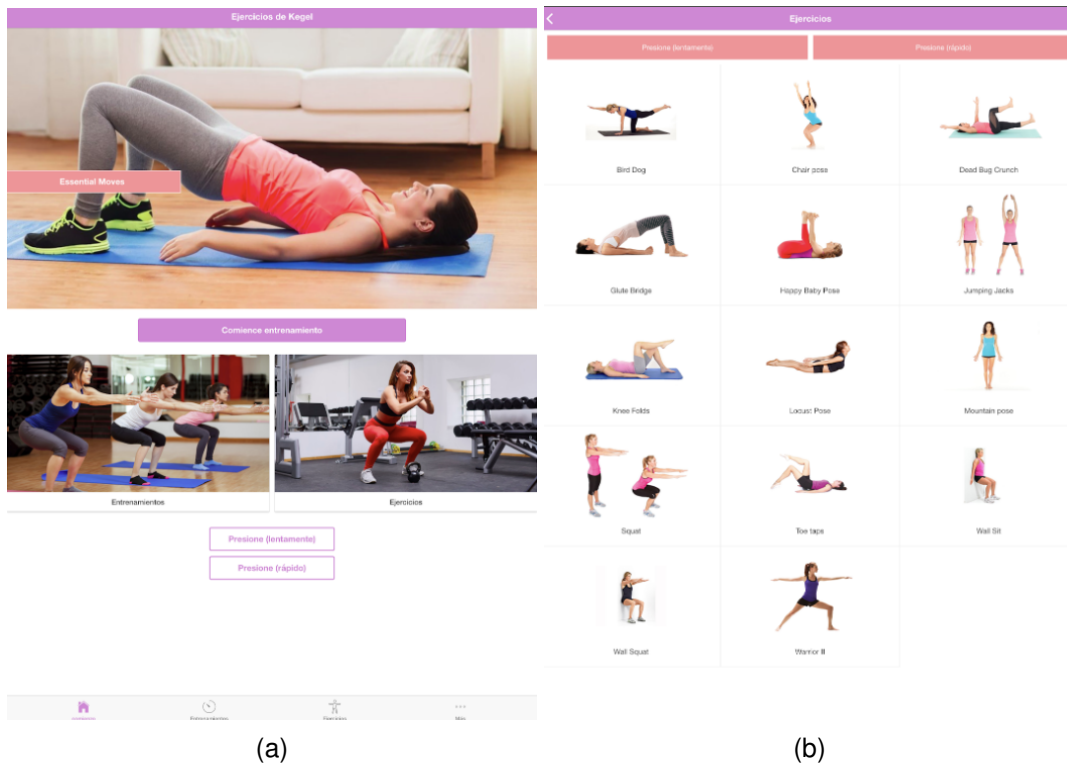


Figura 1.6: Ejercicios Kegel - Fortalecer el suelo pélvico

1.4. Tecnología empleada

En esta sección se describen las tecnologías y herramientas empleadas en la realización de este trabajo. Por un lado se utilizarán Unity y Blender para el desarrollo del modelo tridimensional del suelo pélvico y por otro lado se empleará Android Studio con lenguaje de programación Java para integrar este modelo en una aplicación Android sencilla que simule la ya existente del Hospital Príncipe de Asturias. A mayores, se describe brevemente la sonda vaginal con la que se realizarán las pruebas.

1.4.1. Unity

Unity [8] es un motor de videojuegos multiplataforma creado por Unity Technologies, aunque también se utiliza para crear modelos tridimensionales. Está disponible para su desarrollo en Windows, Mac OS y Linux.

El motor gráfico utilizado es OpenGL que permite incluir sombras dinámicas, muchos tipos de mapeados (de relieve, de reflejos, por paralaje...) y efectos de post-procesamiento de pantalla completa. El control de las funciones de los distintos objetos se lleva a cabo mediante scripts en C#, que deriva su sintaxis de C y C++.

Unity permite crear aplicaciones tanto para sistemas operativos (Windows, Linux y Mac OS) como para Android o Play Station. En este caso su compatibilidad con Android fue fundamental para escoger esta herramienta, ya que el objetivo final es utilizar el modelo en una app de Android. Además, cuenta con simuladores que permiten probar tu aplicación en distintos dispositivos sin necesidad de llegar a construirla del todo. También se puede probar en tu propio teléfono utilizando su aplicación móvil Unity Remote y un cable USB.



Figura 1.7: Logo de Unity[8]

1.4.2. Blender

Blender [9] es un programa multiplataforma, dedicado al modelado, renderizado, iluminación y animación de gráficos en tres dimensiones. Es compatible con todas las versiones de Windows, Mac OS y Linux. Aunque se centra en el modelado y la animación, también tiene herramientas que permiten editar audio y vídeo. El control de varias tareas se realiza a través de lenguaje Python por medio de scripts asociados a los diversos objetos, aunque también permite el uso de un sistema con ladrillos lógicos con una interfaz muy intuitiva.

Sus herramientas avanzadas a la hora de modelar y su compatibilidad con Unity han hecho que sea un programa muy necesario para el trabajo realizado.



Figura 1.8: Logo de Blender[9]

1.4.3. Android

Android [10] es un sistema operativo de código abierto y basado en el kernel de Linux. Desarrollado por Google, es compatible con todo tipo de dispositivos desde móviles o tabletas hasta televisiones inteligentes o sistemas integrados en automóviles. Android tiene una arquitectura por capas (Figura 1.9) que colabora para brindar un entorno de ejecución robusto y flexible. A continuación se explica cada una de ellas:

- **Kernel Linux:** Esta capa proporciona la base del sistema operativo. Se encarga de gestionar los recursos del hardware, como la memoria, los procesos o el sistema de archivos. Gracias a Linux, Android consigue estabilidad, seguridad y soporte.
- **Runtime de Android:** Esta capa proporciona un entorno de ejecución para las aplicaciones Android mediante la máquina virtual ART (Android RunTime) que compila el código de las aplicaciones en un formato ejecutable optimizado para mejorar el rendimiento y la eficiencia de la ejecución de las aplicaciones.
- **Bibliotecas:** Las bibliotecas de Android, escritas principalmente en C y C++, proporcionan funciones y utilidades fundamentales para el desarrollo de aplicaciones. Algunos ejemplos de bibliotecas son las que permiten procesar audio y video, bibliotecas gráficas como OpenGL y SQLite para la gestión de bases de datos. Brindan un soporte esencial al Framework de Aplicaciones de Android.
- **Framework de Aplicaciones:** Esta capa proporciona un conjunto de herramientas y servicios para simplificar el desarrollo de las aplicaciones. Incluye un sistema de vistas para crear interfaces de usuario mediante componentes reutilizables, un administrador de recursos para gestionar recursos como archivos gráficos, un sistema de notificaciones y un administrador de tareas. Además proporciona proveedores de contenido para la comunicación entre aplicaciones y la gestión de servicios.
- **Aplicaciones:** Esta es la capa más alta del sistema donde habitan las aplicaciones que interactúan directamente con los usuarios. Estas están escritas en Java (aunque ahora también admite Kotlin), y tienen acceso a la misma API proporcionada por el Framework de Aplicaciones de Android. De esta forma, las aplicaciones base y las de terceros se ejecutan en el mismo entorno y se comunican entre sí.

1.4.4. Java

Java [11] es un lenguaje de programación diseñado para ser portable, seguro y robusto. Fue desarrollado por Sun Microsystems a mediados de la década de 1990 y adquirido después por Oracle Corporation.

Las principales características de Java son las siguientes:

- **Portabilidad:** Java está diseñado para ser ejecutado en cualquier plataforma a través de una máquina virtual Java (JVM). Se basa en el principio "write once, run anywhere".
- **Orientado a objetos:** Java se basa en el concepto de clases y objetos que interactúan entre sí para realizar tareas.
- **Seguridad:** Java está diseñado con características como el control de acceso y la gestión automática de la memoria para prevenir errores de acceso a la memoria



Figura 1.9: Capas de Android[10]

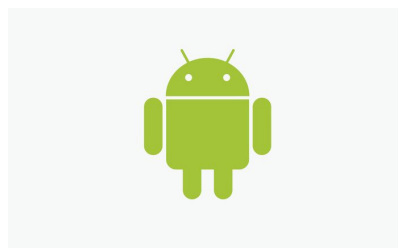


Figura 1.10: Logo de Android[10]

- Robustez: Destaca por su capacidad para manejar errores y excepciones de manera efectiva, lo que lo hace adecuado para aplicaciones críticas.

Por estos motivos Java es ampliamente utilizado en variedad de campos, como el desarrollo de aplicaciones para dispositivos móviles, que es el caso que nos interesa en este trabajo.



Figura 1.11: Logo de Java[11]

1.4.5. Android Studio

Android Studio [12] es un entorno de desarrollo integrado (IDE) utilizado para el desarrollo de aplicaciones móviles para el sistema operativo Android. Fue desarrollado por Google y salió al mercado en 2013 como el IDE oficial para el desarrollo de aplicaciones Android.

Este IDE está basado en el software IntelliJ IDEA de JetBrains y está disponible para Linux, Mac OS, Windows y Chrome OS. Permite utilizar varios lenguajes de programación como Java, Kotlin o C++.

Dispone de un potente emulador de dispositivos Android que permite a los desarrolladores probar sus aplicaciones en una gran variedad de dispositivos, versiones de Android y resoluciones de pantalla.

Para compilar, empaquetar y gestionar las dependencias de las aplicaciones Android, utiliza Gradle, una herramienta de automatización de construcción de proyectos utilizada ampliamente en el desarrollo de software.



Figura 1.12: Logo de Android Studio[12]

1.4.6. Sonda vaginal

A continuación, se describen las características y el funcionamiento de la sonda vaginal con la que se pretende utilizar la aplicación (Figura 1.13). Aunque para el desarrollo del trabajo no haya sido necesario estar en contacto con la sonda, ya que basta con trabajar con los datos que la sonda recoge, no hay que olvidar que es una herramienta fundamental de cara al objetivo final de este proyecto.

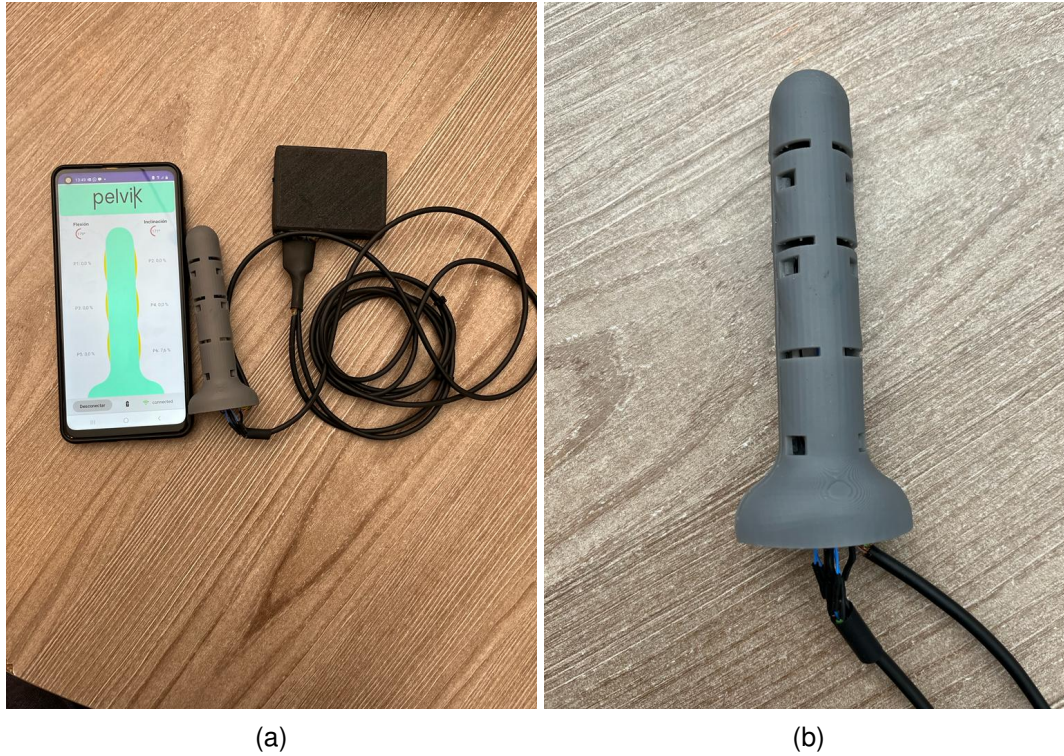


Figura 1.13: Sonda vaginal

La sonda utilizada está basada en la sonda vaginal PR04-A, con una longitud de 100 mm y 20 mm de diámetro. Cuenta con 6 sensores de presión manufacturados por AITEX, cuyas propiedades se recogen en la tabla 1.1.

Descripción	Valor
Forma	Redondeado
Diámetro [mm]	10
Anchura [mm]	4
Rango de presión [mmHg]	5-100
Sensibilidad [bits]	12
Cable de conexión [m]	1.5

Tabla 1.1: Propiedades de los sensores de presión

Los seis sensores están organizados para monitorear la presión del suelo pélvico en su totalidad: dos sensores detectan la presión profunda lateral, que se encargan de la continencia y dar apoyo al órgano pélvico; tres sensores superficiales, que detectan la presión en los músculos encargados de la continencia fecal y urinaria y de las funciones sexuales; y un sensor en la parte superior de la sonda que detecta la presión intraabdominal.

Introducción

Los sensores contienen una tinta conductiva que cambia su resistencia dependiendo de la fuerza aplicada sobre ellos. Este cambio de resistencia provoca un cambio de voltaje que es interpretado por un microcontrolador (STM32F3) que puede operar esta información en tiempo real.

Los resultados obtenidos con la sonda hasta el momento, permiten distinguir, por medio de los sensores, puntos de máxima contracción de los músculos (CM), momentos en los que se mantiene la contracción, presión intraabdominal (PIA), y diferencias entre los músculos profundos y superficiales, y los laterales izquierdos y derechos.

Mediante un software, estos datos recogidos pueden interpretarse en tiempo real en gráficos que muestran la presión para cada uno de los 6 sensores, tal y como muestra la figura 1.14 . Como ya se ha explicado, el objetivo de este proyecto es convertir esta información en algo más fácilmente interpretable para los usuarios y profesionales que trabajan con ello, mediante un modelo tridimensional del suelo pélvico que muestre la presión de los sensores directamente sobre los músculos afectados.

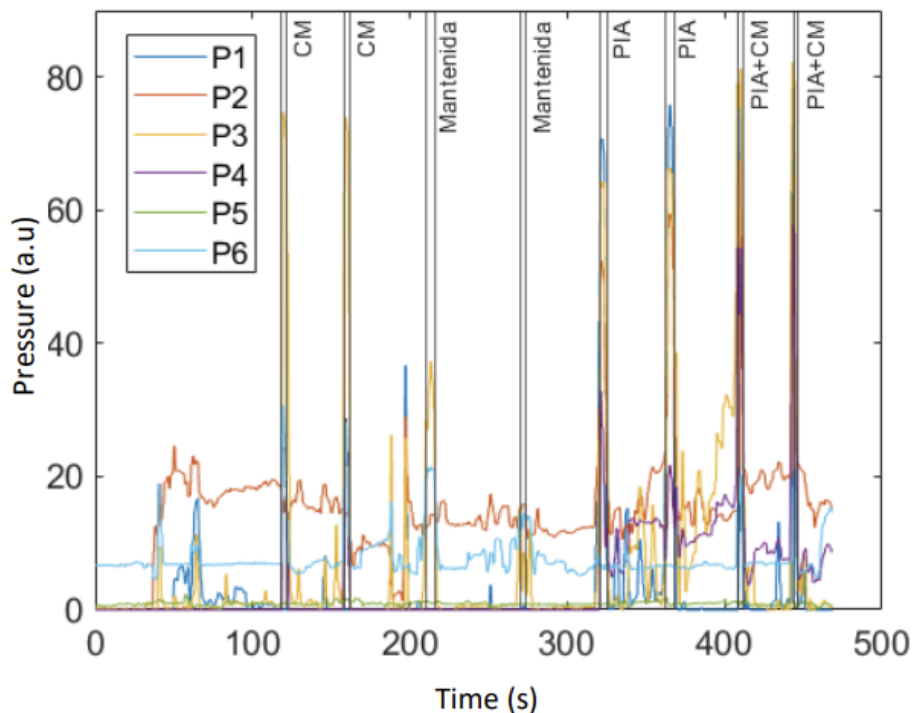


Figura 1.14: Datos recogidos por la sonda vaginal en tiempo real

Por último, mencionar que la sonda cuenta con conexión Bluetooth, que permite conectarla directamente al dispositivo móvil con la aplicación ya existente y obtener los datos allí.

1.5. Objetivos

En esta sección se resumen los principales objetivos de este trabajo.

- Representar un modelo tridimensional del suelo pélvico en el que se puedan identificar por separado los distintos músculos que distinguen los sensores de la sonda vaginal.
- Representar en dicho modelo los músculos que se están ejercitando con colores fríos para contracciones débiles y cálidos para contracciones fuertes.
- Representar en tiempo real datos extraídos de la sonda del modelo.
- Integrar esta aplicación en la aplicación Android ya existente.
- Validar los resultados en un entorno real con pacientes y fisioterapeutas, gracias a la colaboración con el grupo de fisioterapia del Hospital Universitario Príncipe de Asturias de Alcalá de Henares.

1.6. Metodología

Tras analizar los objetivos del proyecto y estudiar las características de este, se decide utilizar una metodología incremental e iterativa. Por un lado, esta metodología se adapta a los requisitos cambiantes, pues estos se revisan tras cada iteración. Por otro lado, otras metodologías ágiles están pensadas para trabajos en grupo y muchas de sus ventajas no son aplicables a trabajos en solitario.

Otra ventaja, es que esta metodología, incluye todas las fases típicas del desarrollo de software en cada iteración, por tanto, crea productos funcionales tras cada una de estas, permitiendo así tener terminada una parte del producto final y cambiar el rumbo del proyecto para cumplir con los objetivos, si fuera necesario. Además, al dividir el proyecto en incrementos más pequeños y manejables, se reduce el riesgo de fracaso del trabajo.

En el siguiente capítulo se explica la planificación inicial y en que consiste cada iteración, estando estas detalladas en el Capítulo 3.

1.7. Estructura de la memoria

La memoria del presente Trabajo Fin de Grado se divide en los siguientes capítulos:

- **Introducción:** Presentación del proyecto de manera general, incluyendo el contexto, las motivaciones, el estado actual de la cuestión, las tecnologías y herramientas utilizadas en el desarrollo, los objetivos, la metodología utilizada y la estructura de la memoria.

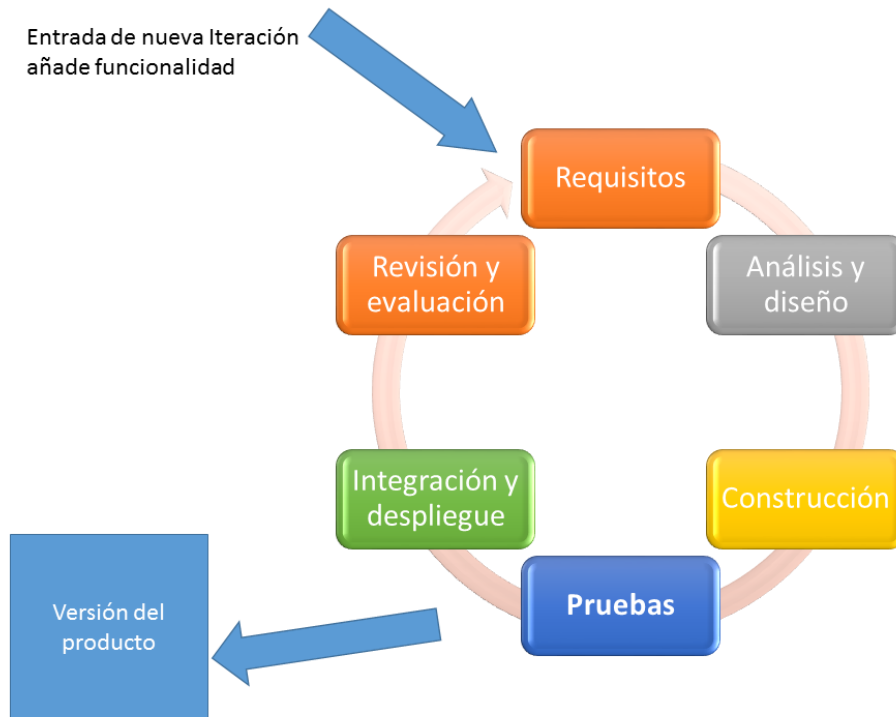


Figura 1.15: Metodología iterativa e incremental[13]

- **Planificación:** Se plantea la planificación del trabajo, incluyendo una planificación inicial por bloques de trabajo e iteraciones, un análisis de los posibles riesgos del proyecto, una descripción del entorno de trabajo, una estimación de costes y la planificación final (comparada con la planificación inicial)
- **Descripción de las iteraciones:** Desarrollo detallado de cada una de las iteraciones mostrando así el trabajo realizado y los problemas encontrados.
- **Estado final de la aplicación:** Descripción de la aplicación una vez finalizado el trabajo con su respectivo análisis y diagramas de diseño.
- **Conclusiones:** Exposición de las conclusiones del proyecto, un resumen general de este y exposición del trabajo futuro.
- **Apéndices:** Documentación adicional al proyecto como manuales de despliegue y uso.
- **Bibliografía:** Referencias bibliográficas utilizadas en el trabajo.

