UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CARRERA EN TELECOMUNICACIONES

INFORME FINAL DEL TRABAJO DE INTEGRACIÓN CURRICULAR, MODALIDAD PRESENCIAL

TEMA:

“SISTEMA INTELIGENTE DE MONITOREO Y CONTROL DE POSTURAS CORPORALES EN LOS EJERCICIOS DE LEVANTAMIENTOS LATERALES Y PRESS MILITAR APLICADO AL GIMNASIO ZENERGYM DE LA CIUDAD DE OTAVALO”

Trabajo de titulación previo a la obtención del título de Ingeniero en Telecomunicaciones

Línea de investigación: Desarrollo, aplicación de software y cybersecurity (seguridad cibernética)

AUTOR:

Erik Alexander Flores Morales

DIRECTOR:

Ing. Carlos Alberto Vásquez Ayala, MSc.

Ibarra, julio 2024

UNIVERSIDAD TÉCNICA DEL NORTE
DIRECCIÓN DE BIBLIOTECA

1. IDENTIFICACIÓN DE LA OBRA

En cumplimiento del Art. 144 de la Ley de Educación Superior, hago la entrega del presente trabajo a la Universidad Técnica del Norte para que sea publicado en el Repositorio Digital Institucional, para lo cual pongo a disposición la siguiente información:

DATOS DE CONTACTO			
CÉDULA DE IDENTIDAD:	1004501894		
APELLIDOS Y NOMBRES:	Flores Morales Erik Alexander		
DIRECCIÓN:	Ibarra		
EMAIL:	eafloresm@utn.edu.ec		
TELÉFONO FIJO:		**TELÉFONO MÓVIL:**	0968172207

DATOS DE LA OBRA	
TÍTULO:	SISTEMA INTELIGENTE DE MONITOREO Y CONTROL DE POSTURAS CORPORALES EN LOS EJERCICIOS DE LEVANTAMIENTOS LATERALES Y PRESS MILITAR APLICADO AL GIMNASIO ZENERGYM DE LA CIUDAD DE OTAVALO.
AUTOR (ES):	Flores Morales Erik Alexander
FECHA: DD/MM/AAAA	29/07/2024
SOLO PARA TRABAJOS DE GRADO	
PROGRAMA:	■ **PREGRADO** ☐ **POSGRADO**
TITULO POR EL QUE OPTA:	**Ingeniero en Telecomunicaciones**
ASESOR /DIRECTOR:	**Ing. Carlos Alberto Vásques Ayala, MSc.**

2. CONSTANCIAS

El autor manifiesta que la obra objeto de la presente autorización es original y se la desarrolló, sin violar derechos de autor de terceros, por lo tanto, la obra es original y que es el titular de los derechos patrimoniales, por lo que asume la responsabilidad sobre el contenido de la misma y saldrá en defensa de la Universidad en caso de reclamación por parte de terceros.

Ibarra, a los 29 días del mes de julio de 2024

EL AUTOR:

Erik Alexander Flores Morales

CERTIFICACIÓN DIRECTOR DEL TRABAJO DE INTEGRACIÓN CURRICULAR

Ibarra, 29 de julio de 2024

Ing. Carlos Alberto Vásquez Ayala, MSc.

DIRECTOR DEL TRABAJO DE INTEGRACIÓN CURRICULAR

CERTIFICA:

Haber revisado el presente informe final del trabajo de Integración Curricular, mismo que se ajusta a las normas vigentes de la Universidad Técnica del Norte; en consecuencia, autorizo su presentación para los fines legales pertinentes.

Ing. Carlos Alberto Vásquez Ayala, MSc.

C.C.: 1002424982

APROBACIÓN DEL COMITÉ CALIFICADOR

El Comité Calificador del trabajo de Integración Curricular "Sistema inteligente de monitoreo y control de posturas corporales en los ejercicios de levantamientos laterales y press militar aplicado al gimnasio ZENERGYM de la ciudad de Otavalo." elaborado por Erik Alexander Flores Morales, previo a la obtención del título de Ingeniero en Telecomunicaciones, aprueba el presente informe de investigación en nombre de la Universidad Técnica del Norte:

Ing. Carlos Alberto Vásquez Ayala, MSc.

C.C.: 1002424982

Ing. Edgar Daniel Jaramillo Vinueza, Msc.

C.C.: 1001545142

DEDICATORIA

Esta Tesis va dedicada a mi padre Segundo Flores y madre Piedad Morales, por su apoyo constante e incondicional, por siempre creer en mí y enseñarme los valores de respeto, perseverancia y esfuerzo para alcanzar los objetivos de la vida.

A mis hermanos, Henry y Franklin por ser mis compañeros de vida y por su aliento y confianza en todo momento.

A mi tía Marlene Morales y familiares por ser parte de mi fuente de inspiración y fuerza para seguir adelante.

Erik Alexander Flores Morales

AGRADECIMIENTO

En primer lugar, agradezco a Dios por darme la salud y las fuerzas necesarias para culminar esta etapa de mi vida.

Agradezco profundamente a mis padres, por su esfuerzo y apoyo que me han brindado. Cada paso, cada desafío superado, ha sido posible gracias a ustedes.

A mis amigos Juan Diego, Cristian, Edin, Marlon, Steven y amiga Katherine por compartir sus conocimientos y apoyo en el transcurso de toda la carrera de estudio.

A mi amigo César Bautista, por su apoyo y por estar siempre presente en los momentos difíciles de la vida.

A todos los docentes de la carrera de ingeniería en Telecomunicaciones, por su guía y sabiduría para inspirarme a seguir adelante y alcanzar mis metas académicas.

Expreso mi más sincero agradecimiento al ingeniero Carlos Vásquez director del trabajo de titulación y al ingeniero Edgar Jaramillo asesor del trabajo de titulación cuyo conocimiento, esfuerzo y orientación fueron fundamentales para el desarrollo de este trabajo.

Expreso mi agradecimiento al ingeniero Jaime Michilena coordinador de la carrera de ingeniería en Telecomunicaciones, por su apoyo y dedicación para asegurar el éxito de todos los estudiantes.

Finalmente, agradezco a los integrantes del gimnasio ZENERGYM en especial a las instructoras Kata Estrella y Alejandra Estrella, por toda su colaboración, y apoyo para la realización de este proyecto.

Erik Alexander Flores Morales

RESUMEN EJECUTIVO

El presente trabajo de titulación aborda la problemática del sedentarismo y los problemas de salud asociados por la falta de actividad física. La creciente necesidad de cuidar la salud ha aumentado la afluencia en el gimnasio ZENERGYM. Sin embargo, la limitada cantidad de instructores impide ofrecer un seguimiento personalizado en los entrenamientos, lo que provoca que los clientes ejecuten los ejercicios de manera incorrecta, desencadenando posibles lesiones por posturas corporales inadecuadas. Además, esta situación genera malestar, inseguridad e inconformidad por parte los clientes. El objetivo general fue diseñar un sistema inteligente de monitoreo y corrección de posturas corporales en los ejercicios físicos de levantamientos laterales y press militar mediante visión artificial para mejorar el control en la ejecución de las rutinas de entrenamiento. Para el desarrollo del sistema, la metodología implementada fue el modelo en cascada, utilizando el estándar ISO/IEC/IEEE 29148:2018 para determinar los requerimientos de hardware como es el dispositivo Kinect v2 y software a utilizar como es el entorno de visual studio 2022 bajo el lenguaje de C#. Mediante el uso del dispositivo Kinect v2, se capturaron los valores de las articulaciones específicas, los cuales fueron utilizados para desarrollar un algoritmo que calcula los ángulos entre los vectores formados por las articulaciones detectadas durante la ejecución de los ejercicios en tiempo real, facilitando la identificación de posturas incorrectas y proporcionando retroalimentación detallada para su corrección. Los resultados obtenidos a partir de una encuesta dirigida a los integrantes del gimnasio ZENERGYM evidenciaron el buen desempeño del sistema en la detección y corrección de posturas incorrectas durante la ejecución de los ejercicios de levantamientos laterales y press militar en tiempo real.

Palabras clave: Kinect, sensores, articulaciones, ángulos entre articulaciones, levantamientos laterales de hombros, press militar de hombros, corrección de posturas.

ABSTRACT

This thesis addresses the problem of sedentary lifestyles and the health problems associated with the lack of physical activity. The growing need to take care of one's health has increased the number of people attending the ZENERGYM gym. However, the limited number of instructors prevents us from offering personalized follow-up in the workouts, which causes clients to perform the exercises incorrectly, triggering possible injuries due to inadequate body postures. In addition, this situation generates discomfort, insecurity and dissatisfaction on the part of the clients. The general objective was to design an intelligent system for monitoring and correcting body postures in the physical exercises of lateral raises and military press through artificial vision to improve control in the execution of training routines. For the development of the system, the methodology implemented was the waterfall model, using the ISO/IEC/IEEE 29148:2018 standard to determine the hardware requirements such as the Kinect v2 device and software to be used such as the visual studio 2022 environment under the C# language. Using the Kinect v2 device, the values of specific joints were captured, which were used to develop an algorithm that calculates the angles between the vectors formed by the joints detected during the execution of the exercises in real time, facilitating the identification of incorrect postures and providing detailed feedback for correction. The results obtained from a survey directed to the members of the ZENERGYM gym showed the good performance of the system in the detection and correction of incorrect postures during the execution of the lateral raises and military press exercises in real time.

Keywords: Kinect, sensors, joints, angles between joint, lateral shoulder raises, shoulder military press, posture correction.

LISTA DE SIGLAS

SDK. Software Development Kit (Kit de Desarrollo de Software)

IoT. Internet of Things (Internet de las cosas)

API. Application Programming Interface (Interfaz de Programación de Aplicaciones)

XAML. Extensible Application Markup Language (Language de Mercado de Aplicaciones Extensibles

WPF. Windows Presentation Foundation (Fundación para la Presentación en Windows)

ÍNDICE DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

ÍNDICE DE ECUACIONES

Capítulo I: ANTECEDENTES

En el presente capítulo se dará a conocer una breve visión general a la propuesta del presente proyecto, planteando el tema, problemática, objetivos, alcance, justificación, contexto y presupuesto.

1.1 Tema

Sistema inteligente de monitoreo y control de posturas corporales en los ejercicios de levantamientos laterales y press militar aplicado al Gimnasio “ZENER GYM” de la ciudad de Otavalo.

1.2 Problema

La salud es un estado de bienestar físico, mental y social, la cual debe tener en cuenta el estado general del cuerpo y la mente. El bienestar mental no puede separarse del bienestar físico. La salud y la enfermedad física y mental representan partes esenciales de la vida que están profundamente interrelacionadas (Gómez Ayala, 2007).

Diferentes investigaciones auspiciadas por la Organización Mundial de la Salud (OMS) han puesto de manifiesto la interrelación entre salud física y salud mental, habiéndose comprobado que la existencia de graves enfermedades físicas influye en el estado mental del afectado y su familia; al mismo tiempo, también se ha puesto de manifiesto la importancia que tienen la salud mental y la salud física en el bienestar global del sujeto (Gómez Ayala, 2007). Remor & Pérez-Llantada Rueda (2007) en su estudio de la relación entre niveles de la actividad física y la experiencia de estrés y de síntomas de mal estar físico, mencionan que estudios epidemiológicos comparando personas físicamente activas con aquellas sedentarias, han observado un aumento en el riesgo de mortalidad de 1.2 a 2 veces más probabilidad de fallecimiento para aquellos individuos no activos físicamente.

De igual manera el Plan de Creación de Oportunidades por medio del eje social, menciona que “en el Ecuador el limitado acceso a servicios de salud se refleja en el bienestar de la sociedad, observándose problemáticas asociadas al consumo de drogas, problemas nutricionales, sedentarismo, suicidios en adolescentes, entre otros” (Secretaría Nacional de Planificación, 2021, p. 62).

Por ende, es importante reconocer que la práctica regular de ejercicio físico no solo mejora la salud física, sino que también contribuye a aumentar la satisfacción con la vida al proporcionar una sensación general de bienestar (De La Guardia Gutiérrez et al., 2020).

Por consiguiente, cada vez más personas están tomando consciencia de que hacer ejercicio o actividad física trae múltiples beneficios para la salud física y mental, implicando mayor afluencia en los centros de acondicionamiento físico.

En este contexto, se tuvo un acercamiento con los gerentes de los gimnasios, Fuerza Extrema, TITAN GYM y ZENER GYM de la ciudad Otavalo los cuales revelaron que la mayor afluencia de personas en los gimnasios se basa en el interés creciente en el cuidado de la salud por medio del ejercicio físico y el servicio que pueda ofrecer cada gimnasio.

Sin embargo, el incremento significativo en la afluencia de personas ha generado la necesidad de ampliar los espacios de entrenamiento, lo que conlleva a una gestión deficiente en la planificación de rutinas de entrenamiento para cada cliente, afectando negativamente el servicio ofrecido.

En el gimnasio “ZENERGYM” de la ciudad de Otavalo la señorita Gerente e Instructora Katerine supo manifestar que la problemática del servicio ofrecido al cliente proviene de dos causas principales.

“Debido a la limitada cantidad de personal de instructores, no es posible ofrecer un seguimiento personalizado en los entrenamientos. Esto resulta en que los clientes ejecuten los ejercicios de manera incorrecta, lo que a su vez puede ocasionar lesiones de diversa gravedad debido a una posición corporal incorrecta, el sobreentrenamiento y la falta de estímulo muscular. Además, esta situación genera malestar, inseguridad e inconformidad por parte de los clientes”

De igual manera supo mencionar que “la falta de monitoreo de la disponibilidad de espacios asignados para la realización de ejercicios físicos obliga al instructor a desplazarse constantemente para verificarlos, lo que dificulta en la planificación de las rutinas de entrenamiento”

Considerando las dos causas mencionadas, se propone el diseño de un sistema inteligente que permita monitorear la ejecución de rutinas de entrenamiento, y la detección y corrección de posturas corporales durante la ejecución de ejercicios. Con la finalidad de sustituir parcialmente la presencia del instructor y mejorar la experiencia brindada por parte del gimnasio.

1.3 Objetivos

1.3.1 Objetivo General

Diseñar un sistema inteligente de monitoreo y corrección de posturas corporales en los ejercicios físicos de levantamientos laterales y press militar mediante visión artificial para mejorar el control en la ejecución de las rutinas de entrenamientos.

1.3.2 Objetivos Específicos

- Describir información bibliográfica acerca del Fisicoculturismo en los ejercicios físicos de levantamientos laterales y press militar.
- Determinar los requerimientos de software y hardware necesarios para el desarrollo del diseño del sistema inteligente.
- Diseñar un sistema inteligente el cual permita captar los movimientos corporales emitidos por los clientes, a fin de monitorear la ejecución de rutinas de entrenamiento y corregir las posturas corporales.
- Realizar pruebas de funcionamiento del diseño del sistema inteligente y evaluar el nivel de aceptación por parte de los clientes e instructores.

1.4 Alcance

El presente proyecto tiene como finalidad, diseñar un sistema inteligente dirigido a los clientes del gimnasio “ZENER GYM” para detectar y corregir las posturas corporales durante la realización de los ejercicios de levantamientos laterales y press militar. Esto se logrará mediante la implementación de una interfaz gráfica con varias ventanas que permitirá al cliente visualizar el conteo de repeticiones en tiempo real de los ejercicios realizados y las demostraciones y explicaciones de la forma correcta que se debe ejecutar. El instructor dispondrá de una interfaz gráfica que inicialmente verificará la disponibilidad del espacio físico de entrenamiento de los ejercicios de levantamientos laterales y press militar lo cual facilitará en la planificación y asignación de rutinas de entrenamiento para cada cliente, de igual manera contará con una alerta cuando el cliente realice varias repeticiones erróneas de las dos variantes de ejercicios.

La metodología utilizada en el presente proyecto sigue un enfoque en cascada, la cual se compone de distintas fases secuenciales, brindando así una estructura organizada y coherente para el desarrollo. A continuación, se describen las 4 fases.

Como primera fase se realizará la recolección de información relacionada al Fisicoculturismo, centrándose específicamente en las características, factores del entrenamiento, métodos de entrenamiento y medios de entrenamiento enfocados a los ejercicios de levantamientos laterales y press militar.

En la segunda fase, se llevará a cabo un análisis exhaustivo del escenario actual para determinar los requerimientos de software y hardware necesarios para el desarrollo del diseño del sistema inteligente. Se detallarán todos los requisitos necesarios para el proyecto, y luego se seleccionará una plataforma en la nube que brinde el soporte para las alertas y el monitoreo de las rutinas de entrenamiento en los ejercicios de levantamientos laterales y press militar.

Una vez completadas las dos fases iniciales, se procederá al desarrollo del diseño de un sistema inteligente utilizando visión artificial mediante el dispositivo Kinect con la integración en el sistema en módulo NVIDIA Jetson Nano, esto implicará la configuración de las herramientas, entornos de desarrollo y bibliotecas. Posteriormente, se desollará una interfaz gráfica para que los clientes puedan acceder fácilmente al control y guía de las dos variantes de ejercicios de entrenamiento.

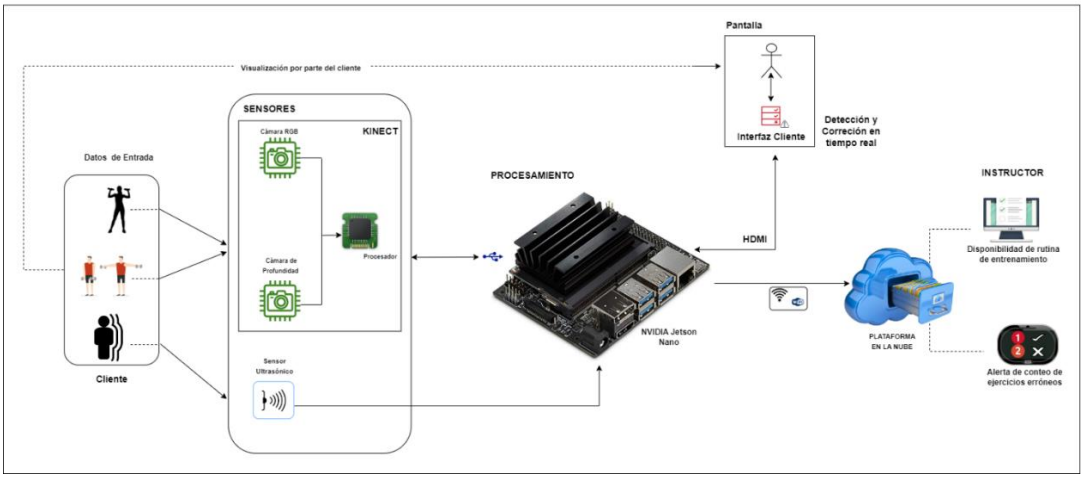
El instructor dispondrá de una interfaz gráfica que inicialmente verificará la disponibilidad del espacio físico de entrenamiento de los ejercicios de levantamientos laterales y press militar. Esto se logrará a través del uso de un sensor ultrasónico que recopilará los datos de presencia de personas, el sistema en módulo NVIDIA Jetson Nano

recibirá estos datos, los cuales se los procesará y programará para su posterior carga en la nube.

La siguiente fase consiste en realizar pruebas para asegurar que el sistema inteligente cumpla con los requerimientos iniciales. Estas pruebas se enfocarán en verificar el monitoreo de la disponibilidad del espacio físico de entrenamiento de los ejercicios de levantamientos laterales y press militar, y el control de las posturas corporales en los clientes del gimnasio "ZENER GYM". Finalmente, se llevará a cabo la evaluación del nivel de aceptación del sistema inteligente por parte de los clientes e instructores.

Figura 1

Arquitectura propuesta para el presente proyecto



Fuente: Elaborado por el autor

Nota: En la figura se plantea la arquitectura propuesta para el sistema de monitoreo y control de posturas en los ejercicios de levantamientos laterales y press militar.

1.5 Justificación

El proyecto actual surge como respuesta a la creciente necesidad de incorporar innovaciones tecnológicas en el ámbito de la salud, los centros de acondicionamiento físico y el fisicoculturismo. Su propósito principal es fomentar la actividad física y mejorar la experiencia del cliente en el gimnasio brindando un control de entrenamiento más acogedor y personalizado.

Desde 1946, cuando la Organización Mundial de la Salud (OMS) definió la salud como «el estado integral de bienestar físico, mental y social, y no solamente la ausencia de enfermedad», se ha llegado paulatinamente a entender que los enfoques terapéuticos más efectivos son los que tratan el cuerpo y la mente en conjunto (Gómez Ayala, 2007). La Organización Mundial de la Salud (OMS) proporciona información en la cual menciona que la actividad física tiene importantes beneficios para la salud del corazón, el cuerpo y la mente, lo que impulsa la búsqueda de métodos para satisfacer estas necesidades de buena salud en todas las personas (Organización Mundial de la Salud, 2022). Uno de estos métodos es la utilización de centros de acondicionamiento físico, los cuales han ganado popularidad entre personas de todas las edades que buscan mejorar su salud o mantener su bienestar. Sin embargo, es importante señalar que los centros de acondicionamiento físico pueden experimentar deficiencias en la calidad del servicio que ofrecen a sus clientes, especialmente durante los meses de alta afluencia. Estas deficiencias se reflejan en la falta de monitoreo sobre los espacios de entrenamiento, lo que genera inseguridad y malestar entre los clientes.

Las deficiencias en la calidad del servicio pueden ser problemáticas para el éxito de un negocio, ya que éste depende en esencia de la demanda de sus clientes. Por lo tanto, es fundamental atender adecuadamente esta demanda y trabajar en la gestión de la calidad, ya

que esto incide directamente en el funcionamiento y éxito de los centros de acondicionamiento (Zavala-Choez & Vélez-Moreira, 2020).

Como solución para abordar estas condiciones, en el presente proyecto se empleará el diseño de un sistema inteligente para brindar un control de entrenamiento aún más acogedor y personalizado a sus clientes, lo que a su vez conlleva a una mayor rentabilidad para el gimnasio.

Capítulo II: FUNDAMENTOS TEÓRICOS

Este capítulo presenta la base conceptual y teórica que sustenta el presente proyecto, incluyendo la fundamentación teórica acerca del fisicoculturismo, ejercicios de entrenamiento y especificaciones de entrenamientos en los ejercicios de levantamientos laterales de hombros y press militar. Además, se revisa aspectos relacionados con el hardware y software del sistema.

2.1 Fisicoculturismo

En la modernidad se está observando una tendencia, cada vez más pronunciada, a promover cambios en nuestros cuerpos, desde el más sutil, obtenido por el uso de cosméticos y el ejercicio físico moderado, hasta el más extremo como la práctica intensiva del fisicoculturismo, el tatuaje y las cirugías plásticas invasivas e irreversibles. Estos cambios en el cuerpo tienen lugar en un entorno cultural cambiante en el que la tecnología juega un rol esencial, convirtiendo al cuerpo humano que quiere transformarse, mudarse, modificarse. (Rodríguez, 2019)

El fisicoculturismo es definido por el diccionario de la Real Academia Española como la práctica de ejercicios gimnásticos encaminada al desarrollo de los músculos. Según esto, es una forma de gimnasia que pretende un cambio en la masa muscular del gimnasta, siendo esa hipertrofia (presentada como exceso) el objetivo y razón de ser de dicha disciplina. (Rodríguez, 2019)

Dicho en términos más claros: el fisicoculturismo es una actividad corporal que se enfrenta a la idea de límite. El fisicoculturismo es una actividad liminar, de intento de cambio ontológico, de una transformación del cuerpo que va más allá de la forma, llegando al desafío límite, como una superación constante. (Rodríguez, 2019)

Para conseguir la masa muscular previamente idealizada se realizarán ejercicios físicos intensos (este ejercicio es una acción localizada sobre un músculo en específico, contra una resistencia pasiva, y biomecánicamente concebida para el desarrollo de la zona muscular implicada en ella), generalmente ejercicios anaeróbicos, consistentes en movimientos musculares con carga de modo que el aumento de la carga utilizada a través de pesas y aparatos conlleve tal hipertrofia muscular. (Rodríguez, 2019)

También se puede definir el fisicoculturismo, como aquella subcultura de hombres y mujeres que, a través de programas rigurosos de entrenamiento con pesas, dietas extremas y uso de productos químicos, construyen sus cuerpos alcanzando formas y tamaños inusuales (Rodríguez, 2019).

2.1.1 Culturismo

El termino culturismo literalmente es la construcción del cuerpo, se debe al francés Marcel Rouet (1909-1982), que lo definió como el arte de practicar la cultura física en todas sus formas para alcanzar la salud y mantenerla por el equilibrio físico (Rodríguez, 2019).

El fisicoculturismo es una actividad deportiva y performativa dramatizada y sobreactuada por la cual se quiere construir un cuerpo bodybuilder, en la que el atleta se integra en una comunidad subcultural, aprendiendo de esa comunidad unos conocimientos esotéricos que pone en práctica por medio de un ejercicio disciplinado que sacraliza el dolor en el rito de las repeticiones, siendo este rito un acto simbólico, y que con ello consigue un cambio corporal. (Rodríguez, 2019)

Con relación a la progresión en los centros de acondicionamiento físico Everett (2020) en su libro de guía completa para deportistas y entrenadores menciona que no existe

una única forma en el aprendizaje de la arrancada con relación a los tiempos de progreso. Cada método que se utilice va relacionado con la tradición que se lleve a cabo en el centro de acondicionamiento físico o el gimnasio de entrenamiento, de igual manera va ligado con la necesidad de cada atleta, llegando a residir siempre en los mismos principios técnicos.

De igual manera Everett (2020) menciona que existe gran cantidad de técnicas para los diferentes levantamientos y cada entrenador enseña según sus tradiciones, tomando en cuenta que existen tipos de clientes como son los levantadores novatos los cuales se deben dar más trabajo en detalles y en los puntos más fundamentales.

2.2 Ejercicios de Entrenamiento

En el mundo de ejercitamiento físico se ha experimentado un auge en la acogida de las personas interesadas en el bienestar, aumento de fuerza y el desarrollo de la figura estética, tomando en cuenta una serie de normas generales a la hora del entrenamiento con pesas. (Buenache, 2009)

El enfoque de estudio se centrará en la práctica de ejercicios diseñados específicamente para fortalecer y desarrollar los músculos de los hombros, comprendiendo su aplicación en la rutina de entrenamiento y así mejorar el rendimiento y bienestar físico.

Los ejercicios esenciales para fortalecer los hombros y los músculos que los rodean incluyen el Press Militar de Hombros con Mancuernas y los Levantamientos Laterales con Mancuernas mencionó la señorita Alejandra Estrella instructora del gimnasio ZENER GYM. Cuando se ejecutan adecuadamente, estos ejercicios no solo trabajan los deltoides, sino también los músculos estabilizadores y otros grupos musculares complementarios.

2.3 Ergonomía

Ergonomía es la disciplina científica relacionada con la comprensión de las interacciones entre los seres humanos y los elementos de un sistema, y la profesión que aplica teórica, principios, datos y métodos de diseño para optimizar el bienestar humano y todo el desempeño del sistema. (Muñoz, 2016)

2.3.1 Ergonomía Deportiva

La Ergonomía en el deporte es la agrupación de estudios a fin de aumentar el rendimiento y la salud de un deportista por medio de diferentes métodos, entre los tipos de ergonomía deportiva se tiene la ayuda de diferentes materiales como es la tecnología deportiva y el diseño de materiales. («La ergonomía en el deporte», 2014)

2.3.1.1 Tecnología deportiva

La Ergonomía en el deporte es la agrupación de estudios a fin de aumentar el rendimiento y la salud de un deportista por medio de diferentes métodos, entre los tipos de ergonomía deportiva se tiene la ayuda de diferentes materiales como es la tecnología deportiva y el diseño de materiales. («La ergonomía en el deporte», 2014)

Diseño de materiales

Se definen diferentes fases para tener un rendimiento adecuado por parte de los deportistas, entre estos:

- Como primer punto la corrección de la técnica
- El uso de materiales
- Equipamiento y seguimiento necesario

2.4 Biomecánica Deportiva

En el libro Biomecánica deportiva y control de entrenamiento, Suarez (2009) menciona que la biomecánica es definida como la ciencia que analiza las fuerzas internas y externas que actúan sobre el cuerpo humano y el efecto que estas generan.

La biomecánica deportiva es una de las aplicaciones de la biomecánica como parte de la investigación de los movimientos de un deportista en la realización de diferentes ejercicios físicos. La evaluación del gesto deportivo, el análisis de la práctica deportiva para mejorar su rendimiento, desarrollar técnicas de entrenamiento, contar con materiales y equipos adecuados, son algunos de los objetivos de la biomecánica deportiva. (Perdomo Ogando et al., 2018)

La Biomecánica deportiva contribuye a una comprensión a detalle de las actividades y ejercicios, así mismo tiempo que interviene en la prevención de lesiones, mejora del rendimiento y el desarrollo de nuevos materiales de entrenamiento. (Perdomo Ogando et al., 2018)

Los principales aportes de la biomecánica deportiva están relacionaos con la solución de:

- Corrección de posturas corporales.
- Prevención de lesiones y dolores de articulaciones.
- Reducción de Fatiga.
- Mejora de rendimiento a corto y largo plazo.
- Diseño y determinación de técnicas e indicadores corporales.