Capítulo 2

Planificación

2.1. Planificación inicial

En la siguiente sección se procede a detallar la planificación inicial del proyecto en iteraciones. Este Trabajo de Fin de Grado ha sido aceptado el día 27 de septiembre de 2023 y se estima su finalización el día 30 de junio de 2024, dedicándole un total de al menos 300 horas acorde a la normativa de la Universidad de Valladolid. El trabajo estará dividido en 4 iteraciones:

- **1ª iteración:** Importación del modelo 3D en Unity. Una vez adquirido el modelo 3D, se adaptará al entorno de trabajo realizando los cambios necesarios y solventando los problemas que surjan para poder realizar las modificaciones posteriores que se precisen. Durante esta iteración también se preparará el entorno y se realizará la documentación necesaria. Se estiman unas 70 horas.
- **2ª iteración:** Aportar funcionalidad al modelo. Se utilizará una escala de colores cálidos y fríos para representar la presión ejercida en los músculos correspondientes a través de los datos obtenidos por la sonda vaginal. Se estiman unas 100 horas necesarias.
- **3ª iteración:** Aplicación móvil. Se integrará la aplicación creada a una aplicación móvil que permita visualizar los músculos trabajados durante el ejercicio en tiempo real desde tu smartphone. Se estiman unas 100 horas necesarias.
- **4ª iteración:** Prueba en un entorno real. Se validarán los resultados obtenidos en un entorno real con pacientes y fisioterapeutas. Esto se conseguirá gracias a la colaboración con el grupo de fisioterapia del Hospital Universitario Príncipe de Asturias de Alcalá de Henares. Posteriormente se analizarán los resultados y se realizarán las correcciones necesarias. Se estiman unas 50 horas.

En la figura 2.1 se puede ver la distribución por horas por cada iteración junto con las horas acumuladas.

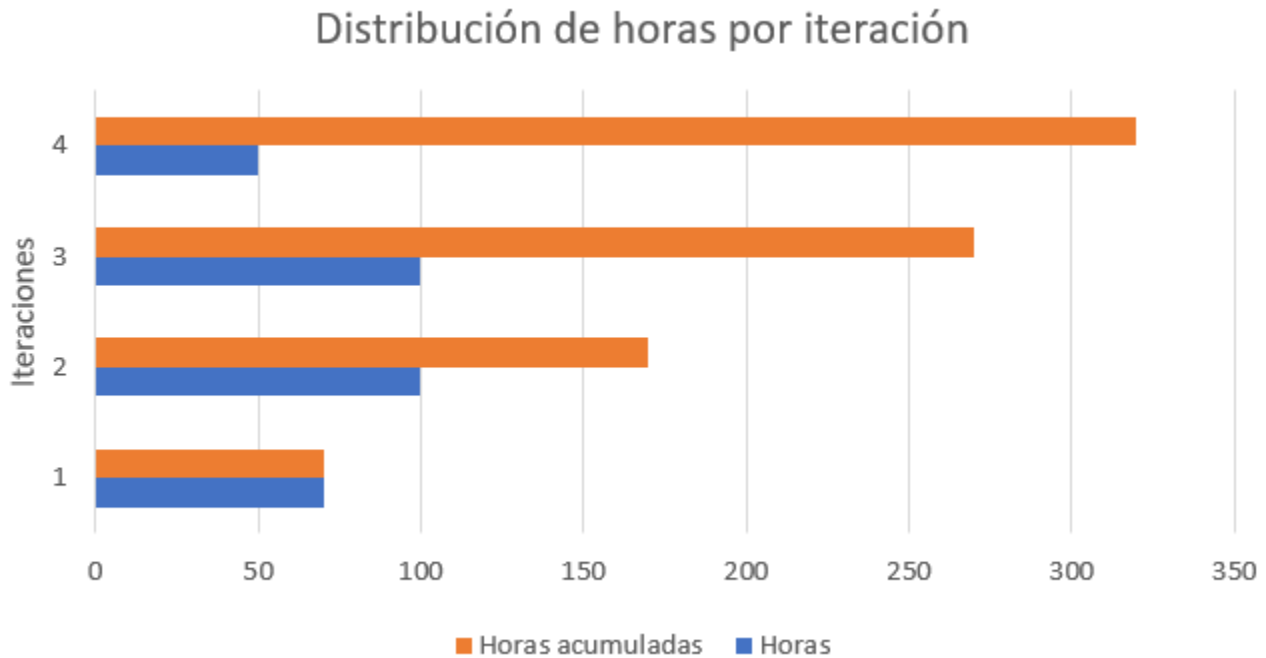


Figura 2.1: Horas estimadas iniciales por iteración

2.2. Análisis de riesgos

Durante la realización de cualquier proyecto, especialmente un trabajo de fin de grado, es esencial identificar, evaluar y gestionar los riesgos que podrían surgir. Los riesgos son eventos o situaciones inciertas que pueden afectar negativamente el progreso del proyecto. En esta sección, se presenta una evaluación detallada de los riesgos identificados, su probabilidad de ocurrencia, el impacto en el proyecto y las estrategias de mitigación y reducción aplicadas para minimizar su impacto en el trabajo.

En la Tabla 2.1 se resumen los riesgos identificados durante la realización del trabajo de fin de grado, junto con su probabilidad de ocurrencia, el retraso estimado que podrían causar y la exposición al riesgo (calculada multiplicando la probabilidad por el impacto).

En la Tabla 2.2, se muestran los riesgos anteriores junto con una medida de mitigación que permita paliar los efectos del riesgo y una medida de reducción que reduzca la probabilidad de que el riesgo se manifieste.

Planificación

ID	Riesgo	Descripción	Probabilidad	Retraso (Días)	Exposición al riesgo
1	Enfermedad	Contraer una enfermedad que no me permita desarrollar el trabajo	0.1	5	0.5
2	Planificación incorrecta	Pérdida de tiempo debido a una mala planificación	0.1	15	1.5
3	Cambio en los requisitos	Los requisitos del proyecto varían cuando este ya se ha comenzado a desarrollar	0.15	15	2.25
4	Perdida de datos	No se guardan los datos correctamente y se pierde parte del trabajo realizado	0.1	5	0.5
5	Falta de experiencia con Unity	Al no haber trabajado con este software previamente, se pueden dar problemas para manejarlo	0.3	3	0.9
6	Problemas para implementar	Dificultades para desarrollar alguna funcionalidad del proyecto que lleven más tiempo del estimado ya sea por la falta de experiencia o por las limitaciones del entorno de trabajo	0.3	7	2.1
7	Problemas con el equipo de trabajo	Fallos con el portátil de trabajo y la imposibilidad de usarlo	0.2	5	1

Tabla 2.1: Descripciones y probabilidades de los riesgos del proyecto

2.3. Entorno de trabajo

En esta sección se describirá cual es son las características de los dispositivos utilizados para el desarrollo de este proyecto y las instrucciones para preparar el entorno de trabajo.

En primer lugar y como herramienta principal se ha utilizado un portátil ASUS Vivobook 15[14] operando sobre Windows 11. Las especificaciones de este ordenador se detallan en la Tabla 2.3.

En segundo lugar, se ha utilizado un dispositivo móvil Xiaomi Redmi Note 15S [15] con Android 12 [16] cuyas características se detallan en la Tabla 2.4.

Para el montaje del entorno de desarrollo, por un lado, se ha seguido el proceso de instalación de Unity, Blender y Android Studio en el portátil utilizado. Por otro lado, para utilizar el dispositivo móvil como entorno de pruebas de desarrollo de Unity ha sido necesario instalar la aplicación Unity Remote y conectar el móvil al portátil mediante un cable USB. Además, es necesario haber activado la Depuración a través de USB en el dispositivo móvil. Esto se logra yendo a Ajustes, a la sección "Sobre el teléfono", y tocar repetidas veces la versión de MIUI para activar las Opciones de desarrollador. Una vez conseguido esto, se puede habilitar la depuración por USB y podremos probar nuestra aplicación Unity en Android sin necesidad de construir un APK e instalarlo en el teléfono. Este mismo procedimiento es necesario

ID	Riesgo	Descripción	Acción de mitigación	Acción de reducción
1	Enfermedad	Contraer una enfermedad que no me permita desarrollar el trabajo	Emplear horas extras en los días posteriores a la enfermedad para recuperar el tiempo perdido	Mantener precauciones y llevar una vida saludable para no enfermarse
2	Planificación incorrecta	Pérdida de tiempo debido a una mala planificación	Trabajar horas a mayores para compensar la pérdida	Estimar las horas con cierta holgura que permitan estas clases de fallos
3	Cambio en los requisitos	Los requisitos del proyecto varían cuando este ya se ha comenzado a desarrollar	Trabajar horas extras para reestructurar el enfoque del trabajo	Mantenerse en contacto con los tutores para ir comprobando que el trabajo va según lo previsto
4	Perdida de datos	No se guardan los datos correctamente y se pierde parte del trabajo realizado	Emplear horas a mayores para recuperar el trabajo perdido.	Realizar copias de seguridad frecuentemente y guardarlas en distintos dispositivos o en algún repositorio online
5	Falta de experiencia con Unity	Al no haber trabajado con este software previamente, se pueden dar problemas para manejarlo	Utilizar tiempo adicional para compensar la lentitud que pueda surgir al realizar las tareas	Aprovechar el tiempo previo al comienzo de la implementación para realizar tutoriales y coger cierta soltura con la herramienta de trabajo
6	Problemas para implementar	Dificultades para desarrollar alguna funcionalidad del proyecto que lleven más tiempo del estimado ya sea por la falta de experiencia o por las limitaciones del entorno de trabajo	Emplear horas a mayores para realizar la tarea	Informarse previamente de las funciones que se pueden realizar con el software utilizado y buscar alternativas
7	Problemas con el equipo de trabajo	Fallos con el portátil de trabajo y la imposibilidad de usarlo	Llevar a reparar y utilizar otro ordenador mientras tanto	Cuidar el portátil, transportarlo siempre en su funda y evitar los golpes y caídas

Tabla 2.2: Medidas de mitigación y reducción de los riesgos del proyecto

para probar nuestra aplicación Android desde Android Studio, aunque esta vez basta con conectar mediante un USB el móvil al ordenador y no hay aplicaciones intermedias como Unity Remote. Cabe mencionar que al tener ya activada la depuración por USB al haberla utilizado en Unity, no es necesario repetir el procedimiento para Android Studio.

El software utilizado más relevante en el entorno de trabajo se describe a continuación:

- **Unity** [8]: Herramienta para construir el modelo 3D.
- **Blender**[9]: Herramienta para llevar a cabo diversos ajustes en el modelo tridimensional.
- **Android Studio**[12]: Entorno de desarrollo oficial de Android con el que se construye la aplicación Android Nativo.
- **Google Chrome**[17]: Navegador web desarrollado por Google.

Ordenador	ASUS Vivobook 15
Procesador	AMD Ryden 7 5800H
Memoria	16 GB
Disco HDD	1 TB
Disco SSD	250 GB
Tarjeta gráfica	AMD Radeon(TM) Graphics
Resolución	1920 x 1080
Sistema operativo	Windows 11
Arquitectura del sistema	64 bits

Tabla 2.3: Especificaciones del portátil de trabajo

Móvil	Xiamoi Redmi Note 15S
Procesador	MTK G95 Octa-core Max 2.05GHz
Memoria	8 GB
Almacenamiento	128 GB
Tarjeta gráfica	250 GB
Tarjeta gráfica	Asus Dual GeForce RTX 3060 Ti
Resolución	720 x 1440
Sistema operativo	Android 12
Batería	5000 mAh

Tabla 2.4: Especificaciones del smartphone

- **OneDrive**[18]: Repositorio remoto de la Universidad de Valladolid, donde se almacenan la aplicación y la librería del proyecto.
- **Teams**[19]: Plataforma de comunicación desarrollada por Microsoft, empleada para mantenerse en contacto con los tutores.
- **Overleaf**[20]: Editor web de textos LaTeX, empleado para escribir la presente memoria.

2.4. Estimación de costes

En esta sección se detalla una estimación de los costes suponiendo que este trabajo se financiara. Una posible división para clasificar los costes [21] es la siguiente: costes humanos, costes de hardware, costes de software (licencias) y costes del espacio de trabajo.

2.4.1. Costes humanos

Los costes humanos del proyecto son aquellos que involucran el salario de los participantes. Por un lado, el sueldo anual medio de un desarrollador junior se encuentra en 21000€ [22]. Este sueldo, a jornada completa con 40 horas semanales, se traduce en 10,77€ por hora. Como se ve en la sección 2.1, el total de horas dedicadas a este proyecto será 320, lo que significa 3446,4€.

A mayores debe considerarse el salario de los supervisores del proyecto, en este caso de los tutores

del TFG. El salario anual de un jefe de proyecto software en España ronda los 47500€ al año, es decir, 24,36€ por hora [23]. Se estima una hora por cada tutoría, con aproximadamente 5 tutorías durante el desarrollo del proyecto, y un añadido de 5 horas en revisión de la memoria y solución de otros imprevistos que pudieran surgir. Como hay dos tutores en este trabajo, el sueldo alcanza un total de 487,2€.

En la tabla 2.5 se resumen los costes de esta sección.

Coste	Total
Salario junior	3446.4€
Salario supervisores	487.2€
Total	3933.6€

Tabla 2.5: Costes humanos

2.4.2. Costes de hardware

El hardware empleado en este trabajo ha sido el portátil y el móvil descritos en la sección 2.3. Podría considerarse también el precio de la sonda vaginal, pero realmente esta no se considera necesaria para el desarrollo del proyecto.

Como ambos dispositivos ya se disponían con anterioridad, su coste se considera amortizado. Por un lado, el portátil empleado tiene un coste de 799€, durante este proyecto se utilizará aproximadamente 320 horas y tiene una vida media de 6 años, utilizándose aproximadamente unas 8 horas diarias. El coste amortizado se calcula a continuación:

$$799 \times \left(\frac{320}{6 \times 8 \times 365} \right) = 14.59$$

Por otro lado, el teléfono móvil tiene un coste de 180€, durante este proyecto se empleará unas 60 horas y tiene una vida media de 4 años. Su coste amortizado es el siguiente:

$$180 \times \left(\frac{60}{4 \times 24 \times 365} \right) = 0.31$$

El resumen de los costes de hardware se encuentra en la tabla 2.6.

Coste	Real	Amortizado
Portátil	799€	14.59€
Móvil	180€	0.31€
Total	979€	14.9€

Tabla 2.6: Costes de hardware

2.4.3. Costes de software

En costes de software se incluirían las licencias de los programas utilizados para el proyecto, pero en este caso todos los programas son de uso gratuito o su licencia la paga la Universidad de Valladolid. Por tanto, no hay gastos de software.

2.4.4. Costes del espacio de trabajo

Los costes del espacio de trabajo incluyen el precio de la luz y el gas durante las 320 horas dedicadas al proyecto. El lugar de trabajo es la propia vivienda del desarrollador así que no se incluyen gastos de alquiler.

Por un lado, el consumo medio de un ordenador es de 60kWh al mes si se trabaja 8 horas diarias [24]. Es decir, unos 60kWh por 240 horas, con un precio de unos 14€. De esta forma, unos 80kWh por 320 horas saldrían a unos 19€.

Por otro lado, el gasto de consumo de gas de una vivienda ronda los 80€ mensuales [25]. Si aproximamos a las 320 horas de duración del proyecto, se gastarán unos 30€ durante la duración de todo el proyecto.

La información de esta sección se resume en la tabla 2.7.

Coste	Total
Luz	19€
Gas	30€
Total	49€

Tabla 2.7: Costes del espacio de trabajo

2.4.5. Costes totales

El coste total del proyecto desglosado por las secciones anteriores se recoge en la tabla 2.8.

Coste	Total
Humano	3933.6€
Hardware	14.9€
Software	0€
Espacio de trabajo	49€
Total	3997.5€

Tabla 2.8: Coste total

2.5. Planificación final

En esta sección se expone la planificación resultante tras el desarrollo de todo el proyecto y se compara con la planificación que se había planteado inicialmente.

Aunque no se ha llevado un seguimiento estricto de las horas, se estima que el proyecto ha alcanzado el tiempo planificado al comienzo aunque con variaciones en el tiempo dedicado a cada iteración. La comparativa entre las horas dedicadas por iteración en la planificación inicial y en la final se encuentra en la Figura 2.2.

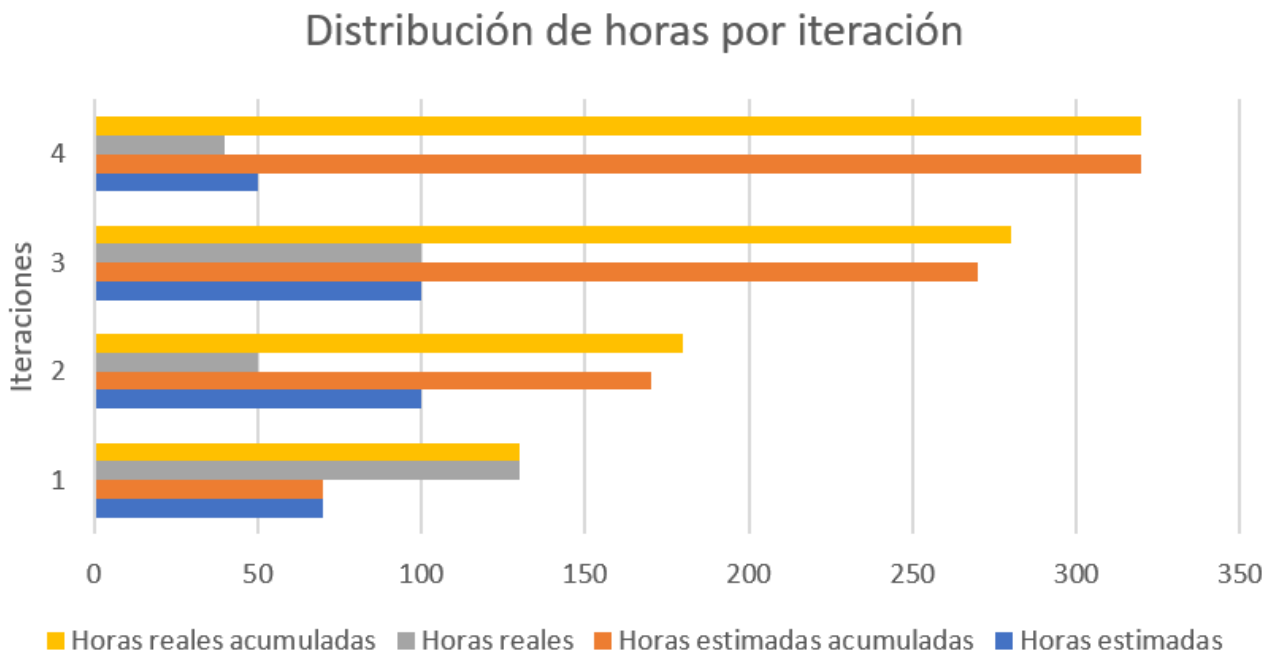


Figura 2.2: Comparación entre planificación inicial y final

Como indica el Capítulo 3, se encontraron varios problemas en la importación del modelo y preparación del entorno de trabajo, tanto que el tiempo dedicado a la primera iteración casi duplica el estimado al comienzo del proyecto. Esto se ha compensado con menos tiempo de desarrollo en la segunda iteración, pues la funcionalidad del modelo no era muy compleja y no hubo grandes problemas para programar en C#. La tercera iteración cumplió el tiempo estimado bastante bien, tratándose de una de las iteraciones más costosas como se describe más adelante. Por último, la iteración de pruebas llevó algo menos del tiempo previsto al no probar la aplicación en un entorno real si no con datos simulados.

Capítulo 3

Descripción de las iteraciones

En el siguiente capítulo se detalla el proceso de desarrollo de cada una de las cuatro iteraciones así como problemas que hayan podido surgir y como se han solucionado.

3.1. Iteración 1

El objetivo de la primera iteración es importar el modelo tridimensional adquirido al entorno de Unity para que este se encuentre listo para aportarle la funcionalidad necesaria en la siguiente iteración.

A pesar de ser aparentemente sencillo, se estimaron un total de 70 horas para esta iteración por no tener conocimientos previos del entorno de desarrollo y las complicaciones que podrían suponer solventar los problemas por simples que fueran. Aun así, la estimación se quedó corta, pues la importación del modelo no fue tan inmediata como parecía.

En primer lugar, el Hospital Universitario Príncipe de Asturias compró el modelo a través de la página web Sketchfab [26] utilizada para comprar y vender modelos tridimensionales. Este modelo fue construido por Aimee Hutchinson para el Centro de Anatomía e Identificación Humana (CAHID) de la Universidad de Dundee.

El primer problema encontrado, fue que este modelo se encontraba en formato *.USDZ* incompatible con Unity. La solución más rápida fue convertirlo a un formato *.FBX* con la ayuda de un convertidor en línea [27]. Una vez hecho, solo había que abrir Unity, crear un nuevo proyecto e importar el nuevo archivo *.FBX*.

Al importar el objeto surgía el siguiente problema, el modelo adquirido no incluía las texturas que se mostraban en la página web, y lo que inicialmente era un modelo realista (Figura 3.1) se había convertido en un objeto monocromático en el que era imposible distinguir los músculos (Figura 3.2).

En la página web decían que el modelo incluía las texturas, pero el Hospital al adquirirlo aseguraban no tenerlas. Tras intentar comunicarme con la creadora del modelo y no obtener respuesta, opté por recrear las texturas con algunas gratuitas encontradas por internet aunque con unos resultados más

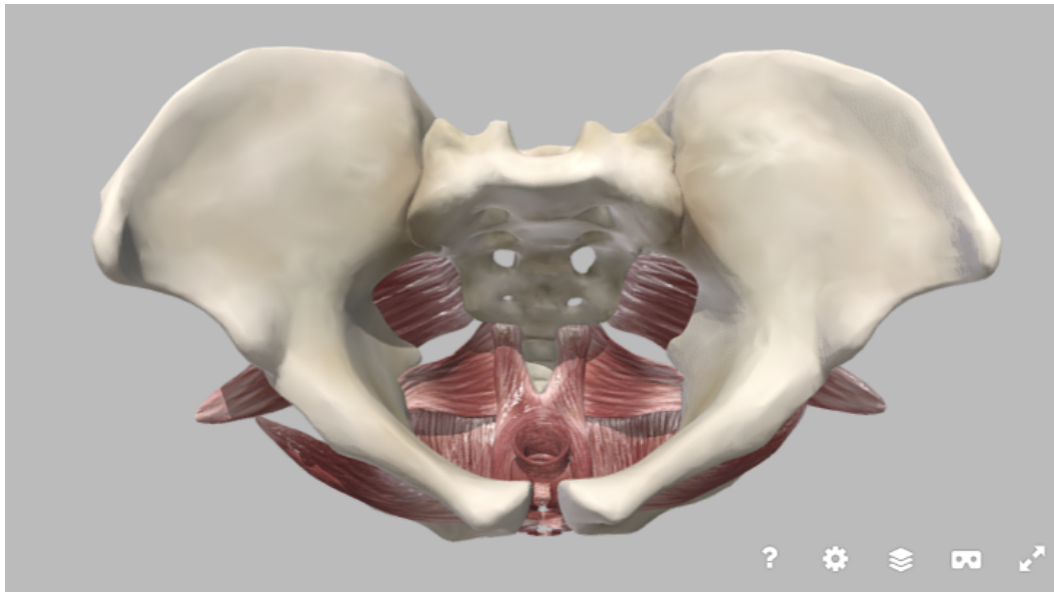


Figura 3.1: Modelo original

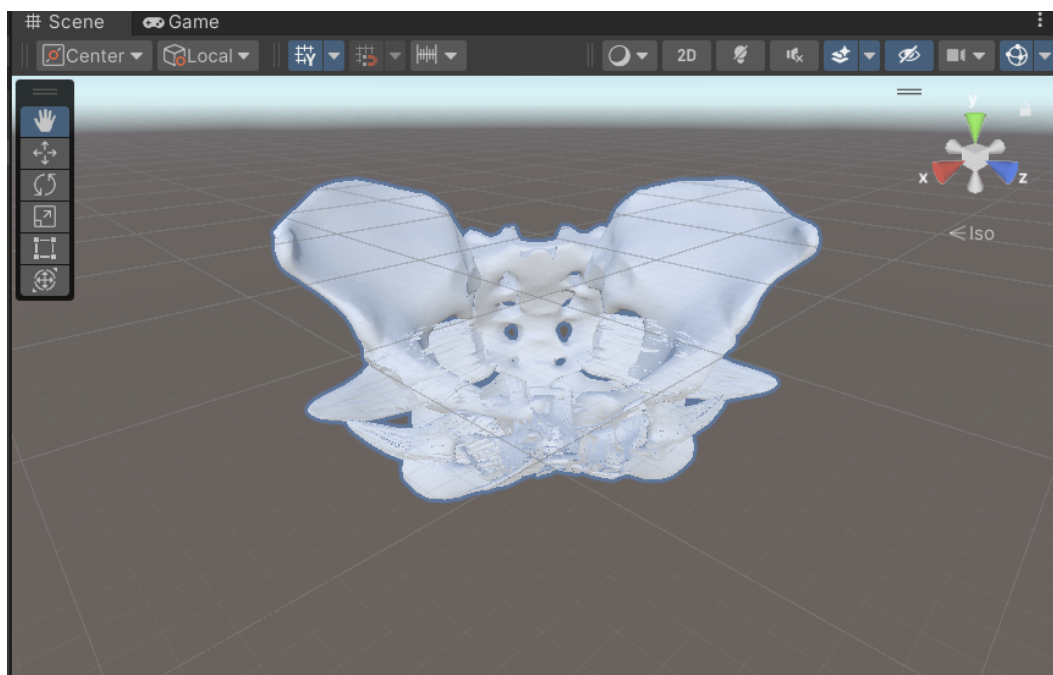


Figura 3.2: Modelo importado

pobres a los originales (Figura 3.3). Estas texturas fueron adquiridas en el siguiente enlace [28].

El último de los problemas a la hora de importar el modelo y el más crucial fue el hecho de que el objeto adquirido solo estaba constituido por dos subobjetos independientes: uno que contenía toda la estructura ósea y otro con todos los músculos. Esto suponía un problema porque para la aplicación a desarrollar era necesario distinguir los músculos por separado para poder trabajar individualmente con cada uno de ellos dándole un color distinto según la presión ejercida sobre él.

Para solventarlo, fue necesario recurrir a documentación adicional sobre la anatomía del suelo pélvico y poder diferenciar los distintos músculos del modelo. En este caso, la propia creadora del modelo

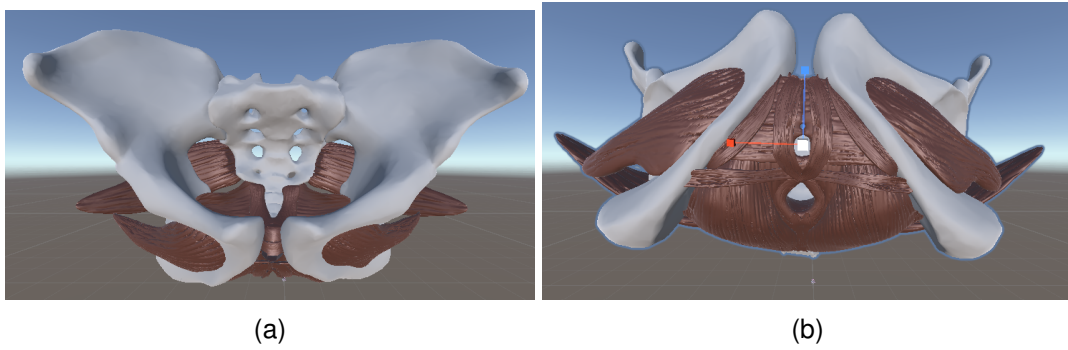


Figura 3.3: Modelo con texturas reconstruidas

dispone en su página web personal [29] de esquemas anatómicos que fueron de gran utilidad (Figura 3.4).

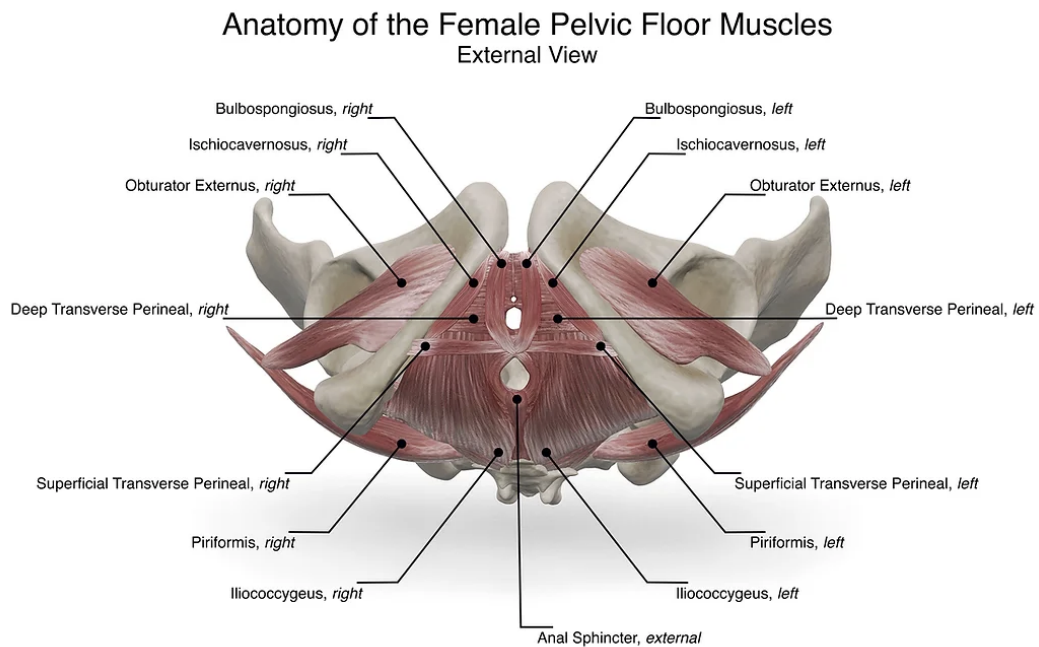


Figura 3.4: Esquema anatómico del suelo pélvico

Por desgracia Unity no facilita herramientas de edición avanzadas para conseguir separar el gran objeto muscular del que se disponía en pequeños músculos individuales, así que fue necesario recurrir a Blender, software de edición de objetos tridimensionales. El objeto estaba formado por una gran malla de triángulos diminutos y a través de Blender se consiguió separarlo en 20 músculos independientes (Figura 3.5).

Finalmente, cuando el Hospital Príncipe de Asturias me hizo llegar la lista de los músculos que detectaba cada sensor de la sonda vaginal, fue necesario realizar algunos ajustes, pues había algunos músculos que debían ser subdivididos en lateral derecho y lateral izquierdo. La información de los músculos finales utilizados en este modelo se resume en la Tabla 3.1.

La información de los músculos detectados por cada sensor se recoge a continuación:

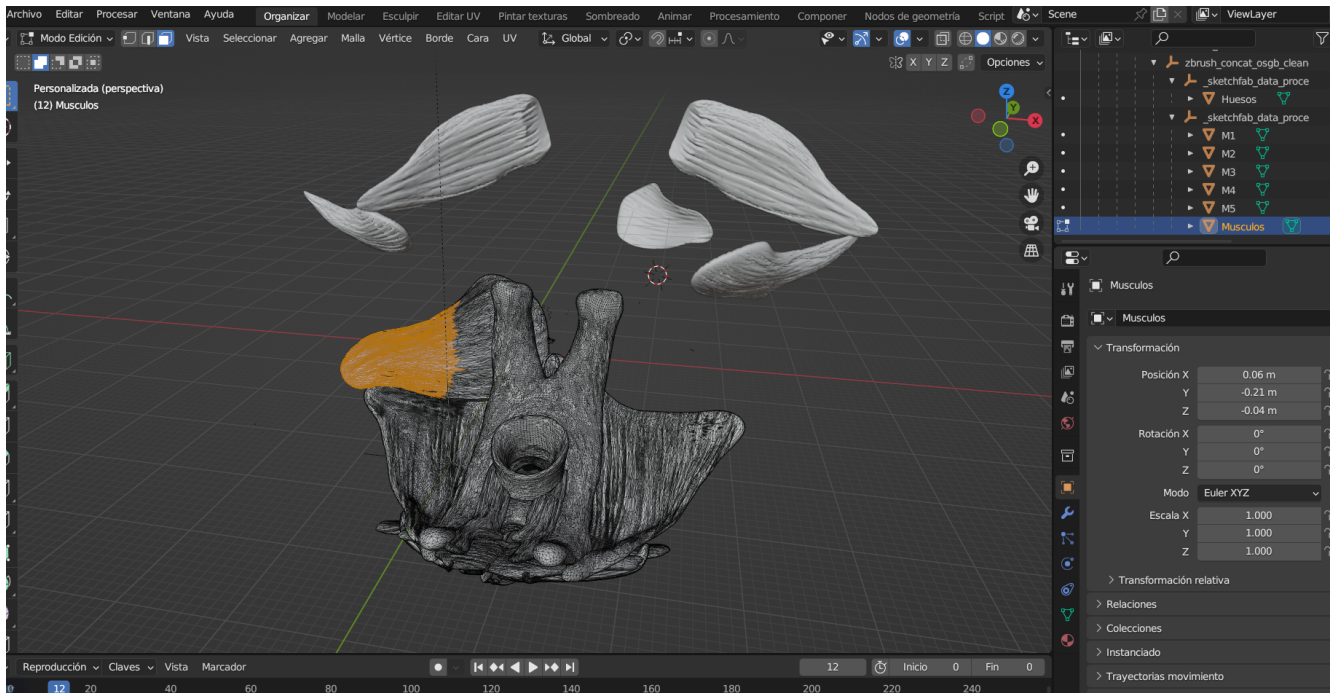


Figura 3.5: Separación de músculos por cota de malla en Blender

Identificador	Músculo
M1	Obturador Externo (derecho)
M2	Obturador Externo (izquierdo)
M3	Piriforme (izquierdo)
M4	Piriforme (derecho)
M5	Isquiococcígeo (izquierdo)
M6	Isquiococcígeo (derecho)
M7	Iliococcígeo (izquierdo)
M8	Iliococcígeo (derecho)
M9	Esfínter Anal Externo
M10	Perineo Superficial Transversal (izquierdo)
M11	Perineo Superficial Transversal (derecho)
M12	Isquiocavernoso (izquierdo)
M13	Isquiocavernoso (derecho)
M14A	Bulboesponjoso (derecho)
M14B	Bulboesponjoso (izquierdo)
M15	Perineo Profundo Transversal
M16	Esfínter Anal Interno
M17A	Pubococcígeo (derecho)
M17B	Pubococcígeo (izquierdo)
M18	Fibras Prerrectales
M19	Puborrectal (derecho)
M20	Puborrectal (izquierdo)

Tabla 3.1: Tabla de identificadores y músculos

- **Sensor 21:** Detecta el aumento de la presión intraabdominal (PIA), producida o bien por un empuje procedente del diafragma o bien por la contracción de la musculatura abdominal. Para una

Descripción de las iteraciones

correcta realización de los ejercicios, este sensor no debería ser presionado.

- **Sensor 22:** Detecta los músculos profundos del suelo pélvico, en concreto fibras del músculo iliococcígeo derecho.
- **Sensor 23:** Detecta los músculos profundos del suelo pélvico, en concreto fibras del músculo iliococcígeo izquierdo.
- **Sensor 24:** Detecta los músculos superficiales del suelo pélvico, en concreto fibras del músculo bulboesponjoso, isquiocavernoso y transverso superficial del periné derechos.
- **Sensor 25:** Detecta los músculos superficiales del suelo pélvico, en concreto fibras del músculo bulboesponjoso, isquiocavernoso y transverso superficial del periné izquierdos.
- **Sensor 26:** Indicará una presión de la sonda contra la sínfisis púbica, siendo un reflejo indirecto de una contracción de los músculos superficiales del suelo pélvico, en concreto fibras del músculo bulboesponjoso, isquiocavernoso y transverso superficial del periné derechos e izquierdos.
- **Sensor 31:** Detecta los músculos profundos del suelo pélvico, en concreto fibras de los músculos pubococcígeo y puborrectal izquierdos.
- **Sensor 32:** Detecta los músculos profundos del suelo pélvico, en concreto fibras de los músculos pubococcígeo y puborrectal derechos.

Con esta información es fácil detectar qué músculos de nuestro modelo se activarán con cada uno de los sensores. Esto se muestra en la Tabla 3.2.

Sensor	Músculos
Sensor 21	Ninguno
Sensor 22	Músculo 8
Sensor 23	Músculo 7
Sensor 24	Músculos 11, 13 y 14A
Sensor 25	Músculos 10, 12 Y 14B
Sensor 26	Ninguno
Sensor 31	Músculos 17A y 19
Sensor 32	Músculos 17B y 20

Tabla 3.2: Relación de músculos y sensores

Como vemos en esta tabla existen dos problemas. En primer lugar, en nuestro modelo no aparece la musculatura abdominal que detecta el sensor 21 y esta no puede ser coloreada según la presión. Como esta no debe ser presionada para una correcta realización del ejercicio, se ha optado por no representarla y en su lugar, mostrar un mensaje en pantalla advirtiendo de que se está detectando presión sobre ella en tal caso.

En segundo lugar, el sensor 26 detecta presión en la sínfisis púbica tratándose esta de una respuesta sobre ciertos huesos que no aparecen en nuestro modelo. En este caso se opta por omitir gráficamente la información aportada por dicho sensor por no llenar la sencilla interfaz de la aplicación con demasiados marcadores de presiones.

De esta manera podemos ver gráficamente la correspondencia por músculos y sensores en las Figuras 3.6, 3.7, 3.8, 3.9, 3.10 y 3.11.

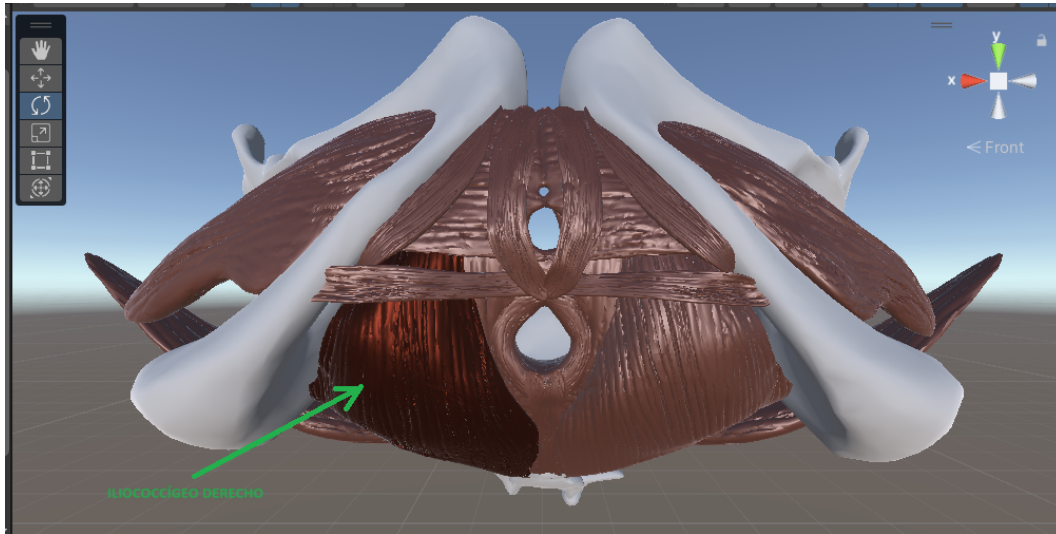


Figura 3.6: Sensor 22

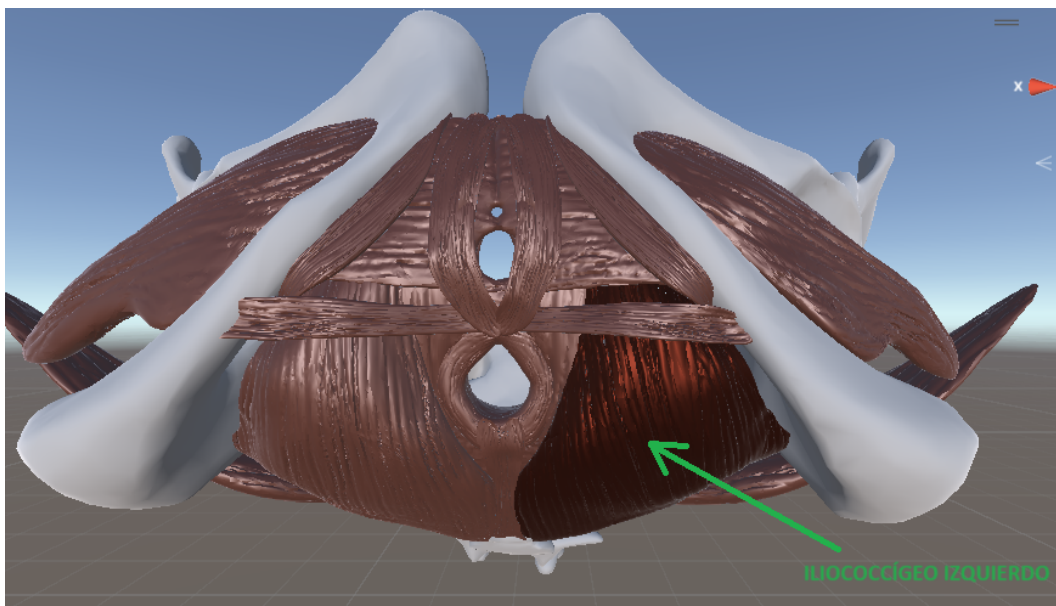


Figura 3.7: Sensor 23

Una vez realizados todos estos ajustes, nuestro modelo estaría listo para comenzar a implementarle su funcionalidad.