Antes de adentrarnos a la biomecánica del hombro y codo los cuales participan en la ejecución de los ejercicios como el press militar y levantamientos laterales, es importante revisar los planos y ejes corporales para entender cómo se llevan a cabo estos movimientos

2.5 Ejes y planos corporales

Partimos de la posición anatómica, el sujeto aparece erguido, con los miembros inferiores y superiores apegados al cuerpo y con las palmas de las manos mirando al frente como se muestra en la Figura 2.

2.5.1 Ejes corporales

2.5.1.1 Eje Vertical

Es aquel que recorre el cuerpo en toda su longitud, pasando desde la cabeza hasta los pies (Navarro, 2007).

2.5.1.2 Eje Transversal

Atraviesa transversalmente el cuerpo, perpendicularmente al eje vertical (Navarro, 2007).

2.5.1.3 Eje Anteroposterior

Atraviesa de delante a atrás el cuerpo de forma perpendicular a los dos anteriores (Navarro, 2007).

2.5.2 Planos corporales

2.5.2.1 Plano Sagital

Plano compuesto por los ejes vertical y anteroposterior, permite ubicar una estructura anatómica ya sea en situación lateral o medial. Por ejemplo, el hombro, se localiza en situación lateral al esternón (Navarro, 2007).

2.5.2.2 Plano Transversal

Plano compuesto por el cruce de los ejes transversales y anteroposterior, es perpendicular al plano sagital. Según este plano, una estructura anatómica podrá situarse en la parte superior o en la parte inferior (Navarro, 2007).

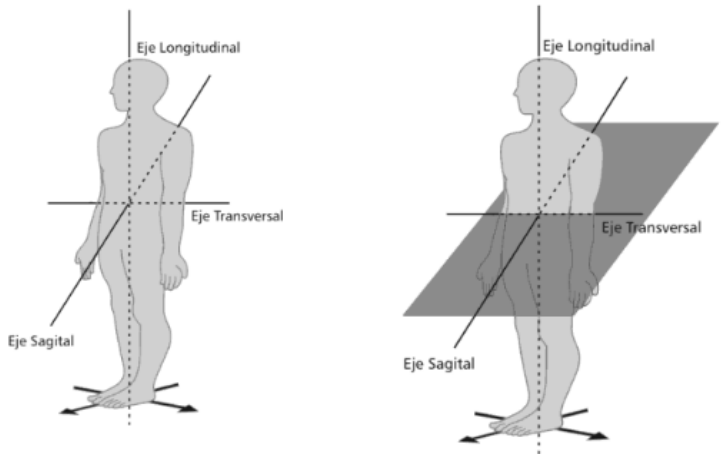
2.5.2.3 Plano Frontal

Plano compuesto por la unión de los ejes vertical y transversal. Según este plano una estructura anatómica podría situarse hacia delante o hacia atrás (Navarro, 2007).

En la Figura 2 se muestra una representación de los ejes y planos corporales en una persona.

Figura 2

Ejes y Planos corporales



Fuente: Elaborado por (Navarro, 2007)

2.5.3 Biomecánica del hombro

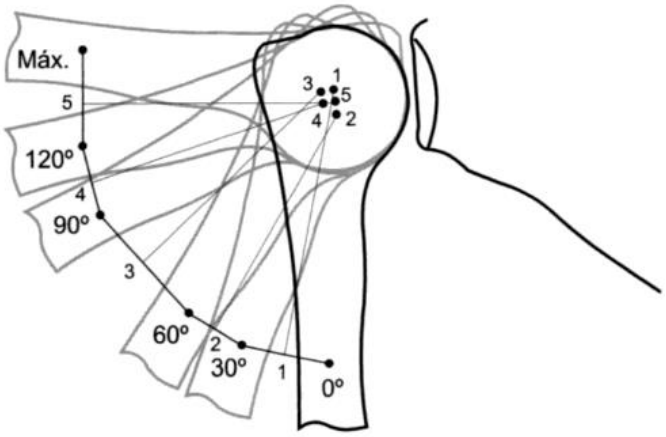
El hombro es muy reconocido por su gran capacidad de movimiento, se considera la articulación más móvil del cuerpo humano, pero también la más inestable (Sanabria & Patiño, 2013).

Conocer el centro instantáneo de rotación (CIR) es importante para comprender como los músculos realizan diversas acciones (Voegeli, 2000).

La localización varía según los distintos autores. Para la articulación escapulohumeral, algunos lo sitúan en el cuello anatómico, otro por dentro y debajo del troquiter, y otros, basándose en análisis cinemáticos, lo sitúan en la mitad inferior de la cabeza humeral si la aducción es de 0 a 50° y cuando la abducción es de 50 a 90° en la mitad superior de la cabeza humeral como se muestra en la Figura 3. (Voegeli, 2000)

Figura 3

Distribución de los diferentes CIR durante la elevación del brazo en el plano escapular.



Fuente: Elaborado por (Voegeli, 2000)

2.5.3.1 Abducción

El movimiento de abducción es fundamental en la función de toda la extremidad superior, esta acción implica en la separación del brazo del tronco hasta los 90°; a partir de 90° el brazo regresa de nuevo al eje del cuerpo (Voegeli, 2000).

2.5.3.2 Aducción

La aducción hace referencia al movimiento de la extremidad superior desde una posición de 90° hacia el costado del cuerpo o regreso al eje del cuerpo (Voegeli, 2000).

2.5.4 Biomecánica del codo

La articulación del codo, aquella que une el brazo con el antebrazo, es la articulación intermedia del miembro superior. Permite al antebrazo aproximarse o separarse del brazo, posición que es importante tomar en cuenta en la ejecución de los ejercicios de hombros, este movimiento se conoce como flexoextensión. (Voegeli, 2000)

Desde el punto funcional, las articulaciones del codo está compuesta por tres articulaciones diferentes: humerocubital, humerorradial radiocubital proximal. Tanto la articulación humerocubital como la humerorradial realizan los movimientos de flexoextensión, mientras que la articulación radiocubital lleva los movimientos de pronosupinación. (Voegeli, 2000)

Con relación a las articulaciones mencionadas, son de suma importancia tomar en cuenta para la ejecución de los ejercicios de press militar y levantamientos laterales de hombros.

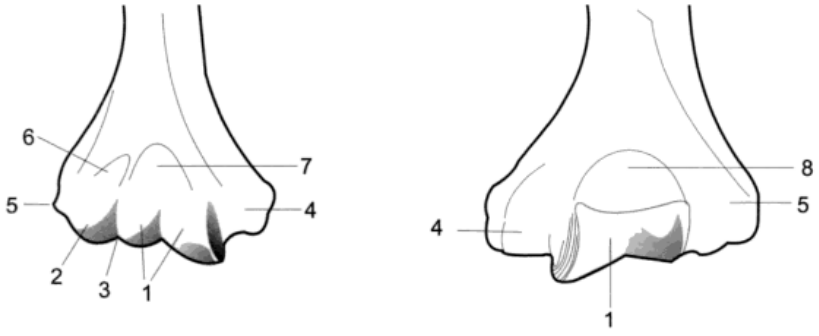
2.5.4.1 Estructuras Óseas

Como se ilustra en la Figura 4, el humero presenta distalmente dos superficies articulares, la tróclea humeral y el cóndilo del húmero o capitulum humeri. El canal cóndilo-troclear separa la superficie troclear del cóndilo humeral (Voegeli, 2000).

La tróclea humeral adopta una forma de polea asimétrica, ya que su borde medial desciende más distalmente. Está recubierta de cartílago hiliano y forma un arco de unos 330°-340°. Sus dos superficies articulares se hallan separadas por una garganta situada en el plano sagital, que no discurre paralela a éste, sino que se dispone de forma elíptica con una dirección de anterolateral a posteromedial. (Voegeli, 2000)

Figura 4

Estructura ósea del codo



Nota. En la parte derecha de muestra la vista anterior de la epífisis del humero y en la parte izquierda se muestra la vista posterior de la epífisis del húmero. Fuente: Elaborado por (Voegeli, 2000)

1. Tróclea
2. Cóndilo
3. Canal cóndilo-troclear

4. Epicóndilo medial
5. Epicóndilo lateral
6. Fosita radial
7. Fosa olecraniana

La existencia de las citadas fosas es indispensable para que los movimientos de flexoextensión del codo tengan el rango de amplitud normal (0-140/160°) (Voegeli, 2000).

2.5.4.2 Movimiento de flexoextensión

La flexión se describe como el movimiento que acerca la cara frontal del antebrazo con la cara anterior del brazo y extensión como el movimiento de retorno del antebrazo a su posición anatómica (Voegeli, 2000).

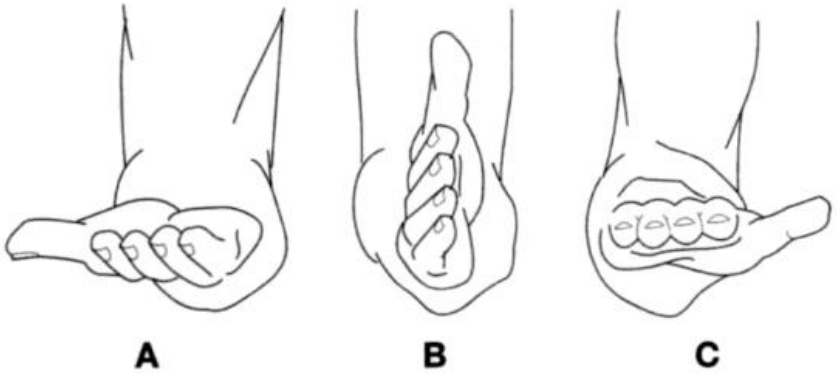
El rango de movimiento oscila entre 0° haciendo referencia a la extensión completa y 140-160° que sería la flexión máxima (Voegeli, 2000). Los movimientos de flexión están limitados por los siguientes factores:

2.5.4.3 Movimiento de pronosupinación

Estableciendo el codo a 90° de flexión la pronación es el movimiento de rotación media que sitúa el pulgar hacia dentro y la palma de la mano hacia abajo (Figura 5). La supinación implica el movimiento de llevar al pulgar hacia fuera y la palma abajo. La supinación implica llevar al pulgar hacia fuera y la palma de la mano mirando hacia arriba. Es crucial valorar estos movimientos con el codo en flexión para evitar la influencia rotacional. (Voegeli, 2000)

Figura 5

Movimientos Pronosupinación



Nota. A supinación, B posición intermedia, C Pronación. Fuente: Elaborado por (Voegeli, 2000)

2.6 Rutinas de entrenamiento

Iniciar una rutina de culturismo implica mantener un nivel constante de compromiso tanto en el entrenamiento como en el cuidado que se requiere día a día.

El entrenamiento debe ser lento, permitiendo tener un buen movimiento del músculo que se esté trabajando con el cual permita quemar muchas más calorías (Linares, 2014).

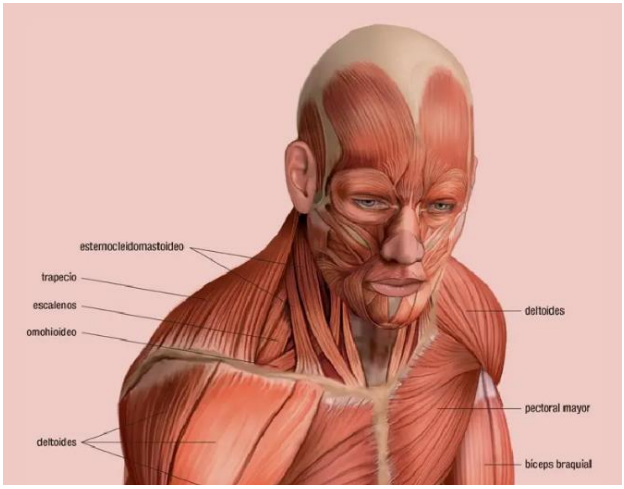
Considerando a las personas principiantes como aquellas que no tienen experiencia en el entrenamiento con pesas, todos ellos deben empezar con un programa en el que la resistencia sea ligera y centrarse en la forma apropiada de realizar los ejercicios. La adaptación debe llegar de modo gradual con el esfuerzo del entrenamiento, deben basarse en las capacidades del individuo y los diferentes objetivos, estos guiados por un instructor o persona a cargo del centro de acondicionamiento físico. (Linares, 2014)

Es importante tomar en cuenta Atlas del sistema muscular el cual se mencionará en el desarrollo de la investigación.

En la Figura 6 se puede apreciar los diferentes músculos los cuales se implican en los ejercicios enfocados en los hombros, Linares (2014) menciona que el hombro es la parte superior lateral del tronco en la cual se une el brazo con el torso y el deltoides es el músculo que cubre la articulación y que da la apariencia de redondez a esta parte del cuerpo.

Figura 6

Sistema muscular implicados en los ejercicios de hombros

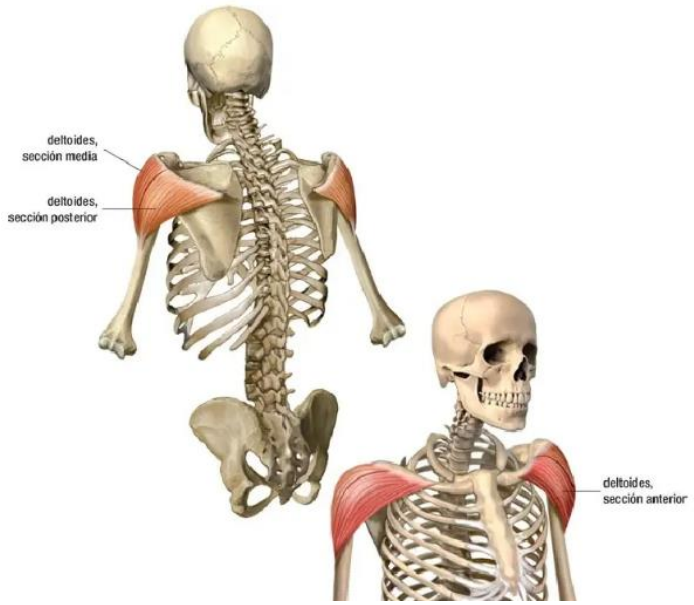


Fuente: Elaborado por (Linares, 2014)

La Figura 7 presenta las partes musculares del deltoides que se involucran en los ejercicios de press militar de hombros con mancuernas y levantamientos laterales con mancuernas que se dan a conocer a continuación.

Figura 7

Partes de Deltoides



Fuente: Elaborado por (Linares, 2014)

2.6.1 Press militar de hombros con mancuerna

El press militar es un ejercicio en el que se incluye la fuerza de torso, se considera como un ejercicio compuesto en el que trabajan la fuerza de los hombros, obteniendo beneficios como fortalecimiento del core, mejora en la movilidad, por lo cual todas las personas atletas deberían incorporar ejercicios de Press Militar en las rutinas diarias de entrenamiento. (R, 2023)

2.6.1.1 Medios de entrenamiento

Los medios de entrenamiento en los gimnasios se refieren a las diferentes herramientas, equipos o recursos con los cuales se pueda llevar a cabo un entrenamiento, los medios que se utilicen sirven para realizar los ejercicios con variedad y a diferentes intensidades.

En los ejercicios de press militar se tiene dos variantes; el press militar con barra y el press militar con mancuernas. El press militar con mancuernas es un ejercicio para el trabajo

unilateral, corregir asimetrías y puede ser más cómodo para personas que inician en el mundo del fisicoculturismo. (R, 2023)

2.6.1.2 Factores de Entrenamiento

La señorita Katherine, la cual cumple la función de instructora del Gimnasio ZENER GYM, supo manifestar que los factores de entrenamiento hacen referencia a una serie de variables que pueden ajustarse durante la rutina de entrenamiento, los cuales son muy importantes para personalizar y optimizar el programa de entrenamiento según las necesidades de los usuarios. Tomando en cuenta lo mencionado se debe hacer referencia a la intensidad, volumen y la duración en cada repetición a realizar en los ejercicios de press militar. Se debe hacer conocer al usuario que entre las repeticiones existe un tiempo de descanso y recuperación para permitir que los músculos involucrados en los ejercicios se recuperen y puedan dar el mejor rendimiento en cada repetición.

2.6.1.3 Métodos de Entrenamiento

Tomando en cuenta lo mencionado en párrafos anteriores, el press militar de hombros con mancuerna es un ejercicio de entrenamiento de fuerza centrado en desarrollar los músculos de los hombros.

La señorita Katherine instructora en el gimnasio ZENER GYM menciona que existen varios métodos de entrenamiento como son; entrenamiento de fuerza, entrenamiento de hipertrofia, entrenamiento de resistencia muscular, entrenamiento de potencia, entre otros.

En los entrenamientos de fuerza el objetivo es ganar fuerza en los músculos involucrados, utilizando un peso exigente y así alcanzar entre 1 a 6 repeticiones por serie.

En el entrenamiento de hipertrofia se pretende aumentar el tamaño de los músculos deltoides, las repeticiones a realizar por serie son entre 6 y 12 con un peso moderado según el nivel de resistencia de la persona.

En el entrenamiento de resistencia muscular el objetivo es alcanzar la mayor cantidad de repeticiones (por lo general entre 12 a 20 repeticiones por serie).

El entrenamiento de potencia se enfoca en mejorar la capacidad de generar fuerza explosiva y rápida en los músculos. Para lograrlo, es importante utilizar un peso moderado que permita realizar cada repetición de press militar con la máxima velocidad posible

2.6.1.4 Descripción de posturas de entrenamiento

La señorita Katherine instructora del gimnasio ZENER GYM manifestó las instrucciones para la ejecución del ejercicio.

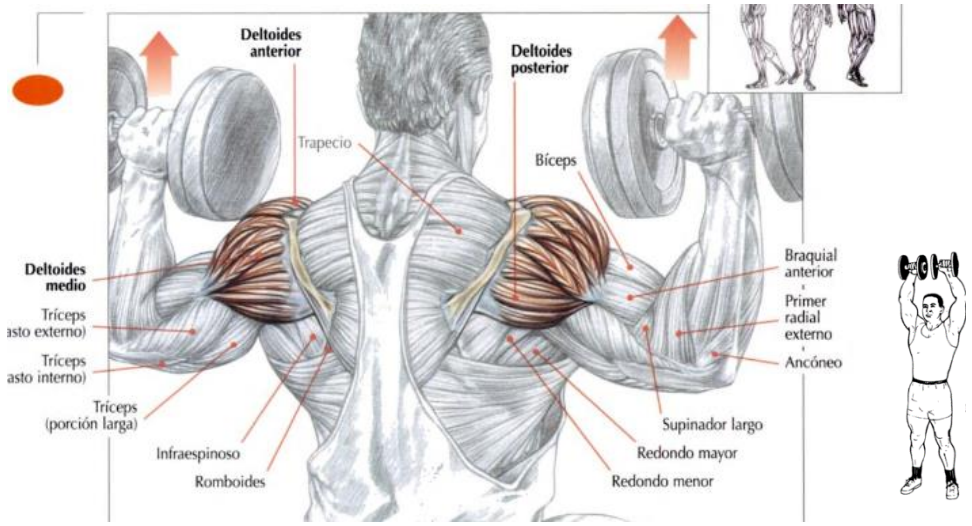
Posición inicial: Comience de pie con los pies separados a la anchura de los hombros. Elija un peso adecuado para garantizar un movimiento correcto y completo. Sujete las mancuernas, manteniendo los hombros ligeramente inclinados y alineados con el torso. Asegúrese de que las manos tengan una separación ligeramente mayor que el ancho de los hombros.

Técnica: Inicie el ejercicio extendiendo los brazos para elevar las mancuernas de manera lateral hasta el punto en el que los codos estén completamente estirados, pero no bloqueados. A continuación, realice un descenso controlado del peso hasta que las manos se encuentren por encima de los hombros. Durante todo el ejercicio, mantenga una tensión constante en los hombros y evite arquear la espalda.

Ángulos: En la posición inicial como se muestra en la figura 8, las palmas de las manos deben mirar hacia delante o modo pronación, los codos deben estar flexionados formando un ángulo entre 80 a 90 grados. En el ascenso los codos deben estar extendidos levantando las mancuernas hacia arriba sin llegar a la extensión completa, en la posición final los brazos deben estar parcialmente rectos, pero sin bloquear las articulaciones.

Figura 8

Ejecución de ejercicios de press militar



Fuente: Elaborado por (Delavier, 2007)

2.6.2 Levantamientos laterales con mancuernas

Para la ejecución del ejercicio de levantamiento laterales con mancuernas, tomando en cuenta lo explicado en la sección de biomecánica del hombro, este ejercicio implica la abducción y aducción de los brazos, el cual es un movimiento que aleja el brazo o miembro superior del tronco y se realiza en el plano frontal en torno a un eje anteroposterior formando un ángulo de 90° aproximadamente y el regreso a su posición inicial (Figura 9).

De igual manera señorita instructora Katherine menciona que los ejercicios de levantamientos laterales con mancuernas implican el trabajo de los deltoides laterales.

2.6.2.1 Medios de entrenamiento

Pearl (2008) En su libro menciona varios medios de entrenamiento para los levantamientos con mancuernas, entre los más destacados se tiene; levantamiento lateral sentado, levantamiento lateral con la espalda apoyada, levantamiento lateral en el suelo, levantamiento lateral sobre banco inclinado, y levantamientos laterales con mancuernas de pie.

2.6.2.2 Factores de entrenamiento

Para los ejercicios de levantamientos laterales, existen varios factores de entrenamiento que pueden variar en las rutinas de entrenamiento, teniendo en cuenta el objetivo de la persona que está entrenando y el progreso que desea alcanzar. Entre los factores de entrenamiento se incluyen; la intensidad, volumen, rango de movimiento, velocidad de ejecución, la técnica y los diferentes tiempos de descanso. (Buenache, 2009)

2.6.2.3 Métodos de entrenamiento

Entre los diferentes métodos de entrenamiento se tienen:

Entrenamiento de fuerza

Entrenamiento de hipertrofia

Entrenamiento de resistencia

Superseries

Entre los métodos mencionados, es importante tener en cuenta un enfoque adicional, como la concentración, ya que esto puede llevar a obtener mejores resultados y mejorar la conexión entre la mente y el músculo involucrado en el entrenamiento. (Buenache, 2009)

2.6.2.4 Descripción de posturas de entrenamiento

En el libro de tratado general de la musculación Linares (2014) menciona el proceso de ejecución y posturas del ejercicio de levantamiento lateral posterior, de pie. Como paso inicial se debe tomar una mancuerna en cada mano, tomando en cuenta que el peso debe ser el mismo para cada mano.

Posición inicial: La postura de la persona implica que la persona se encuentre en posición vertical, con las piernas discretamente distanciadas y alineadas con los hombros. Las rodillas deben estar ligeramente flexionadas, los codos con una pequeña inclinación, y las mancuernas deben estar posicionadas frente al cuerpo (Linares, 2014).

Técnica: Se procede a elevar los brazos al mismo tiempo, levantando las mancuernas hacia los lados mediante una abducción del hombro. Es importante mantener una suave flexión en los codos a lo largo de todo el movimiento (ver Figura 9). Una vez que los brazos estén alineados y formen un ángulo de 90° con el torso, se inicia el movimiento inverso con un control preciso del descenso de las mancuernas. (Linares, 2014)

Es importante tomar en cuenta que los codos no deben estar completamente rectos por lo que deben conservar una pequeña flexión, generalmente alrededor de 5 a 10 grados.

Figura 9

Ejecución de ejercicio de levantamientos laterales

Fuente: Elaborado por (Linares, 2014)

Finalizada la revisión teórica acerca del fisicoculturismo y las dos variantes de ejercicios de hombros, se procede con la revisión de los fundamentos teóricos acerca de los dispositivos electrónicos a usar en el presente proyecto.

2.7 Sensor Kinect

Para la creación del sistema, se emplea el sensor Kinect, el cual posibilita la detección y seguimiento de las distintas articulaciones del cuerpo humano y la generación de los huesos para la detección del esqueleto humano.

2.7.1 Descripción de Hardware

Como se muestra en la Figura 10, el Kinect se constituye como un dispositivo compuesto por múltiples sensores, y la característica primordial de ambas versiones reside en su capacidad para identificar a la persona ubicada frente a él, reconocer su voz y registrar

sus movimientos. La segunda versión, en particular, experimentó mejoras significativas en diversos aspectos, tales como la amplitud de su campo de visión y la calidad de las imágenes capturadas, entre otros. Además, se añadieron funcionalidades adicionales, como la detección de movimientos en entornos con poca luz, la medición del ritmo cardíaco y la estimación de la fuerza muscular. (Guzm‡n & Sierra, s. f.)

Figura 10

Sensor Kinect v1 y Sensor Kinect v2

Fuente: Elaborado por (Paul et al., 2016)

La Tabla 1 resume la comparación de las dos versiones del Kinect.

Tabla 1

Comparativa entre Kinect v1 y Kinect v2.

Características	Kinect v1	Kinect v2
Video	640 x 480 @ 30 fps	1920 x 1080 @ 30 fps Full High Definition
Profundidad	320x240, 640x480 Distancia 0.8 m a 4m	512x424 Distancia 0.5 a 4.5 metros

Rastreo del cuerpo	2 usuarios 20 puntos por cada usuario	6 usuarios 25 puntos por cada uno
Ángulos de visión	57° horizontal, 43° vertical	70° horizontal, 60° vertical
Latencia	102 ms	20 ms
Distancia mínima de uso	1.82 metros	1.37 metros
Sistema Operativo	Windows Linux Mac OS	Windows Linux Mac OS
Conexión	USB 2.0	USB 3.0
Sensores	RGB, IR, Depth, 4 micrófonos	RGB, IR, Depth, 4 micrófonos

Fuente: Elaborado por el autor

El sensor está equipado con varias cámaras, incluyendo las de tipo RGB, Depth Image e infrarroja, cada una de las cuales captura información única. Al procesar esta información de manera conjunta, se logra la capacidad de distinguir entre personas y objetos, así como calcular la distancia y profundidad. (Guzm‡n & Sierra, s. f.)

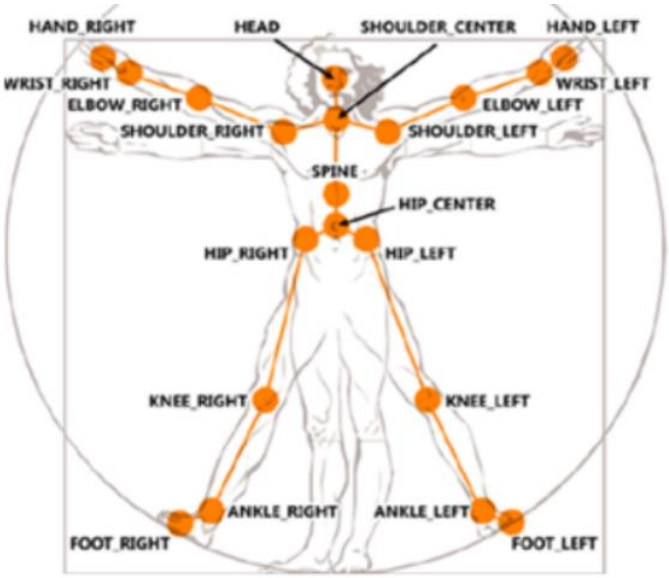
Cuando hay varias personas presentes en una habitación o cuando los colores del fondo se asemejan mucho a los de las personas, la tarea de distinguirlas se vuelve complicada. Para abordar esta situación, se garantiza que ambas cámaras estén enfocadas en la misma entidad, y aquí es donde entra en funcionamiento la cámara infrarroja. Esta cámara divide el campo de visión en una especie de matriz de 2x2, asegurando que las cámaras siempre estén apuntando hacia la misma celda. (Guzm‡n & Sierra, s. f.)

En el Kinect v2, se introduce una modificación en la forma en que se mide la profundidad. En esta versión, se emplea la técnica de tiempo de vuelo, que implica la determinación del tiempo que la luz infrarroja tarda en viajar hacia un objeto y regresar. Esto proporciona una mayor precisión en la recopilación de datos en la detección de movimientos de una persona. (Guzm‡n & Sierra, s. f.)

La identificación de personas se lleva a cabo en base a una técnica denominada 'skeleton', que implica la detección de los puntos clave de articulación del cuerpo y la conexión de estos puntos para crear una representación que se asemeja a un muñeco de palitos (ver Figura 11). Esta representación es altamente efectiva para analizar posturas corporales completas, pero puede generar incertidumbre al abordar regiones específicas del cuerpo, como las manos, donde solo se reconocen dos puntos. (Guzm‡n & Sierra, s. f.)

Figura 11

Puntos Skeleton

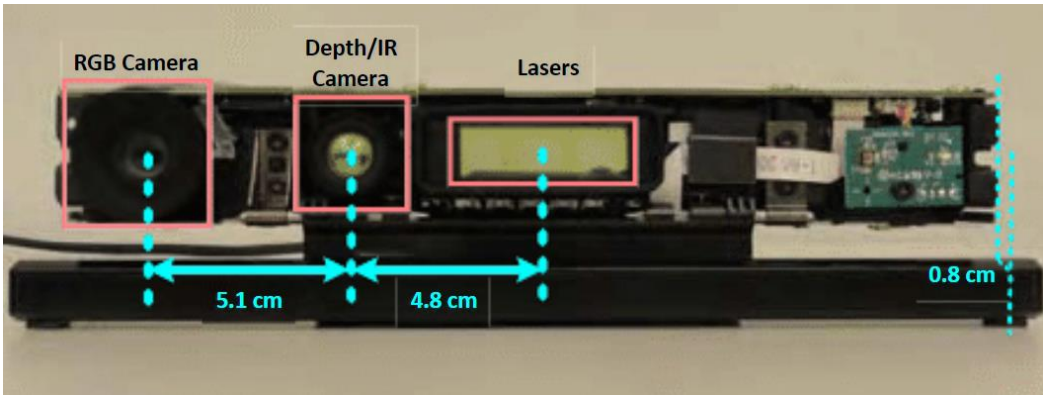


Fuente: Elaborado por (Guzm‡n & Sierra, s. f.)