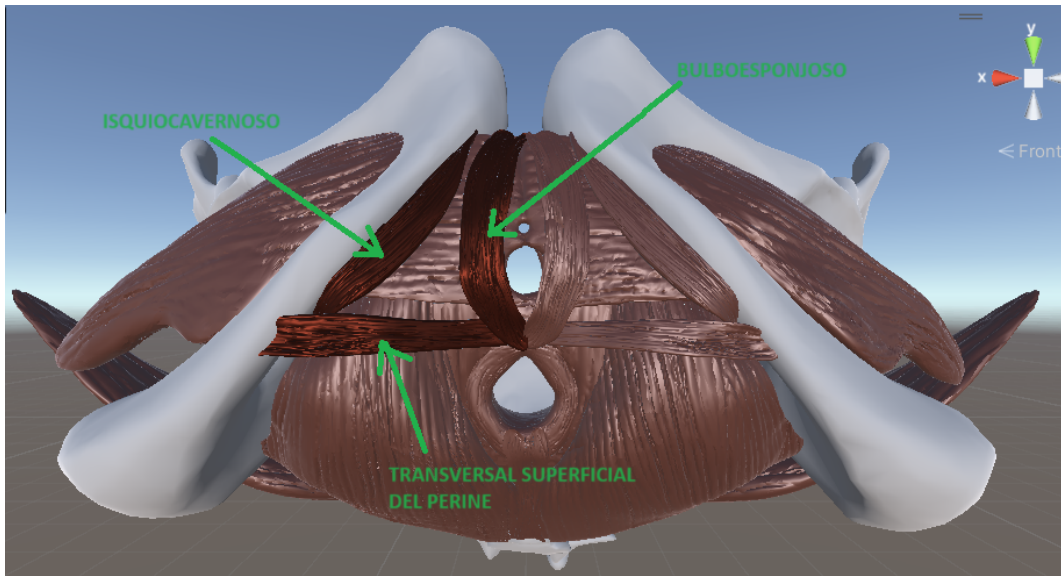


Figura 3.8: Sensor 24

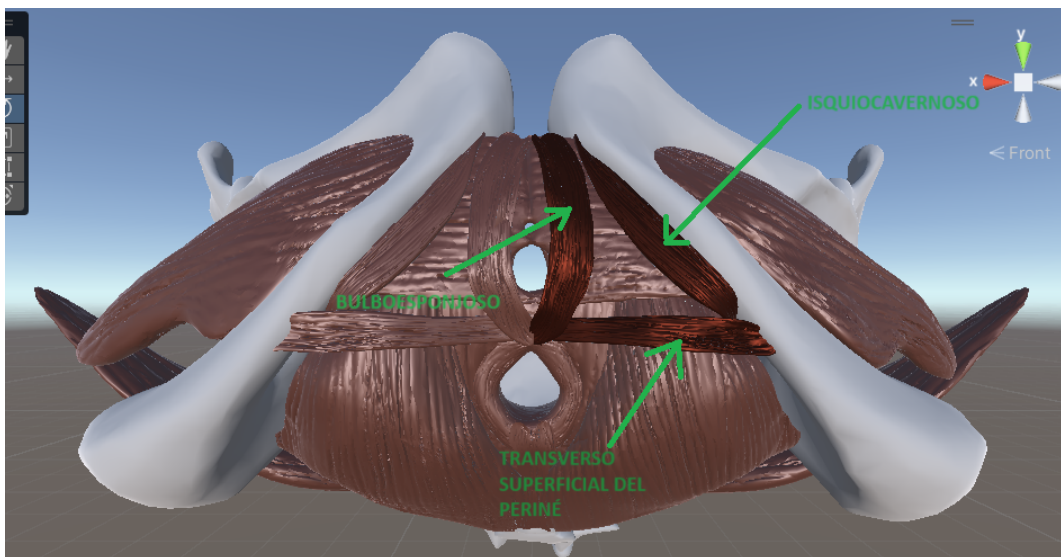


Figura 3.9: Sensor 25

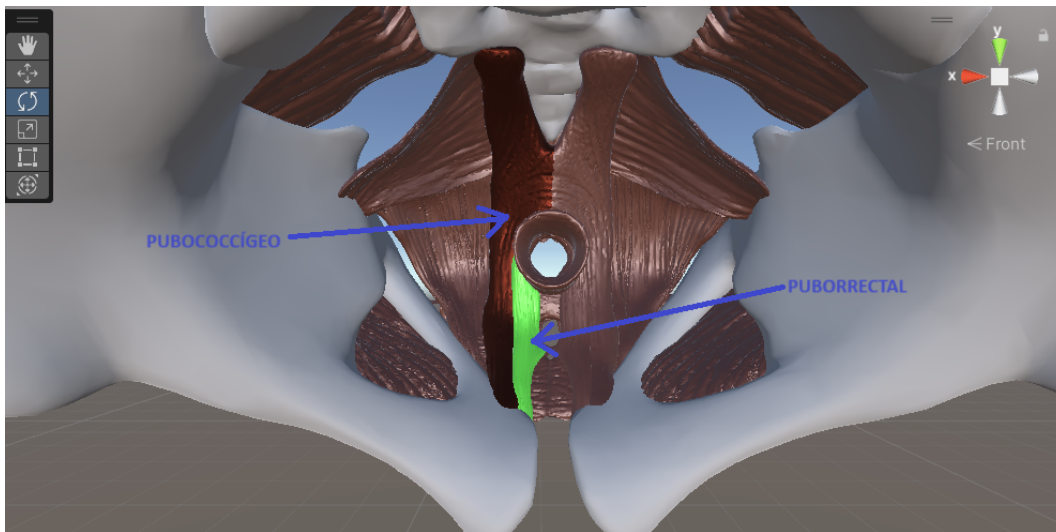


Figura 3.10: Sensor 31

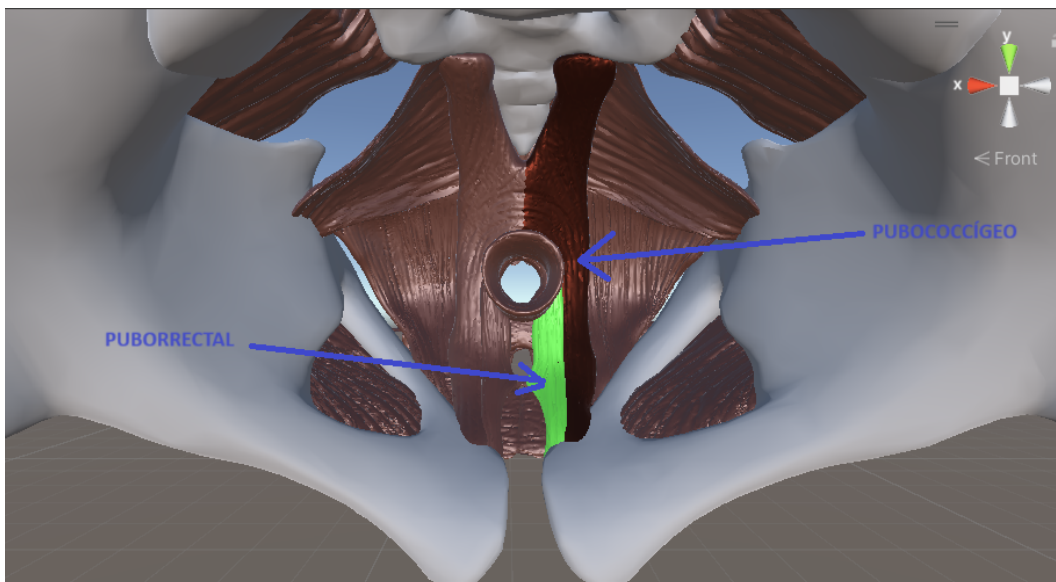


Figura 3.11: Sensor 32

3.2. Iteración 2

Los requisitos iniciales del modelo exigían que este fuera capaz de representar la presión obtenida en los sensores de la sonda vaginal a través de una escala de colores con tonos fríos para presiones bajas y tonos cálidos para presiones altas, con un tono neutro para presiones intermedias.

Durante esta iteración no se conoce cuáles son los valores de presión que alcanzan los sensores, por lo que se opta por escalarlos al intervalo [0,1], asociando un color azul al 0 (presión baja), un color blanco al 0.5 (presión intermedia) y un rojo al 1 (presión alta). El resto de valores intermedios serán la progresión de ir de uno de los colores fijados a otro.

Para conseguir este funcionamiento, se ha optado por asignar una textura distinta a cada grupo de músculos que se presionan con cada sensor de la sonda. Es decir, por ejemplo, los músculos 11, 13 y 14A, tendrán una misma textura denominada *Sensor 24* que actualizará su color cuando reciba cambios de presión en dicho sensor. El código para conseguir estos cambios se desarrolla en scripts C# asociados a estos objetos de Unity y puede ser similar al mostrado en el Listing 3.1.

```
1 public Material material1;
2 float presion1 = float.Parse(columns[1]);
3 float ajusteVariable1 = presion1 / 100f;
4 float tono1 = Mathf.Lerp(240f, 0f, ajusteVariable1);
5 float saturacion1 = Mathf.Lerp(0f, 1f, Mathf.Abs(ajusteVariable1 - 0.5f) * 2f);
6 Color nuevoColor1 = HSVToRGB(tono1, saturacion1, 1f);
7 material1.color = nuevoColor1;
```

Listing 3.1: Cambio de color según la presión

En el código 3.1, se muestra como se cambia el color al leer una presión de un supuesto fichero en el que el valor de la presión se encuentra en la segunda columna. En primer lugar se ajusta la presión a un rango de 0 a 1, suponiendo que los valores leídos van de 0 a 100. Después se ajusta el tono asignando azul al 0 y rojo al 1 (en la escala HSV, el tono azul se consigue con el 240 y el rojo con el 0). Luego se ajusta la saturación para que sea 1 en el centro de la escala y 0 en los extremos, es decir, que cuando la presión sea 0.5, el color sea blanco. Por último, mediante una función convertimos este color en la escala HSV a RGB que es la utilizada en Unity y se lo aplicamos al material original.

Nótese que este proceso se realizará de forma similar para los 6 sensores y sus correspondientes músculos cada vez que se reciba un nuevo dato de presión en su sensor. Como todavía no se disponen de datos de prueba ni se conoce el formato del fichero, este código queda pendiente de reajustar cuando se obtenga la información necesaria.

A mayores, durante esta iteración se le añadió código a la aplicación para permitir que el usuario pudiera interactuar con el modelo mediante controles táctiles intuitivos. Con un simple script de C# se consiguió que el modelo pudiera rotar. Además se añadió un botón que al ser pulsado restablece el ángulo original del modelo para poder volver a su situación inicial fácilmente.

3.3. Iteración 3

En esta iteración, el objetivo era transportar la aplicación creada a través del software de Unity a una aplicación móvil para poder ejecutarla desde nuestro smartphone. Además, los requisitos se ampliaron y se pidió una documentación detallada de como se podría realizar la integración de esta aplicación móvil a otra aplicación ya existente, para que esta pueda servir como tutorial al Hospital Príncipe de Asturias a la hora de querer integrar mi aplicación Unity a la suya en Android Nativo. Como no se cuenta con la aplicación del hospital, se ha generado una aplicación sencilla de prueba en Android Nativo que sirva como ejemplo. Los pasos detallados de esta integración se muestran en el Anexo A, por lo que aquí solo se describirá el proceso de forma general.

El primer paso fue probar la aplicación Unity en un dispositivo Android exportándola directamente. Este proceso es sencillo, ya que Unity cuenta con herramientas que lo realizan de forma automática.

Una vez asegurado el funcionamiento de la aplicación de Unity, se realiza el proceso de integrarla en otra aplicación en Android Nativo. Comencé creando una aplicación Android sencilla con un único botón, que al pulsarlo, abriera la aplicación Unity. Tras exportar la aplicación Unity como una librería Android, seguí diversos tutoriales que mostraban como integrarlo. Surgieron muchos problemas como incompatibilidades de versiones que hicieron de esta, la parte más dura del proyecto.

Por último, fue necesario desarrollar el sistema de comunicación entre ambas aplicaciones, para permitir que la aplicación Android envíe los datos que recibe a través de la sonda a la aplicación Unity, la cual los utilizará para actualizar el modelo tridimensional. Esta parte se realizó utilizando un servicio Android, es decir, una tarea que se ejecuta en segundo plano sin necesidad de una interfaz visual, cuyo objetivo es enviar datos a Unity con la frecuencia con la que los recibe de la sonda vaginal.

3.4. Iteración 4

Inicialmente el objetivo en esta última iteración era probar la aplicación en un entorno real, facilitado por el Hospital Príncipe de Asturias. Debido a retrasos en los tiempos, hasta mediados de abril no se dispusieron de datos de pruebas para simular la aplicación. Además, estos datos pertenecían a una versión anterior de la sonda, con menos sensores de los descritos inicialmente. Por ejemplo, en esta sonda no existía un sensor que avisara de cuando se ejercía presión intraabdominal, pero estos datos de prueba fueron suficientes para simular el funcionamiento de la aplicación. Por tanto esta iteración se ha utilizado para probar el modelo con datos simulados y no en un entorno real.

A continuación se describe brevemente la sesión de pruebas realizada por el equipo de fisioterapia del hospital, del que se disponen datos. Durante esta sesión se realizaron 4 ejercicios:

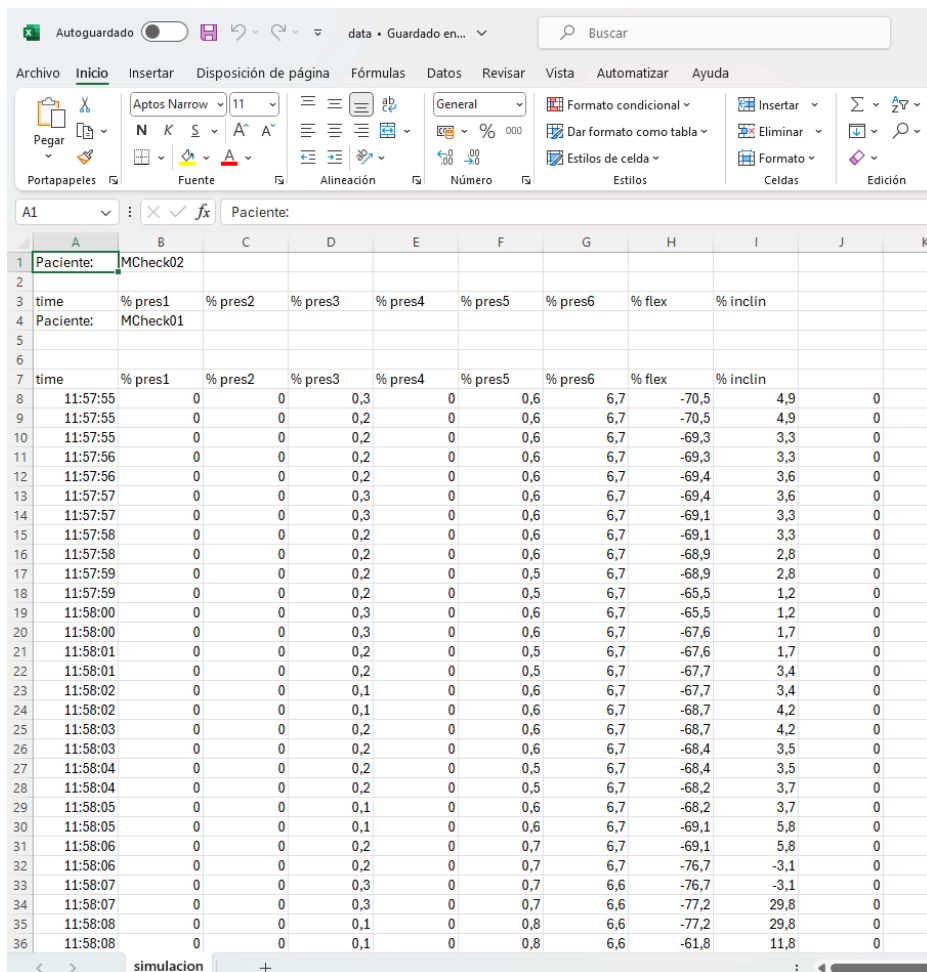
- CM: contracción máxima durante 3 segundos y posterior relajación.
- CR: contracción mantenida cogiendo aire inicialmente, contracción respirando durante 5 segundos y posterior relajación.

Descripción de las iteraciones

- PIA: presión intrabdominal. Se coge aire y mientras se retiene, se empuja hacia fuera durante 5 segundos.
- PIA+CM: se contrae fuerte y luego se tose y se relaja.

Estos ejercicios se realizarán dos veces rotando la sonda 180° entre el primer bloque y el segundo. En el primer bloque se realiza cada ejercicio dos veces consecutivas tras un breve descanso, mientras que en el segundo sólo se realiza una vez cada ejercicio.

Centrándonos en los datos para la aplicación, estos estaban en un fichero Excel, en los que en cada fila se obtenían las presiones ejercidas en cada uno de los 6 sensores por cada instante de tiempo, como muestra la figura 3.12. La última columna de este Excel ha sido añadida artificialmente para simular el sensor de presión intraabdominal y poder probarlo, tomando el valor 0 cuando no se encuentra activo y el valor 1 cuando sí que lo está, como se indicaba en la sección 3.1.



	A	B	C	D	E	F	G	H	I	J	K
1	Paciente:	MCheck02									
2											
3	time	% pres1	% pres2	% pres3	% pres4	% pres5	% pres6	% flex	% inclin		
4	Paciente:	MCheck01									
5											
6											
7	time	% pres1	% pres2	% pres3	% pres4	% pres5	% pres6	% flex	% inclin		
8	11:57:55	0	0	0,3	0	0,6	6,7	-70,5	4,9	0	
9	11:57:55	0	0	0,2	0	0,6	6,7	-70,5	4,9	0	
10	11:57:55	0	0	0,2	0	0,6	6,7	-69,3	3,3	0	
11	11:57:56	0	0	0,2	0	0,6	6,7	-69,3	3,3	0	
12	11:57:56	0	0	0,2	0	0,6	6,7	-69,4	3,6	0	
13	11:57:57	0	0	0,3	0	0,6	6,7	-69,4	3,6	0	
14	11:57:57	0	0	0,3	0	0,6	6,7	-69,1	3,3	0	
15	11:57:58	0	0	0,2	0	0,6	6,7	-69,1	3,3	0	
16	11:57:58	0	0	0,2	0	0,6	6,7	-68,9	2,8	0	
17	11:57:59	0	0	0,2	0	0,5	6,7	-68,9	2,8	0	
18	11:57:59	0	0	0,2	0	0,5	6,7	-65,5	1,2	0	
19	11:58:00	0	0	0,3	0	0,6	6,7	-65,5	1,2	0	
20	11:58:00	0	0	0,3	0	0,6	6,7	-67,6	1,7	0	
21	11:58:01	0	0	0,2	0	0,5	6,7	-67,6	1,7	0	
22	11:58:01	0	0	0,2	0	0,5	6,7	-67,7	3,4	0	
23	11:58:02	0	0	0,1	0	0,6	6,7	-67,7	3,4	0	
24	11:58:02	0	0	0,1	0	0,6	6,7	-68,7	4,2	0	
25	11:58:03	0	0	0,2	0	0,6	6,7	-68,7	4,2	0	
26	11:58:03	0	0	0,2	0	0,6	6,7	-68,4	3,5	0	
27	11:58:04	0	0	0,2	0	0,5	6,7	-68,4	3,5	0	
28	11:58:04	0	0	0,2	0	0,5	6,7	-68,2	3,7	0	
29	11:58:05	0	0	0,1	0	0,6	6,7	-68,2	3,7	0	
30	11:58:05	0	0	0,1	0	0,6	6,7	-69,1	5,8	0	
31	11:58:06	0	0	0,2	0	0,7	6,7	-69,1	5,8	0	
32	11:58:06	0	0	0,2	0	0,7	6,7	-76,7	-3,1	0	
33	11:58:07	0	0	0,3	0	0,7	6,6	-76,7	-3,1	0	
34	11:58:07	0	0	0,3	0	0,7	6,6	-77,2	29,8	0	
35	11:58:08	0	0	0,1	0	0,8	6,6	-77,2	29,8	0	
36	11:58:08	0	0	0,1	0	0,8	6,6	-61,8	11,8	0	

Figura 3.12: Fichero de datos de simulación

De este fichero se extrae la frecuencia con la que la sonda vaginal toma datos, cada medio segundo. Esto es importante a la hora de simular la prueba en nuestro modelo, pues indica la velocidad a la que debe leer una nueva línea de datos. Viendo el formato del fichero, podemos ahora reajustar el código para conseguir simular la prueba.

Su función es recibir las líneas del fichero Excel a través de Android y separarlas en un array donde cada elemento es el valor de una columna. Después, ajusta estas presiones a una escala del 0 al 1 y le asigna un color a este valor, siendo azul el 0, rojo el 1 y una escala progresiva para valores intermedios.

A continuación se muestran algunas imágenes que confirman que la aplicación funciona correctamente (Figuras 3.13 3.14).