Para el desarrollo del trabajo de titulación se hace uso del sensor Kinect v2, o también conociendo como Kinect XBOX, el cual está compuesto por la cámara RGB, sensor de profundidad, y el proyector de la luz infrarroja (Figura 12).

Figura 12

Vista frontal de sensores Kinect v2



Fuente: Elaborado por (Yang et al., 2015)

Como se detalló en la Tabla 1, el Kinect v2 consta de algunas mejoras como son:

- Mayor campo de visión y mejor seguimiento del cuerpo.
- Mayor resolución con más detalles.
- Mejor rango de profundidad del sensor.
- Aumento de la velocidad en el puerto USB 3.0
- Mejora en la captación de sonidos.
- Captación de movimientos a oscuras

2.7.2 Funcionamiento del Kinect

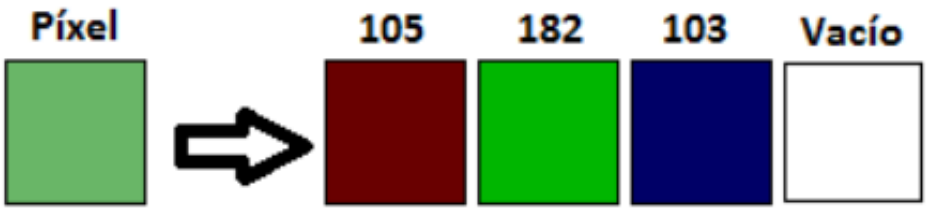
Es fundamental considerar que el esqueleto generado por Kinect proporciona información sobre las articulaciones y no sobre los huesos. Esta distinción es crucial, ya que

afecta la interpretación de los movimientos corporales. Cada articulación cuenta con las propiedades de color (x,y), profundidad (x,y), cámara (x,y,z) y orientación (x,y,z,w). (Jamhoury, 2022)

Asimismo, es importante considerar la configuración de una imagen, que se forma por una serie de píxeles. La Figura 13 muestra que cada píxel de la imagen consta de cuatro componentes que representan los valores de los colores RGB (rojo, verde, azul), además de un valor correspondiente a la transparencia en el caso de RGBa, o un valor nulo si se trata de RGB estándar.

Figura 13

Componentes de un pixel



Fuente: Elaborado por (De la Fuente Garrido, 2012)

Cada elemento del píxel adquiere un valor en el rango de 0 a 255, equivalente a un byte. Así, el conjunto de bytes obtenido del sensor, específicamente de la cámara RGB, representa la disposición de esos píxeles de manera organizada, de arriba hacia abajo y de izquierda a derecha. En este arreglo (Figura 14), los primeros cuatro elementos del vector corresponden a los valores de rojo, verde, azul y alfa, mientras que los 4 últimos serán del píxel de abajo a la derecha. (De la Fuente Garrido, 2012)

Figura 14

Estructura del vector de bytes de imagen

Fuente: Elaborado por (De la Fuente Garrido, 2012)

Cuando se emplean cámaras de profundidad, se genera un vector de bytes. En este caso, los bytes no representan los valores de los componentes de un píxel, sino la distancia desde el píxel hasta el sensor. (De la Fuente Garrido, 2012)

2.7.2.1 Coordenadas de color (X,Y)

El Kinect está equipado con dos cámaras: una cámara a color RGB y una cámara de profundidad, ambas con resoluciones distintas. Según lo indicado en la tabla 1, la cámara a color tiene una resolución de 1920 x 1080, mientras que la cámara de profundidad tiene una resolución de 512 x 424. Esto implica la existencia de coordenadas de color (x, y) y coordenadas de profundidad (x, y). (Jamhoury, 2022)

Las coordenadas de color (x, y) representan la ubicación de la articulación en la imagen capturada por la cámara a color.

Según Jamhoury (2022) en su estudio del sistema de coordenadas de Kinect v2, menciona que un punto del espacio de color describe un punto 2D en la imagen de fila/columna de un píxel en la imagen, donde x=0, y=0 es el pixel en la parte superior

izquierda de la imagen en color, y x=1919, y=1079 (ancho-1, altura-1) corresponde a la parte inferior derecha

En otras expresiones, el Kinect proporciona un resultado que varía de 0 a 1 para las coordenadas de color (x, y). Estos valores están ajustados en una escala de 0 a 1, correspondiente a la resolución de la cámara, que es de 1920 x 1080. (Jamhoury, 2022)

2.7.2.2 Coordenadas de profundidad (x,y)

Las coordenada de la articulación en la imagen de la cámara de profundidad se dan por medio de las coordenadas (x,y), en el cual se debe tomar en cuenta el espacio de profundidad el cual se utiliza para describir la ubicación 2D en la imagen de profundidad. Se puede pensar como la ubicación de una fila/columna de un pixel en el cual "x" es la columna e "y" es la fila. Entonces x = 0, y = 0, corresponden a la esquina superior izquierda de la imagen, y x = 511, y = 423 (ancho-1, alto-1) es la esquina inferior derecha de la imagen. (Jamhoury, 2022)

2.7.2.3 Coordenadas de la cámara (x,y,z,w)

Las coordenadas de la cámara del sensor Kinect hace uso del sensor infrarrojo para localizar los puntos tridimensionales de las articulaciones en el espacio. Estas coordenadas se manejan de manera distinta a las coordenadas de color y profundidad (Jamhoury, 2022).

El término utilizado como "espacio de la cámara" se refiere al sistema de coordenadas tridimensionales empleado por el Kinect (ver Figura 15). La definición de este sistema de coordenadas es la siguiente:

Tomando en cuenta el origen (x=0, y=0, z=0) es la ubicación en centro del sensor IR en el Kinect.

El incremento de la coordenada X sucede en dirección hacia la izquierda desde, la perspectiva del sensor.

La coordenada Y hace referencia a la dirección vertical de la cámara, se basa en la inclinación del sensor hacia arriba y hacia abajo.

La coordenada Z aumenta en la dirección hacia la que mira el sensor, en otras palabras, representa la distancia desde la cámara del sensor al punto en el espacio.

1 unidad = 1 metro

Figura 15

Coordenadas del espacio de la cámara del sensor Kinect

Fuente: Elaborado por (Jamhoury, 2022)

En el espacio de la cámara del Kinect v2, las coordenadas son expresadas en metros. Las coordenadas (x,y) pueden tomar valores positivos como valores negativos, ya que su rango de extensión va en ambas direcciones desde el sensor. (Jamhoury, 2022)

Jamhoury (2022) en su estudio del sistema de coordenadas de Kinect v2 menciona que el rango de profundidad del Kinect es de 8 metros, pero el rango de seguimiento del esqueleto es de 0.5m a 4.5m, y este tiene problemas para encontrar un esqueleto a menos de

1.5m debido al campo de visión de la cámara, por lo cual el valor de la cámara Z normalmente se debe encontrar entre 1.5 metros y 4.5 metros.

De igual manera Jamhoury (2022) menciona que el valor de la coordenada “y” también puede ser negativa o positiva ya que el valor positivo se da cuando está por encima del sensor, y el valor negativo cuando se encuentra por debajo de sensor.

2.7.3 Software con sensor Kinect

Para el uso del sensor Kinect se puede receptar los datos mediante el sistema operativo Windows, el cual se instalará en un ordenador Lenovo con procesador x64, bajo el lenguaje de programación C# y el uso de varios entornos de desarrollo entre los cuales se tiene:

- Visual Studio
- Processing
- Visual Studio Code

Dentro del entorno de programación se tiene el lenguaje de programación entre los más conocidos:

- C#
- C++
- JavaScript

De igual manera se tendrá las diferentes librerías a emplear:

- SDK para Kinect
- NUI Skeleton Tracking
- OpenPose

- OpenCV
- OpenNI/Nite

2.7.4 Limitaciones Kinect

Mediante el estudio del sensor Kinectv2, se recolectó información acerca de la compatibilidad entre la arquitectura del Kinect con la arquitectura del sistema en módulo Jetson Nano "ARM" planteada inicialmente. Se concluyó que no existe compatibilidad para la programación y el uso de todas sus funcionalidades que brinda el creador de Kinect "Microsoft".

Por lo cual se llevó a cabo un reajuste del hardware de programación con relación a la compatibilidad del sensor kinectv2, así como para aprovechar al máximo las herramientas disponibles a través de su SDK y las adaptaciones necesarias para el desarrollo de aplicaciones.

2.7.4.1 Hardware empleado

El Desarrollo del Proyecto se llevó a cabo en un ordenador personal, considerando que el dispositivo es compatible con el sistema operativo Windows.

Microsoft lanzó gratuitamente el software Development Kit (SDK) para Kinect, específicamente para entornos Windows, con los siguientes requisitos mínimos de hardware y software.

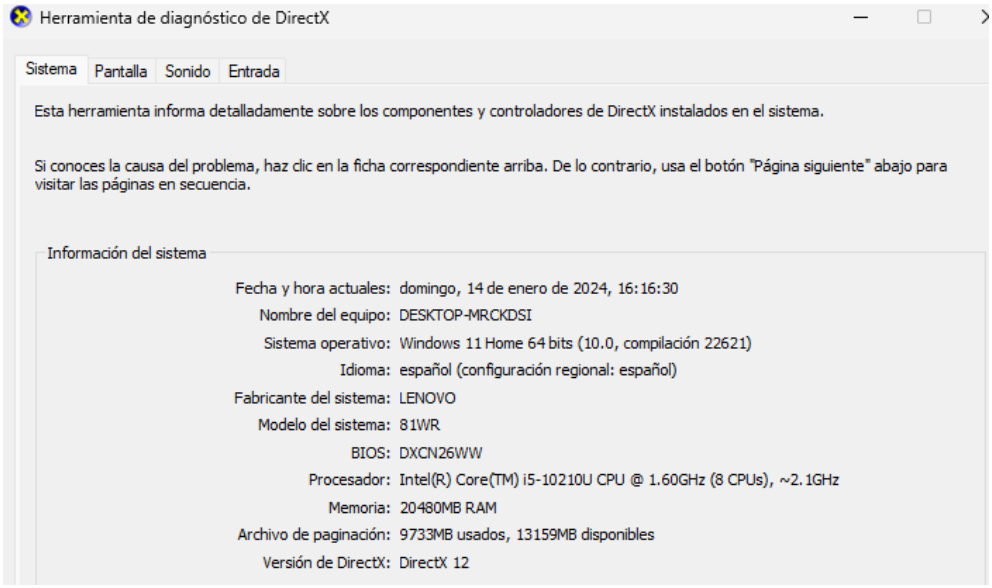
- Procesador de 64 bits.
- Controlador USB 3.0
- Adaptador de Kinect de conexión con sistema operativo Windows.
- 4GB de RAM

- Tarjeta gráfica que admita DirectX 11.
- Windows 8 o superior

En la Figura 16 se especifican las características del equipo Lenovo empleado para la programación y el testeo del sistema, presentando los siguientes detalles.

Figura 16

Diagnóstico Equipo Lenovo



Fuente: Elaborado por el autor

2.8 Metodología de Desarrollo

Una metodología de desarrollo de proyecto se refiere al conjunto estructurado de procesos utilizados para llevar un control del desarrollo de un proyecto.

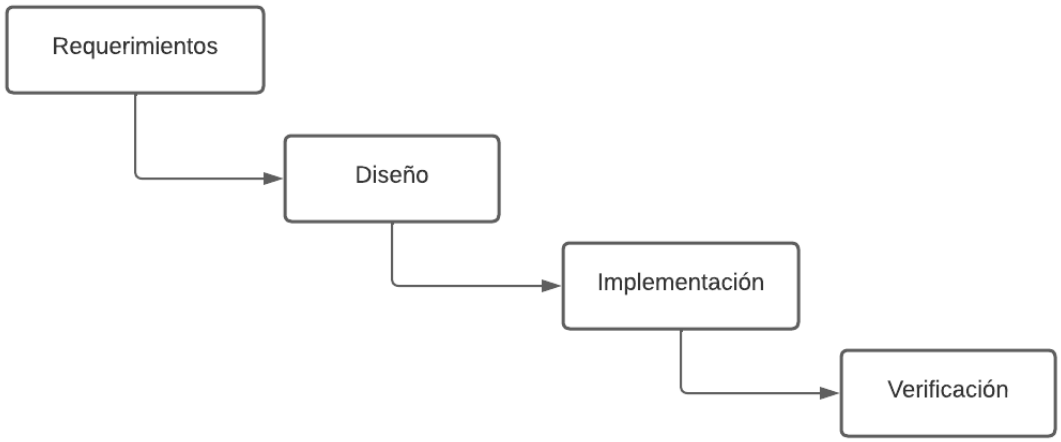
2.8.1 Metodología lineal o en Cascada

Donde no se inicia una etapa o fase hasta que se complete la anterior. Cada vez que una fase finaliza el cual debe ser aprobado el cual sirve como punto de partida para la siguiente fase (Areba, 2001).

En la Figura 17 se presenta la secuencia de la metodología en cascada a implementar en sistema.

Figura 17

Metodología en cascada



Fuente: Elaborado por el autor

Pantaleo & Rinaudo(2015) mencionan que el proceso incluye una serie de etapas en el siguiente orden:

1. Análisis de Requerimientos: Conlleva el entendimiento del proyecto a desarrollar, esto incluye tomar en cuenta comportamientos, requisitos, relación con partes interesadas tanto externas como internas
2. Diseño: Es la forma en que la solución se implementará por medio de estrategias y tareas que se definen en la fase anterior. El enfoque del diseño se centra en componentes, entornos de trabajo y herramientas a utilizar, obteniendo como resultado un resultado preliminar del diseño del sistema.

3. Implementación: Se trata de la implementación del sistema conforme a lo planificado durante la fase de diseño, incluyendo la detección de errores y la realización de pruebas unitarias. Este proceso culmina en el sistema final, que será verificado en la fase subsiguiente.
4. Pruebas-Verificación: En la fase de pruebas se asegura que el sistema satisfaga los requerimientos establecidos, en otras palabras, implica confirmar que se comporte de acuerdo con lo esperado.

La ventaja de emplear el modelo en cascada reside en la detección temprana de riesgos, gracias a la definición y fijación inicial de los requerimientos establecidos (Pantaleo & Rinaudo, 2015).

Capítulo III: DISEÑO DE SISTEMA INTELIGENTE

En el presente capítulo, se detalla el desarrollo del diseño del sistema, aplicando la metodología “Modelo en Cascada”. En esta metodología, se definen las fases secuenciales a cumplir, lo que permite un desarrollo estructurado y organizado.

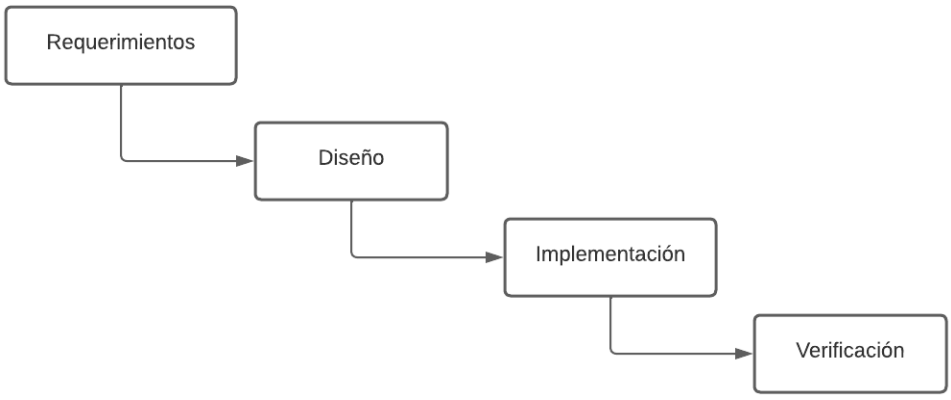
3.1 Metodología de diseño

Considerando que una metodología es un marco el cual sirve como orientación y guía para las actividades a desarrollar, organiza el proceso mediante la definición de pautas a seguir y restricciones a cumplir (Pantaleo & Rinaudo, 2015).

El modelo por utilizar en el presente proyecto es el denominado “Modelo en Cascada” por su enfoque secuencial, facilitando una planificación estructurada y una implementación por medio de fases. En la Figura 18 se muestran las cuatro fases que conforman del modelo en Cascada a cumplir en el proyecto.

Figura 18

Fases Modelo en Cascada



Nota. El modelo en cascada se conforma por cuatro fases secuencias, iniciando desde la fase de requerimientos, seguida de la fase de diseño, fase de implementación y finalmente la fase de verificación. Fuente: Elaborado por el autor

3.2 Requerimientos del sistema

En la fase de requerimientos, se detallan los requisitos específicos del sistema basándose en el estándar ISO/IEC/IEEE 29148:2018, lo que facilita la definición precisa de los requerimientos de hardware y software a utilizar. Durante este proceso, se abordan aspectos clave como el propósito del sistema y los beneficiarios a los que servirá. Este análisis sienta las bases para el desarrollo y una comprensión clara de las funciones necesarias y las características requeridas.

3.2.1 Generalidades

> La ingeniería de requisitos es una función interdisciplinaria que mide entre los dominios del adquirente y del proveedor para establecer y mantener los requisitos que debe cumplir el sistema, software o servicio de interés («ISO/IEC/IEEE International Standard - Systems and software engineering – Life cycle processes – Requirements engineering», 2018).

En el contexto actual, la relevancia de esta práctica se destaca aún más, especialmente al considerar las necesidades específicas y el escenario de aplicación del sistema.

3.2.1.1 Escenario Actual

Este proyecto se llevará a cabo en el gimnasio "ZENER GYM", situado en el barrio Monserrath de la ciudad de Otavalo, en las calles Luis Alberto de la Torre y Jacinto Collahuazo. El gimnasio es de un piso y cuenta con diversas áreas de entrenamiento.

En la Figura 19 se presenta la entrada principal del gimnasio ZENER GYM.

Figura 19

Gimnasio ZENERGYM

Fuente: Gimnasio ZENERGYM

El gimnasio es un lugar al que las personas acuden regularmente en busca de cuidado tanto físico como mental. El gimnasio se esfuerza por proporcionar un servicio que satisfaga las necesidades de todos sus clientes.

Durante una entrevista dirigida a la señorita Katerine Estrella, gerente e instructora del gimnasio, la cual mencionó que uno de los inconvenientes para ofrecer un servicio acogedor al cliente, se asocia con el seguimiento y monitoreo continuo de la ejecución de ejercicios, especialmente aquellos que no involucran máquinas, como los ejercicios de hombros con mancuernas o barras.

Al no brindar un seguimiento continuo de la ejecución de los ejercicios a los diferentes clientes, estos se frustran, desaniman y pierden el interés por volver al gimnasio,

optando por abandonar de forma definitiva aquel gimnasio, o muchas veces buscan otro gimnasio el cual les brinde el seguimiento continuo para tener una rutina de entrenamiento más satisfactorio. Mencionó la señorita Katherine Estrella, instructora del gimnasio ZENERGYM.

Los ejercicios de hombros son comunes tanto para hombres como para mujeres. Sin embargo, la ejecución correcta depende en gran medida de corregir la postura al comenzar las repeticiones. Con un aumento en el número de clientes, no siempre es posible corregir a cada uno de manera continua. Muchas veces, los clientes pueden sentir timidez al preguntar si están realizando correctamente o no el ejercicio debido a la rigidez de sus movimientos corporales, expresó la señorita Katerine.

Por lo cual es crucial abordar aspectos importantes tales como el propósito del proyecto, la identificación de los beneficiarios y la definición de los requisitos necesarios.

3.2.2 Propósito del sistema

El presente proyecto tiene como finalidad, diseñar un sistema inteligente dirigido a los clientes del gimnasio “ZENERGYM” para detectar y corregir las posturas corporales durante la realización de los ejercicios de levantamientos laterales y press militar. Esto se logrará mediante la implementación de una interfaz gráfica que permitirá al cliente visualizar el conteo de repeticiones en tiempo real de los ejercicios realizados y las demostraciones y explicaciones de la forma correcta que se debe ejecutar. El instructor dispondrá de una interfaz gráfica que inicialmente verificará la disponibilidad del espacio físico de entrenamiento de los ejercicios de levantamientos laterales y press militar lo cual facilitará en la planificación y asignación de rutinas de entrenamiento para cada cliente, de igual

manera contará con una alerta cuando el cliente realice varias repeticiones erróneas en las dos variantes de ejercicios.

3.2.3 Beneficiarios Stakeholders

Los stakeholders, o partes interesadas son aquellos individuos, instituciones, o grupos que tienen un interés directo o indirecto en el desarrollo, implementación y resultados del sistema a desarrollar. Los beneficiaros directos incluyen a los clientes del gimnasio ZENERGYM que ejecutan las rutinas de entrenamiento en las variantes de hombros, de igual manera las señoritas instructoras al monitorear la disponibilidad del área de entrenamiento y recibir las notificaciones de alertas sobre los ejercicios ejecutados incorrectamente por parte del cliente.

En la tabla 2 se muestra las partes interesadas o stakeholders presentes para el desarrollo del proyecto.

Tabla 2

Lista de Stakeholders

LISTA DE STAKEHOLDERS
1. Clientes del gimnasio ZENERGYM
2. Gerente del gimnasio ZENERGYM
3. Instructores del gimnasio ZENERGYM
4. MSc. Carlos Vásquez (Director del trabajo de titulación)
5. MSc. Edgar Jaramillo (Asesor del trabajo de titulación)
6. Flores Erik (Desarrollador del trabajo de titulación)

Fuente: Elaborado por el autor

3.2.4 Construcción de atributos de los requerimientos

Es esencial abordar la definición de las características de los atributos de los requisitos, fundamentada en tres aspectos clave: Stakeholders, Sistema y Arquitectura. Estos componentes deben ser desarrollados de manera efectiva para lograr el cumplimiento exitoso del objetivo del proyecto.

3.2.4.1 Nomenclatura de los requerimientos

En la tabla 3 se plantea la descripción de los requerimientos del sistema con sus respectivos acrónimos, permitiendo tener un manejo más controlado de la información.

Tabla 3

Definición de Acrónimos

Descripción del requerimiento	Acrónimo
Stakeholders	STSR
Sistema	SYSR
Arquitectura	SRSH

Fuente: Elaborado por el autor

3.2.5 Requerimientos de Stakeholders

El estándar ISO/IEC/IEEE 29148:2018 da a conocer que el propósito del proceso de definición de Necesidades y Requisitos es definir los requisitos de las partes interesadas para un sistema, que pueda proporcionar las capacidades que necesitan los usuarios y otras partes interesadas en un entorno definido. («ISO/IEC/IEEE International Standard - Systems and software engineering – Life cycle processes – Requirements engineering», 2018)

Para la obtención de requisitos y necesidades es una actividad iterativa. En la cual se considera varias técnicas para identificar, que incluyen:

- Talleres estructurados con lluvia de ideas
- Entrevista, cuestionarios
- Observación del entorno o patrones de trabajo
- Revisión de documentación técnica
- Análisis de mercado o evaluación del sistema competitivo
- Simulación, creación de prototipos

Para establecer los requerimientos de Stakeholders en este proyecto se hace uso de una de las opciones mencionadas, que son los cuestionarios, aplicados a partes interesadas de forma directa.

3.2.5.1 Encuesta de Requerimientos

Al realizar encuestas, es esencial considerar la cantidad de clientes a los que se les aplicarán, especialmente aquellos clientes que asisten regularmente al gimnasio ZENERGYM. El objetivo es recopilar datos sobre los requisitos que los clientes desearían tener en sus rutinas de entrenamiento, específicamente en las dos variantes de ejercicios de hombros.

3.2.5.2 Tamaño de la muestra

El cálculo del tamaño de la muestra hace referencia al número de clientes del gimnasio ZENERGYM necesarios para que la muestra sea una representación del total de los clientes regulares que asisten al gimnasio.

Para obtener el valor del tamaño ideal de la muestra, se aplicará la fórmula planteada por Murrya y Larry para una población finita y conocida:

Ecuación 1: Tamaño ideal de la muestra

$$n = \frac{N\sigma Z^2}{(N-1)e^2 + \sigma Z^2}$$

En la cual:

n= tamaño de la muestra

N= tamaño de la población

Z= nivel de confianza, valor constante dependiendo el valor de nivel de confianza que se quiera. 99% corresponde al valor más alto y equivale a 2.58 y 95% corresponde al valor de 1.96, valor mínimo admitido.

e= error muestral, se establece el margen de error permitido.

σ= desviación estándar de la población, valor constante 0.5

3.2.5.3 Determinación de Parámetros

Se establecen los valores de las variables que se trabajarán, para luego aplicarlos en la fórmula correspondiente.

Para la obtención del valor de N se realizó un formulario a la señorita gerente e instructora del gimnasio, la cual tenía conocimiento del valor del número de clientes regulares en el gimnasio ZENERGYM, se valida esta información en el ANEXO 1.

N= 50 clientes regulares en el gimnasio ZENERGYM

Z= 1.96

e= 0.09

σ= 0.5

Ecuación 2: Obtención del tamaño de muestra

$$n = \frac{50(0.5^2)(1.96^2)}{(50-1)0.09^2 + (0.5^2)(1.96^2)}$$

$$n = 35.37 \cong 35$$

Como resultado del tamaño de muestra, se tiene que la encuesta debe ser realizado a 35 clientes del gimnasio ZENERGYM.

La encuesta dirigida hacia los clientes del gimnasio (Anexo2) y a la señorita gerente del gimnasio (Anexo4), se la realiza por medio de la herramienta proporcionada en Google Forms. Los clientes tendrán acceso a la encuesta por medio de un código QR que redirecciona directamente a esta.

A través de los resultados obtenidos de las encuestas realizadas (Anexo3, Anexo5) a partes interesadas directas, incluyendo instructores/as y clientes del gimnasio ZENER GYM, se presentan los requisitos operativos y de usuarios en la Tabla 4. Esta tabla especifica de manera clara los requisitos y expectativas percibidas de cada stakeholder, brindando una visión completa de las necesidades percibidas.

Tabla 4

Requerimientos Stakeholders

STSR				
#	Requerimiento	Prioridad		
		Alta	Media	Baja
Requerimientos operacionales				
STSR1	El sistema inteligente debe ser implementado dentro de un gimnasio	X		
STSR2	Sistema compacto		X	
STSR3	El sistema inteligente debe contar con una base de datos para el registro de usuarios			X
STSR4	Ubicación del sistema inteligente frontal al usuario	X		
Requerimientos de usuario				
STSR5	Facilidad de uso del sistema inteligente para el usuario	X		
STSR6	Visualización en tiempo real de ejecución de ejercicios.	X		
STSR7	Acceso a información de guía de ejecución de ejercicios	X		
STSR8	Notificación de alertas automáticas dirigidas al instructor/a			X
STSR9	Conteo total de repeticiones correctas e incorrectas por cada ejecución de ejercicio	X		
STSR10	Verificación de disponibilidad de área de entrenamiento de ejercicios de hombros			X
STSR11	Acceso a elección de tipo de ejercicio a realizar y cantidad de repeticiones		X	

Nota. En la tabla se puede apreciar los requerimientos operacionales y del usuario, en base al levantamiento de información por medio de formularios. Fuente: Elaborado por el autor

Los requerimientos presentados en la tabla 4 abarcan las necesidades expresadas tanto por los clientes e instructores del gimnasio ZENER GYM durante observaciones de parte

directa y cuestionarios realizados, a fin de tener una visión amplia del sistema y su ambiente de funcionamiento e interacción con los usuarios e instructores.

3.2.6 Requerimientos del Sistema

El estándar ISO/IEC/IEEE 29148:2018 da a conocer el propósito del proceso de definición de requisitos del sistema, el cual es transformar la visión de las capacidades deseadas orientada al usuario y de las partes interesadas en una visión técnica de una solución que satisfaga las necesidades operativas del usuario. («ISO/IEC/IEEE International Standard - Systems and software engineering – Life cycle processes – Requirements engineering», 2018). En la Tabla 5 se describe los requerimientos relacionados con las funciones a realizar del sistema.

Tabla 5

Requerimientos del sistema

SYSR					
#	**Requerimiento**	**Prioridad**			**Relación**
		Alta	**Media**	**Baja**	
	Requerimientos de uso				
SYSR1	El sistema debe encontrarse dentro un gimnasio con mancuernas de entrenamiento.	X			STSR1
SYSR2	Comunicación entre sensor Kinect y Windows	X			
SYSR3	Lectura de datos en tiempo real	X			STSR6
SYSR4	El sistema cuenta con una pantalla para la visualización de video de las posturas de los ejercicios ejecutados por los clientes	X			
SYSR5	Ubicación del sistema frontalmente al usuario	X			STSR4
SYSR6	Sistema conectado con una base de datos para el registro de usuarios			X	STSR3

Requerimientos de performance					
SYSR7	Alerta cuando la mayoría de las repeticiones de ejercicios sean realizadas incorrectamente		X		STSR8
SYSR8	Notificación de alerta dirigida al instructor			X	STSR8
SYSR9	Replica de posturas corporales en la pantalla de visualización del usuario		X		STSR6
SYSR10	Determinación de posturas incorrectas realizadas por los usuarios		X		STSR9
SYSR11	Acceso a información de guía de ejecución de ejercicios	X			STSR7
Requerimientos de estados					
SYSR12	El sistema debe permanecer activo cuando detecte una persona.	X			
SYSR13	Reinicio automático de conteo de repeticiones una vez que el usuario haya finalizado las repeticiones programadas	X			
SYSR14	El sistema debe permitir apagar el Kinect para ahorrar recursos	X			
Requerimientos de Interfaz					
SYSR15	El sistema inteligente debe interactuar con los sensores del dispositivo Kinect.	X			
SYSR16	Obtención de video en tiempo real de las posturas corporales del usuario	X			STSR6
SYSR17	El sistema inteligente cuenta con una interfaz gráfica interactiva con el usuario		X		STSR5
SYSR18	El ordenador debe contener un puerto USB3.0 para la conexión con el Kinect	X			
SYSR19	Ordenador con RAM mínima 4 GB	X			
SYSR20	Ordenador con procesador 64 bits	X			
Requerimientos Físicos					
SYSR21	La ubicación del sistema debe estar frontal al usuario	X			STSR4
SYSR22	El sistema final debe ser compacto		X		STSR2

SYSR23	Ubicación del sensor Kinect para la detección del cuerpo total del usuario	X

Nota. En la tabla se da a conocer los requerimientos del sistema, dividido en 5 requerimientos esenciales para el desarrollo del sistema. Fuente: Elaborado por el autor

Los requerimientos del sistema (SYSR) presentados en la tabla 5 engloban las funcionalidades necesarias de operación del sistema, en la cual se toma en cuenta los requerimientos de uso, performance, estados, interfaz y requerimientos físicos, los cuales se vinculan con las expectativas y requerimientos de las partes interesadas.

3.2.7 Requerimientos de Arquitectura

En la Tabla 6 se da a conocer los requerimientos de arquitectura (SRSH), lo cuales contienen información acerca de los requisitos de diseño del sistema, relacionados al software, hardware y eléctricos, los mismo que facilitan a la elección de los componentes de hardware y software a emplear en el sistema. Los requerimientos de arquitectura establecen las pautas fundamentales para la elección de hardware y software que se incorporará en el desarrollo del sistema.

Tabla 6

Requerimientos de Arquitectura

SRSH					
#	**Requerimiento**	**Prioridad**			**Relación**
		Alta	**Media**	**Baja**	
	Requerimientos lógicos				
SRSH1	Compatibilidad de la versión del SO Windows con el Kinect.	X			
SRSH2	Compatibilidad de SO con librerías de conexión entre el ordenador y dispositivo kinect.	X			

SRSH3	Compatibilidad de SO con arquitectura Kinect	X			
SRSH4	Conexión de base de datos con el ordenador.		X		STSR3
SRSH5	Ubicación del sistema inteligente en el rango de detección del dispositivo Kinect.	X			
SRSH6	Compatibilidad de puertos de entrada y salida del kinect con el ordenador	X			
	Requerimientos de diseño				
SRSH7	Ubicación específica del sistema inteligente para prevenir daños por contacto directo.		X		
SRSH8	Accesibilidad de interfaz para interacción con el usuario	X			SYSR17
SRSH9	El dispositivo Kinect debe colocarse al frente del usuario	X			
SRSH10	Las cámaras del sensor Kinect deben enfocar el cuerpo completo del usuario	X			
SRSH11	El dispositivo Kinect debe detectar una persona ubicada en el área de ejercicio.	X			
SRSH12	Los adaptadores del Kinect deben ser ubicado en un lugar estratégico.			X	
	Requerimientos de Software				
SRSH13	Se requiere un sistema operativo con lenguaje de programación de código abierto y acceso a creación de aplicaciones	X			
SRSH14	Se requiere compatibilidad con librerías de detección de cuerpo humano por medio de Kinect.	X			SRSH3
SRSH15	Se requiere compatibilidad con librerías para la creación de una interfaz gráfica y aplicaciones.	X			
SRSH16	Se requiere una plataforma para la generación de alertas.	X			SYSR8
SRSH17	Se requiere un entorno de programación compatible con Windows.	X			
SRSH18	Se requiere DirectX 11 mínima.		X		
SRSH19	Se requiere el SDK de Kinect en el ordenador			X	
SRSH20	Plataforma en la nube compatible con .NET	X			

Requerimientos de Hardware				
SRSH21	El ordenador requiere de puertos usb 3.0 para la comunicación con el dispositivo Kinect	X		SRSH6
SRSH22	El sistema inteligente debe ser compatible con SO Windows.	X		SRSH1
SRSH23	El ordenador requiere compatibilidad con el dispositivo Kinect	X		SRSH2
SRSH24	Se requiere adaptador para el conector del dispositivo Kinect	X		
SRSH25	La versión del Kinect debe tener precisión en la detección corporal del usuario.	X		
SRSH26	El dispositivo Kinect debe detectar la mayor cantidad de articulaciones del esqueleto humano		X	
SRSH27	El dispositivo Kinect debe contar con una buena calidad de imagen	X		
Requerimientos Eléctricos				
SRSH28	Fuente de alimentación del sistema	X		
SRSH29	Fuente de alimentación para el dispositivo Kinect	X		
SRSH30	Adaptador de conector de Kinect para Windows	X		
SRSH31	Regulador de voltaje 12v para Kinect	X		

Nota. Se presenta cinco requerimientos de arquitectura necesarios para la elección de hardware y software del sistema. Fuente: Elaborado por el autor

3.2.8 Benchmarking (Selección de Hardware y Software)

El estándar ISO/IEC/IEEE 29148:2018 menciona que la obtención de requisitos y necesidades es una actividad iterativa, considerando diferentes técnicas para identificar los requisitos entre estas se tiene el Benchmarking («ISO/IEC/IEEE International Standard - Systems and software engineering – Life cycle processes – Requirements engineering», 2018).

La elección de hardware y software se basará en los requisitos mencionados previamente para satisfacer las necesidades de las partes interesadas (STRS, SYSR, SRSH). Se realizará una evaluación de los distintos componentes a utilizar en el proyecto mediante el uso de tablas comparativas, definiendo dos valores de puntuación: “2” para indicar el cumplimiento total del requerimiento, “1” para indicar el cumplimiento parcial y “0” cuando no se cumpla dicho requerimiento.

3.2.8.1 Hardware

La selección de hardware se realiza considerando la información presentada en la Tabla 6, la cual detalla los requisitos de arquitectura. Se tiene en cuenta las herramientas que se utilizarán en el proyecto, como la placa de desarrollo y las versiones del dispositivo Kinect.

Hardware de desarrollo

Por medio de la Tabla 7 se da a conocer la comparación de los diferentes hardware de desarrollo, los cuales hacen referencia a los requerimientos planteados. Se seleccionará el hardware que cumpla con los requerimientos necesarios con relación a la compatibilidad con el dispositivo Kinect y el diseño del sistema, para lo cual se plantea 5 opciones relacionadas con los requerimientos y así obtener una valoración de cada requerimiento para la elección del hardware de desarrollo.

Tabla 7

Comparación para elección de la placa de desarrollo

Selección de la placa de desarrollo						
Hardware	**Requerimientos**					
	SRSH3	**SRSH14**	**SRSH18**	**SRSH21**	**SRSH23**	**Valor total**
Ordenador - Windows	2	2	2	2	2	10
NVIDIA Jetson Nano	0	0	0	2	1	3
Cumple totalmente 2 **Cumple parcialmente 1** **No cumple 0**						
Elección: La tabla muestra que el ordenador Windows es la mejor opción como requerimiento de hardware a utilizar para el proyecto, un ordenador Windows tiene incluido DirectX 11, permitiendo la programación de APIs desarrolladas por Microsoft. Al tener las limitaciones con compatibilidad del Kinect con arquitecturas diferentes a la de sistema operativo Windows, un sistema integrado con arquitectura ARM no es una opción accesible.						

Nota. La tabla muestra la puntuación para la elección de la placa de desarrollo del sistema, obteniendo como mejor opción el ordenador Windows. Fuente. Autoría propia.

Sensor de movimiento

Para la elección del sensor de movimiento se ha seleccionado el dispositivo Kinect en sus dos versiones, las cuales ofrece ventajas en comparación con otros sensores o métodos

de captura de movimiento, de igual manera es ideal para la implementación en aplicaciones de interfaz de usuario basadas en gestos y movimientos corporales.

El dispositivo Kinect está equipado con un conjunto de sensores, incluyendo una cámara RGB, sensor de profundidad, cámara infrarroja y micrófonos. Esta combinación de tecnologías permite contar con una amplia gama de funcionalidades, desde la detección de personas en un espacio en específico, así como la detección de articulaciones de las partes del cuerpo humano.

En la Tabla 8 se muestra la comparación de los requerimientos cumplidos por cada versión de Kinect, en la cual se han establecido cuatro requerimientos claves. Estos requerimientos son evaluados en base a su valoración total, lo cual permite determinar la versión a elegir.

Tabla 8

Comparación de sensor de movimiento Kinect v1 y Kinect v2

Selección de Dispositivo Kinect					
Hardware	**Requerimientos**				
	SRSH23	**SRSH25**	**SRSH26**	**SRSH27**	**Valor total**
Kinect v1	2	1	1	1	5
Kinect v2	2	2	2	2	8
Cumple totalmente 2					
Cumple parcialmente 1					
No cumple 0					

Elección: Mostrada la comparación de las dos versiones del Kinect se llega a concluir que la mejor opción es el Kinect v2, cumpliendo con el mayor número de requerimientos establecidos, determinando mejoras con su versión anterior, como es mayor detección de articulaciones corporales.

Nota. En la tabla se puede apreciar la comparación entre las dos versiones del dispositivo Kinect, llegando a concluir que la mejor opción es la versión 2 del dispositivo kinect. Autoría propia.

La tabla 9 presenta las características técnicas del sensor Kinect v2, el cual se ha seleccionado como la opción a utilizar en el desarrollo del sistema.

Tabla 9

Especificaciones técnicas del sensor Kinect v2.

Características	Kinect v2
Video	1920 x 1080 @ 30 fps Full High Definition
Profundidad	512x424 Distancia 0.5 a 4.5 metros
Rastreo del cuerpo	6 usuarios 25 puntos por cada uno
Ángulos de visión	70° horizontal, 60° vertical
Latencia	20 ms
Distancia mínima de uso	1.37 metros
Sistema Operativo	Windows

	Linux
	Mac OS
Conexión	USB 3.0
Sensores	RGB, IR, Depth, 4 micrófonos

Fuente: Autoría propia

3.2.9 Software

Una vez seleccionado el hardware apropiado a utilizar, se realiza la selección del software, la cual se basa en los criterios establecidos en los requerimientos de arquitectura de la Tabla 6. Es importante tomar en cuenta que se trabajará con el dispositivo Kinectv2, por lo cual se debe tener una comunicación exitosa entre el sensor Kinectv2 y el entorno de programación junto al lenguaje de programación y las librerías a utilizar.

3.2.9.1 Entorno de Programación

Para la elección del entorno de programación se toma en cuenta 4 requerimientos con relación al software del sistema. La Tabla 10 muestra la valoración de cada requerimiento establecido para la elección del entorno de programación.

Tabla 10

Comparación de entornos de programación

Selección de entorno de programación					
Hardware	**Requerimientos**				
	SRSH13	**SRSH14**	**SRSH15**	**SRSH17**	**Valor total**
Visual Studio Code	1	1	1	2	5

Visual Studio	2	2	2	2	10
Processing	1	1	1	2	5

Cumple totalmente 2

Cumple parcialmente 1

No cumple 0

Elección: Para la elección del entorno de programación, Visual Studio obtuvo la valoración más alta. Ofrece una plataforma versátil, equipada con una amplia gama de herramientas y extensiones útiles para manipular los datos del sensor Kinect. Además, cuenta con un soporte robusto de Microsoft junto a su integración con el ecosistema de .NET, lo cual facilita la conexión mediante el SDK de Microsoft.

Nota. La información de la tabla representa la mejor opción para la selección del entorno de programación la cual es Visual Studio, obteniendo la máxima puntuación en función a los requerimientos de software establecidos. Fuente: Autoría propia.

3.2.9.2 Lenguaje de Programación

Para seleccionar el lenguaje de programación, se establecen tres criterios fundamentales basados en los requerimientos de software, los cuales se utilizan para comparar diferentes lenguajes de programación, permitiendo así evaluar cada uno de manera específica y facilitando la elección final.

Se consideran lenguajes de programación los cuales permitan utilizar de forma correcta y precisa los recursos que brinda el dispositivo Kinect v2, especialmente en el entorno de Windows dado que Microsoft proporciona SDKs oficial y amplia documentación para el desarrollo de proyectos.

La Tabla 11 muestra los lenguajes de programación a comparar y su valoración total.

Tabla 11

Comparación de lenguajes de programación

Selección de lenguaje de programación				
Hardware	**Requerimientos**			
	SRSH13	**SRSH14**	**SRSH19**	**Valor total**
Python	1	1	0	2
C++	1	1	2	4
C#	2	2	2	6
JavaScript	1	2	0	3
Cumple totalmente 2				
Cumple parcialmente 1				
No cumple 0				

Elección: Como lenguaje de programación se obtuvo la mejor opción el uso de C#. Por medio del SDK de Kinect brindado por Microsoft se tiene ejemplos de guía del uso de los sensores del Kinect en el lenguaje C# y así codificar según los requerimientos del presente proyecto. C# es un lenguaje de programación moderno el cual tiene acceso a las bibliotecas y herramientas las cuales permiten aplicar lógica de la aplicación a desarrollar en lugar de los detalles de bajo nivel como es el manejo de los datos del sensor.

Nota. Se da a conocer el cumplimiento de los parámetros para la elección del lenguaje de programación, obteniendo como mejor opción C#. Fuente: Autoría propia.

3.2.9.3 Plataforma en la nube

Establecidos los requerimientos de hardware y software, se seleccionará una plataforma en la nube, que permita monitorear la disponibilidad del espacio de entrenamiento, así como visualizar la cantidad de las repeticiones realizadas de forma correcta e incorrecta de los ejercicios de levantamientos laterales y press militar.

La integración del sistema con una plataforma en la nube amplía las funcionalidades de la aplicación, permitiendo un seguimiento más eficaz por parte del instructor. Para ello, en la Tabla 12 se establecen tres requerimientos específicos que deben cumplirse.

Tabla 12

Comparación de plataforma en la nube

Selección de lenguaje de programación				
Hardware	**Requerimientos**			
	STSR9	**STSR10**	**SRSH19**	**Valor total**
ThingSpeak	1	2	2	5
Ubidots	2	2	2	6
AWS	1	2	2	5
Firebase	1	2	2	5
Cumple totalmente 2				
Cumple parcialmente 1				
No cumple 0				

Elección: En la selección de la plataforma en la nube, conforme a los requerimientos previamente establecidos, Ubidots obtuvo la mayor valoración. Esta

plataforma en la nube proporciona una API REST que es accesible desde cualquier cliente en .NET, facilitando su integración mediante el llamado de librerías.

Nota. La tabla presentada da a conocer la valoración de los requerimientos de las diferentes plataformas en la nube, destacando a Ubidots como la plataforma con la más alta valoración. Fuente: Autoría propia.

Ubidots es una de las plataformas más conocidas en el ámbito del Internet de las Cosas IoT, destacando por sus servicios tanto gratuitos como de pago. Ofrece la capacidad de integrar datos de diversos sensores, con diversos entornos de programación (Mínguez, 2020). Diferentes datos pueden ser manejados por medio de dispositivos y variables las cuales permiten crear controles gráficos para su visualización.

3.3 Diseño del sistema

Definidos los componentes tanto en hardware y software del sistema, se procede a fijar las directrices del diseño del sistema. Se consideran los criterios evaluados previamente, los cuales facilitarán el desarrollo del sistema de corrección de posturas corporales en los ejercicios de levantamientos laterales y press militar.

Durante esta fase se especifica las funciones del sistema por medio de diagramas de bloques y diagramas de flujos, los cuales brindan una estructura clara para orientar los procesos de codificación y la ejecución del sistema final.

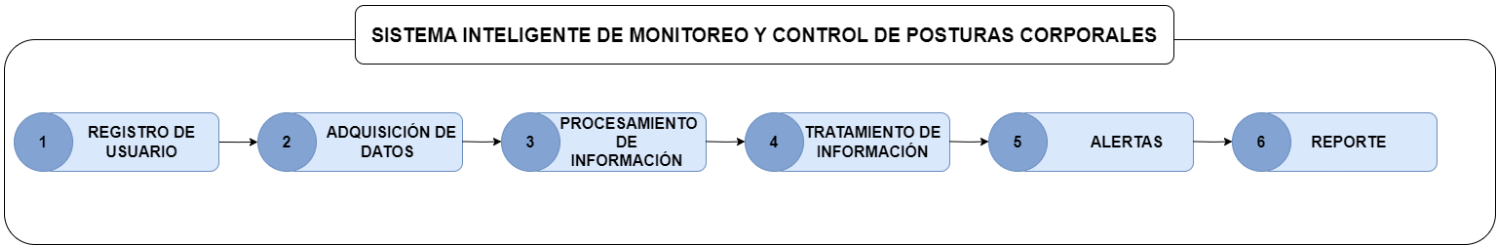
3.3.1 Diagrama de bloques general del sistema

Se plantea 6 fases las cuales engloban el proceso de funcionamiento del sistema de monitoreo y control de posturas corporales. Cada fase comprende subprocesos relacionados con la función principal a cumplir.

Por medio de un diagrama de bloques general, en la Figura 20 se ilustra las fases que permiten tener una compresión clara de las actividades realizadas en las seis fases mencionadas de la arquitectura del sistema, las cuales se explorarán a detalle en las secciones subsiguientes.

Figura 20

Diagrama de bloques general del sistema



Nota. La Figura muestra las seis fases principales del sistema. Fuente: Elaborado por el autor.

En el bloque 1, los usuarios se encuentran con la opción de iniciar sesión o registrarse. A través de una interfaz gráfica, pueden registrarse proporcionando un nombre de usuario y contraseñas personales. La información introducida se almacenará en una base de datos local cargada en MySQL. Al iniciar sesión correctamente, el sistema accederá a la información del nombre del usuario para especificar quién genera una alerta en el envío de notificaciones.

En el segundo bloque, correspondiente a la adquisición de datos, se realiza mediante la activación de los sensores y cámaras integradas en el dispositivo Kinect para la captura datos en tiempo real. El proceso comienza con el sensor de profundidad de tiempo de vuelo, que emite pulsos de luz infrarroja para detectar personas. De igual manera se activa la cámara de color para la captura de imágenes de alta definición. Esta cámara es esencial para el seguimiento de entrenamiento en tiempo real.

En el tercer bloque, centrado en el procesamiento de información, se inicializan un lector de marcos de color y un lector de marcos corporales. Estos lectores capturan datos en tiempo real, facilitando la detección y el seguimiento tridimensional de articulaciones. Estas articulaciones, a su vez, permiten la formación de huesos, los cuales se crean a partir de la unión entre ellas.

Los datos de las articulaciones capturados permiten mapear coordenadas en tres dimensiones y formar vectores específicos. Estos vectores son esenciales para calcular diferentes ángulos, facilitando la determinación de rangos angulares para las posiciones inicial y final de cada ejercicio. Este análisis permite validar la completitud de cada repetición y definir los rangos angulares para identificar repeticiones como correctas o incorrectas.

El cuarto bloque se enfoca en el tratamiento de la información, donde se clasifica cada repetición del ejercicio como “correcta” o “incorrecta”. La cantidad de repeticiones se registra en tiempo real en la interfaz gráfica del usuario y en la interfaz gráfica de Ubidots, lo que permite al instructor realizar un seguimiento en tiempo real sin necesidad de estar físicamente presente en el lugar específico del entrenamiento. Paralelamente, el usuario tiene acceso a verificar los detalles de los ejercicios ejecutados erróneamente, incluyendo el número de la repetición mal ejecutada, el estado de la posición (ya sea “inicial” o “final”), y una descripción detallada del error en la postura.

El quinto bloque se enfoca en la generación de alertas desencadenadas por eventos específicos. Este evento se activa al contabilizar un número de repeticiones incorrectas que supera un umbral determinado en relación con el total de repeticiones. Cuando esto sucede, el sistema automáticamente envía una notificación de alerta al instructor de entrenamiento

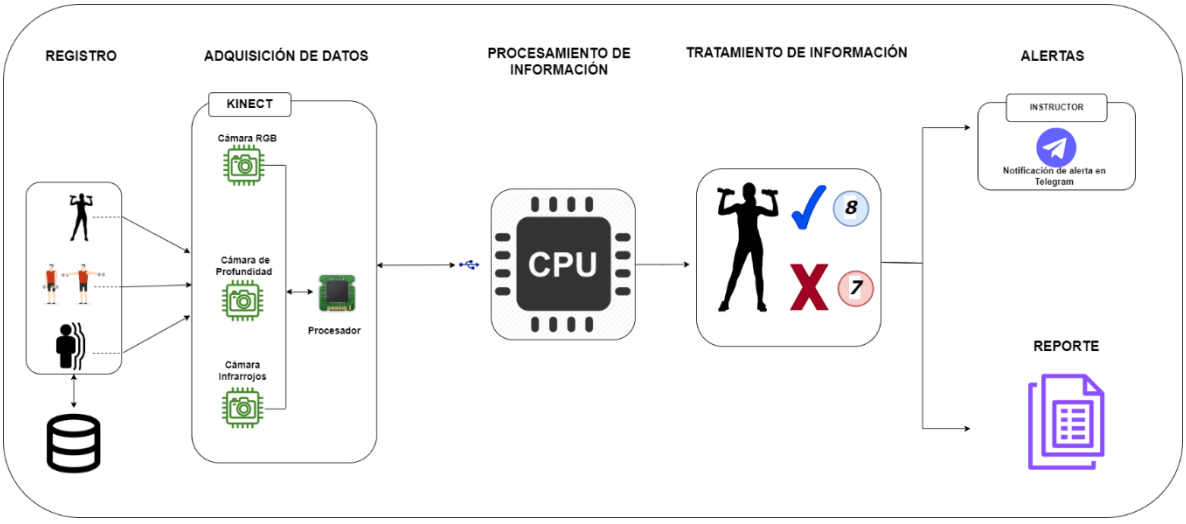
por medio del uso de Telegram, facilitando así la intervención oportuna y el apoyo necesario al usuario.

El sexto bloque se dedica a la generación de reportes, que proporcionan un resumen detallado de todas las repeticiones realizadas. Cada reporte incluye el total de repeticiones, desglosando los detalles de las posiciones inicial y final, junto con el valor de la moda de los ángulos capturados en estas dos posiciones durante las repeticiones de los ejercicios. Los reportes generados pueden ser exportados en formato PDF para facilitar un seguimiento más detallado.

La representación gráfica de la Figura 21 ilustra con mayor detalle los aspectos previamente mencionados.

Figura 21

Arquitectura general del sistema



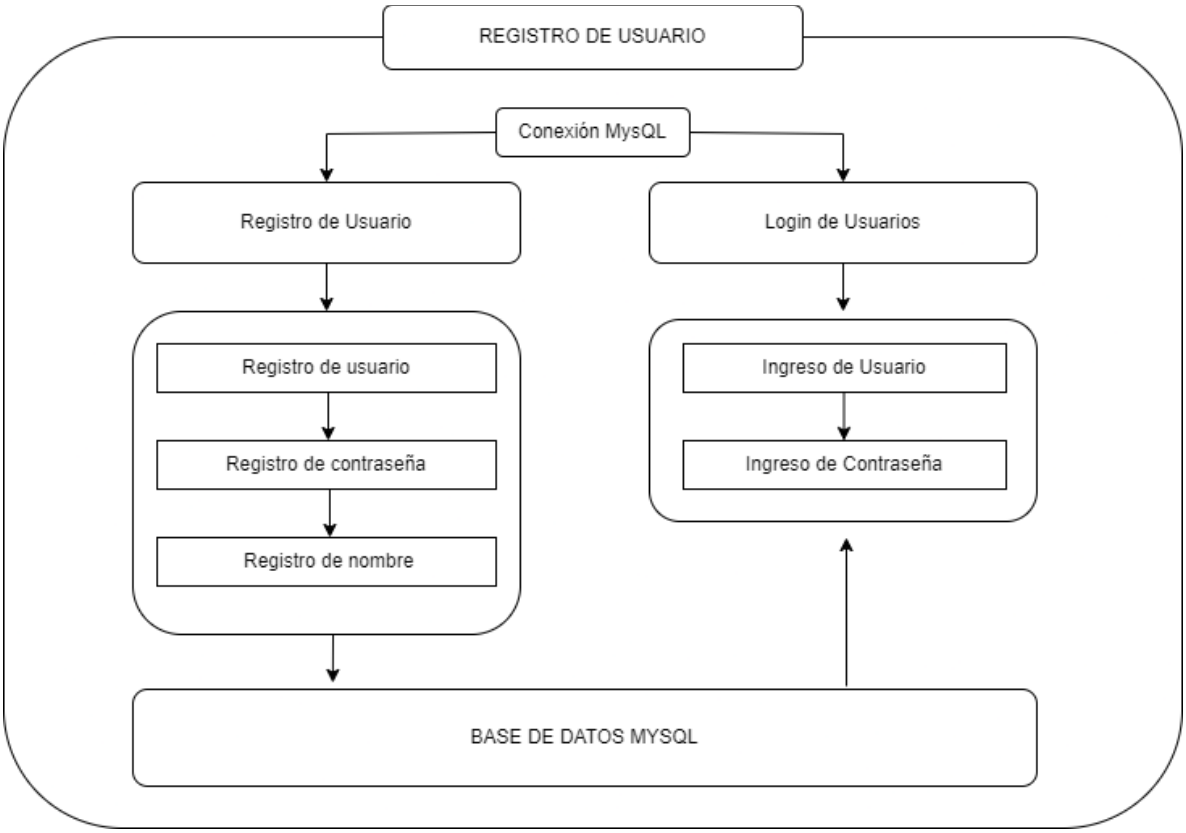
Nota. La Figura muestra la arquitectura general del sistema. Fuente: Elaborado por el autor

3.3.1.1 Registro

La base de datos se encarga de almacenar la información de usuarios, facilitando el acceso al sistema y uso del dispositivo Kinect. Esto permite ofrecer una experiencia personalizada en sus rutinas de entrenamiento y un seguimiento más a detalle por medio de alertas dirigidas a la señorita instructora cuando el usuario ejecute cierta cantidad de repeticiones de forma errónea. En la Figura 22 ilustra el diagrama de bloques que describe el funcionamiento de la base de datos para el registro de usuarios.

Figura 22

Diagrama de bloques Login/Registro de usuarios



Nota. El diagrama de bloques da a conocer la conexión con la base de datos MySQL, así como las dos funciones principales del sistema, el registro de usuarios e inicio de sesión. Fuente: Elaborado por el autor

Como paso inicial se tiene la conexión de MySql con el sistema, definiendo una clase interna para el manejo de la conexión en una base de datos creada en MySQL mediante el uso del método GetConnection. Se establecen las diferentes variables que contienen los detalles necesarios para la conexión, se asigna el servidor “localhost” indicando que el sistema se encuentra dentro del mismo ordenador, se establece el puerto de comunicación “3306” el cual es el puerto predeterminado para MySQL, seguido se establece el usuario “root” y la contraseña para el ingreso a la base de datos con su respectivo nombre “sistema_usuarios”. Posterior se construye una cadena de conexión concatenando las variables con las etiquetas específicas para establecer la conexión. Finalmente se crea una instancia del objeto MySqlConnection para establecer y manejar la conexión con la base de datos.

En la Figura 23 se ilustra el proceso de programación para establecer la conexión entre el sistema y la base de datos MySQL.

Figura 23

Establecimiento de conexión del sistema y base de datos

```
using MySql.Data.MySqlClient;

namespace Proyecto
{
    3 referencias
    internal class Conexion
    {
        3 referencias
        public static MySqlConnection GetConnection()                    //Agreagacion de un metodo publoc
        {
            string servidor = "localhost";                               //Nombre del servidor
            string puerto = "3306";                                      //Puerto de comunicación
            string usuario = "root";                                     //Nombre de usuario
            string password = "eafloresm1";                              //Contraseña de ingreso
            string bd = "sistema_usuarios";                              //Nombre de la base de datos
            //Establecimiento de cadena de conexión
            string cadenaconexion = "server =" + servidor + "; port =" + puerto + "; user id =" + usuario + "; password =" + password + "; database =" + bd;

            MySqlConnection conexion = new MySqlConnection(cadenaconexion);  //Creación de objeto para la conexion
            return conexion;                                                 //Devuelve el método de conexión.
        }
    }
}
```

Nota. Líneas de programación para establecer conexión entre el sistema y la base de datos MySQL. Fuente: Elaborado por el autor

En la fase de login y registro de usuario se tiene dos partes principales las cuales son:

El registro de usuario, el cual dispone de una interfaz gráfica que permite introducir datos personales, incluyendo usuario, contraseña y el nombre. Estos valores se almacenan en la base de datos en MySQL, los cuales una vez guardados permiten el acceso a las interfaces de entrenamiento del sistema.