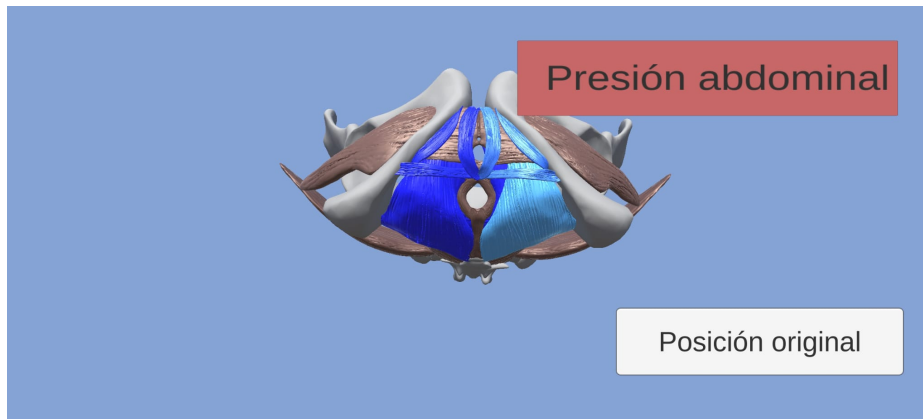


Figura 3.13: Simulación de ejercicios (1)

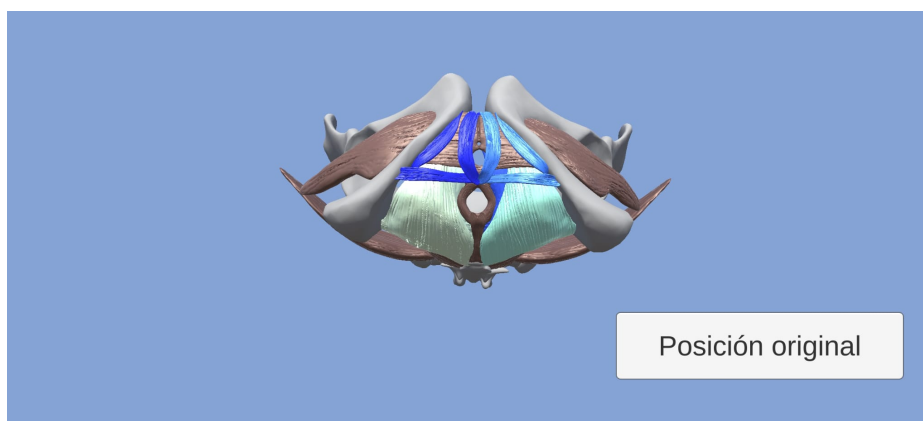


Figura 3.14: Simulación de ejercicios (2)

Capítulo 4

Estado final de la aplicación

En este capítulo se desarrollan las fases de análisis y diseño que han concluido en el estado final de la aplicación.

4.1. Análisis

4.1.1. Análisis de requisitos

En esta parte, se explican los diversos requisitos que se establecieron antes de comenzar a desarrollar el Trabajo de Fin de Grado en la reunión inicial con los tutores. Un requisito se define como una exigencia relativa al sistema que se detalla, y se clasifican en tres categorías: Funcionales, No Funcionales y de Información [30] [31]. Estos requisitos son esenciales para entender las necesidades del sistema en profundidad y guiar el proceso de desarrollo de manera efectiva.

Requisitos funcionales

ID	Nombre	Descripción
RF01	Modelo 3D interactivo	El sistema será capaz de mostrar un modelo 3D interactivo del suelo pélvico
RF02	Músculos ejercitados	El sistema será capaz de detectar que músculo se está ejercitando en cada momento
RF03	Presión ejercida	El sistema será capaz de detectar la presión ejercida en cada músculo y representarla a través de una escala de colores

Tabla 4.1: Requisitos funcionales

En la Tabla 4.1 se detallan los diversos requisitos funcionales necesarios para el proyecto. Estos requisitos delinean el comportamiento esperado del sistema que va a ser desarrollado. La tabla presenta cada una de estos requisitos con un identificador único, su respectivo nombre y una breve descripción que proporciona una comprensión inicial de la misma.

Requisitos no funcionales

ID	Nombre	Descripción
RNF01	Sistema de desarrollo	El sistema será desarrollado a través de Unity 2022.3.22f1
RNF02	Portabilidad	El sistema será portable a un smartphone con sistema operativo Android 12 para poder utilizarse desde él
RNF03	Conexión con aplicación Android	El sistema trabajará con una aplicación Android ya existente de la que obtendrá los datos necesarios para su funcionamiento
RNF04	Conexión Bluetooth con la sonda	El sistema recibirá datos a través de una conexión Bluetooth con la sonda vaginal

Tabla 4.2: Requisitos no funcionales

En la Tabla 4.2 se exponen los requisitos no funcionales del proyecto. Estos requisitos son las restricciones impuestas al sistema que definen sus atributos de calidad. Son los encargados de que el sistema satisfaga las necesidades del usuario. El formato de la tabla consta de un identificador único, el nombre del requisito y una breve descripción del mismo.

Requisitos de información

ID	Nombre	Descripción
RI01	Músculos	El sistema almacenará un identificador único para cada uno de los músculos
RI02	Asociación músculos-sensor	El sistema almacenará los músculos que deben activarse al recibir presión en cada uno de los sensores
RI03	Presión	El sistema almacenará la presión ejercida en cada sensor para representarla en el modelo tridimensional

Tabla 4.3: Requisitos de información

En la Tabla 4.3 se describen los requisitos de información del proyecto. Estos requisitos describen la información que es gestionada y almacenada por el sistema. La tabla presenta un identificador único para cada requisito junto con su nombre y una breve descripción de este.

4.1.2. Modelo de dominio

El modelo de dominio de un proyecto es una representación gráfica de las entidades de información del mismo y sus relaciones. De esta forma se puede entender fácilmente la interacción entre los componentes del dominio del sistema. En la Figura 4.1 se muestra el modelo de dominio de este proyecto.

Por un lado, la sonda vaginal está compuesta por varios sensores, los cuáles tienen un identificador único y recogen un valor según la presión que se ejerce sobre ellos. La sonda, a su vez, se comunica con el modelo tridimensional del suelo pélvico, al que le manda los valores de presión recogidos por

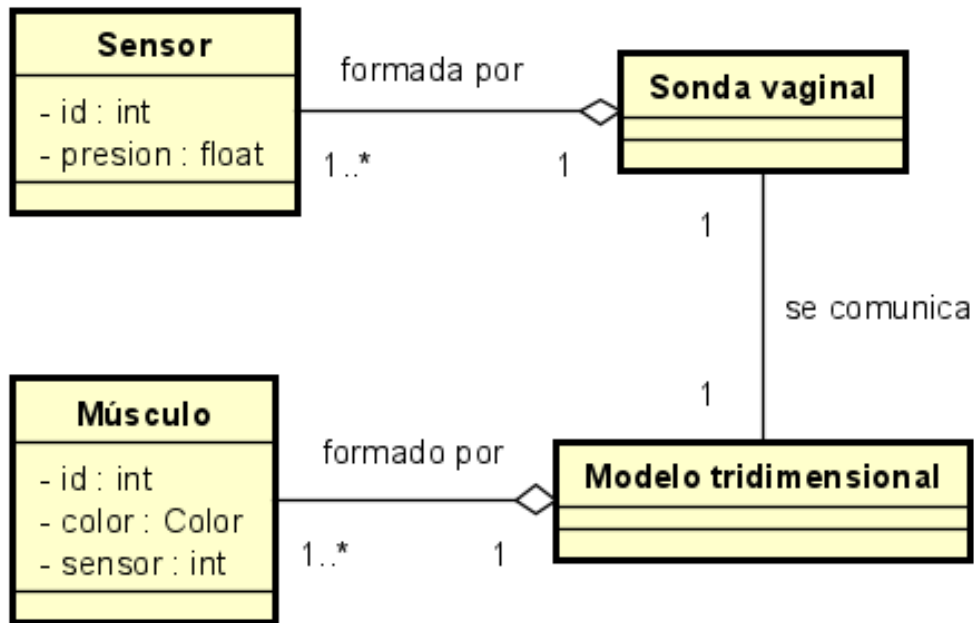


Figura 4.1: Modelo de dominio

los sensores. Este modelo está formado por diversos músculos, que variarán su color dependiendo del valor de presión que reciban. Además, estos músculos cuentan con un atributo que indica con qué sensor están relacionados, así, actualizarán su color únicamente con el valor de presión recogido por dicho sensor.

4.1.3. Historias de usuario

En esta sección se resumen los requisitos de la aplicación mediante historias de usuarios, que permiten recoger de forma coloquial los objetivos que debe cumplir la aplicación y una forma de validarlos. A mayores, se recoge un identificador único, la prioridad de cumplir ese objetivo y el riesgo de que fracase en la implementación.

ID	HU01
Nombre	Representación de músculos ejercitados
Prioridad	Alta
Riesgo	Bajo
Descripción	Como paciente quiero poder ver gráficamente en un modelo comprensible qué músculos estoy ejercitando
Validación	- Quiero ver un modelo tridimensional del suelo pélvico donde se identifiquen los músculos - Quiero que en el modelo se muestren con colores los músculos ejercitados

Tabla 4.4: Historia de usuario HU01

ID	HU02
Nombre	Representación de la presión ejercida
Prioridad	Alta
Riesgo	Bajo
Descripción	Como paciente quiero poder ver gráficamente la presión que ejerzo en cada músculo
Validación	- Quiero que los músculos se representen con una gama de colores fríos-cálidos dependiendo de la presión ejercida

Tabla 4.5: Historia de usuario HU02

ID	HU03
Nombre	Interacción con el modelo
Prioridad	Media
Riesgo	Bajo
Descripción	Como paciente quiero poder interactuar con el modelo
Validación	- Quiero poder girar el modelo tridimensional mientras realizo el ejercicio - Quiero poder volver a la posición original del modelo pulsando un botón

Tabla 4.6: Historia de usuario HU03

ID	HU04
Nombre	Visualización en tiempo real
Prioridad	Alta
Riesgo	Medio
Descripción	Como paciente quiero que la representación del ejercicio sea en tiempo real
Validación	- Quiero poder ver los músculos ejercitados y sus presiones al tiempo al que realizo el ejercicio

Tabla 4.7: Historia de usuario HU04

4.1.4. Diagrama de actividad

Los diagramas de actividad son una forma de representar el flujo de ejecución de un sistema junto con las interacciones del usuario. En esta sección se muestra el diagrama de actividad que expone como se realiza la comunicación entre las aplicaciones Android y Unity (Figura 4.2).

En primer lugar, el usuario inicia la aplicación Android y está muestra la pantalla inicial con el botón que permite abrir la aplicación Unity. Cuando este es pulsado, la aplicación Android ejecuta el servicio que le permite enviar datos a Unity en segundo plano. Después, la aplicación Unity se ejecuta y muestra el modelo tridimensional a la vez que actualiza este con los datos recibidos de Android.

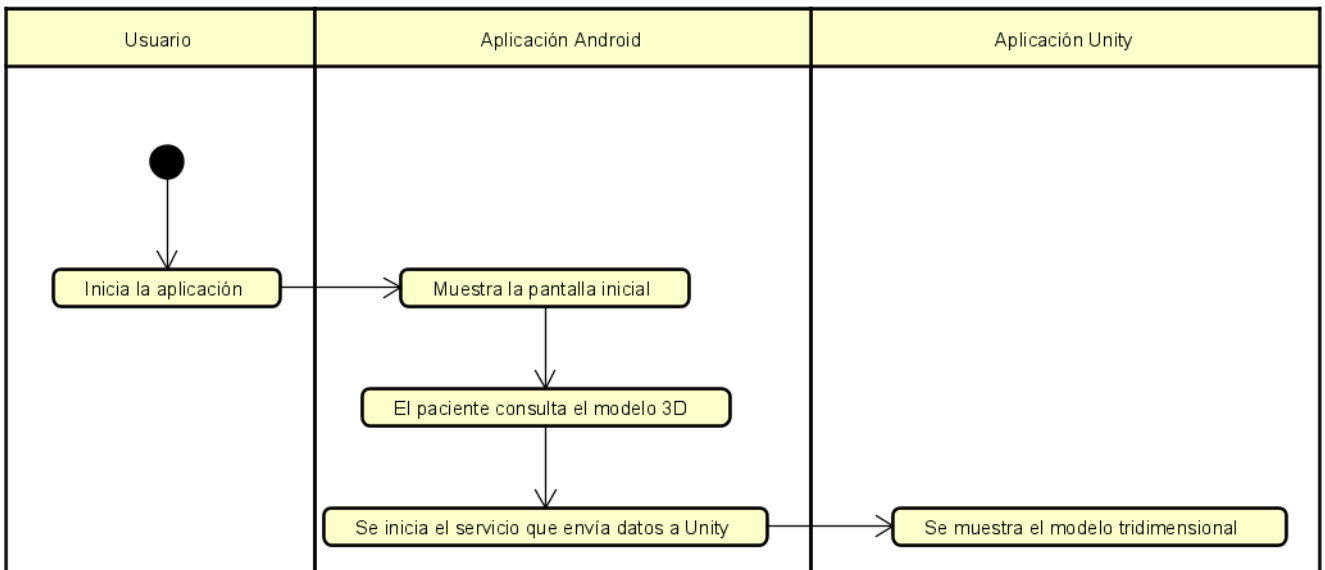


Figura 4.2: Diagrama de actividad

4.2. Diseño

En esta sección se detalla el diseño final de la aplicación. Se incluyen a continuación el diagrama de paquetes, el diagrama de despliegue, la estructura del proyecto, los patrones de diseño utilizados y las interfaces de la aplicación.

4.2.1. Diagrama de paquetes

El diagrama de paquetes tiene la función de mostrar las dependencias de los paquetes que forman la aplicación. En la Figura 4.3 se ve el de la aplicación Android y en la Figura 4.4 el de la aplicación Unity.

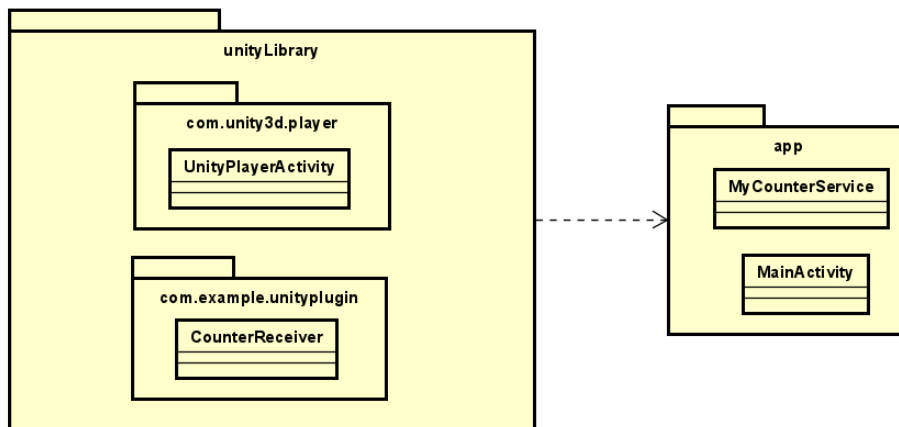


Figura 4.3: Diagrama de paquetes Android

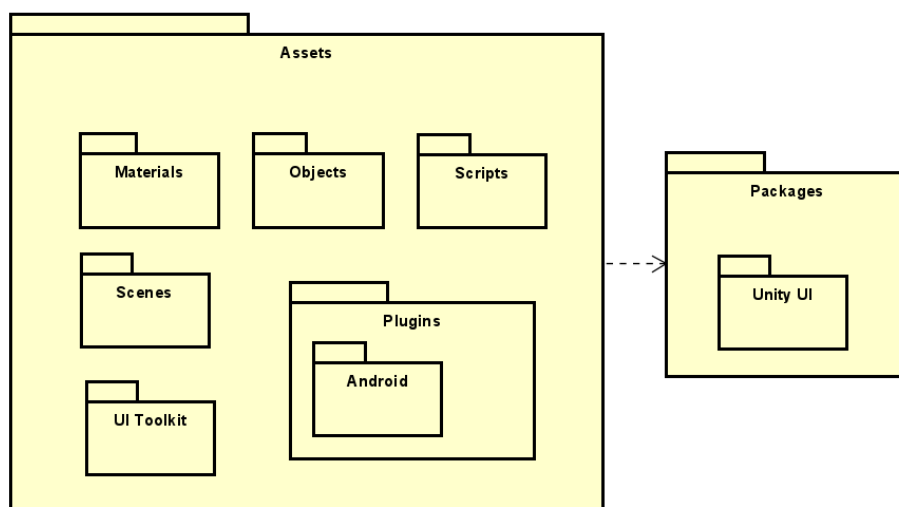


Figura 4.4: Diagrama de paquetes Unity

4.2.2. Diagrama de despliegue

El diagrama de despliegue sirve para mostrar como se conectan los dispositivos hardware del proyecto a la hora de ser utilizados. Este se observa en la Figura 4.5.

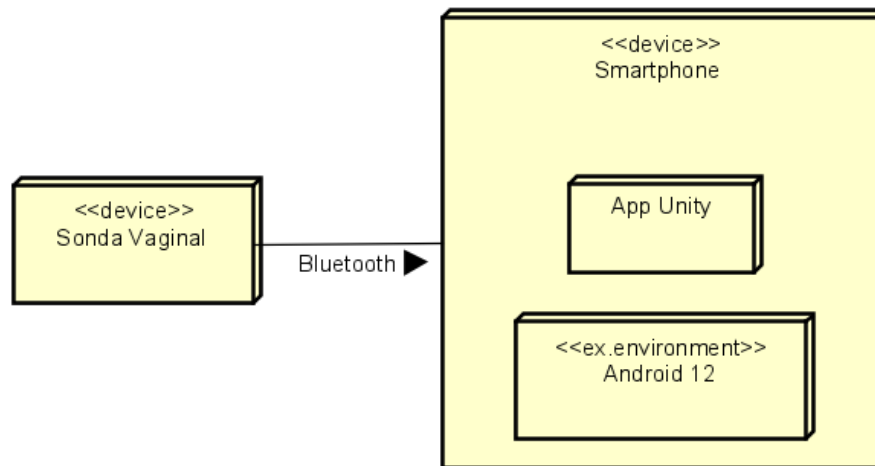


Figura 4.5: Diagrama de despliegue

En el diagrama de despliegue podemos ver como la sonda vaginal se comunica a través de una conexión Bluetooth con el teléfono móvil. Este, con un entorno Android 12, contiene la aplicación Unity que utilizará los datos que la sonda envíe.

4.2.3. Estructura del proyecto

El proyecto tiene la estructura por defecto configurada en cualquier aplicación diseñada con Android Studio. Los principales archivos son los siguientes:

- **Manifiesto de Android:** Existen dos manifiestos de Android, uno para la aplicación Android Nativo y otro para la aplicación Unity. En estos documentos se almacena información de la aplicación como permisos o los servicios y actividades que se utilizan.
- **Main Activity:** Es la clase Java principal de la aplicación en Android Nativo. En ella se gestiona la llamada a la aplicación Unity desde Android.
- **MyCounterService:** Es el servicio Android encargado de recibir los datos y avisar mediante un Intent de que hay datos nuevos para ser enviados a Unity.
- **activity_main.xml:** Esta clase gestiona la interfaz gráfica de la aplicación Android.
- **Counter Receiver:** Clase Java encargada de recibir las notificaciones del servicio Android y llamar a los métodos que envíen estos datos a Unity.

- **Unity Player Activity:** Clase principal de la aplicación Unity, con los métodos encargados de enviar los datos a Unity desde Android y de recibir en Android los que Unity envíe.
- **Build Gradle:** Los tres archivos denominados así guardan información de la configuración del proyecto, como los directorios del SDK o el NDK para compilar la aplicación.

La estructura completa del proyecto se puede ver en la Figura 4.6. A continuación se describe la estructura de la aplicación Unity y sus directorios más relevantes:

- **Materials:** Contiene las texturas utilizadas para los músculos y huesos del modelo. Cada conjunto de músculos que se activa con un mismo sensor tiene una textura distinta para facilitar su actualización.
- **Plugins:** Esta carpeta se utiliza para almacenar bibliotecas y complementos externos que extienden la funcionalidad de Unity. En este caso contiene el método Java que recibe los datos del servicio Android.
- **Scenes:** Carpeta donde se almacenan las escenas del proyecto, es decir, el conjunto de objetos, cámara, luces y sus propiedades.
- **Scripts:** En esta carpeta se guardan los ficheros que aportan funcionalidad mediante código al proyecto, como el fichero que aporta el control táctil o el que permite actualizar el color de los músculos del modelo.
- **UI Toolkit:** Contiene los recursos destinados a diseñar y gestionar la interfaz gráfica, como ajustes del panel.

La estructura completa de la aplicación Unity se puede ver en la Figura 4.7.

4.2.4. Patrones de diseño

Los patrones de diseño son soluciones habituales a problemas comunes cuando se diseña software. En este proyecto se ha utilizado principalmente el Patrón Observador.