El inicio de sesión de usuario, ubicado en la interfaz principal del sistema, permite el acceso a las interfaces de entrenamiento. Para ingresar, el usuario debe ingresar su nombre y contraseña, los cuales fueron establecidos durante el proceso de registro.

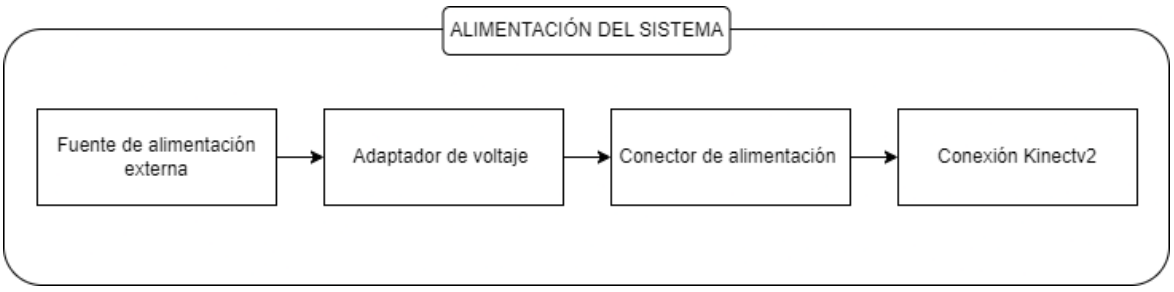
3.3.1.2 Adquisición de datos

Se presenta el proceso de desarrollo del segundo bloque, correspondiente a la adquisición de datos, en el cual el dispositivo Kinect se activa por medio del suministro de energía del adaptador del sistema activando sus sensores y recepción de datos en tiempo real.

En la Figura 24 se describe la secuencia del proceso del suministro de alimentación del sistema.

Figura 24

Diagrama de bloques sistema de alimentación



Nota. Diagrama de bloques que muestra la fuente de alimentación del sistema. Fuente: Elaborado por el autor

El diagrama presentado consta de cuatro partes secuenciales que son necesarias para el funcionamiento adecuado del sistema en conjunto con el sensor Kinect.

El primer bloque representa el inicio del sistema de alimentación, proporcionando la energía necesaria para activar el sistema a partir de una fuente externa, como un tomacorriente

El segundo bloque corresponde al adaptador de voltaje, que regula la corriente proveniente de la fuente de alimentación externa a los niveles requeridos por el dispositivo Kinect. Este adaptador proporciona una salida de 12-15V y un amperaje de 2.67A, garantizando que la corriente entregada cumple con las especificaciones técnicas necesarias para el correcto funcionamiento del dispositivo Kinect.

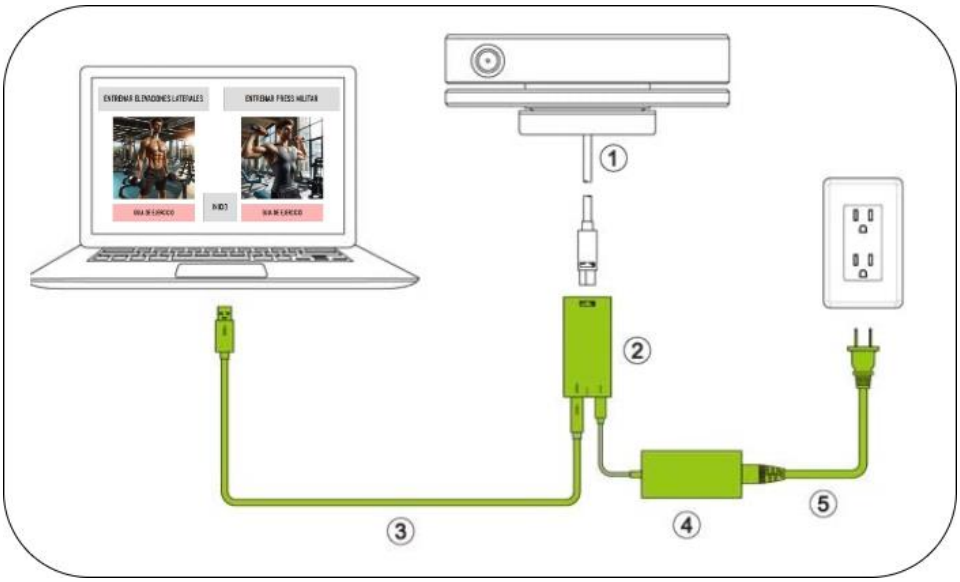
El tercer bloque corresponde al conector de alimentación, este sirve para unir el sensor Kinect con su fuente de energía. Este conector encaja específicamente en el puerto Kinect del sensor y facilita el paso de la corriente eléctrica necesaria para su funcionamiento. Además, se requiere que el Kinect esté conectado a un ordenador a través de un puerto USB 3.0 para la transmisión de datos.

El cuarto bloque representa la conexión física del sensor Kinect a la computadora y su integración con el Software Development Kit (SDK) proporcionado por Microsoft. Esta conexión no solo establece la comunicación entre el hardware y el software, sino que también permite la compatibilidad con una variedad de aplicaciones que aprovechan las capacidades avanzadas del sensor Kinect.

En la Figura 25 se ilustra el diagrama de conexión del sistema de alimentación del dispositivo Kinect y la conexión con el ordenador para el inicio de captura de datos.

Figura 25

Diagrama del suministro de energía al sistema



Nota. Se muestra el diagrama de suministro de energía del sistema, en el cual se tienen valores especificando cada parte del sistema de alimentación. Fuente: Elaborado por el autor

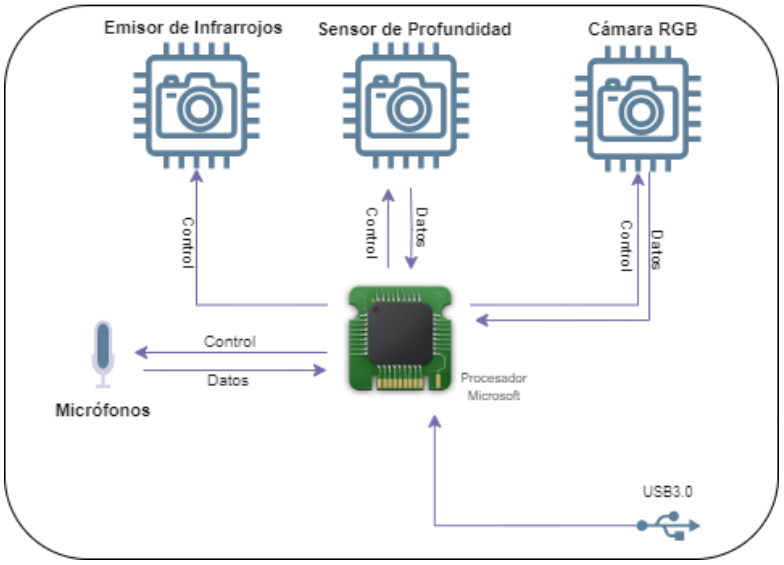
El valor 1 señala el puerto de conexión del sensor Kinect hacia el adaptador, conocido como “Kinect port”. El valor 2 se refiere al adaptador del Kinect, que dispone de dos conexiones: una para el cable USB 3.0, indicada con el valor 3, y otra para la entrada de energía, controlada por el adaptador y señalada con el valor 4. Por último, la conexión a la fuente de energía externa está marcada con el valor 5.

Tras completar el proceso de suministro de energía al dispositivo Kinect, se procede con el desarrollo del proceso de adquisición de datos provenientes de las diferentes cámaras y sensores que integra el dispositivo.

En la Figura 26 se ilustra el diagrama esquemático de la adquisición de datos mediante el dispositivo Kinect.

Figura 26

Diagrama esquemático de adquisición de datos.



Nota. Diagrama esquemático que muestra la interconexión de los sensores y su activación por medio del procesador Microsoft. Fuente: Elaborado por el autor

En la Figura 26 se ilustra cómo el procesador principal, X871141, coordina el intercambio de información con los diversos sensores conectados: el emisor de infrarrojos, el sensor de profundidad, la cámara RGB y los micrófonos, todos cruciales para la adquisición de datos. La conexión del Kinect con el ordenador, necesaria para la transferencia de datos, se realiza mediante un puerto USB 3.0.

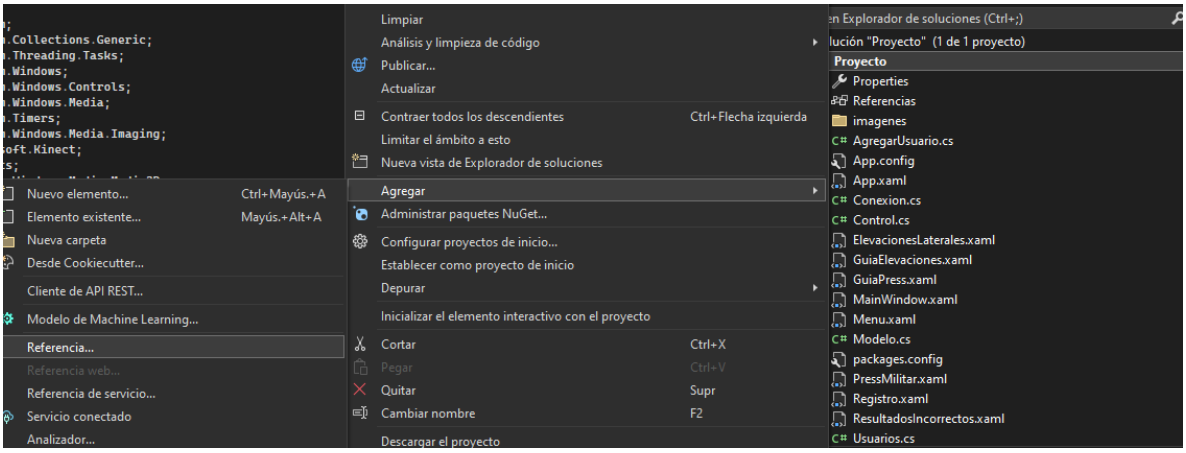
La activación de la cámara RGB, el sensor de profundidad y el sensor de infrarrojos se realiza a través del SDK proporcionado por Microsoft, que facilita la interfaz y control de estos componentes del Kinect. Utilizando el entorno de programación Visual Studio, se emplea la librería Microsoft.Kinect, la cual proporciona acceso a diversas clases, propiedades y métodos necesarios para controlar cada componente del Kinect.

En el entorno de programación en VisualStudio 2022 se agrega la referencia Microsoft.Kinect dentro del proyecto, y así llamar a la directiva *Microsoft.Kinect* para hacer uso de las clases, métodos y propiedades.

En la Figura 27 se ilustra el proceso para agregar la referencia dentro del proyecto de desarrollo.

Figura 27

Agregación de referencia Microsoft.kinect



Fuente: Elaborado por el autor

Posterior se obtiene una instancia de la clase KinectSensor. Esta clase permite establecer la conexión con las funciones del dispositivo Kinect. Una vez creada la instancia, se verifica que el dispositivo Kinect esté correctamente conectado. Se inicializa el sensor mediante el método Open(), un paso crítico para comenzar a operar y recibir datos. Este proceso asegura que el sistema esté listo para leer cualquier dato capturado por el dispositivo.

```
this.SensorKinect = KinectSensor.GetDefault();  //Obtención de instancia de la clase KinectSensor
```

Seguido se inicializa un objeto que permite leer los datos de los cuadros de seguimiento corporal emitidos por el sensor. “BodyFrameSource” es una propiedad del

sensor Kinect que permite tener acceso a los datos continuos del seguimiento corporal. De igual manera se tiene el método OpenReader() para la creación de una instancia de BodyFrameReader. Este proceso permite que el lector reciba continuamente los cuadros con la información del seguimiento corporal.

```
this.bodyFrameReader = this.SensorKinect.BodyFrameSource.OpenReader();//Acceso a datos de seguimiento corporal
```

Para procesar las imágenes a color. Se inicializa un lector mediante la propiedad "ColorFrameSource" del Kinect, el cual proporciona acceso a los cuadros de color capturados por la cámara RGB del dispositivo Kinect. Se crea una instancia del lector utilizando el método OpenReader(), que permite suscribirse a eventos que notifican la llegada de nuevos cuadros, esto asegura que el sistema pueda recibir y procesar continuamente la información visual en color generada por la cámara.

```
this.LecturaFrameColor = this.kinect.ColorFrameSource.OpenReader(); //Acceso a datos de cámara RGB
```

De igual manera se establece una descripción del cuadro de color, utilizando el método "CreateFrameDescription", especificando el formato BGRA (Blue, Green, Red, Alpha channel), este proceso permite generar metadatos sobre los cuadros, incluyendo información sobre su tamaño y formato.

```
FrameDescription colorFrameDescription = this.kinect.ColorFrameSource.CreateFrameDescription(ColorImageFormat.Bgra);//Descripcion del cuadro a color
```

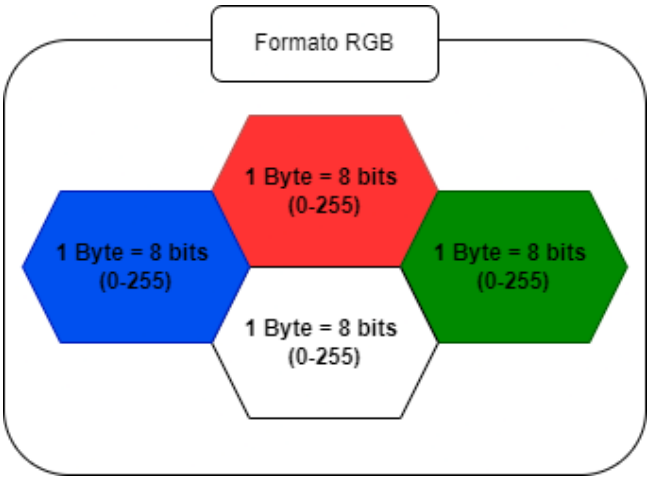
El formato que se ha configurado es Bgr32, el cual indica que cada pixel se compone de 32 bits en la memoria, estos representan los canales azul, verde, rojo y un canal no utilizado. Este proceso se logra por medio de la creación del objeto WriteableBitmap y se toma las dimensiones explicadas en el paso anterior "colorFrameDescription".

```
this.MapaBits = new WriteableBitmap(colorFrameDescription.Width, colorFrameDescription.Height, 96.0, 96.0, PixelFormats.Bgr32, null);
```

Como se muestra en la línea de configuración de formato se utiliza el formato Bgr32, el cual es el formato más sencillo de controlar. Cada pixel se puede representar como el esquema de la Figura 28, cada color representado tiene un valor de 8 bits y cada bit tiene un valor de un 0 o un 1, al tener 8 bits, eso quiere decir que se tiene una combinación de 8 ceros y unos, con estos valores de 0 y 1 se puede lograr hasta 256 combinaciones diferentes. Estas combinaciones son las que permiten crear diferentes tonos y brillos de colores

Figura 28

Formato RGB



Nota. Diagrama esquemático que muestra la representación del formato RGB para formar las imágenes a color del sistema. Fuente: Elaborado por el autor

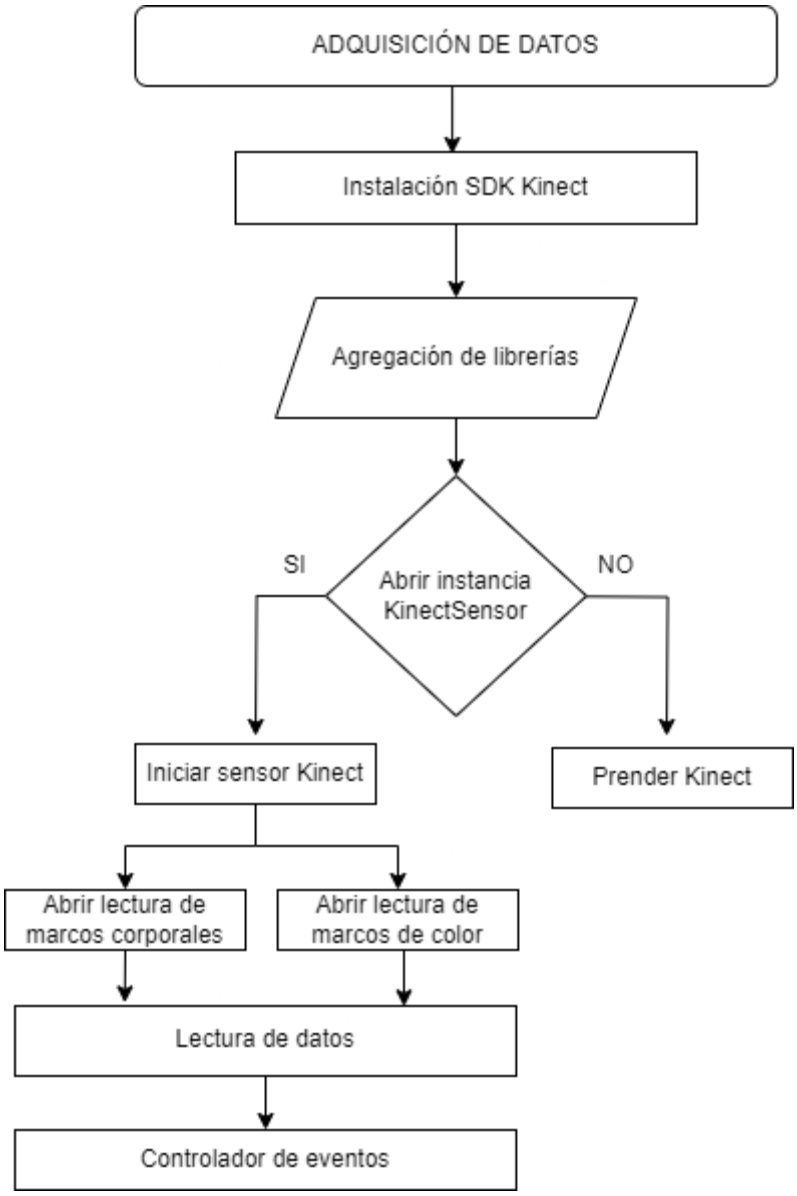
Finalmente se crea un controlador de eventos del lector de cuadro de color, esto permite llamar al método de lectura de cuadros de color cada vez que el dispositivo Kinect recepte un cuadro de color disponible y así realizar el procesamiento en tiempo real.

```
this.LecturaFrameColor.FrameArrived += LecturaFrameColor_FrameArrived; //Creacion de evento para cada cuadro de color
```

Bajo este contexto en la Figura 29 se muestra un diagrama de flujo del bloque de adquisición de datos.

Figura 29

Diagrama de flujo adquisición de datos



Nota. Diagrama de flujo del proceso de adquisición de datos mediante los sensores y cámaras del dispositivo Kinect. Fuente: Elaborado por el autor

Tras activar los sensores y sus respectivos lectores, el sistema inicia la lectura de datos, los cuales son gestionados por el controlador de eventos, encargado de procesar la

información en tiempo real. Esto prepara el sistema para avanzar al siguiente bloque del proceso general.

3.3.1.3 Procesamiento de datos

Esta sección detalla el proceso del tercer bloque del sistema general correspondiente al procesamiento de datos.

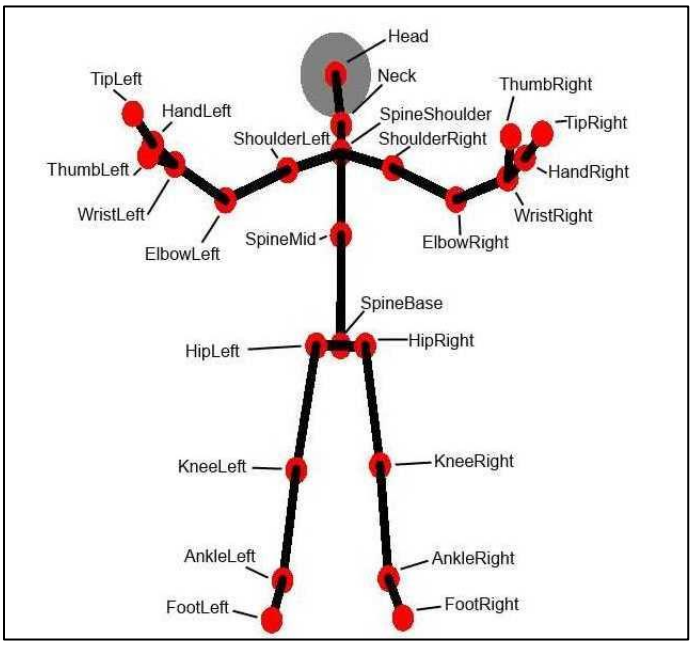
El procesamiento de datos se subdivide en tres diagramas de flujo que describen el proceso de formación del esqueleto humano mediante la captura de datos tridimensionales. Además, incluyen el procesamiento para la validación de las dos variantes de ejercicios y finalmente el control de repeticiones realizadas.

Procesamiento formación de esqueleto humano

El dispositivo Kinect procesa imágenes de color y mapas de profundidad en tiempo real, incorporando algoritmos avanzados de visión artificial para el seguimiento esquelético. Este sistema identifica 25 puntos articulares del cuerpo humano, facilitando una detección precisa de las mismas. En la Figura 30 se ilustra las 25 articulaciones rastreadas por el sensor Kinectv2.

Figura 30

Articulaciones detectadas por el Kinect



Nota. La figura muestra los 25 puntos de las articulaciones detectadas por el sensor Kinect. Fuente: Elaborado por (*Reconocimiento de posturas con Kinect I*, s. f.)

Cada articulación es un punto representado por tres coordenadas (X, Y, Z), situados en el punto (0.0.0) del sensor de profundidad.

X es el valor de la distancia lateral desde el centro del sensor hacia la izquierda o derecha.

Y es el valor de la altura desde el sensor hacia arriba o abajo.

Z es el valor de la profundidad, o distancia desde el sensor hacia adelante o atrás

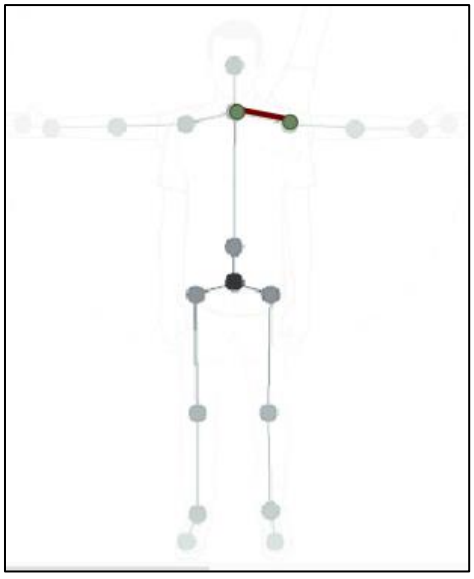
Se obtiene la descripción de los frames de profundidad capturados por el sensor Kinect. De igual manera que el paso anterior se hace referencia a una instancia de la clase Kinect, el cual permite dar acceso a la fuente de los frames de profundidad, estos frames son

imágenes que contienen información con relación a las distancias de los objetos tomando como referencia el sensor de profundidad, el cual cada pixel representa la distancia desde el punto de vista del sensor hasta un objeto en específico como es una persona. “DepthFramesource” proporciona metadatos de los frames, como es la resolución del frame en ancho y alto. Habilitado el flujo de datos de profundidad y color se realiza el procesamiento para la formación de la estructura esquelética del cuerpo humano en tiempo real. Se inicializa una lista de “tuplas” en la cual cada tupla representa un hueso definido como una conexión entre dos articulaciones “JoinType”.

En la Figura 31 se ilustra la formación de un hueso por medio de la unión de dos articulaciones.

Figura 31

Formación de hueso entre dos articulaciones



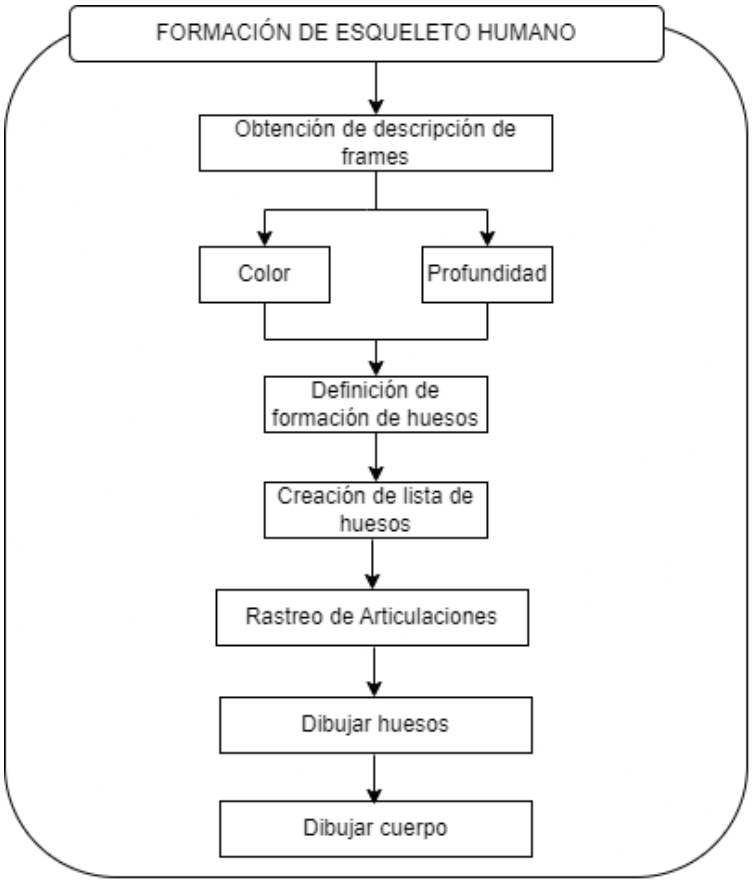
Nota. Formación de hueso por medio de la unión de dos articulaciones. Fuente: Elaborado por el autor

La formación del esqueleto se agrega por pares de articulaciones “bones” que representan un hueso. La estructura se divide en secciones como son: torso, brazos, piernas.

Se establece el color del cuerpo detectado y se crea una instancia de la clase DrawingGroup para la actualización de los gestos del cuerpo. En la Figura 32 se presenta el diagrama de flujo que ilustra el proceso de detección del esqueleto humano, validando así las explicaciones proporcionadas en esta sección.

Figura 32

Diagrama de flujo formación de esqueleto humano



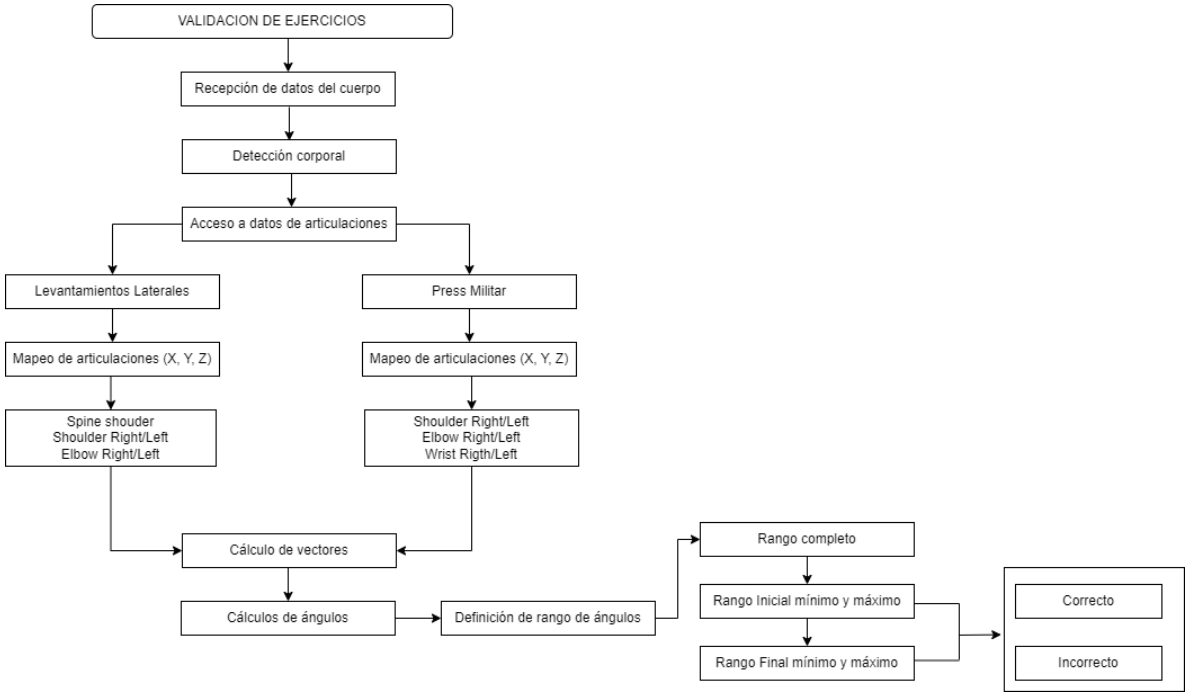
Nota. Diagrama de flujo del proceso de formación del esqueleto humano. Fuente: Elaborado por el autor

Definido el proceso de detección del esqueleto humano se procede con el procesamiento de los datos en las dos variantes de ejercicios.