Todas las aplicaciones orientadas a eventos tienen presente el **Patrón Observador**. Este patrón permite definir un mecanismo de suscripción para notificar a varios objetos sobre cualquier evento que le suceda al objeto que están observando [32]. Cualquier botón de la aplicación es un ejemplo de este patrón, que con ejecuta el método `onClick` al ser pulsado. Los `BroadcastReceivers` que se utilizan para enviar los datos que llegan de la sonda vaginal a la aplicación Unity también son un ejemplo del Patrón Observador. El servicio Android notifica cuando se recibe un dato nuevo de la sonda a través del Intent `com.example.myapplication.UPDATE`. El `BroadcastReceiver`, suscrito a este filtro, se encarga de enviárselo a la aplicación Unity.

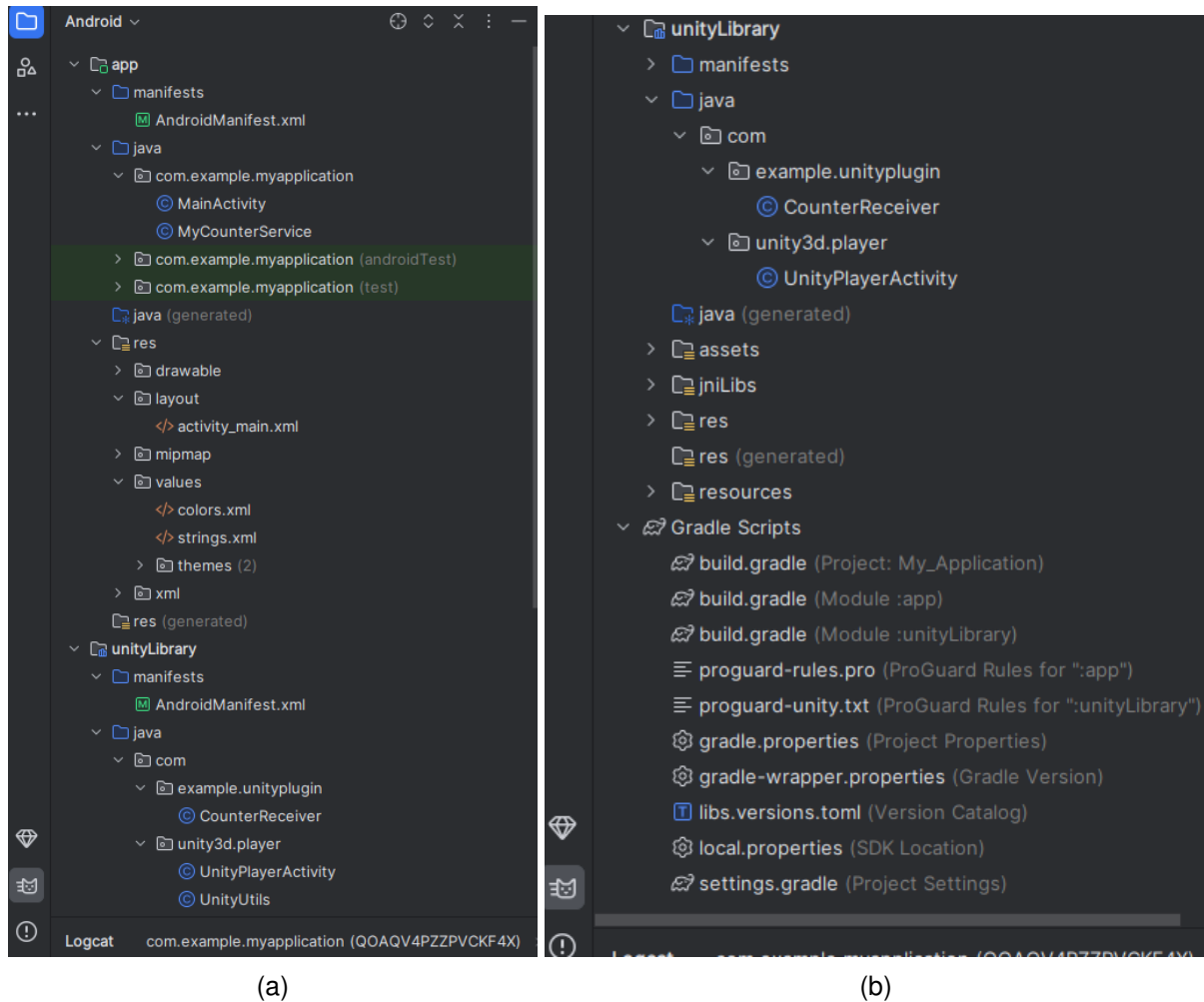


Figura 4.6: Estructura del proyecto en Android

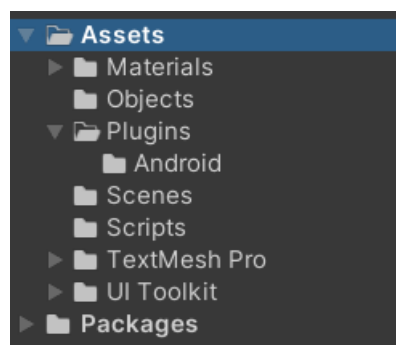


Figura 4.7: Estructura del proyecto en Unity

4.2.5. Interfaces

En esta sección se muestran las interfaces de la aplicación mediante capturas de pantalla.

- **Figura 4.8:** Esta es la pantalla que aparece al ejecutar la aplicación. Esta pantalla es temporal, solo se ha utilizado para simular una aplicación Android sencilla en la que integrar la aplicación Unity, por eso, no aparecerá cuando se integre en la aplicación ya existente del hospital. Solo

cuenta con un botón que te lleva al modelo 3D de la aplicación Unity.

- **Figura 4.9:** Esta es la pantalla que aparece al pulsar el botón Modelo 3D de la interfaz anterior. Es la única interfaz con la que cuenta la aplicación Unity y en ella se muestra el modelo con la representación gráfica del ejercicio en tiempo real. El botón de la izquierda Posición original, sirve para hacer volver el modelo a la posición en la que estaba al ejecutar la aplicación después de haberse rotado. El botón de la derecha Volver, permite regresar a la aplicación Android, es decir, la pantalla anterior.

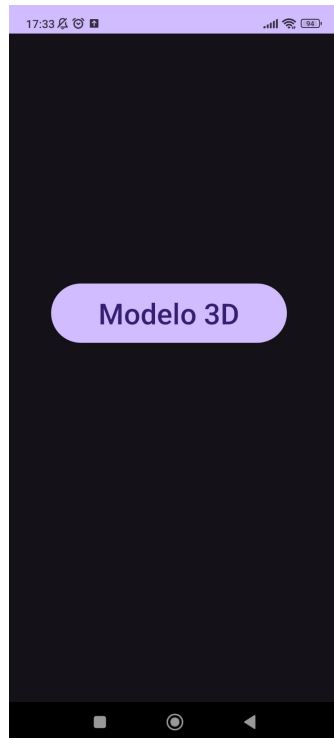


Figura 4.8: Interfaz Android

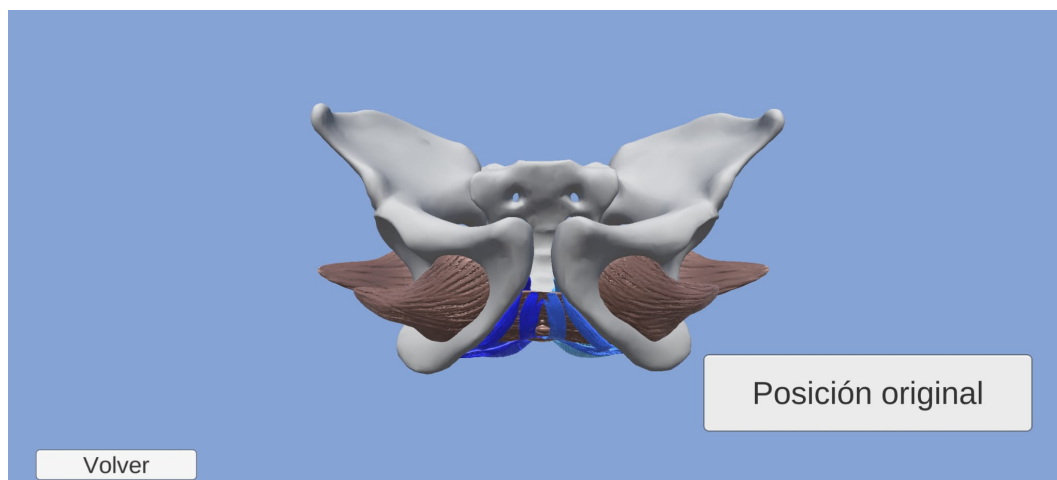


Figura 4.9: Interfaz Unity

Capítulo 5

Conclusiones

En este capítulo se resume el trabajo realizado y el aprendizaje recibido a lo largo de este, junto con una valoración personal.

Este trabajo representa una herramienta de utilidad para la realización de los ejercicios de fortalecimiento de suelo pélvico. A través de un modelo tridimensional de la musculatura, se consigue una visualización interactiva en tiempo real que permite a los pacientes y fisioterapeutas mejorar la técnica y aprovechar al máximo estos ejercicios para obtener unos mejores resultados. Además, su interfaz intuitiva y sencilla no debería suponer grandes problemas para ningún tipo de usuario, incluidos aquellos que no están tan familiarizados con la tecnología.

A nivel personal, creo que tuve suerte al encontrar este TFG. Por un lado, el hecho de desarrollar una aplicación que se va a utilizar en el campo de la medicina me hace sentir que he hecho un trabajo útil para la salud de las personas. Por otro lado, a nivel tecnológico, he aprendido y profundizado en distintos campos de la informática. El descubrimiento de programas de modelado 3D como Blender o Unity me ha abierto un mundo con el que no estaba familiarizado y he disfrutado iniciándome en él. También he aprendido como funciona una aplicación Android, la estructura general de esta y como funcionan sus servicios que permiten realizar acciones en segundo plano. La integración de la aplicación Unity en Android y el paso de datos entre ellas han sido tareas tediosas pero me han permitido entender procesos como la configuración del entorno de desarrollo, la gestión de herramientas SDK para la compilación de aplicaciones, la compatibilidad entre versiones Android de distintos programas o la depuración de problemas. Además, mis tutores D.º Mario Corrales y D.º Alfonso Bahillo me han sabido guiar durante todo el proyecto y ayudarme cuando he encontrado problemas que no sabía como abordar.

Por último, destacar que de los objetivos iniciales, no se pudo probar el trabajo en el Hospital Príncipe de Asturias, y que además, no se pudo integrar la aplicación Unity en la aplicación ya existente de este hospital si no que se realizó en una aplicación de prueba y quedó documentada. Los motivos de que dichos objetivos no se cumplieran fueron la falta de tiempo y el hecho de no contar con el código fuente de la aplicación ya existente del hospital, respectivamente.

5.1. Trabajo futuro

En esta sección se resume el trabajo futuro que queda pendiente:

- En primer lugar, como ya se ha indicado, el retraso de la nueva versión de la sonda vaginal, ha hecho que se trabaje con una versión anticuada. Para probar la aplicación con esta nueva sonda, sería necesario adaptar el modelo para asociar los músculos de este a los detectados por los distintos sensores de la sonda.
- En segundo lugar, es necesario integrar el modelo en la aplicación Android ya existente. Para ello, bastaría con seguir los pasos del Apéndice A, donde se describe la integración en una sencilla aplicación Android de prueba.
- En tercer lugar, se debería probar la aplicación en un entorno real, para valorar la utilidad que encuentran los pacientes y fisioterapeutas en ella y corregir posibles problemas.
- Por último, faltaría sustituir la lectura de datos a través de un fichero por la conexión Bluetooth con la sonda vaginal que ya está funcionando en la aplicación actual del Hospital.

Apéndices

Apéndice A

Manual de integración

En la primera parte de este manual se explicará como se exporta directamente nuestra aplicación Unity a Android, en la segunda veremos como integrar nuestra aplicación a una sencilla app en Android Nativo y en la última se detalla el proceso de comunicación entre ambas aplicaciones en tiempo real a través de un servicio Android.

A.1. Exportar aplicación Unity a Android

Este proceso es muy sencillo ya que Unity cuenta con herramientas que lo realizan automáticamente. Una vez que tengamos nuestra aplicación desarrollada completamente, basta con ir a la barra superior de herramientas del menú Unity, y en el apartado *Build Settings...* seleccionar la plataforma Android en vez de la plataforma Windows, Mac, Os que viene por defecto, como muestra la Figura A.1.

Si no tenemos instalado el plugin necesario, lo hacemos a través del botón *Install with Unity Hub*. Una vez hecho, cambiamos el entorno pulsando el botón *Switch platform*. Ahora, en el menú *Player Settings* situado en la parte inferior izquierda de la anterior ventana, ajustamos los detalles de nuestra aplicación como nombre, versión o icono, como se ve en la Figura A.2.

Cuando esté todo listo pulsaremos el botón *Build* y esto construirá automáticamente un APK en la ruta que indiquemos. Trasladando este APK a nuestro móvil e instalándolo podremos utilizar nuestra aplicación perfectamente.

A.2. Integrar aplicación Unity en otra aplicación Android Nativo

Este proceso es más largo y complejo que el anterior, pues a pesar de que Unity tiene herramientas para facilitarlos, hará falta utilizar Android Studio y modificar diversos scripts. Para esta sección he seguido algunos tutoriales [33] [34] que aunque difieran en detalles debido a los cambios de versiones han sido indispensables para conseguir el resultado.

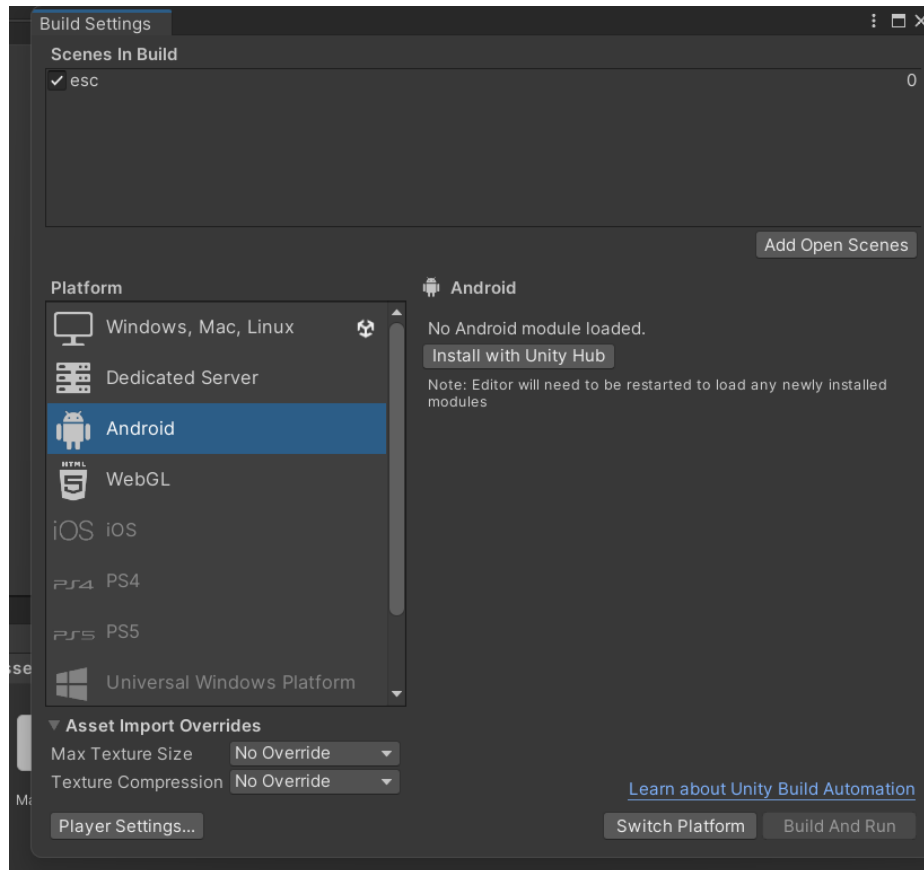


Figura A.1: Menú Build Settings en Unity

Los pasos que se seguirán son: exportar nuestra aplicación Unity como una librería en Android, crear una aplicación sencilla en Android Studio, importar nuestra librería en esta aplicación, configurar el acceso a nuestra aplicación a través de la app Android y configurar el paso de datos entre las dos aplicaciones. Aunque en esta sección se detalle todo el proceso, a la hora de integrarlo no hará falta hacer los primeros pasos, pues en el repositorio OneDrive de este proyecto [35] se encuentra la aplicación Unity exportada como librería Android.

En primer lugar, antes de exportar nuestra aplicación Unity, debemos realizar unas pequeñas modificaciones que nos permitirán posteriormente comprobar el intercambio de datos entre esta y la aplicación Android. Este paso de datos es crucial, ya que es la aplicación Android la que dispone de los datos de presión e identificadores de sensores con los que nuestra aplicación mostrará los resultados a través del modelo tridimensional. Para visualizar este intercambio de datos, debemos crear un script asociado al objeto Canvas de nuestra aplicación y crear en él un InputField que al crearse llamará a un método GetDataForUnity, que se definirá más tarde. De esta manera, al abrir nuestra aplicación Unity desde la aplicación Android, se mostrará en el InputField los datos introducidos previamente. El código de este script puede ser algo como el mostrado en el Listing A.1.

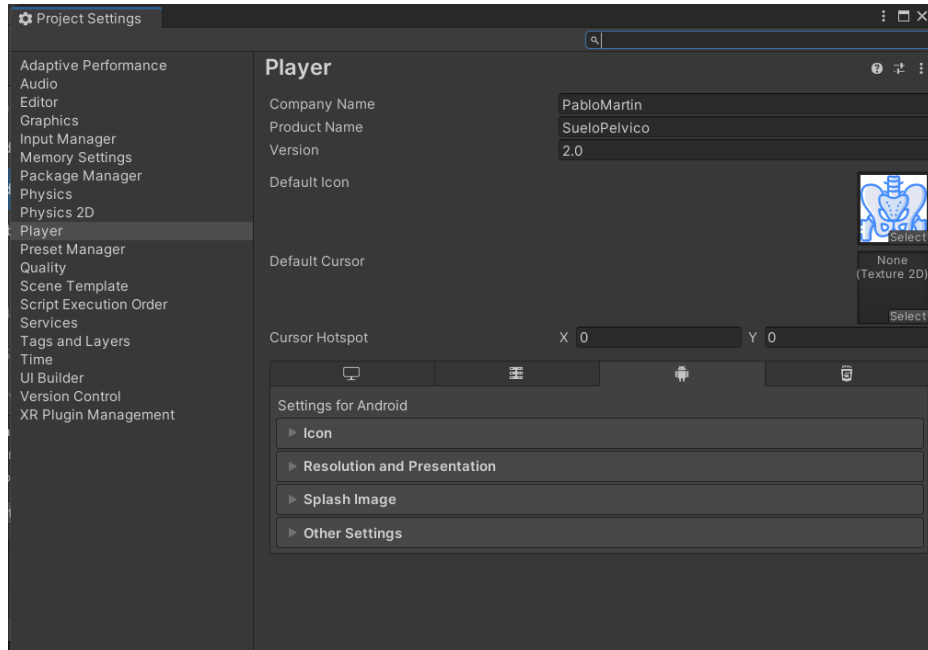


Figura A.2: Menú Player Settings en Unity

```
1 public InputField fld;
2 void Start(){
3     AndroidJavaClass ajc = new AndroidJavaClass("com.unity3d.player.UnityPlayer");
4     AndroidJavaObject ajo = new AndroidJavaObject("currentActivity");
5     fld.text = ajo.Call<string>("GetDataForUnity");
6 }
```

Listing A.1: Métodos para pasar datos en Unity

Para exportar ahora nuestra aplicación, tenemos que modificar algunos ajustes de configuración en el menú *Build Settings*.... Accederemos al menú *Player Settings* que hay en este y luego a *Player* y *Configuration*. Buscaremos la opción que dice *Scripting Backend* y elegiremos la opción *IL2CPP* como compilador back-end. También nos aseguraremos de que en *Target Architectures*, *ARMv7* esté marcado, y seleccionaremos a mayores *ARM64* (Figura A.3).

Con estos cambios realizados, volvemos al menú *Build Settings*... y seleccionamos la opción que dice *Exportar proyecto*. Después pulsamos *Export*. Esto generará una carpeta con nuestra aplicación en la ruta seleccionada. Esta carpeta se encuentra en el repositorio OneDrive de este proyecto [35] lista para su uso para no tener que realizar estos pasos previos.

Ahora crearemos una aplicación sencilla en Android Nativo para simular la app de la que dispone el hospital e integrar en ella nuestra aplicación Unity. Para esto utilizaremos Android Studio. Al iniciar el programa, elegiremos crear un nuevo proyecto y seleccionaremos la opción *Empty Activity* (Figura A.4) y el lenguaje Java. También se puede escoger Kotlin, similar a Java, pero el código utilizado en esta sección es Java. Si se desea emplear Kotlin, el propio Android Studio puede transcribir el código a dicho lenguaje.