El procesamiento de cada ejercicio comienza con la adquisición de datos corporales, para extraer las coordenadas del esqueleto humano. Posterior, se calculan vectores entre las articulaciones involucradas, y se aplica cálculo vectorial mediante el producto escalar para determinar el ángulo entre dos vectores en las posiciones inicial y final de cada ejercicio. Los rangos de ángulos preestablecidos permiten evaluar si estas posiciones cumplen con los criterios definidos para considerar si una repetición es correcta o incorrecta, lo cual determina la ejecución adecuada de cada repetición. En la Figura 33 se ilustra el diagrama de flujo con el procesamiento de validación de las dos variantes de ejercicios de hombros.

Figura 33

Diagrama de flujo validación de ejercicios



Nota. Diagrama de flujo del proceso de validación de ejercicios. Fuente: Elaborado por el autor

Procesamiento de validación de ejercicio levantamientos laterales

Mediante la suscripción de un método a un evento de lectura de marcos corporales, se inicia el proceso de recepción de datos, disparándose cada vez que un nuevo frame de datos está disponible para ser procesado. El uso asincrónico del evento se debe a que dentro de este proceso se realizan operaciones asincrónicas, como es el envío de datos a la plataforma en la nube u operaciones de entrada/salidas llamadas APIs.

Una vez obtenidos los datos de los frames corporales, se verifica la detección del cuerpo. Si la detección es exitosa, se obtiene acceso a los datos de todas las articulaciones del cuerpo.

Para el ejercicio de levantamientos laterales, se consideran los valores de las siguientes articulaciones: spine shoulder, shoulder right/left y elbow right/left, que corresponden a la columna vertebral, los hombros y los codos, respectivamente.

En la Figura 34 se ilustra las líneas de programación para la obtención de los valores de las articulaciones implicadas en el ejercicio de levantamientos laterales.

Figura 34

Programación para la obtención de articulaciones levantamientos laterales

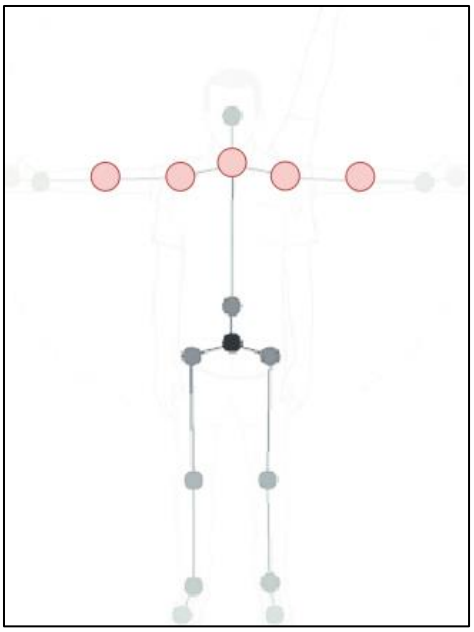
```
Joint SpineShoulder = body.Joints[JointType.SpineShoulder];     //Obetener articulaciones del SpineShoulder
Joint hombroDerecho = body.Joints[JointType.ShoulderRight];     // Obtener las articulaciones del hombro derecho
Joint codoDerecho = body.Joints[JointType.ElbowRight];          // Obtener las articulaciones del hombro codo derecho
Joint hombroIzquierdo = body.Joints[JointType.ShoulderLeft];    // Obtener las articulaciones del hombro izquierdo
Joint codoIzquierdo = body.Joints[JointType.ElbowLeft];         // Obtener las articulaciones del hombro codo izquierdo
```

Nota. Líneas de programación para la obtención de valores de articulaciones en levantamientos laterales. Fuente: Elaborado por el autor

En la Figura 35 se ilustra las articulaciones utilizadas para el cálculo de vectores en el ejercicio de levantamientos laterales.

Figura 35

Articulaciones consideradas en el ejercicio de levantamientos laterales.



Fuente: Elaborado por el autor

Con la obtención de estos valores el sistema utiliza las coordenadas en tres dimensiones para el cálculo de vectores entre los puntos de las respectivas articulaciones. En la obtención de estos vectores se toma como vértice o punto de origen la articulación spine shoulder extendiéndose hacia shoulder (hombro) y elbow (codo). La formación del vector del hombro se expresa por medio de la ecuación 3.

Ecuación 3: Vector spine shoulder con el hombro

$$VectorAB = (ShoulderRight/Left.X - SpineShoulder.X,$$
$$ShoulderRight/Left.Y - SpineShoulder.Y,$$
$$ShoulderRight/Left.Z - SpineShoulder.Z)$$

La formación del vector del codo se expresa por medio de la siguiente ecuación 4.

Ecuación 4: Vector spine shoulder con el codo

$$VectorAC = (ElbowRight/Left.X - SpineShoulder.X,$$
$$ElbowRight/Left.Y - SpineShoulder.Y,$$
$$ElbowRight/Left.Z - SpineShoulder.Z)$$

De esta manera, se forman dos vectores: uno dirigido hacia el hombro y el otro hacia el codo. Estos vectores son cruciales para calcular el ángulo entre ellos, proporcionando así los valores angulares en las posiciones inicial y final durante la ejecución de la repetición del ejercicio.

En la ecuación 5, se muestra el producto escalar entre dos vectores para el cálculo del ángulo entre ellos.

Ecuación 5: Producto escalar entre dos vectores.

$$\vec{A}.\vec{B} = |\vec{A}\,||\vec{B}|\cos(\theta)$$

En la cual el producto escalar o producto punto es igual a la multiplicación de las dos magnitudes de los vectores por el coseno del ángulo formado entre estos dos vectores. Reemplazando la terminología con la del sistema, la ecuación para el cálculo del ángulo se denota de la siguiente forma.

Ecuación 6: Cálculo de ángulo entre dos vectores.

$$\cos(\theta) = \frac{\overrightarrow{vectorAB}.\overrightarrow{vectorAC}}{|\overrightarrow{vectorAB}\,||\overrightarrow{vectorAC}|}$$

El resultado del ángulo que se obtendrá será en radianes por lo que se aplica la ecuación 7 para obtener el valor en grados.

Ecuación 7: Transformación de radianes a grados.

$$Agrados = Aradianes(\frac{180}{\pi})$$

En la Figura 36 se ilustra las líneas de programación aplicadas en el sistema para el cálculo del ángulo entre los dos vectores.

Figura 36

Proceso cálculo de ángulo en las dos variantes de ejercicios

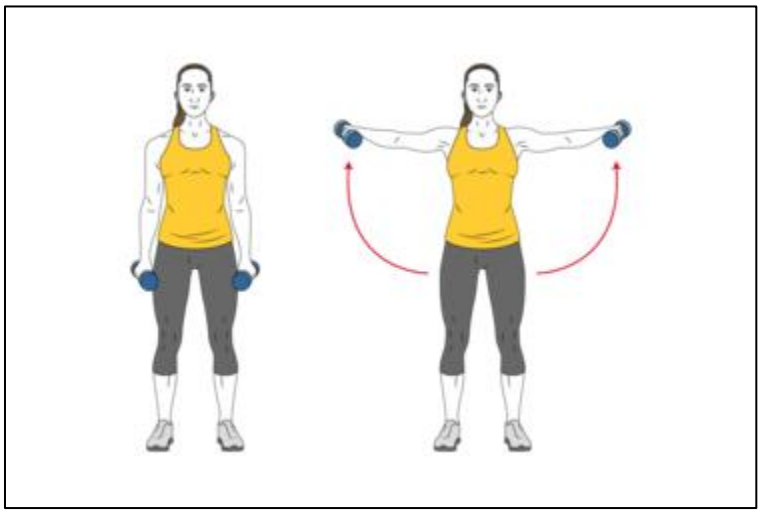
```
public int CalcularAngulo(CameraSpacePoint puntoA, CameraSpacePoint puntoB, CameraSpacePoint puntoC)
{
    // Calcula los vectores a partir de los puntos
    Vector3D vectorAB = new Vector3D(puntoB.X - puntoA.X, puntoB.Y - puntoA.Y, puntoB.Z - puntoA.Z);     //Vector hacia el hombro
    Vector3D vectorAC = new Vector3D(puntoC.X - puntoA.X, puntoC.Y - puntoA.Y, puntoC.Z - puntoA.Z);     //Vector hacia el codo

    // Calcula de ángulo
    double productoPunto = Vector3D.DotProduct(vectorAB, vectorAC);                                     //Operación del producto punto entre los dos vectores
    double magnitudAB = vectorAB.Length;                                                                //Calculo de la magnitud del vector hacia el hombro
    double magnitudAC = vectorAC.Length;                                                                //Calculo de la magnitud del vector hacia el codo
    double cosenoAngulo = productoPunto / (magnitudAB * magnitudAC);                                    //Calculo del coseno del ángulo
    double anguloRadianes = Math.Acos(cosenoAngulo);                                                    //Calculo del coseno en radianes
    // Convierte el ángulo a grados
    double anguloGrados = anguloRadianes * (180.0 / Math.PI);                                           //Conversion de radianes a grados
    return (int)Math.Truncate(anguloGrados);                                                               // Redondeo y converción a entero
}
```

Nota. Líneas de programación para el cálculo de ángulo por medio del producto escalar. Fuente: Elaborado por el autor

Al obtener el valor del ángulo entre los dos vectores, se obtiene un vector cada 30 frames por segundo, para lo cual se realizan pruebas para definir los ángulos en las diferentes posiciones de ejecución de las repeticiones del ejercicio. En la figura 37 se ilustra la posición inicial y final de ejecución del ejercicio de levantamientos laterales.

Figura 37

Posición inicial y final del ejercicio de levantamientos laterales



Nota. La parte izquierda de la figura representa la posición inicial del ejercicio mientras que la parte derecha de la figura representa la posición final. Fuente: (*Elevación lateral de hombros con mancuernas de pie brazos estirados*, s. f.)

En la figura 38 se define el rango de ángulos en la posición inicial y el rango de ángulos en la posición final del ejercicio de levantamientos laterales.

Figura 38

Rango de ángulos para una repetición completa de levantamientos laterales

```
// Ángulos para la posición inicial y final
private const double AnguloInicialMin = 20; // Mínimo ángulo aceptable en posición inicial
private const double AnguloInicialMax = 40; // Máximo ángulo aceptable en posición inicial
private const double AnguloFinalMin = 1; // Mínimo ángulo aceptable en posición final
private const double AnguloFinalMax = 20; // Máximo ángulo aceptable en posición final
```

Nota. Definición de ángulos mínimos y máximos en la posición inicial y final que valida una repetición completa del ejercicio de levantamientos laterales Fuente: Elaborado por el autor

En la figura se puede apreciar que se tiene un valor mínimo y un valor máximo para cada posición, inicial y final respectivamente. Estos valores abarcan tanto rangos de ángulos

de repeticiones correctas como incorrectas y así validar cuando el usuario ejecute una repetición completa del ejercicio de levantamientos laterales.

Dentro del procesamiento de datos se establece un método el cual establece una condición lógica que verifica si el ángulo calculado está en el rango de los valores predefinidos para validar como una repetición completa, repetición correcta o repetición incorrecta.

$$valorAnguloCalculado \geq valorMinimo \;\&\&\; valorAnguloCalculado \leq valorMaximo$$

El valor del ángulo calculado debe ser mayor o igual que el valor mínimo predefinido y menor o igual que el valor máximo predefinido, tanto en la posición inicial como en la posición final.

Validado los ángulos y el estado del ejercicio tanto en la posición inicial como en la posición final. Se realiza la definición del rango de valores para la validación de una repetición realizada correcta e incorrectamente.

En la Figura 39 se ilustra los valores de los rangos de ángulos en la posición correcta del ejercicio de levantamientos laterales

Figura 39

Rango de ángulos para una repetición correcta de levantamientos laterales

```
//Angulos de posiciones correctas
private const double AnguloInicialCorrectoMin = 25;
private const double AnguloInicialCorrectoMax = 35;
private const double AnguloFinalCorrectoMin = 5;
private const double AnguloFinalCorrectoMax = 11;
```

Nota. Definición de ángulos mínimos y máximos en la posición inicial y final que valida una repetición correcta de levantamientos laterales. Fuente: Elaborado por el autor

En la figura 40 se ilustra los valores de los rangos de ángulos incorrectos en la posición inicial del ejercicio de levantamientos laterales.

Figura 40

Rango de ángulos para determinar una repetición incorrecta en la posición inicial de levantamientos laterales

```
private const double AnguloInicialBajoIncorrectoMin = 20;        //Separado al torso
private const double AnguloInicialBajoIncorrectoMax = 24;        //Separado al torso
private const double AnguloInicialAltoIncorrectoMin = 36;         //Junto al torso
private const double AnguloInicialAltoIncorrectoMax = 40;        //Junto al torso
```

Nota. Definición de ángulos mínimos y máximos en la posición inicial que valida una repetición incorrecta de elevaciones laterales. Fuente: Elaborado por el autor

En la figura 41 se ilustra los valores de los ángulos incorrectos en la posición final del ejercicio de elevaciones laterales.

Figura 41

Rango de ángulos para determinar una repetición incorrecta en la posición final de levantamientos laterales

```
private const double AnguloFinalBajoIncorrectoMin = 1;          //Posicion muy baja de lo brazos
private const double AnguloFinalBajoIncorrectoMax = 4;          //Posicion muy baja de los brazos
private const double AnguloFinalAltoIncorrectoMin = 12;         //Posicion muy alta de los brazos
private const double AnguloFinalAltoIncorrectoMax = 20;         //Posicion muy alta de los brazos
```

Nota. Definición de ángulos mínimos y máximos en la posición final que valida una repetición incorrecta de levantamientos laterales. Fuente: Elaborado por el autor

Por lo tanto, para que una repetición sea clasificada como correcta o incorrecta, las posiciones de los brazos deben encontrarse dentro del rango total de los ángulos predefinidos. Si no se cumple este criterio, la posición no será reconocida como parte del entrenamiento.

Procesamiento de validación de ejercicio press militar

Con relación al ejercicio de press militar se toma en cuenta los valores de las articulaciones en tres dimensiones de: shoulder right/left, Elbow right/left y Wrist right/left estas articulaciones corresponden al hombro derecho/izquierdo, codo derecho/izquierdo y muñeca derecha/izquierda respectivamente.

La elección de estas tres articulaciones se debe al rango de movimiento en el ejercicio de press militar, ya que se realiza un movimiento de extensión de los brazos, a diferencia del ejercicio de levantamientos laterales con su movimiento de elevación de brazos.

Para la obtención de los vectores se toma como vértice o punto de origen la articulación shoulder right y shoulder left extendiéndose hacia elbow (codo) y wrist (muñeca).

En la Figura 42 se ilustra las líneas de programación para la obtención de los valores de las articulaciones implicadas en el ejercicio de levantamientos laterales.

Figura 42

Programación para la obtención de articulaciones Press militar

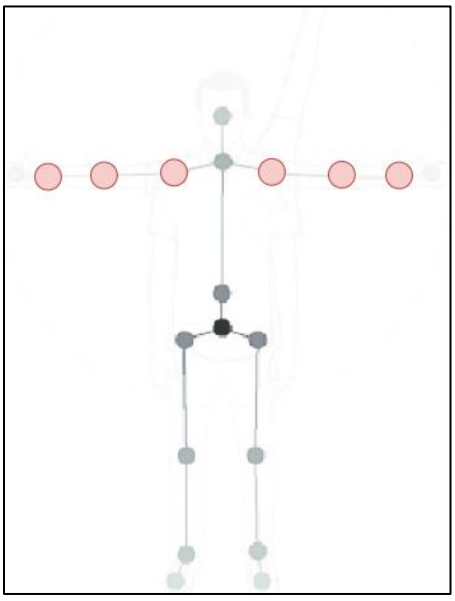
```
Joint hombroDerecho = body.Joints[JointType.ShoulderRight];          //Obetener articulaciones del hombro derecho
Joint hombroIzquierdo = body.Joints[JointType.ShoulderLeft];         //Obetener articulaciones del hombro izquierdo
Joint codoDerecho = body.Joints[JointType.ElbowRight];               // Obtener las articulaciones del codo derecho
Joint munecaDerecha = body.Joints[JointType.WristRight];             // Obtener las articulaciones de la mano derecha
Joint codoIzquierdo = body.Joints[JointType.ElbowLeft];              // Obtener las articulaciones del codo izquierdo
Joint munecaIzquierda = body.Joints[JointType.WristLeft];            // Obtener las articulaciones de la mano izquierda
```

Nota. Líneas de programación para la obtención de valores de articulaciones en press militar. Fuente: Elaborado por el autor

En la figura 43 se ilustra las articulaciones utilizadas para el cálculo de vectores en el ejercicio de Press militar.

Figura 43

Articulaciones consideradas en el ejercicio de Press militar.



Fuente: Elaborado por el autor

La formación del vector del hombro hacia el codo se expresa en la siguiente ecuación.

Ecuación 8: Vector entre hombro y codo

$$VectorAB = (ElbowRight/Left.X - ShoulderRight/Left.X,$$
$$ElbowRight/Left.Y - ShoulderRight/Left.Y,$$
$$ElbowRight/Left.Z - ShoulderRight/Left.Z)$$

La formación del vector de la muñeca se expresa de la siguiente manera.

Ecuación 9: Vector entre hombro y muñeca

$$VectorAB = (WristRight/Left.X - ShoulderRight/Left.X,$$
$$WristRight/Left.Y - ShoulderRight/Left.Y,$$
$$WristRight/Left.Z - ShoulderRight/Left.Z)$$

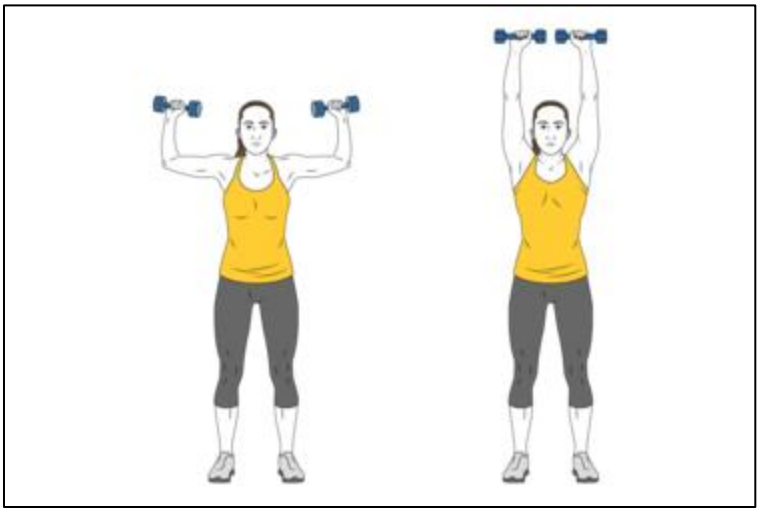
Con los valores de los vectores formados se realiza las mismas operaciones que se utilizó en el proceso de levantamientos laterales por medio de las ecuaciones: Ecuación 10, Ecuación 11 y Ecuación 12.

La lógica presentada en la Figura 36 se aplica directamente a las líneas de código que gestionan la creación de vectores y el cálculo vectorial subsiguiente para determinar el ángulo entre dos vectores con relación al ejercicio de press militar. Bajo este proceso se determina los rangos de ángulos para la validación de una repetición, tomando en cuenta los estados de posición, en la posición inicial y en la posición final respectivamente.

En la figura 44 se ilustra la posición inicial y final de ejecución del ejercicio de press militar.

Figura 44

Posición inicial y final del ejercicio de press militar



Nota. La parte izquierda de la figura representa la posición inicial del ejercicio mientras que la parte derecha de la figura representa la posición final. Fuente:(*Press militar o de hombros con mancuernas de pie*, s. f.)

En la figura 45 se define el rango de ángulos en la posición inicial y el rango de ángulos en la posición final del ejercicio de press militar.

Figura 45

Rango de ángulos para una repetición completa de press militar.

```
// Rango de ángulos para la posición inicial y final
private const int AnguloInicialMin = 25;                    // Angulo minimo aceptable en posición inicial
private const int AnguloInicialMax = 50;                    // Angulo maximo aceptable en posición inicial
private const int AnguloFinalMin = 2;                       // Angulo minimo aceptable en posición final
private const int AnguloFinalMax = 22;                      // Angulo maximo aceptable en posición final
```

Nota. Definición de ángulos mínimos y máximos en la posición inicial y final que valida una repetición completa del ejercicio de press militar. Fuente: Elaborado por el autor

En la figura se muestra los valores de los ángulos predefinidos para establecer la posición inicial y final de la repetición de press militar, si no cumple este criterio, no se detectará una posición de entrenamiento. De igual manera dentro de estos rangos se debe establecer los rangos para repeticiones correctas e incorrectas.

Se establece el mismo método de los levantamientos laterales para definir una condición lógica que verifica si el ángulo calculado está en el rango de los valores predefinidos para validar como una repetición completa, repetición correcta o repetición incorrecta aplicado al ejercicio de press militar.

$$valorAnguloCalculado \geq valorMinimo \;\&\&\; valorAnguloCalculado \leq valorMaximo$$

Validado los ángulos y el estado del ejercicio tanto en posición inicial como en posición final. Se realiza la definición del rango de valores de ángulos para la validación de una repetición realizada de forma correcta e incorrecta.

En la figura 46 se ilustra los valores de los rangos de ángulos en la posición correcta del ejercicio de press militar.

Figura 46

Rango de ángulos de validación de una repetición correcta de press militar

```
// Angulos Posicion Correctas
private const int AnguloInicialCorrectoMin = 33;
private const int AnguloInicialCorrectoMax = 43;
private const int AnguloFinalCorrectoMin =7;
private const int AnguloFinalCorrectoMax = 14;
```

Nota. Definición de ángulos mínimos y máximos en la posición inicial y final que valida una repetición correcta de press militar. Fuente: Elaborado por el autor

En la Figura 47 se ilustra los valores de los rangos de ángulos incorrectos en la posición inicial del ejercicio de press militar.

Figura 47

Rango de ángulos de validación de una repetición incorrecta en la posición inicial de press militar.

```
//Angulos Posiciones Incorrectas
private const int AnguloInicialBajoIncorrectoMin = 25;          //Manos muy separadas del hombro
private const int AnguloInicialBajoIncorrectoMax = 32;          //Manos muy separadas del hombro
private const int AnguloInicialAltoIncorrectoMin = 44;          //Mano muy junta al hombro
private const int AnguloInicialAltoIncorrectoMax = 50;          //Mano muy junta al hombro
```

Nota. Definición de ángulos mínimos y máximos en la posición inicial que validan una repetición incorrecta de press militar. Fuente: Elaborado por el autor

En la Figura 48 se ilustra los valores de los rangos de ángulos incorrectos en la posición final del ejercicio de press militar.

Figura 48

Rango de ángulos de validación de una repetición incorrecta en la posición final de press militar.

```
private const int AnguloFinalBajoIncorrectoMin = 15;          //Brazos muy recogidos
private const int AnguloFinalBajoIncorrectoMax = 22;          //Brazos muy recogidos
private const int AnguloFinalAltoIncorrectoMin = 2;           //Brazos muy extendidos
private const int AnguloFinalAltoIncorrectoMax = 6;           //Brazos muy e3xtendidos
```

Nota. Definición de ángulos mínimos y máximos en la posición final que validan una repetición incorrecta de press militar. Fuente: Elaborado por el autor

Una vez establecidos los rangos de ángulos que determinan si una repetición se ha ejecutado de forma completa, correcta o incorrecta, se detalla el procedimiento para el procesamiento del control y conteo de repeticiones. Este paso es esencial para monitorear el desempeño y progreso durante la sesión de entrenamiento y brindar un control detallado para el usuario.

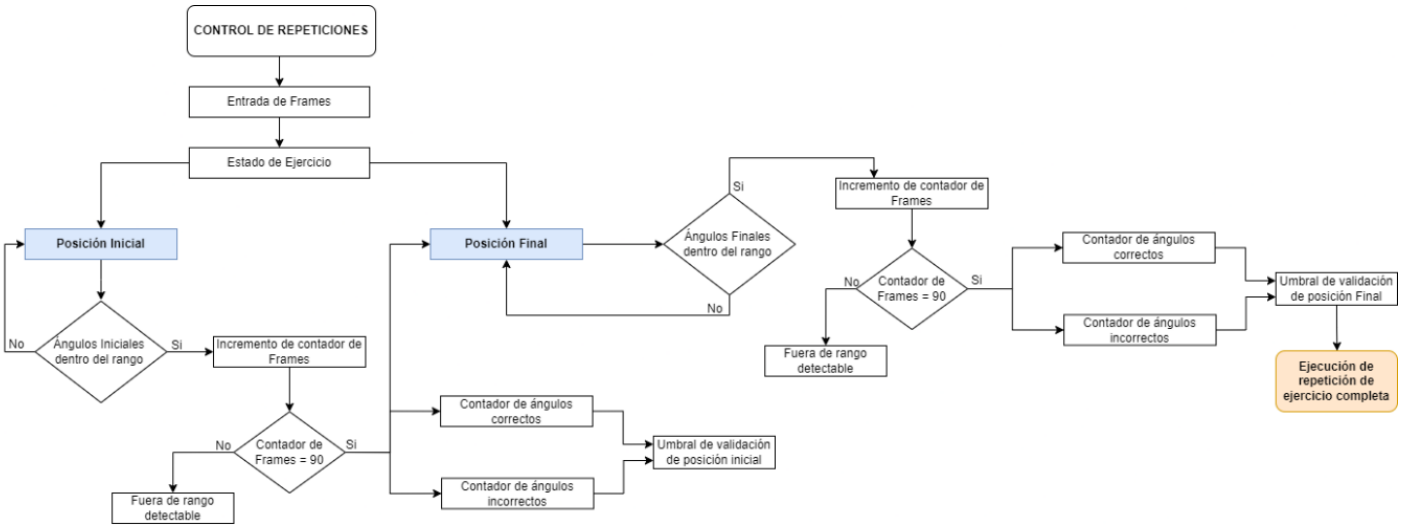
Procesamiento de control de repeticiones

Inicialmente, el usuario especifica la cantidad de repeticiones que desea realizar en una de las dos variantes de ejercicios de hombros. Cada repetición se supervisa mediante varias verificaciones, que incluyen el análisis de la cantidad de frames para determinar los estados de posición y compararlos con ángulos predefinidos previamente. Esto se realiza mediante contadores que ayudan a validar el tipo de repetición realizada.

La Figura 49 muestra el diagrama de flujo que detalla el procesamiento del control de repeticiones para las dos variantes de ejercicios de hombros.

Figura 49

Diagrama de flujo del procesamiento de control de ejecución de repeticiones



Nota. Diagrama de flujo del proceso del control de ejecución de repeticiones. Fuente: Elaborado por el autor

El diagrama de flujo ilustra el control de repeticiones para la ejecución de dos variantes de ejercicios de hombros, organizado en varias etapas claves. Inicialmente, la entrada de frames activa los estados del ejercicio, los cuales son validados según los rangos de ángulos en las posiciones inicial o final.

Una vez verificado el estado del ejercicio, se activa un contador de frames que varía desde el número 1 hasta 90. Este contador facilita la ejecución controlada del ejercicio, teniendo en cuenta que se reciben 30 frames por segundo, lo que significa que la posición inicial o final debe mantenerse durante 3 segundos para que el sistema realice la detección adecuada.

Tras esta fase, se activan contadores para los ángulos correctos e incorrectos. Además, se establece un umbral para determinar si la posición inicial del ejercicio se valida como correcta o incorrecta.

Si el contador de ángulos correctos en la posición inicial alcanza o supera el valor de 46, se incrementa un contador que valida esta posición como correcta. Si el contador de ángulo incorrectos en la posición inicial supera al valor de 44 se incrementa un contador que valida la posición inicial como incorrecta. De esta manera se cubre los 90 ángulos calculados y se valida la posición inicial como correcta o incorrecta.

En la Figura 50 se ilustra la lógica de programación para el control de la posición inicial como correcta o incorrecta.

Figura 50

Lógica de programación para el control de la posición inicial.

```
// Proceso para contar una repetición
if (estadoEjercicio == EstadoEjercicio.EnPosicionInicial && EstaEnRango(anguloDerecho, AnguloInicialMin, AnguloInicialMax) &&
    EstaEnRango(anguloIzquierdo, AnguloInicialMin, AnguloInicialMax))                                    //Activacion de estado en posicion i
{
    contadorFramesI++;                                                  //Aumento de contador de Frames
    if (contadorFramesI >= 1 && contadorFramesI <= 90)                  //Condicion para contar 90 frames
    {
        if (posicionInicialCorrecta)                                    //condición de posicion correcta
        {
            contadorAnguloInicialCorrectos++;                           //Aumento de contador de ángulos correctos
        }
        if (posicionInicialIncorrecta)                                  //Condicion de posicion incorrecta
        {
            contadorAnguloInicialIncorrectos++;                         //Aumento de contador de ángulos incorrectos
        }
    }
    if (contadorFramesI >= FramesParaTresSegundos)                      //Condicion cuando los frames son igual a 90
    {
        if (contadorAnguloInicialCorrectos >= 46)                       //Condición para activar el contador de posicion inicial correcta
        {
            correctas++;                                                //Aumento de contador para la posicion correcta
        }
        if (contadorAnguloInicialIncorrectos > 44)                      //Condición para activar el contador de posicion inicial incorrecta
        {
            incorrectasI++;                                             //Aumento de contador para la posicion incorrecta.
```

Nota. Líneas de programación para el control de la posición inicial del ejercicio de levantamientos laterales. Fuente: Elaborado por el autor

Seguido a este paso se tiene la activación del estado en posición final, y desactivación del estado en posición inicial, en la cual se aplica la misma lógica para el contador de frames,

las verificaciones de condiciones y la evaluación de la posición final como correcta e incorrecta.

En la Figura 51 se ilustra la lógica de programación para el control de la posición final como correcta o incorrecta.

Figura 51

Lógica de programación para el control de la posición inicial.

```
if (estadoEjercicio == EstadoEjercicio.EnPosicionFinal && EstaEnRango(anguloDerecho, AnguloFinalMin, AnguloFinalMax) &&
    EstaEnRango(anguloIzquierdo, AnguloFinalMin, AnguloFinalMax))                    //Activación de posición final
{
    contadorFramesF++;                                                               //Aumento de contador de Frames
    if (contadorFramesF >= 1 && contadorFramesF <= 90)                               //Condicion para contar 90 frames
    {
        if (posicionFinalCorrecta)                                                   //Condición de posicion correcta
        {
            contadorAnguloFinalCorrectos++;                                          //Aumento de contador de ángulos correctos
        }
        if (posicionFinalIncorrecta)                                                 //Condición de posición incorrecta
        {
            contadorAnguloFinalIncorrectos++;                                        //Aumento de contador de ángulos incorrectos
        }
    }
    if (contadorFramesF >= FramesParaTresSegundos)                                   //Condicion cuando los frames son igual a 90
    {
        if (contadorAnguloFinalCorrectos >= 46)                                      //Condición para activar el contador de posicion inicial correcta
        {
            correctas++;                                                             //Aumento de contador para la posicion correcta
        }
        if (contadorAnguloFinalIncorrectos > 44)                                     //Condición para activar el contador de posicion inicial incorrecto
        {
            incorrectasF++;
```

Nota. Líneas de programación para el control de la posición final del ejercicio de levantamientos laterales. Fuente: Elaborado por el autor

Una vez finalizado el proceso de validación en la posición final de la repetición del ejercicio, se evalúan los contadores de posición correcta en estado inicial y final. Para validar una posición como correcta, ambas posiciones (inicial y final) deben generar un contador de "1". Si la suma de estos contadores es "2", la repetición se considera correcta. En cambio, si alguno de los contadores de estado de posición inicial o final es igual a "1", la repetición se clasifica como incorrecta. Tras evaluar, el estado del ejercicio regresa a la posición inicial y así comenzar una nueva repetición. Además, existe un contador de repeticiones que se restablece al valor inicial una vez que se completan todas las repeticiones definidas por el usuario.

Concluida la explicación del bloque de procesamiento de información, el cual valida cada repetición ejecutada y controla las repeticiones correctas e incorrectas, se avanza al siguiente bloque. Este se centra en el tratamiento de la información para su visualización por parte del usuario y entrega de datos cruciales al instructor, facilitando el seguimiento y monitoreo del entrenamiento en tiempo real.

3.3.1.4 Tratamiento de información

El proceso de tratamiento de información se refiere al manejo de los datos ya procesados, los cuales tienen dos aspectos fundamentales como es la parte del usuario e instructor/a.

Por medio de XAML (Extensible Application Markup Language) se desarrolla las interfaces de usuario en aplicaciones de Windows. XAML es parte del framework de desarrollo.NET y es muy utilizado en tecnologías como Windows Presentation Foundation (WPF) para la creación de interfaces de usuario (UI) en aplicaciones de escritorio. WPF consta de dos archivos.

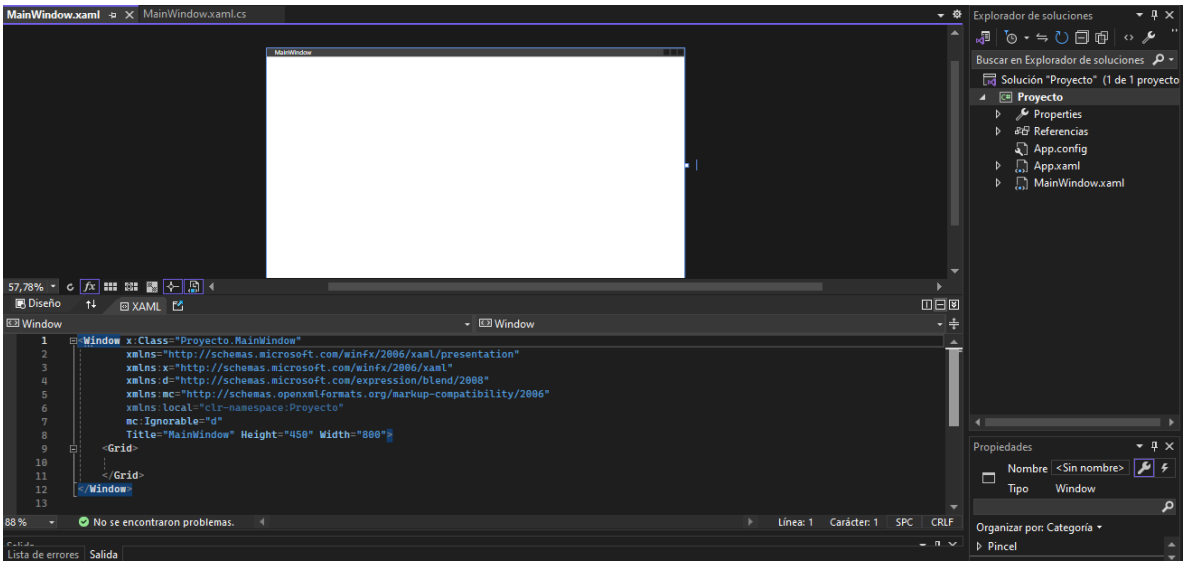
XAML que soporta diferentes componentes como es la integración de gráficos, animaciones, video, audio, los cuales permiten definir la estructura y apariencia de la interfaz de usuario. En el archivo XAML, se define diferentes elementos como cuadros de texto, botones, layouts y más, incluyendo la configuración de sus propiedades como colores, tipo de letra, tamaños y posiciones.

En la ventana de diseño mostrada en la Figura 52, es una representación visual que permite configurar y apreciar los cambios en tiempo real. En la parte inferior de la figura se

tiene el código XAML, donde se escribe o modifica los elementos que componen la interfaz de usuario.

Figura 52

Creación de interfaz de usuario WPF



Nota. Visualización de archivo XAML. Fuente: Elaborado por el autor

El otro archivo que compone WPF es el archivo de código ".cs", este archivo contiene el código C# que maneja la lógica de programación detrás de la interfaz de usuario definida en el XAML. Por medio del lenguaje de programación C# se maneja diferentes eventos y acciones, haciendo que la interfaz sea interactiva, cerrando y abriendo nuevas ventanas o respondiendo a las acciones como un click.

El sistema cuenta con diversas ventanas de interacción para el usuario, como la ventana inicial, el registro de usuarios, el menú de entrenamiento, guías de ejercicios para las dos variantes propuestas, y los entrenamientos en tiempo real.

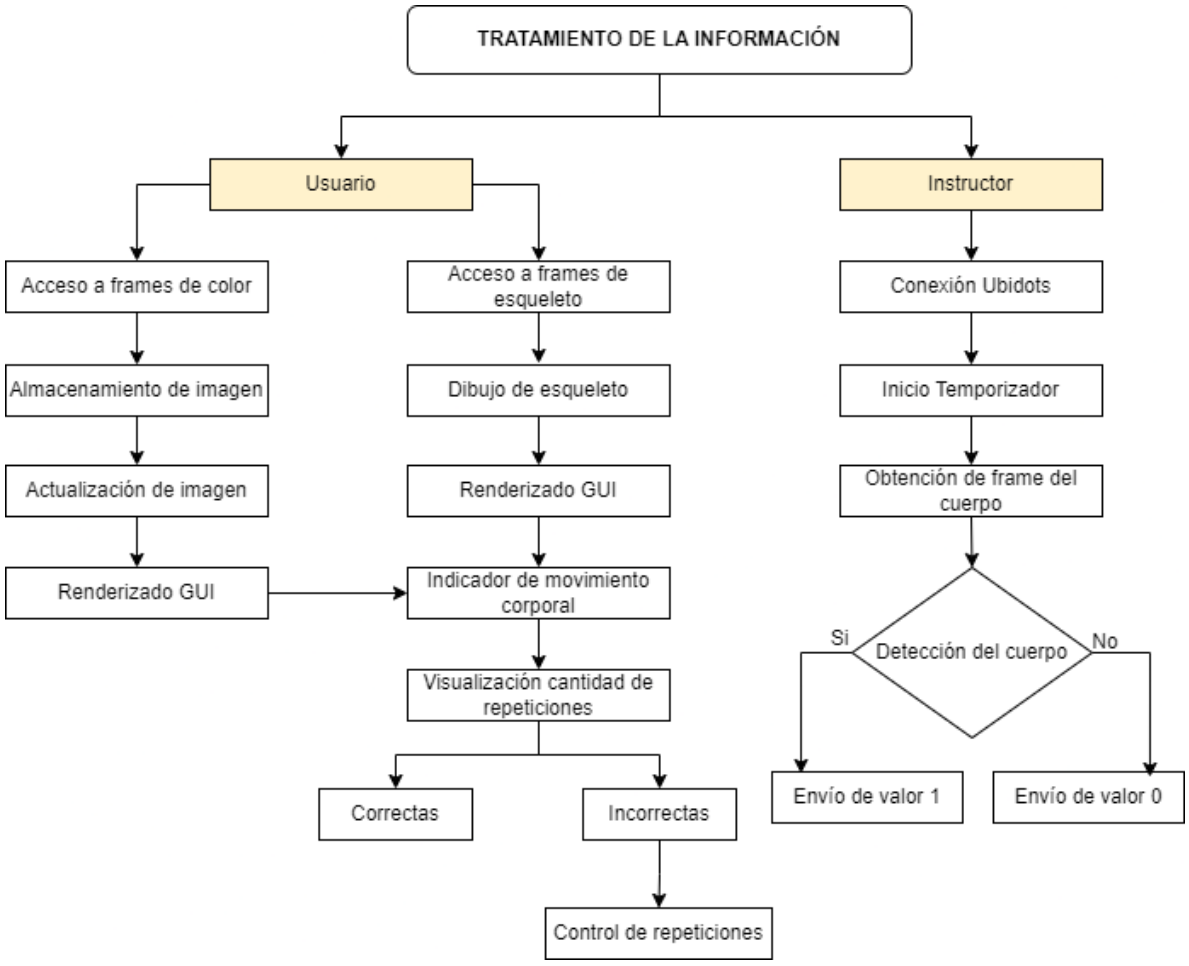
Mediante las interfaces creadas en XAML, el usuario puede visualizar su esqueleto superpuesto en tiempo real, facilitando el seguimiento de los movimientos durante la ejecución de las dos variantes de ejercicios de hombros. Esto se logra mediante los datos obtenidos de la cámara RGB y el sensor de profundidad. Además, el sistema proporciona indicadores de movimientos de los brazos para asegurar que cada repetición sea ejecutada de manera controlada y detectada correctamente, permitiendo así la visualización en tiempo real del número de repeticiones realizadas correcta e incorrectamente. Una vez completadas todas las repeticiones programadas, el usuario puede acceder a los detalles de las repeticiones que se realizaron incorrectamente.

Por su parte, el instructor cuenta con acceso a una interfaz gráfica alojada en la nube a través de la plataforma Ubidots, la cual permite monitorear la disponibilidad del espacio de entrenamiento para los ejercicios de hombros. Además, esta interfaz facilita el seguimiento en tiempo real de la cantidad de repeticiones realizadas correcta e incorrectamente por el usuario.

En la Figura 53 se ilustra el diagrama de bloques que muestra el tratamiento de información para su visualización tanto por el usuario como por el instructor.

Figura 53

Diagrama de bloques del tratamiento de información



Nota. Diagrama de bloques del proceso de tratamiento de la información para el usuario y para el instructor. Fuente: Elaborado por el autor

Para la visualización de video en tiempo real se configura un controlador de eventos que se dispara cada vez que llega un nuevo cuadro de color.

Se obtiene los cuadros de colores disponibles captados por el dispositivo Kinect mediante la cámara RGB. Si se ha obtenido un cuadro de color con éxito se accede a la

descripción del cuadro de color incluyendo los detalles como son el ancho y el alto de la imagen, y otros metadatos necesarios para el procesamiento de la imagen.

Los datos del cuadro de color se copian directamente en el búfer de la imagen y se convierten a BGRA para la visualización en tiempo real por medio de la interfaz del usuario.

En la Figura 54 se especifica las líneas de programación aplicadas para la visualización de video en tiempo real.

Figura 54

Programación aplicada a la visualización de imágenes a color

```
public void LecturaFrameColor_FrameArrived(object sender, ColorFrameArrivedEventArgs e)                    //Control de eventos para frames de color
{
    using (ColorFrame colorFrame = e.FrameReference.AcquireFrame()) //se adquiere el frame mas reciente
    {
        if (colorFrame != null)                                     //condicion cuando se obtiene un frame
        {
            FrameDescription colorFrameDescription = colorFrame.FrameDescription;  //Se obtiene la descripcion del frame
            using (KinectBuffer colorBuffer = colorFrame.LockRawImageBuffer()) ;   //bloqueo de bufer cuando se esta procesando un frame
            {
                this.MapaBits.Lock();                                               //Bloqueo de bitmapa
                if ((colorFrameDescription.Width == this.MapaBits.PixelWidth) && (colorFrameDescription.Height == this.MapaBits.PixelHeight)) //Verificacion
                {
                    //Copia de datos al bitmapa
                    colorFrame.CopyConvertedFrameDataToIntPtr(                  //Llamado de metodo para copiar los datos de la imagen al bufer
                            this.MapaBits.BackBuffer,                           //Buffer para dibujar los datos de imagen
                            (uint)(colorFrameDescription.Width * colorFrameDescription.Height * 4), //Cantidad de datos a copiar
                            ColorImageFormat.Bgra);                                                 //Se especifica el formato de color a usar
                    this.MapaBits.AddDirtyRect(new Int32Rect(0, 0, this.MapaBits.PixelWidth, this.MapaBits.PixelHeight)); //Actualizacion de interfaz al usu
                }
                this.MapaBits.Unlock();                                         //Desbloqueo de bitmapa
            }
        }
    }
}
```

Nota. Líneas de programación para visualizar las imágenes de color captadas por la cámara RGB. Fuente: Elaborado por el autor

En el proceso de acceso a los frames del esqueleto, cuando llega un frame de cuerpo, cada articulación del cuerpo detectado tiene una posición en el espacio tridimensional representada por un CameraSpacePoint o punto espacial de la cámara. Estos puntos necesitan ser convertidos a DepthSpacePoint o punto espacial de profundidad para que puedan ser visualizados en una pantalla en el espacio bidimensional.

Finalizado este proceso se utiliza la información de las articulaciones convertidas para dibujar el esqueleto en un contexto de dos dimensiones. Esto incluye dibujar huesos y articulaciones.

Los puntos en el espacio de la cámara se definen en tres dimensiones (X, Y, Z) donde cada punto representa la posición real en el espacio medido desde el dispositivo Kinect.

Los puntos en el espacio de profundidad simplifican las coordenadas a un plano bidimensional (X, Y) en el cual los valores representan la distancia desde el dispositivo Kinect en términos de pixeles en la imagen de profundidad.

En la Figura 55 se ilustran las líneas de programación utilizadas para representar el esqueleto humano en la interfaz de usuario.

Figura 55

Programación aplicada a la visualización de esqueleto humano

```
//Coversion de las articulaciones de un espacio de 3D a 2D para visualizacion del usuario
foreach (JointType jointType in joints.Keys)                                    //Iteracion de cada articulacion en el diccionario
{
    CameraSpacePoint position = joints[jointType].Position;                     //Variable para representar una posicion tridimencional
    if (position.Z < 0)                                                         //Verificacion que el valor de la componente z menor a 0
    {
        position.Z = InferredZPositionClamp;                                    //Ajuste de valor predeterminado, para que el valor z no sea confiable
    }

    DepthSpacePoint depthSpacePoint = this.coordinateMapper.MapCameraPointToDepthSpace(position); //Estructura para representar la posicion en tridimensional en bidimensional
    jointPoints[jointType] = new Point(depthSpacePoint.X, depthSpacePoint.Y);   //Almacenamiento de las posiciones convertidas y poder accerde mediante Point
}
this.DibujarCuerpo(joints, jointPoints, dc, dibujarEsfero);
```

Nota. Líneas de programación para visualizar esqueleto humano en la interfaz del usuario. Fuente: Elaborado por el autor

3.3.1.5 Alertas

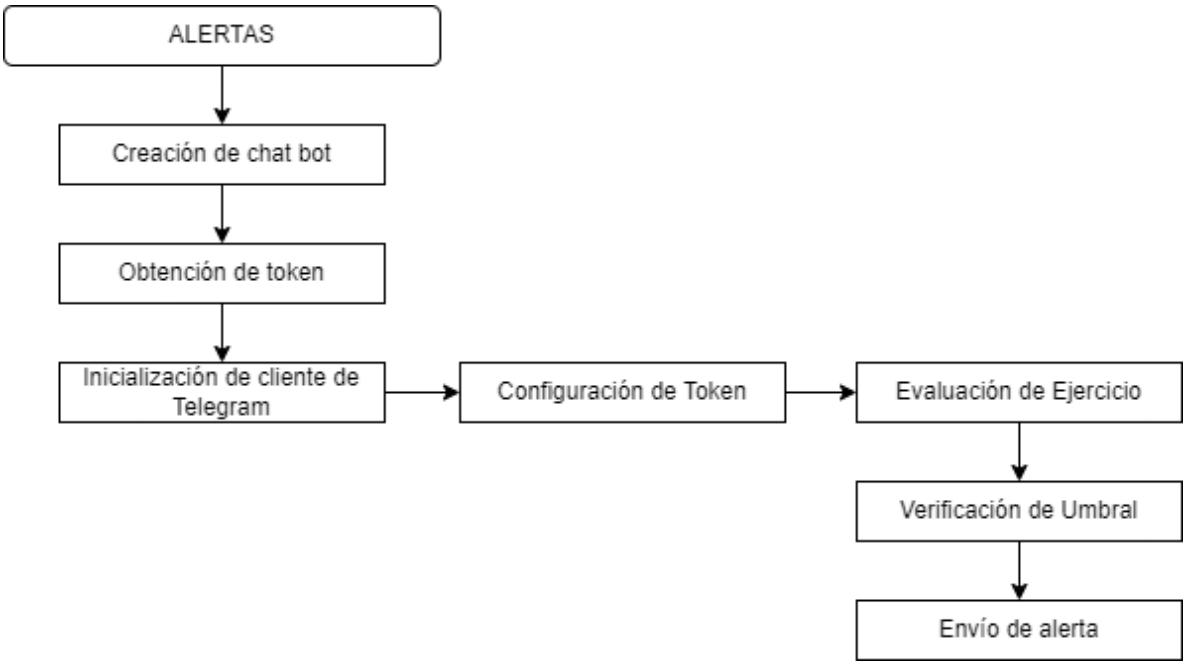
El proceso de generación de alertas se integra mediante el uso de Telegram. Esta plataforma permite crear un chatbot que se puede conectar con diversos entornos de programación, incluido Visual Studio. En Visual Studio, se utiliza la librería Telegram Bot para inicializar un cliente, configurándolo con el token correspondiente.

Las alertas se generan evaluando la cantidad de repeticiones realizadas incorrectamente. Al superar un umbral preestablecido, se activa una notificación que se envía directamente al instructor mediante el chat ID en Telegram.

En la Figura 56, se presenta el diagrama de bloques que ilustra el proceso de generación de alertas del sistema, detallando cómo se evalúan las condiciones que desencadenan estas notificaciones.

Figura 56

Diagrama de bloques generación de alertas del sistema



Nota. Diagrama de bloques del proceso de generación de alertas dirigidas al instructor. Fuente: Elaborado por el autor

El diagrama de bloques inicia con la creación del chat bot a través del BotFather en Telegram, un proceso que permite obtener un token esencial para configurar el sistema de envío de alertas.

Posteriormente, se declara una variable del tipo TelegramBotClient, que se utiliza para interactuar con la API de Telegram. Esta variable se inicializa mediante un constructor que crea una nueva instancia de TelegramBotClient utilizando un token. El token de identificación funciona como una clave secreta la cual permite la autenticación con la API de Telegram.

```
botClient = new TelegramBotClient("6722450347:AAFR9q-qn_eJ38U9yk4pHx6UNnSC7_lTIGM");                    //Coonfiguracion de token
```

Una vez configurado el token, el sistema evalúa cada serie de repeticiones. Si el total de repeticiones incorrectas supera el 50% del total realizado, se envía una alerta a través del chat ID correspondiente. Para el cálculo de porcentaje se aplica la relación del total de repeticiones incorrectas sobre el total de repeticiones realizadas.

En la Figura 57 se muestra las líneas de programación que define el umbral para el envío de alertas por medio del cliente de Telegram y el ChatId correspondiente.

Figura 57

Programación aplicada para envío de alertas

```
if (porcentajeIncorrectas > 0.5) // Umbral para envio de alerta de telegram
{
    //Envio de alerta
    await botClient.SendTextMessageAsync(chatId: "5599983300", text: $"El usuario {nombreUsuario} ha completado {repeticionesIncorrectas} de {totalRepeticionesRealizadas} repeticiones de elevaciones laterales de manera incor
}
```

Nota. Líneas de programación para envío de alertas por medio del bot cliente de telegram. Fuente: Elaborado por el autor

3.3.1.6 Reporte

El último componente esencial del sistema es la generación de reportes. Este bloque maneja el flujo de información y toma decisiones críticas durante el proceso para capturar y mostrar los ángulos ejecutados en cada posición de los ejercicios y en cada brazo del usuario.

El proceso de generación de reportes comienza con la captura de datos durante la sesión de ejercicios. Esto incluye la detección de movimientos correctos e incorrectos, así como el cálculo de los ángulos.

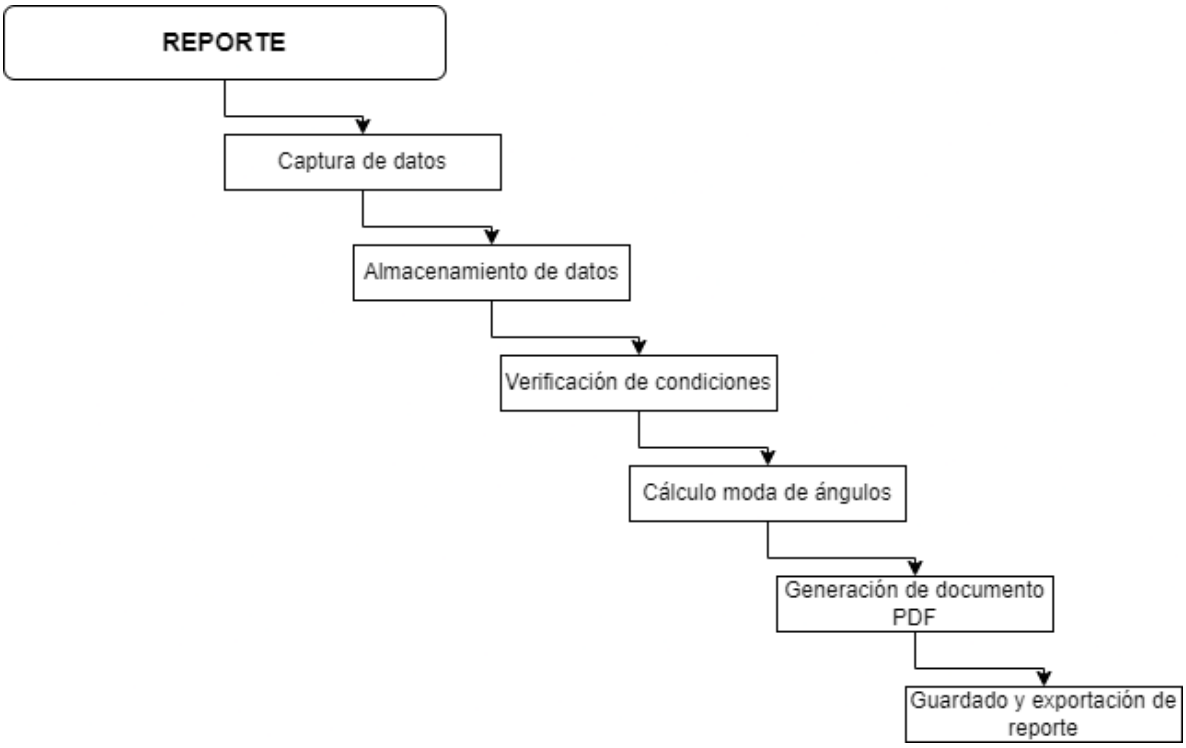
Después de la captura de datos, el siguiente paso es almacenar esta información en una estructura organizada, como es la creación de listas. Este proceso es fundamental para el manejo eficiente de los datos. Posterior, se deben verificar decisiones clave, como asegurar que se haya completado el número total de repeticiones. También se define en qué condiciones se aplicará el cálculo de la moda de ángulos, ya sea en estados iniciales o finales, y tanto en posiciones correctas como incorrectas.

Obtenidos los ángulos, se genera el reporte en formato PDF. Este paso incluye la creación del documento y la presentación de los ángulos en las tablas específicas para cada posición de entrenamiento. Finalmente, el último paso informa al usuario que el documento PDF ha sido generado y almacenado en una ubicación específica del ordenador.

En la Figura 58 se ilustra el diagrama de bloques para la generación de reportes PDF.

Figura 58

Diagrama de bloques de generación de reporte PDF



Nota. Diagrama de bloques del proceso de generación de reporte PDF. Fuente: Elaborado por el autor.

La captura de datos inicia cuando se detecta el esqueleto del usuario y se obtienen los valores de las articulaciones para cada variante de ejercicio, los valores de las articulaciones son necesarios para el cálculo de los ángulos, como se explicó en procesos anteriores.

Se crean dos clases: una para controlar el reporte de la posición inicial y otra para el reporte de la posición final. Dentro de las clases se crean listas y variables, las cuales almacenan la información relacionada con las repeticiones de los ejercicios y los ángulos involucrados en cada posición de ejecución.

En las clases y variables se definen propiedades de “get y set” indicando que tiene métodos getter y setter de forma automática, lo que permite obtener y establecer el valor de la propiedad. Se crean listas para almacenar los ángulos medidos del brazo derecho e izquierdo, que están categorizados según si las condiciones iniciales son correctas o incorrectas. De igual manera, se obtiene el estado de las evaluaciones de las dos posiciones de los ejercicios, esto mediante la propiedad tipo cadena. Se emplea la propiedad “get”, que permite calcular y retornar un valor basado en los datos de otras clases, como son los ejercicios de Levantamientos Laterales y Press Militar.

En la Figura 59 se ilustra las líneas de programación para la creación de variables y listas con las propiedades de get y set.

Figura 59

Creación de variables y listas.

```
public class ControlReportePosicionInicial
{
    4 referencias
    public int NumeroRepeticionT { get; set; }                          // Variable número de repeticion
    4 referencias
    public string EstadoPosicionInicial { get; set; }                   // Variable para estado inicial o final de ejercicio
    6 referencias
    public List<int> AngulosDerechosIC { get; set; } = new List<int>();   // Lista para guardar los ángulos del brazo derecho
    6 referencias
    public List<int> AngulosIzquierdosIC { get; set; } = new List<int>(); // Lista para guardar los ángulos del brazo izquierdo
    8 referencias
    public List<int> AngulosDerechosII { get; set; } = new List<int>();   // Lista para guardar los ángulos del brazo derecho
    6 referencias
    public List<int> AngulosIzquierdosII { get; set; } = new List<int>(); // Lista para guardar los ángulos del brazo izquierdo
    12 referencias
    public string EstadoEvaluacionI { get; set; }                       // Variable para guardar el estado correcto o incorrecto
    6 referencias
    public int ModaAnguloDerechoIC { get; set; }                        // Variable para almacenar el promedio de angulos brazo derecho Pos.IC
    6 referencias
    public  int ModaAnguloIzquierdoIC { get; set; }                     // Variable para almacenar el promedio de angulos brazo izquierdo Pos.IC
    6 referencias
    public int ModaAnguloDerechoII { get; set; }                        // Variable para almacenar el promedio de angulos brazo izquierdo Pos.II
    6 referencias
    public int ModaAnguloIzquierdoII { get; set; }                      // Variable para almacenar el promedio de angulos brazo izquierdo Pos.II
}
```

Nota. Líneas de programación para la creación de variables y listas de almacenamiento de valores. Fuente: Elaborado por el autor

La información que almacenarán estas clases se basa en dos condiciones para cada repetición. Se calculan y guardan los diferentes ángulos en la posición inicial correcta e

incorrecta del brazo derecho para la primera clase. Para la segunda clase, se obtienen y guardan los ángulos en la posición inicial correcta e incorrecta del brazo izquierdo.

Este proceso se realiza tanto para la posición inicial como para la posición final de cada repetición de ejercicio.