A.2. Integrar aplicación Unity en otra aplicación Android Nativo

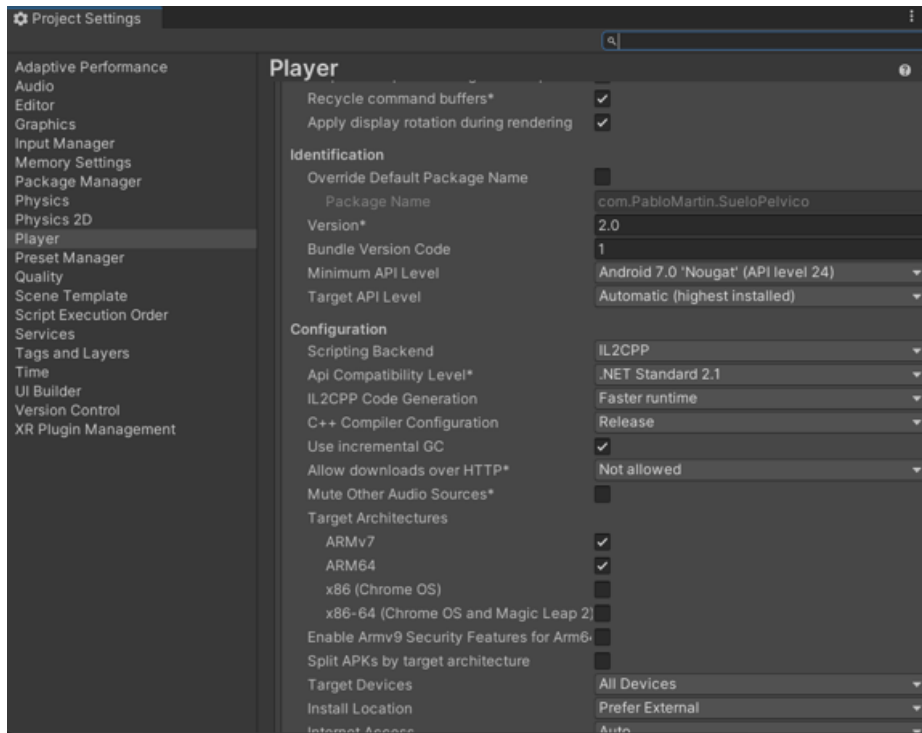


Figura A.3: Menú Configuration en Player Settings

Lo primero que haremos será abrir el script *settings.gradle* a través del menú que muestra la estructura del proyecto en la parte izquierda de la pantalla. En este script, añadiremos al final del todo las dos líneas A.2 tal y como se muestra en la Figura A.5.

```
1 include ':unityLibrary'  
2 project(':unityLibrary').projectDir=new  
   File('...Path...\AndroidStudioProjects\AndroidProject\sueloPelvico')
```

Listing A.2: Script settings.gradle

Con estas líneas, podremos leer posteriormente la librería con nuestra aplicación Unity, en este caso denominada *sueloPelvico*. Es importante cambiar la ruta al lugar en el que tenemos nuestros proyectos Android Studio tal y como se muestra en la Figura A.5.

A continuación, en el script *build.gradle(Module:app)* y copiamos las líneas de código A.3 en la sección de dependencias como se muestra en la Figura A.6.

```
1 implementation project(':unityLibrary')  
2 implementation fileTree(dir: project(':unityLibrary').getProjectDir().toString() + ('\\libs'), include:  
   ['*.jar'])
```

Listing A.3: Script build.gradle(Module:app)

Tras estos pasos podemos construir el proyecto ejecutando en la parte superior de la pantalla. Si con el explorador de Documentos vamos a la ruta en la que tenemos nuestro proyecto Android Studio, veremos que aparece una carpeta denominada *sueloPelvico* que hemos creado con el código anterior

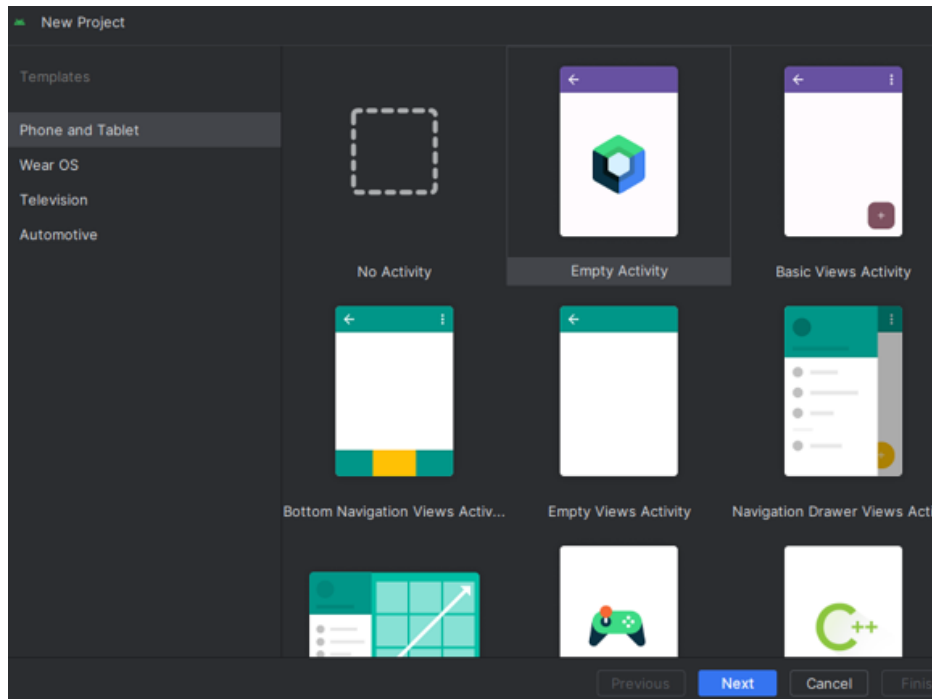


Figura A.4: New Project en Android Studio

```
14 }
15
16 rootProject.name = "My Application"
17 include ':app'
18
19 include ':unityLibrary'
20 project(':unityLibrary').projectDir=new File('C:\\Users\\M1503\\AndroidStudioProjects\\MyApplication\\sueloPelvico')
21
```

Figura A.5: Script settings.gradle

pero que está vacía. Lo que debemos hacer, es sustituir esta carpeta por la librería Unity que habíamos exportado desde Unity. Para ello, en la carpeta generada por Unity, copiamos únicamente la carpeta *sueloPelvico* como se puede ver en la Figura A.7.

Debemos copiar esta carpeta y cambiarla por la que tenemos vacía en la ubicación del proyecto Android Studio.

De vuelta a Android Studio, si intentamos construir de nuevo el proyecto podemos obtener un error generado por fallos de dependencias con los servicios de Google y Unity. Esto se soluciona copiando la línea de código A.4 en el script *gradle.properties(Project properties)* como muestra la Figura A.8.

```
1 unityStreamingAssets=.unity3d, google-services-desktop.json, google-services.json,
  GoogleService-Info.plist
```

Listing A.4: Script gradle.properties(Project properties)

Ahora, copiaremos el Código A.5 en *build.gradle(module: unityLibrary)* tal y como muestra la línea 14 de la Figura A.9.

```

build.gradle • settings.gradle • UnityPlayerHelper.java
C:\Users\M1503> AndroidStudioProjects> MyApplication> app> build.gradle
5  android {
9    defaultConfig {
14      versionName "1.0"
15
16      testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
17    }
18
19    buildTypes {
20      release {
21        minifyEnabled false
22        proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
23      }
24    }
25    compileOptions {
26      sourceCompatibility JavaVersion.VERSION_1_8
27      targetCompatibility JavaVersion.VERSION_1_8
28    }
29  }
30
31  dependencies {
32
33    implementation 'androidx.appcompat:appcompat:1.6.1'
34    implementation 'com.google.android.material:material:1.9.0'
35    implementation 'androidx.constraintlayout:constraintlayout:2.1.4'
36    testImplementation 'junit:junit:4.13.2'
37    androidTestImplementation 'androidx.test.ext:junit:1.1.5'
38    androidTestImplementation 'androidx.test.espresso:espresso-core:3.5.1'
39
40    implementation project(':unitylibrary')
41    implementation fileTree(dir: project(':unitylibrary').getProjectDir().toString() + '\\libs', include: ['*.jar'])
42  }

```

Figura A.6: Script build.gradle(Module:app)

```
1 namespace 'com.unity3d.player'
```

Listing A.5: Script build.gradle(module: unityLibrary)

A continuación, sustuiremos todo el código del script *AndroidManifest.xml* por el Código A.6. Este manifiesto es el documento que permite describir la funcionalidad y los requisitos de una aplicación Android.

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3     xmlns:tools="http://schemas.android.com/tools">
4   <uses-feature android:glEsVersion="0x00030000" />
5   <uses-feature android:name="android.hardware.vulkan.version" android:required="false" />
6   <uses-feature android:name="android.hardware.touchscreen" android:required="false" />
7   <uses-feature android:name="android.hardware.touchscreen.multitouch" android:required="false" />
8   <uses-feature android:name="android.hardware.touchscreen.multitouch.distinct" android:required="false" />
9   </uses-feature>
10  <application android:extractNativeLibs="true">
11    <meta-data android:name="unity.splash-mode" android:value="0" />
12    <meta-data android:name="unity.splash-enable" android:value="True" />
13    <meta-data android:name="unity.launch-fullscreen" android:value="True" />
14    <meta-data android:name="unity.allow-resizable-window" android:value="False" />
15    <meta-data android:name="notch.config" android:value="portrait|landscape" />
16    <meta-data android:name="unity.auto-report-fully-drawn" android:value="true" />
17    <activity android:name="com.unity3d.player.UnityPlayerActivity"
18      android:theme="@style/UnityThemeSelector" android:screenOrientation="fullUser"
19      android:launchMode="singleTask" android:configChanges=
20      "mcc|mnc|locale|touchscreen|keyboard|keyboardHidden|navigation|orientation|
21      screenLayout|uiMode|screenSize|smallestScreenSize|fontScale|layoutDirection|density"
22      android:resizeableActivity="false" android:hardwareAccelerated="false" android:exported="true">
23      <!-- <intent-filter>-->

```

Nombre	Fecha de modificación	Tipo	Tamaño
.gradle	06/03/2024 21:14	Carpeta de archivos	
.idea	06/03/2024 21:14	Carpeta de archivos	
app	06/03/2024 20:30	Carpeta de archivos	
gradle	06/03/2024 20:30	Carpeta de archivos	
sueloPelvico	06/03/2024 20:58	Carpeta de archivos	
.gitignore	06/03/2024 20:30	Archivo de origen ...	1 KB
build	06/03/2024 20:30	Archivo de origen ...	1 KB
gradle	06/03/2024 20:30	Archivo de origen ...	2 KB
gradlew	06/03/2024 20:30	Archivo	6 KB
gradlew	06/03/2024 20:30	Archivo por lotes ...	3 KB
local	06/03/2024 20:46	Archivo de origen ...	1 KB
settings	06/03/2024 20:51	Archivo de origen ...	1 KB

Figura A.7: Librería Unity exportada

```

19     <!-- <category android:name="android.intent.category.LAUNCHER" />-->
20     <!-- <action android:name="android.intent.action.MAIN" />-->
21     <!-- </intent-filter>-->
22     <meta-data android:name="unityplayer.UnityActivity" android:value="true" />
23     <meta-data android:name="notch_support" android:value="true" />
24 </activity>
25 </application>
26 </manifest>

```

Listing A.6: Script AndroidManifest.xml

Por último, pegaremos la línea de código A.7 en el script *strings.xml* de nuestra aplicación Android. Este script se encuentra en la carpeta *values*, dentro de la carpeta *res* que está en la carpeta *app*, como muestra la Figura A.10.

```

1 <string name="game_view_content_description">Game view</string>

```

Listing A.7: Script strings.xml

Al llegar a este punto, ya tendremos lista la integración de nuestra aplicación Unity en la aplicación Android. Es probable que si tratamos de construir nuestro proyecto obtengamos algún error relacionado con el NDK de Android. El NDK es un kit de desarrollo nativo que permite desarrollar software directamente desde una plataforma sin utilizar una máquina virtual [36]. Para solucionarlo, en primer lugar debemos instalar el NDK si no lo tenemos instalado, a través del propio Android Studio tal y como este nos sugiere. En segundo lugar, debemos modificar la línea del script *build.gradle(Module: unityLibrary)* que hace referencia a la lectura del NDK. Se debe sustituir la línea 68 de la Figura A.11 por la línea 69 (Listing A.8) para que el NDK se lea de la variable *ndkPath* en la que se encuentra el NDK que

A.2. Integrar aplicación Unity en otra aplicación Android Nativo

```
gradle.properties X settings.gradle UnityPlayerHelper.java
C:\> Users > M1503 > AndroidStudioProjects > MyApplication > gradle.properties
1 # Project-wide Gradle settings.
2 # IDE (e.g. Android Studio) users:
3 # Gradle settings configured through the IDE "will override"
4 # any settings specified in this file.
5 # For more details on how to configure your build environment visit
6 # http://www.gradle.org/docs/current/userguide/build_environment.html
7 # Specifies the JVM arguments used for the daemon process.
8 # The setting is particularly useful for tweaking memory settings.
9 org.gradle.jvmargs=-Xmx2048m -Dfile.encoding=UTF-8
10 # When configured, Gradle will run in incubating parallel mode.
11 # This option should only be used with decoupled projects. More details, visit
12 # http://www.gradle.org/docs/current/userguide/multi_project_builds.html#sec:decoupled_projects
13 # org.gradle.parallel=true
14 # AndroidX package structure to make it clearer which packages are bundled with the
15 # Android operating system, and which are packaged with your app's APK
16 # https://developer.android.com/topic/libraries/support-library/androidx-rn
17 android.useAndroidX=true
18 # Enables namespacing of each library's R class so that its R class includes only the
19 # resources declared in the library itself and none from the library's dependencies,
20 # thereby reducing the size of the R class for that library
21 android.nonTransitiveRClass=true
22 unityStreamingAssets=.unity3d, google-services-desktop.json, google-services.json, GoogleService-Info.pli
```

Figura A.8: Script gradle.properties(Project properties)

```
build.gradle X settings.gradle UnityPlayerHelper.java
C:\> Users > M1503 > AndroidStudioProjects > MyApplication > unityLibrary > build.gradle
1 apply plugin: 'com.android.library'
2
3
4 dependencies {
5     implementation fileTree(dir: 'libs', include: ['*.jar'])
6
7 }
8
9 android {
10     ndkPath "C:/Program Files/Unity/Hub/Editor/2022.3.20f1/Editor/Data/PlaybackEngines/AndroidPlayer/NDK"
11
12     compileSdkVersion 32
13     buildToolsVersion '32.0.0'
14     namespace 'com.unity3d.player'
15     compileOptions {
16         sourceCompatibility JavaVersion.VERSION_11
17         targetCompatibility JavaVersion.VERSION_11
18     }
19
20     defaultConfig {
21         minSdkVersion 24
22         targetSdkVersion 32
23         ndk {
24             abiFilters 'armeabi-v7a', 'arm64-v8a'
25         }
26         versionCode 1
27         versionName '0.1'
28         consumerProguardFiles 'proguard-unity.txt'
29     }
30
31     lintOptions {
32         abortOnError false
33     }
34 }
```

Figura A.9: Script build.gradle(module: unityLibrary)

acabamos de instalar. La línea a escribir es la siguiente.

```
1 CommandLineArgs.add("--tool-chain-path="+android.ndkPath)
```

Listing A.8: Script Module: unityLibrary

Con estos pasos, la integración está completa. Ahora haremos que se pueda acceder a la aplicación Unity al pulsar un botón en nuestra aplicación Android. Para ello se creará un botón nuevo o se utilizará uno ya existente. El primer paso es ir al script *activity_main.xml* en la ruta */app/java/res/layout*. Allí

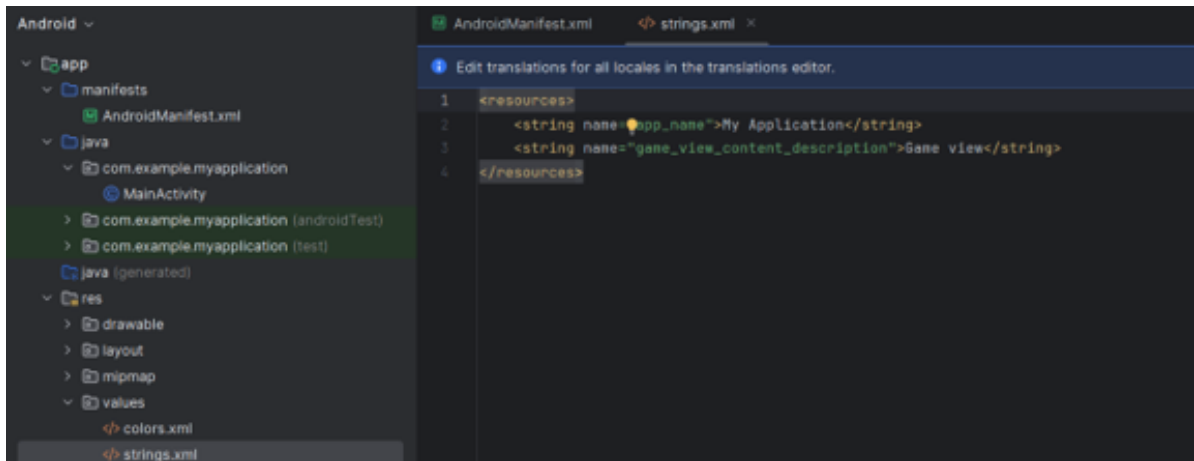


Figura A.10: Script strings.xml

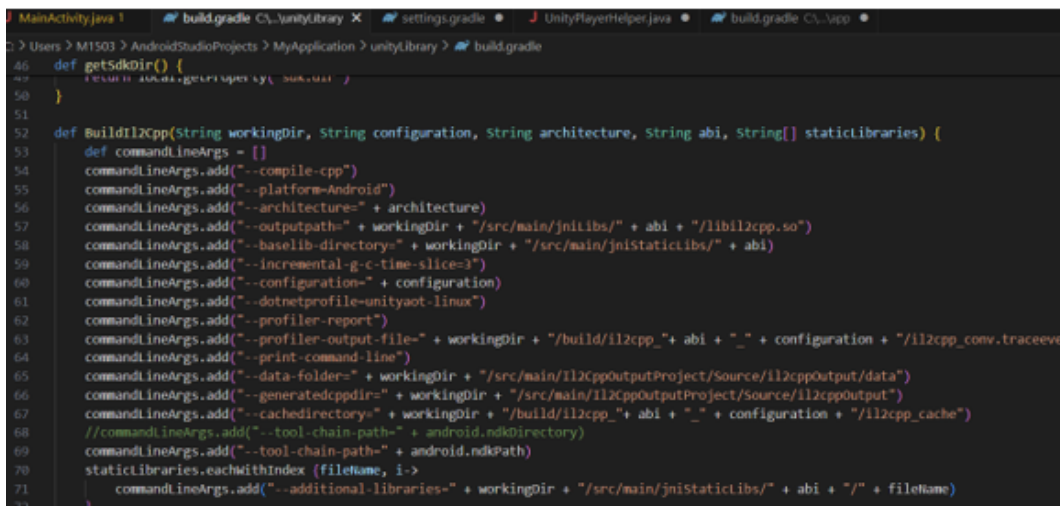


Figura A.11: Script build.gradle(Module: unityLibrary)

añadiremos un botón al que le pondremos el id “@+id/buttonClick” para poder utilizarlo después. El segundo paso será navegar hasta el script *mainActivity.java* y darle la funcionalidad que deseemos.

En este ejemplo, crearemos una función *onCreate* donde definiremos el listener que permanece escuchando desde que se crea el botón. Al pulsar el botón (método *onClick*) se generará un Intent que abrirá nuestra aplicación Unity. Además, para probar el paso de datos, crearemos un string que diga "hello from android." en la variable *result*. Así conseguiremos pasar los datos de las presiones de los sensores a la aplicación Unity.

En el método *onActivityResult*, se encuentra código para leer información proveniente de la aplicación Unity, aunque para nuestro ejemplo no se necesitará, pues la comunicación es unilateral, y Unity no le manda en ningún momento datos a Android. El código a generar en *mainActivity.java* es el mostrado en el Listing A.9.

```
1 import com.unity3d.player.UnityPlayerActivity;
2 public class MainActivity extends AppCompatActivity {
3     int LAUNCH_SECOND_ACTIVITY=1;
4 }
```

A.2. Integrar aplicación Unity en otra aplicación Android Nativo

```
5  @Override
6  protected void onCreate(Bundle savedInstanceState) {
7      super.onCreate(savedInstanceState);
8      setContentView(R.layout.activity_main);
9
10     Button buttonClick=findViewById(R.id.button);
11     buttonClick.setOnClickListener(new View.OnClickListener(){
12         @Override
13         public void onClick(View view) {
14             Intent i=new Intent(MainActivity.this, UnityPlayerActivity.class);
15             i.putExtra("result", "hello form android");
16             startActivityForResult(i, LAUNCH_SECOND_ACTIVITY);
17         }
18     });
19 }
20
21 @Override
22 protected void onActivityResult(int requestCode, int resultCode, Intent data) {
23     try {
24         super.onActivityResult(requestCode, resultCode, data);
25         if (requestCode == LAUNCH_SECOND_ACTIVITY && resultCode == RESULT_OK) {
26             String requiredValue = data.getStringExtra("result");
27             EditText edit = (EditText) findViewById(R.id.editTextText);
28             edit.setText(requiredValue);
29         }
30     }
31     catch (Exception ex) {
32         Toast.makeText(MainActivity.this,ex.toString(), Toast.LENGTH_SHORT).show();
33     }
34 }
```

Listing A.9: Script MainActivity.java

Para completar la comunicación, debemos añadir los métodos del código A.10 en la app de Unity (script *UnityPlayerActivity* en la carpeta *com.unity.3d.player*) que permitan recibir y enviar datos.

```
1  public String GetDataForUnity(){
2      return getIntent().getStringExtra("result");
3  }
4
5  public void SetDataFromUnity(String cmdLine){
6      Log.d("unity",cmdLine);
7
8      Intent returnIntent=new Intent();
9      returnIntent.putExtra("result",cmdLine);
10     setResult(Activity.RESULT_OK,returnIntent);
11     finish();
12 }
```

Listing A.10: Script UnityPlayerActivity

El método *GetDataForUnity* recibe el valor de la variable *result*. Este método es llamado en la primera parte de esta guía desde Unity para mostrarse en el *InputField* que habíamos creado. De esta forma,

podríamos pasar los datos a Unity y modificando el código en este, hacer con ellos lo que queramos.

Si probamos a ejecutar la aplicación, conectando mediante un cable USB nuestro dispositivo móvil al portátil de trabajo, aparecerán las siguientes pantallas.

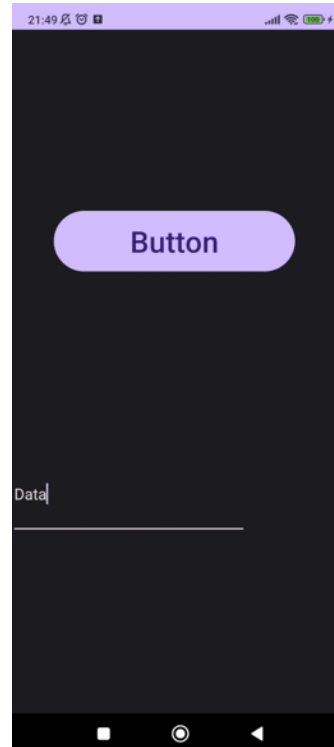


Figura A.12: Pantalla aplicación Android

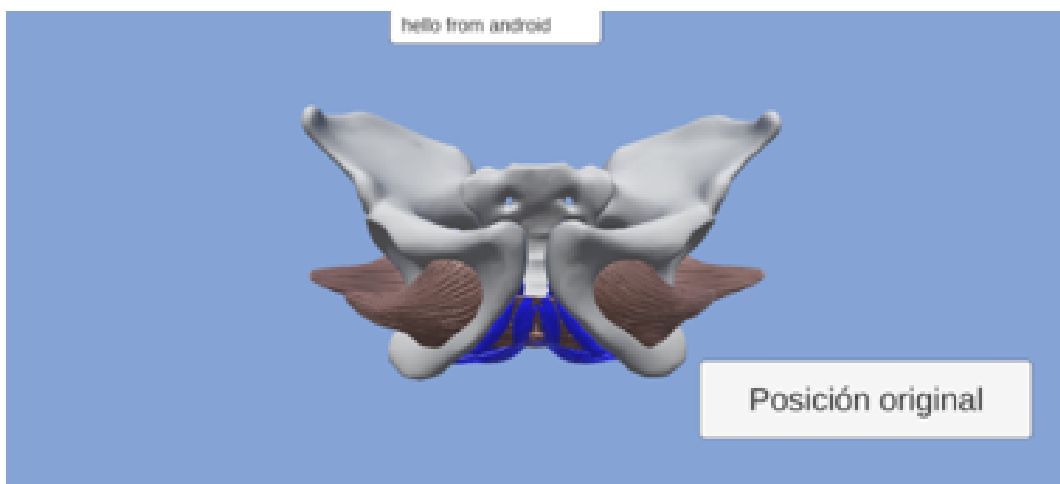


Figura A.13: Pantalla aplicación Unity

En la primera pantalla (Figura A.12) veremos la aplicación Android, con un campo para introducir datos, aunque a este no le hemos dado funcionalidad en el ejemplo. Si pulsamos el botón, se abrirá la aplicación Unity y pasaremos a la segunda pantalla (Figura A.13), en la que en la parte superior vemos el mensaje recibido desde Android.

A.3. Comunicación entre Android y Unity

En la sección anterior se describe un ejemplo simple de paso de datos entre Android y Unity, en el que desde la aplicación Android, al pulsar el botón que inicia la aplicación Unity, se envía un mensaje *hello from android* hasta Unity que lo muestra por pantalla. Este paso de mensajes no sirve para nuestro propósito, pues para enviar los mensajes, la aplicación Android necesita tener el foco y no estar en segundo plano, mientras que lo que queremos es que la aplicación Android, que estará constantemente recibiendo datos de la sonda vaginal vía Bluetooth, sea capaz de enviar estos datos a Unity mientras está se ejecuta, es decir, en segundo plano.

Para conseguir esto, se utilizará un servicio de Android. Los servicios Android son programas que se ejecutan en segundo plano sin interactuar con el usuario, por lo que no tienen interfaz gráfica [37]. Como durante el desarrollo del TFG no se dispone de la sonda vaginal, se simulará su funcionamiento mediante la lectura de datos de un archivo Excel con intervalos de medio segundo. Los datos simulados en este archivo, provienen de un fichero de pruebas facilitado por el Hospital Príncipe de Asturias, que recoge las presiones captadas por los sensores durante la realización de un entrenamiento.

Los siguientes pasos se han conseguido siguiendo diversos tutoriales de Internet sin los que hubiera sido imposible conseguir el trabajo [33] [34].

El primer paso para poner en marcha nuestro servicio será crear una clase Java en nuestro proyecto Android que se encargue de leer periódicamente una fila de un fichero, convertirla a String y avisar por medio de un Intent que hay un nuevo mensaje para que los Receivers suscritos a ese servicio reciban la notificación. El código completo de esta clase se encuentra en el repositorio OneDrive del proyecto junto con todo el proyecto Android.

```

1 private void sendBroadcastMessage(String message) {
2     Intent intent = new Intent("com.example.myapplication.UPDATE");
3     intent.putExtra("counter", message);
4     sendBroadcast(intent);
5     Log.d("MyCounterService", "Message sent: " + message);
6 }

```

Listing A.11: Método sendBroadcastMessage

En el código A.11, se encuentra el método encargado de notificar la actualización a través de la función sendBroadcast y el uso de un Intent. La lectura de datos de un archivo Excel se realiza mediante las librerías apache.poi, tal como muestran los códigos A.12 y A.13.

```

1 private void loadExcelFile() {
2     try {
3         File file = new File(getExternalFilesDir(null), "data.xls");
4         if (!file.exists()) {
5             Log.e("MyCounterService", "El archivo data.xls no se encuentra en " +
6                 file.getAbsolutePath());
7             return;
8         }
9         FileInputStream fis = new FileInputStream(file);

```



```
9     POIFSFileSystem fileSystem = new POIFSFileSystem(fis);
10     HSSFWorkbook workbook = new HSSFWorkbook(fileSystem);
11     sheet = workbook.getSheetAt(0);
12
13     fis.close();
14 } catch (IOException e) {
15     e.printStackTrace();
16 }
17 }
```

Listing A.12: Lectura de fichero mediante apache.poi

```
1 private void startCounter() {
2     handler.postDelayed(new Runnable() {
3         @Override
4         public void run() {
5             if (sheet != null && rowIndex <= sheet.getLastRowNum()) {
6                 HSSFRow row = sheet.getRow(rowIndex++);
7                 if (row != null) {
8                     StringBuilder rowData = new StringBuilder();
9                     for (int i = 0; i < row.getPhysicalNumberOfCells(); i++) {
10                        HSSFCell cell = row.getCell(i);
11                        if (cell != null) {
12                            rowData.append(cell.toString()).append(";");
13                        }
14                    }
15                    sendBroadcastMessage(rowData.toString());
16                }
17            }
18            handler.postDelayed(this, 500);
19        }
20    }, 500);
21 }
```

Listing A.13: Lectura periódica de líneas

El siguiente paso es crear una clase Java encargada de recibir las notificaciones de este servicio. Esta clase debe hallarse en la aplicación Unity dentro de nuestro proyecto, y tendrá un método para registrar el Receiver la primera vez que se lanza y otro método para recoger los datos enviados por el servicio y mandárselos a Unity.

```
1 public static void registerReceiver(Context context) {
2     if (instance == null) {
3         instance = new CounterReceiver();
4         IntentFilter filter = new IntentFilter("com.example.myapplication.UPDATE");
5         context.registerReceiver(instance, filter);
6     }
7 }
8
9 @Override
10 public void onReceive(Context context, Intent intent) {
11     if (intent.getAction().equals("com.example.myapplication.UPDATE")) {
```

```

12     String counter = intent.getStringExtra("counter");
13     UnityPlayer.UnitySendMessage("Canvas", "OnCounterUpdate", counter);
14     Log.d("CounterReceiver", "Message sent");
15 }
16 }

```

Listing A.14: Métodos del BroadcastReceiver

En el código A.14, vemos ambos métodos. Por un lado, `registerReceiver` crea un `Receiver` que se asocia al filtro del servicio previamente creado. Por otro lado, `onReceive` recibe las notificaciones del servicio `UPDATE`, y envía el contenido de estas a un objeto de Unity mediante el método `UnitySendMessage`. Por tanto, es importante tener un objeto en Unity (en este caso el propio `Canvas` de la aplicación) que cuente con métodos para recibir estos datos y procesarlos. Estas funciones se muestran en el código A.15.

```

1 void Start() {
2     if (Application.platform == RuntimePlatform.Android) {
3         AndroidJavaClass unityPlayer = new AndroidJavaClass("com.unity3d.player.UnityPlayer");
4         unityActivity = unityPlayer.GetStatic<AndroidJavaObject>("currentActivity");
5
6         // Registrar el BroadcastReceiver
7         AndroidJavaClass pluginClass = new
8             AndroidJavaClass("com.example.unityplugin.CounterReceiver");
9         pluginClass.CallStatic("registerReceiver", unityActivity);
10        Debug.Log("Receiver registred");
11    }
12 }
13
14 // Metodo que se llamar desde el BroadcastReceiver
15 public void OnCounterUpdate(string counterValue) {
16
17     // Separar la lnea en columnas
18     string[] columns = counterValue.Split(';');
19
20     // Utilizar los datos para actualizar el modelo tridimensional
21 }

```

Listing A.15: Métodos en Unity para recibir el servicio Android

Por último, para que nuestro servicio funcione, es importante registrarlo en el Manifiesto Android de la aplicación Android, y registrar el `Receiver` en el Manifiesto Android de la aplicación Unity. También debemos dar permisos a la aplicación Android para poder leer archivos externos, es decir, nuestro fichero de datos. Todo esto se muestra en los códigos A.16 y A.17.

```

1 <service android:enabled="true" android:name="com.example.myapplication.MyCounterService" />
2 <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />

```

Listing A.16: Registro del servicio en Manifiesto Android y actualización de permisos

```
1 <receiver android:name="com.example.unityplugin.CounterReceiver"
2     android:exported="true">
3     <intent-filter>
4         <action android:name="com.example.myapplication.UPDATE" />
5     </intent-filter>
6 </receiver>
```

Listing A.17: Registro del Receiver en Manifiesto Android

De esta forma queda expuesta la integración de una aplicación Unity en una en Android Nativo y un ejemplo de como pueden comunicarse en tiempo real mediante el uso de un servicio Android.

Apéndice B

Manual de despliegue

En este manual se explica el proceso para desplegar la aplicación para su desarrollo o instalación. En primer lugar se debe acceder al repositorio de OneDrive de la UVA con los archivos necesarios. Este es el enlace:

OneDrive-UVA

En este enlace encontraremos dos carpetas y un fichero:

- **Unity:** Esta es la aplicación Unity exportada sin su integración en Android. Es necesaria para su posterior integración en la aplicación del Hospital de Alcalá de Henares, pero no para este despliegue.
- **Android:** Esta es la carpeta que debemos descargar, en la que ya se encuentra integrada la aplicación Unity a la aplicación Android de prueba.
- **data.xls:** Este fichero contiene datos utilizados para las pruebas realizadas pertenecientes a una sesión de ejercicios real, aportación del Hospital.

Los pasos son los siguientes:

1. Descargar e instalar Android Studio.
2. Descargar el proyecto del repositorio que se indica en el enlace anterior y descomprimirlo.
3. Abrir el proyecto desde Android Studio.
4. Compilar el proyecto pulsando el botón "Build Project".
5. Instalar la aplicación en el smartphone. Para ello, basta con conectar nuestro móvil al ordenador mediante un cable USB y pulsar en Run". Puede que previamente se necesite habilitar la opción de Depuración USB en nuestro móvil para permitir instalar aplicaciones de este método. Esto se consigue buscando el número de compilación de nuestro teléfono en Ajustes y pulsando siete

veces sobre él. Se nos abrirá el menú de Herramientas de desarrolladores, y podremos activar la Depuración USB.

6. La aplicación se ejecutará automáticamente.

Apéndice C

Manual de uso

En el siguiente manual se describe como utilizar la aplicación una vez desplegada.

Antes de comenzar a utilizar la aplicación, se necesitará colocar el fichero del que queremos leer los datos en la carpeta *Android/com.example.myapplication/files* de nuestro dispositivo móvil. En el repositorio se comparte el fichero *data.xls* utilizado en las diversas pruebas de este proyecto. Es importante que el fichero utilizado tenga el mismo nombre y formato si no se quiere modificar el método de lectura en el código.

El uso de esta aplicación es muy sencillo pero aun así se recoge en este manual. Si se está utilizando la versión del repositorio mencionado en el manual anterior, al abrirse la aplicación únicamente habrá un botón con la etiqueta "Modelo 3D". Debemos pulsarlo para ejecutar la aplicación Unity.

Automáticamente la simulación de datos a través de un fichero se comenzará a ejecutar y podremos ver como cambian los colores de los músculos en función de las presiones ejercidas en el ejercicio simulado. Podemos rotar el modelo de forma intuitiva y devolverlo a su posición original con el botón "Posición original". Para volver a la aplicación anterior basta con pulsar el botón "Volver".

Cuando se haya integrado la aplicación Unity en la ya existente del Hospital Alcalá de Henares, el funcionamiento será similar. Se accederá a la aplicación Unity a través de un botón y asegurandonos de que la sonda vaginal esté correctamente conectada al dispositivo mediante Bluetooth para dar el mismo funcionamiento que el fichero de la versión anterior.

Bibliografía

- [1] Ejercicios de kegel. Visitado: 2024-01-06. [Online]. Available: <https://play.google.com/store/apps/details?id=com.jsdev.pfei&hl=es&gl=US&pli=1>
- [2] Propiocepción, rae. [Online]. Available: <https://dle.rae.es/propiocepci%C3%B3n>
- [3] Suelo pélvico vista frontal. Visitado: 2024-04-08. [Online]. Available: <https://fisiotorreperogil.com/index.php/fisioblog/22-que-es-el-suelo-pelvico>
- [4] Suelo pélvico vista de perfil. Visitado: 2024-04-08. [Online]. Available: <https://espanol.babycenter.com/thread/1806929/curso-preparto-introducci%C3%B3n-y-parte-1-nociones-de-anatom%C3%ADa-y-ejercicios-de-kegel?page=2>
- [5] Emy. Visitado: 2024-01-06. [Online]. Available: <https://play.google.com/store/apps/details?id=com.fizimed.emy.app&hl=es&gl=US>
- [6] Easy kegel. Visitado: 2024-01-06. [Online]. Available: <https://play.google.com/store/apps/details?id=com.lescadeaux.kegel&hl=es&gl=US>
- [7] Ejercicios kegel - fortalecer el suelo pélvico. Visitado: 2024-01-06. [Online]. Available: <https://play.google.com/store/apps/details?id=com.stefanroobol.pelvicfloor&hl=es&gl=US>
- [8] (2020) Unity. [Online]. Available: [https://es.wikipedia.org/wiki/Unity_\(motor_de_videojuego\)](https://es.wikipedia.org/wiki/Unity_(motor_de_videojuego))
- [9] Blender. [Online]. Available: <https://es.wikipedia.org/wiki/Blender>
- [10] Android. [Online]. Available: <https://es.wikipedia.org/wiki/Android>
- [11] Java. [Online]. Available: [https://es.wikipedia.org/wiki/Java_\(lenguaje_de_programaci%C3%B3n\)](https://es.wikipedia.org/wiki/Java_(lenguaje_de_programaci%C3%B3n))
- [12] Android studio. [Online]. Available: https://es.wikipedia.org/wiki/Android_Studio
- [13] Esquema metodología iterativa e incremental. [Online]. Available: <http://alaricoi.esy.es/wp-content/uploads/2016/04/iterativo-incremental-1.png>
- [14] Especificaciones asus vivobook 15. [Online]. Available: <https://www.asus.com/es/laptops/for-home/vivobook/vivobook-15-k513-11th-gen-intel/>
- [15] Especificaciones xiaomi redmi note 15s. [Online]. Available: <https://specs-tech.com/es/xiaomi-redmi-note-15s/>

- [16] Android 12. [Online]. Available: https://www.android.com/intl/es_es/android-12/
- [17] Google chrome. [Online]. Available: https://es.wikipedia.org/wiki/Google_Chrome
- [18] Onedrive. [Online]. Available: https://es.wikipedia.org/wiki/Microsoft_OneDrive
- [19] Teams. [Online]. Available: https://en.wikipedia.org/wiki/Microsoft_Teams
- [20] Overleaf. [Online]. Available: <https://www.overleaf.com/learn>
- [21] M. O. Vivo, *Automatización de toma de flujometrías sonoras*. Universidad de Valladolid, 2023. [Online]. Available: <https://uvadoc.uva.es/handle/10324/63015>
- [22] Salario medio desarrollador junior. [Online]. Available: <https://es.talent.com/salary?job=programador+junior#:~:text=El%20salario%20programador%20junior%20promedio,hasta%20%E2%82%AC%2027.000%20al%20a%C3%B1o.>
- [23] Salario medio jefe de proyecto. [Online]. Available: <https://es.talent.com/salary?job=jefe+de+proyecto+software#:~:text=El%20salario%20jefe%20de%20proyecto,hasta%20%E2%82%AC%2047.500%20al%20a%C3%B1o.>
- [24] Consumo medio de un ordenador. [Online]. Available: <https://www.repsol.es/particulares/asesoramiento-consumo/cuanto-consume-ordenador/#:~:text=Por%20lo%20general%2C%20un%20ordenador,cuenta%20el%20tiempo%20de%20uso.>
- [25] Consumo medio gas de una vivienda. [Online]. Available: [https://ganaenergia.com/blog/cual-es-consumo-promedio-gas-casa-espanola/#:~:text=En%20t%C3%A9rminos%20monetarios%20el%20consumo,%E2%82%AC%20y%20los%2080%E2%82%AC.&text=De%20hecho%2C%20la%20factura%20media,las%20tarifas%203.1%20\(o%20RL.](https://ganaenergia.com/blog/cual-es-consumo-promedio-gas-casa-espanola/#:~:text=En%20t%C3%A9rminos%20monetarios%20el%20consumo,%E2%82%AC%20y%20los%2080%E2%82%AC.&text=De%20hecho%2C%20la%20factura%20media,las%20tarifas%203.1%20(o%20RL.)
- [26] Sketchfab. Visitado: 2023-12-01. [Online]. Available: <https://sketchfab.com/3d-models/female-pelvic-floor-muscles-8565b034185c4f9988026c93b65a6f8a>
- [27] Conversor usdz-fbx. Visitado: 2023-11-02. [Online]. Available: <https://products.groupdocs.app/es/conversion/usdz-to-fbx>
- [28] Texturas gratuitas. Visitado: 2023-11-09. [Online]. Available: <https://freepbr.com/t/unity3d/>
- [29] Web personal. Aimee Hutchinson. Visitado: 2023-11-20. [Online]. Available: <https://www.aimeehutchinson.com/>
- [30] Información sobre los requisitos de un proyecto. [Online]. Available: <https://www.saraclip.com/requerimientos-de-un-proyecto/>
- [31] Tipos de requisitos de un proyecto. [Online]. Available: <https://administracionderequerimientos.wordpress.com/2014/08/26/clasificacion-y-tipos-de-requerimientos/>
- [32] Patrón observador. [Online]. Available: <https://refactoring.guru/es/design-patterns/observer>
- [33] Tutorial para integrar app unity en app android. [Online]. Available: <https://www.youtube.com/watch?v=rB52ild5A9U&t=286s>

BIBLIOGRAFÍA

- [34] Tutorial para integrar app unity en app android. [Online]. Available: <https://www.youtube.com/watch?v=Ww4NblubHTc>
- [35] Repositorio one drive. [Online]. Available: https://uvaes-my.sharepoint.com/:f/g/personal/pablo_martin_estudiantes_uva_es/EqXfYePpBrxOiaNXKQjTfVMB5fgx5LhI_CcTwH4XrfszyA?e=8TKcdX
- [36] Ndk android. [Online]. Available: https://en.wikipedia.org/wiki/Android_NDK
- [37] Servicios android. [Online]. Available: <https://dev.to/dragosb/servicios-en-android-1g9j>