En la Figura 60 se ilustra las líneas de programación para acceder a los ángulos y guardarlos en las listas de la posición inicial de la repetición del ejercicio.

Figura 60

Programación de acceso y guardado de valores a listas

```
contadorFramesI++;                                                    //Aumento de contador de Frames
if (contadorFramesI >=1 & contadorFramesI <= 90)                     //Condicion para contar 90 frames
{
    if (posicionInicialCorrecta)                                      //condición de posicion correcta
    {
        // Calcula el ángulo
        anguloDerechoC = CalcularAngulo(SpineShoulder.Position, hombroDerecho.Position, codoDerecho.Position); //Calcula angulos correctos
        anguloIzquierdoC = CalcularAngulo(SpineShoulder.Position, hombroIzquierdo.Position, codoIzquierdo.Position);//Calcula angulos incorrectos
        contadorAnguloInicialCorrectos++;                                 //Aumento de contador de ángulos correctos
        ControlRepeticionesIniciales.LastOrDefault()?.AngulosDerechosIC.Add(anguloDerechoC);    //Agregacion de angulos a la lista
        ControlRepeticionesIniciales.LastOrDefault()?.AngulosIzquierdosIC.Add(anguloIzquierdoC);//Agragacion de ángulos a la lista
    }
    //Condicion de posicion incorrecta
    if (posicionInicialIncorrecta)
    {
        // Calcula el ángulo
        anguloDerechoI = CalcularAngulo(SpineShoulder.Position, hombroDerecho.Position, codoDerecho.Position); //Calcula angulos correctos
        anguloIzquierdoI = CalcularAngulo(SpineShoulder.Position, hombroIzquierdo.Position, codoIzquierdo.Position);//Calcula angulos incorrectos
        contadorAnguloInicialIncorrectos++;                               //Aumento de contador de ángulos incorrectos
        ControlRepeticionesIniciales.LastOrDefault()?.AngulosDerechosII.Add(anguloDerechoI);        //Agregacion de angulos a la lista
        ControlRepeticionesIniciales.LastOrDefault()?.AngulosIzquierdosII.Add(anguloIzquierdoI);    //Agregacion de angulos a la lista
    }
```

Nota. Líneas de programación para acceder y guardar los valores de los ángulos, bajo las condiciones de estado de posición correcta e incorrecta. Fuente: Elaborado por el autor

De esta manera, las clases se llenan con los valores de los diferentes ángulos para cada estado de posición. A estos ángulos, almacenados en las listas, se calcula la moda, basándose en si el estado ha sido validado como correcto o incorrecto. Durante este proceso, los valores se actualizan continuamente a través de la colección para su inclusión en la generación del archivo PDF.

En la Figura 61 se ilustra las líneas de programación para el cálculo de la moda de los ángulos guardados en la lista correspondiente a la posición inicial.

Figura 61

Programación de cálculo de moda de la lista de ángulos.

```
if (contadorFramesI == FramesParaTresSegundos)                                    //Condicion cuando los frames son igual a 90
{
    anguloDerecho = 0;
    anguloIzquierdo = 0;
    // Calcula el promedio de los ángulos capturados
    if (contadorAnguloInicialCorrectos >= 46)//Condición para activar el contador de posicion inicial correcta
    {
        modaAnguloDerechoIC = Moda(ControlRepeticionesIniciales.LastOrDefault().AngulosDerechosIC);      //Cálculo de moda angulosderechosIC
        modaAnguloIzquierdoIC = Moda(ControlRepeticionesIniciales.LastOrDefault().AngulosIzquierdosIC);   //Cálculo de moda angulosizquierdosIC
        estadoEvaluacionI = "Correcto";
        correctas++;//Aumento de contador para la posicion correcta
        Dispatcher.Invoke(() =>
        {   //Actualización de valores en PDF
            ControlRepeticionesIniciales.LastOrDefault().EstadoEvaluacionI = estadoEvaluacionI;
            ControlRepeticionesIniciales.LastOrDefault().ModaAnguloDerechoIC = modaAnguloDerechoIC;
            ControlRepeticionesIniciales.LastOrDefault().ModaAnguloIzquierdoIC = modaAnguloIzquierdoIC;
        });
    }
    if (contadorAnguloInicialIncorrectos > 44)//Condición para activar el contador de posicion inicial incorrecta
    {
        modaAnguloDerechoII = Moda(ControlRepeticionesIniciales.LastOrDefault().AngulosDerechosII);     //Cálculo de moda angulosderechosII
        modaAnguloIzquierdoII = Moda(ControlRepeticionesIniciales.LastOrDefault().AngulosIzquierdosII); //Cálculo de moda angulosizquierdosII
        estadoEvaluacionI = "Incorrecto";
        incorrectasI++;//Aumento de contador para la posicion incorrecta
        string detalle = DetalleErrorPosicionInicial(modaAnguloDerechoII, modaAnguloIzquierdoII);    //Creación de variable, almacena el detalle de Pos.I
        var estadoIncorrectoI = new ControlRepeticionesI                                              //Creación de instancia para asignar informacion
        {
            NumeroRepeticionI = numeroTotalRepeticiones,                                              //Agregación de valor de repeticion tabla control
            EstadoPosicionI = "Posicion Inicial",                                                     //Agregación de valor de posicion tabla control
            DetalleAnguloI = detalle                                                                  //Agregación de detalle de posicion tabla control
        };
        Dispatcher.Invoke(() =>                                                                       //Actualización UI
```

Nota. Líneas de programación para calcular la moda de los ángulos en la lista de estado inicial. Fuente: Elaborado por el autor

El proceso descrito se aplica tanto a la lista de estados finales como a los dos ejercicios de variantes de hombros. Finalmente, se crea un método para la generación del reporte en formato PDF, y se actualizan los valores de las columnas con los datos almacenados en las variables y listas.

Para la demostración de los datos se trabaja con cinco columnas, la primera columna muestra el número de repeticiones realizadas, la segunda columna muestra el estado de posición (posición inicial o posición final), la tercera columna muestra los valores de la moda de los ángulos en el brazo derecho, la cuarta columna muestra los valores de la moda de los

ángulos en el brazo izquierdo, la quinta y última columna muestra la evaluación del estado de la posición, definiéndola como correcta o incorrecta.

En la Figura 62 se ilustran las líneas de programación utilizadas para añadir información en cada columna y generar las tablas en el reporte PDF.

Figura 62

Generación de tablas de reporte

```
// Se añade subtitulo para la primera tabla
Paragraph subPosicionInicial = new Paragraph("Estado de repetición en Posicion Inicial", FontFactory.GetFont("Times New Roman", 12, Font.BOLD));
subPosicionInicial.Alignment = Element.ALIGN_CENTER;
document.Add(subPosicionInicial);
document.Add(new Paragraph("\n"));//Espacio
Paragraph rangocorrectoI = new Paragraph("Rango correcto 25° - 35°", FontFactory.GetFont("Times New Roman", 12, Font.BOLD));
rangocorrectoI.Alignment = Element.ALIGN_CENTER;
document.Add(rangocorrectoI);
document.Add(new Paragraph("\n"));
// Numero de columnas y nombres
PdfPTable tableI = new PdfPTable(5);
tableI.AddCell("Número");                                   // Columna número de repeticiones
tableI.AddCell("Posición");                                 // Columna de posicion inicial o final
tableI.AddCell("Ángulo de brazo derecho");                  // Columna para ángulos c/i brazo derecho
tableI.AddCell("Ángulo de brazo Izquierdo");                // Columna para ángulos c/i brazo izquierdo
tableI.AddCell("Estado de Posición");                       // Columna para posicion correcta o incorrecta
foreach (var item in ControlRepeticionesIniciales)          // Inicio de bucle
{
    tableI.AddCell(item.NumeroRepeticionT.ToString());      // Se añade los valores de numero de repeticion del 1-8
    tableI.AddCell(item.EstadoPosicionInicial);             // Se añade el estado de posicion
    if (item.EstadoEvaluacionI == "Correcto")               // Condicion si la posicion es correcta
    {
        tableI.AddCell(item.ModaAnguloDerechoIC.ToString() + "°");      // Se añade ángulos derechos correctas
        tableI.AddCell(item.ModaAnguloIzquierdoIC.ToString() + "°");    // Se añade ángulos izquierdos correctos
    }
    else if (item.EstadoEvaluacionI == "Incorrecto")        // Condicion si la posicion es incorrecta
    {
        tableI.AddCell(item.ModaAnguloDerechoII.ToString() + "°");      //Se añade ángulos izquierdos incorrectos
        tableI.AddCell(item.ModaAnguloIzquierdoII.ToString() + "°");    //Se añade ángulos izquierdos incorrectos
    }
    else
    {
        tableI.AddCell("");   // En caso de que no haya un estado definido, añadir una celda vacía
        tableI.AddCell("");   // En caso de que no haya un estado definido, añadir una celda vacía
    }
    tableI.AddCell(item.EstadoEvaluacionI);                 // Se añade el estado de posicion
}
document.Add(tableI);                                       // Se agrega la tabla al documento pdf
// Espacio entre tablas
document.Add(new Chunk("\n"));
```

Nota. Líneas de programación para añadir la información en las tablas de reporte de entrenamiento. Fuente: Elaborado por el autor

Capítulo IV: RESULTADOS

En este capítulo, se presenta las pruebas de funcionamiento global del sistema llevadas a cabo en el gimnasio "ZENERGYM". Estas pruebas tienen como objetivo verificar la efectividad en el monitoreo, detección y corrección de las dos variantes de ejercicios planteados. Además, se realiza una encuesta de evaluación para determinar el nivel de aceptación del sistema por parte de los clientes e instructores.

4.1 Casos de pruebas

Se describen cada uno de los casos de pruebas identificados como necesarios para comprobar el funcionamiento final del sistema.

4.1.1 Test Eléctrico	**Subsistema Prueba**	**Eléctrico**
	¿Prueba de despliegue?	**Si/No**
Descripción: Pruebas eléctricas del sistema con la activación del dispositivo Kinect y los sensores.		
Prerrequisitos: **1. Adquisición de adaptador de voltaje Kinect.** **2. Conexión entre adaptador de voltaje y adaptador de cable USB 3.0**		
Pasos: ✓ Verificación de conexión entre partes del adaptador de voltaje Kinect. ✓ Comprobación de suministro de voltaje del adaptador Kinect. ✓ Verificación de encendido de dispositivo Kinect.		

✓ Verificación de activación sensores dispositivo Kinect.
Resultado esperado: Disponer todos los componentes necesarios para el suministro de energía del sistema, activar el dispositivo Kinect por medio del adaptador de voltaje.

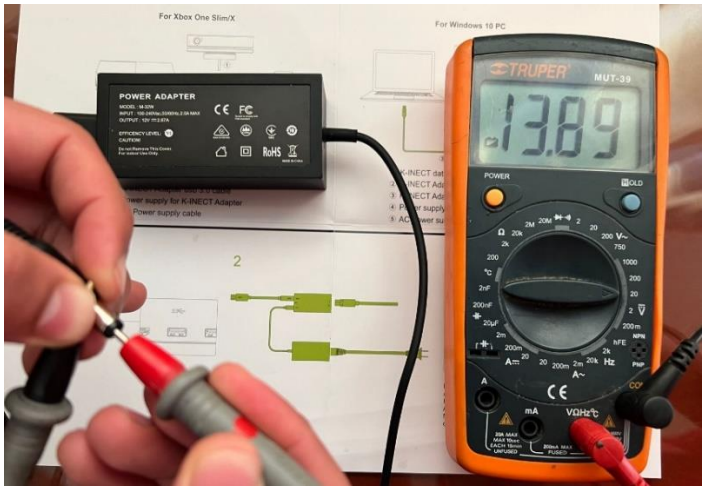
Resultado obtenido:

Las pruebas eléctricas del sistema se inician por medio de la activación del dispositivo Kinect, el valor de voltaje para el funcionamiento correcto del dispositivo Kinect es de 12 - 15v. Este valor se verifica mediante del uso del multímetro, midiendo en el pin de la salida de voltaje.

En la Figura 63 se ilustra el valor de voltaje medido en la salida del adaptador de poder del dispositivo Kinect.

Figura 63

Medición salida de voltaje adaptador de poder



Fuente: Elaborado por el autor

El indicador de activación del dispositivo Kinect es por medio del encendido de su logo de color blanco en la parte derecha, la activación de los sensores se verifica mediante el encendido de tres líneas de color rojo en la parte central del dispositivo.

En la Figura 64 se ilustra la activación del dispositivo Kinect y encendido de sus sensores.

Figura 64

Activación de dispositivo Kinect y sensores

Fuente: Elaborado por el autor

4.1.2 Test de Hardware	**Sistema Prueba**	**Hardware**
	¿Prueba de despliegue?	**Si/No**

Descripción:

Prueba de dispositivos electrónicos utilizados para el desarrollo del sistema.

Prerrequisitos:

1. Adquisición de dispositivo Kinect.

2. Revisión de puertos USB 3.0 en el ordenador.

3. Estudio de funcionamiento de dispositivo Kinect

4. Instalación SDK Kinect

Pasos:

- ✓ Verificación de conexión entre Kinect y ordenador.
- ✓ Verificación de SDK de Kinect para Windows.
- ✓ Verificación de funcionamiento de sensores.

Resultado esperado:

El ordenador debe ser capaz de activar los sensores del dispositivo Kinect por medio del SDK instalado en el ordenador.

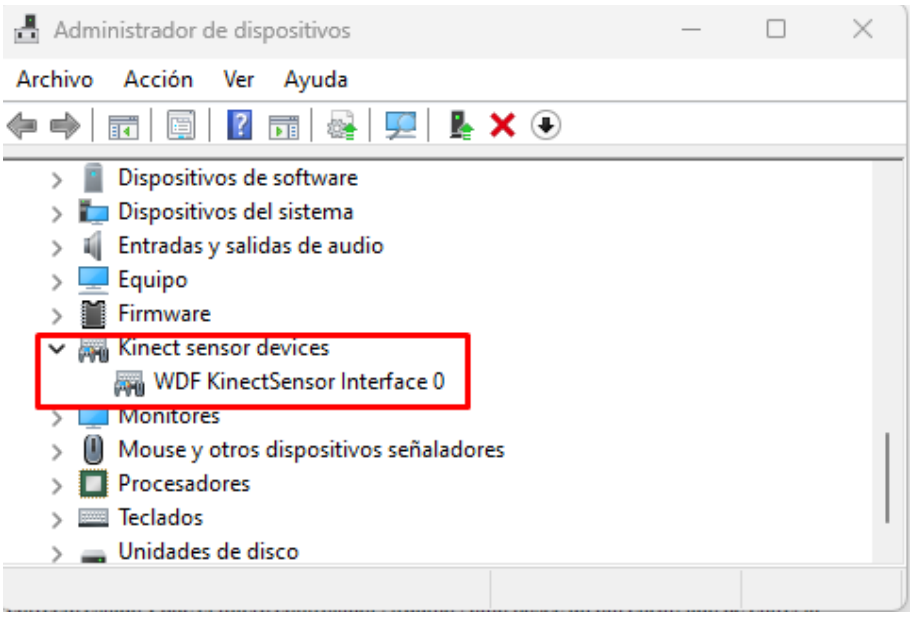
Resultado Obtenido

La verificación de conexión entre el dispositivo Kinect y el ordenador se ejecuta por medio del administrador de dispositivos, WDF KinectSensor Interface 0 es el controlador en específico que permite que el sistema operativo del ordenador se comunique con el hardware

del dispositivo Kinect, accediendo al uso de sus sensores. En la Figura 65 se verifica el controlador Kinect en el ordenador.

Figura 65

Controlador Kinect en el ordenador

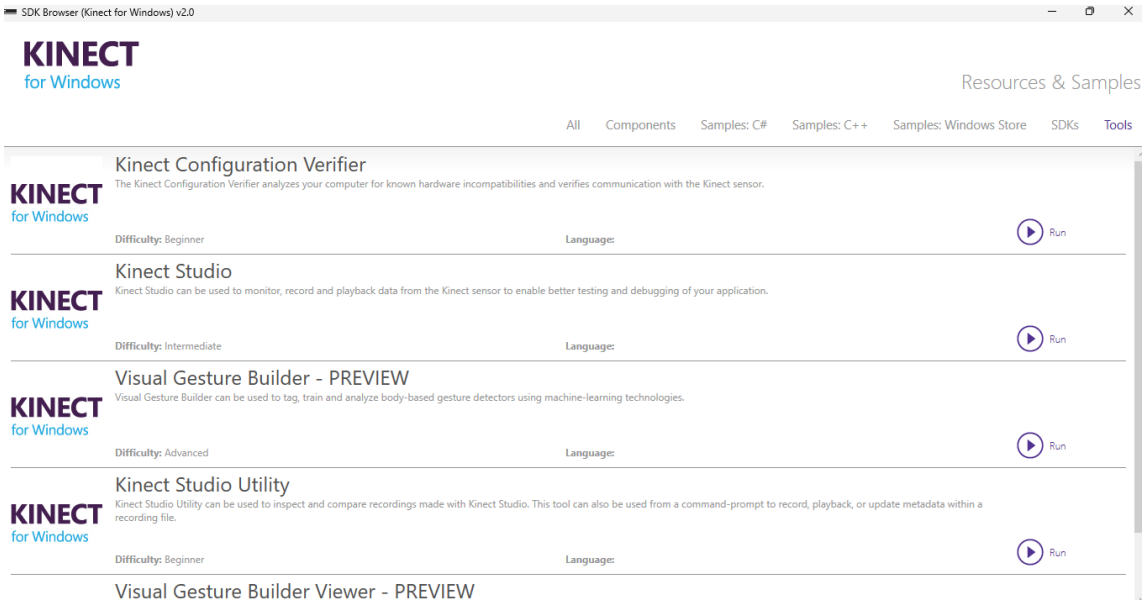


Fuente: Elaborado por el autor

La instalación del SDK 2.0 lanzado por Microsoft se realizó siguiendo una serie de pasos que el asistente de instalación detalla. Una vez completada la instalación, el kit incluye documentación sobre el dispositivo Kinect y diversos ejemplos ejecutables en el entorno de programación de Visual Studio. Estos ejemplos abarcan las distintas funcionalidades del dispositivo Kinect los cuales permiten verificar la funcionalidad de los sensores. En la Figura 66 se muestra el contenido del kit de desarrollo Kinect.

Figura 66

Kit de Desarrollo Kinect

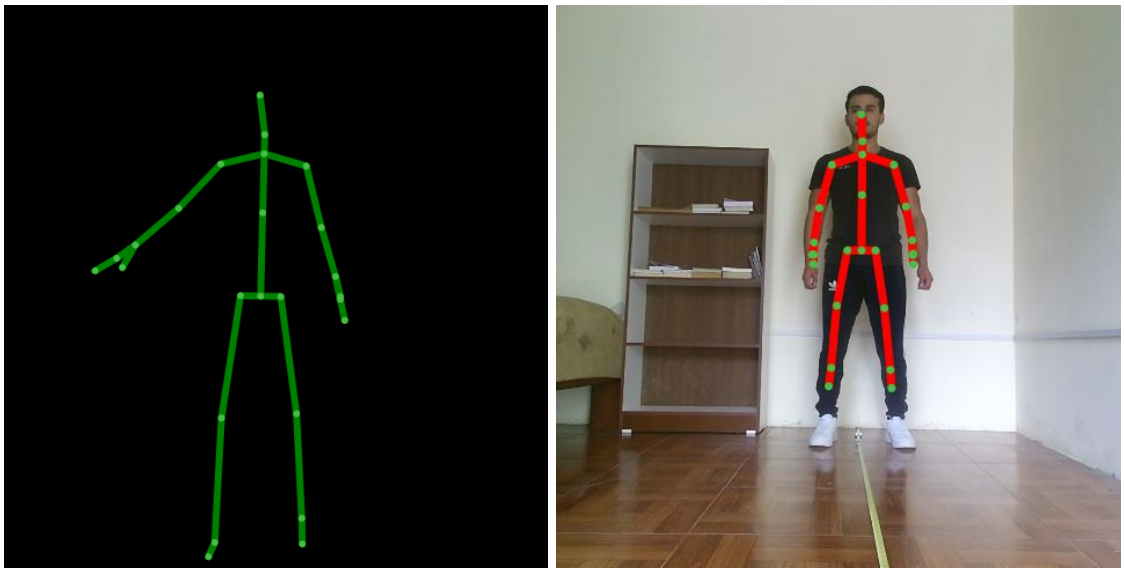


Fuente: Elaborado por el autor

Se ejecutó los ejemplos relacionados con los sensores de la cámara RGB, sensor infrarrojo, y el sensor de profundidad, verificando la correcta funcionalidad. Como se observa en la Figura 67, en la parte izquierda se tiene la detección del esqueleto humano mediante el uso de los sensores de profundidad e infrarrojo, y en la parte izquierda se visualiza la unión del funcionamiento de la cámara RGB con los sensores de profundidad e infrarrojo.

Figura 67

Verificación de funcionalidad de hardware-sensores



Fuente: Elaborado por el autor

4.1.3 Test de Software	**Sistema Prueba**	**Software**
	¿Prueba de despliegue?	**Si/No**
Descripción: Pruebas funcionales del software y visualización de datos para la implementación del sistema.		
Prerrequisitos: **1. Análisis de requerimientos del sistema.** **2. Estudio de entorno de programación Visual Studio y lenguaje de programación C#.**		

3. Estudio de creación de aplicaciones WPF.

4. Estudio de cálculo vectorial (producto escalar) en tres dimensiones.

Pasos:

- ✓ Visualización de adquisición de datos del dispositivo Kinect y sus sensores.
- ✓ Visualización de registro de usuarios.
- ✓ Visualización de procesamiento de información.
- ✓ Visualización de tratamiento de información.
- ✓ Visualización de generación de alertas.
- ✓ Visualización generación de reporte pdf.

Resultado esperado:

El sistema cumplirá con los pasos de programación mencionados, se inicia con la verificación de registro de usuarios por medio de la base de datos MySQL, mediante el login del usuario. El sistema inicia con la adquisición de los datos de las articulaciones del usuario, posterior se verificará los valores de los ángulos calculados, los cuales son comparados con ángulos preestablecidos, validando las repeticiones como correctas e incorrectas, de la misma manera el sistema contendrá una interfaz gráfica para el monitoreo de entrenamiento, con indicadores de las posiciones de ejecución de los ejercicios, paralelo a este proceso el sistema contará con una interfaz gráfica en la nube dirigida al instructor para el monitoreo en tiempo real de la ejecución de ejercicios, de la misma manera el sistema generará alertas dirigidas al instructor cuando la cantidad de repeticiones ejecutadas erróneamente por el usuario supere el 50% del total de repeticiones. Finalmente, el sistema será capaz de generar reportes en formato PDF para el seguimiento de los valores de los ángulos en cada postura de entrenamiento detectada, de igual manera el usuario

tendrá acceso a tablas de información con los detalles de las repeticiones realizadas incorrectamente.

Resultado obtenido:

Los resultados obtenidos se demuestran por medio de la interfaz de inicio del sistema ZENERBOT la cual se diseñó para facilitar el acceso a las funcionalidades de monitoreo y control de posturas. En el lado izquierdo de la pantalla principal se estableció tres botones de control del dispositivo Kinect, apagar el dispositivo, prender y salir del sistema.

Al lado derecho se tienen botones para la creación de un nuevo usuario, y posterior ingresar con la información guardada en la base de datos. En la Figura 68 ilustra la ventana inicial del sistema.

Figura 68

Prueba inicio del sistema

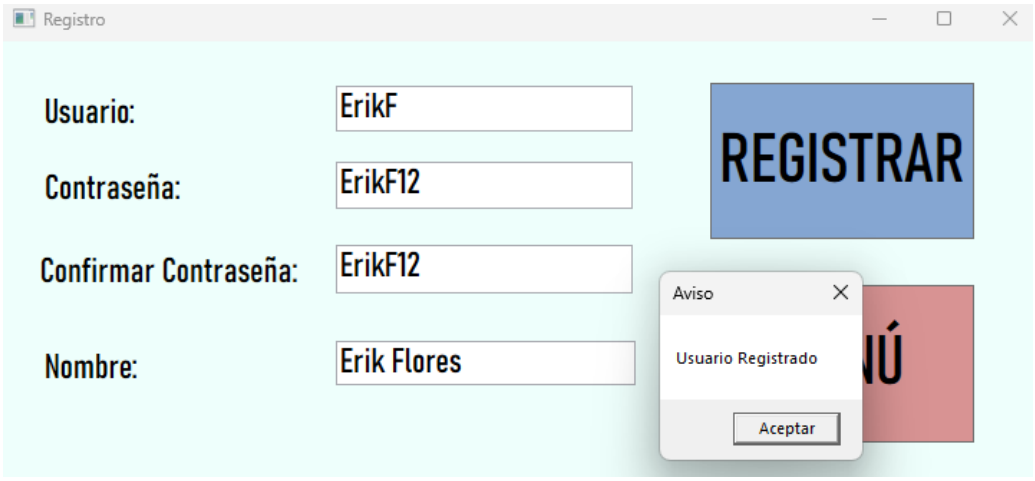
Fuente: Elaborado por el autor

4.1.3.1 Pruebas registro de usuario

Al presionar el botón de creación de nuevo usuario, se abre una ventana con 4 campos por rellanar; usuario, contraseña, confirmación de contraseña y el nombre. En la Figura 69 se ilustra el registro de un nuevo usuario.

Figura 69

Prueba registro de usuario

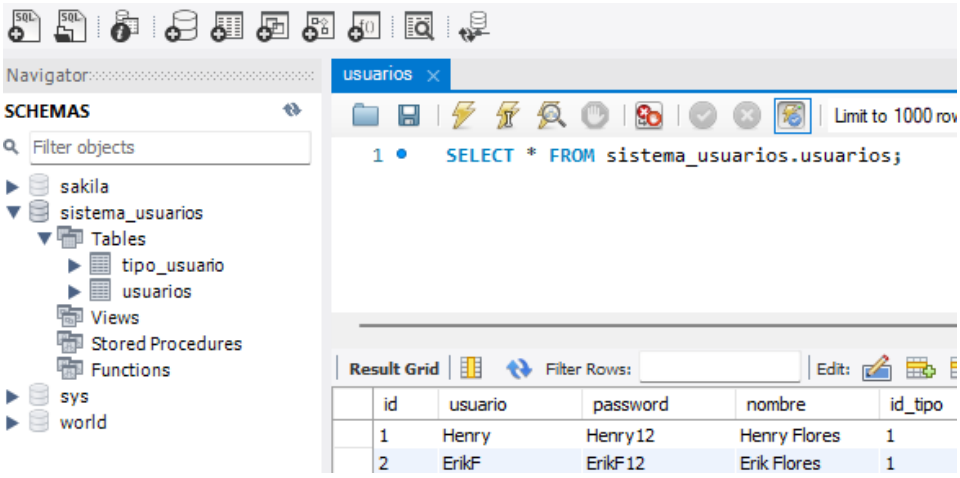


Fuente: Elaborado por el autor

Si los campos de relleno son completados correctamente, el usuario se registrará sin inconvenientes. Por el contrario, si el usuario ya existe o no completa todos los campos, se mostrará un cuadro de aviso con el detalle respectivo. En la Figura 70 se muestra el usuario registrado en la base de datos MySQL.

Figura 70

Prueba registro de usuario en la base de datos MySQL.



Fuente: Elaborado por el autor

Los usuarios registrados en la base de datos acceden a las funcionalidades del sistema por medio del ingreso de los campos (Usuario y Contraseña).

4.1.3.2 Pruebas procesamiento de información.

Las pruebas de procesamiento de información consisten en verificar la obtención de frames de las cámaras RGB y el sensor de profundidad. Con la captura de los frames se tiene acceso a los valores de las articulaciones y posterior se demuestra los ángulos calculados y diversos contadores que validan las repeticiones de los dos ejercicios ejecutados.

Los valores de ángulos calculados son fundamentales para validar las repeticiones en cada ejercicio, así como su correcta ejecución en las posturas de entrenamiento. Al verificar los ángulos, se pueden determinar el rango total del ejercicio, los rangos correctos en las posiciones iniciales y finales, y los rangos incorrectos tanto en el límite inferior como en el superior del rango correcto. Esto permite generar un control detallado de las fallas en cada repetición del ejercicio, proporcionando al usuario un mensaje claro y comprensible.

En la tabla 13 se ilustran los valores de los rangos de ángulos en la posición inicial y final del ejercicio de levantamientos laterales.

Tabla 13

Rango de ángulos Levantamientos Laterales

Posición	Rango total	Rango correcto	Límite inferior incorrecto	Límite superior incorrecto
Inicial	20° - 40°	25° - 35°	20° - 24°	36° - 40°
Final	1° - 20°	5° - 11°	1° - 4°	12° - 20°

Nota. Rango de ángulos establecidos para validaciones del ejercicio de levantamientos laterales. Fuente: Elaborado por el autor

Los ángulos detallados en la variante de levantamientos laterales son utilizados en el sistema para garantizar la detección de cada repetición del ejercicio realizado. En la posición inicial de la repetición del ejercicio, los valores incorrectos de los ángulos, tanto en los límites inferiores como en los superiores indican la separación de los brazos respecto al torso del cuerpo. En la posición final de la repetición del ejercicio, los valores incorrectos de los ángulos, tanto en los límites inferiores y superiores, indican la altura de los brazos en relación con la de los hombros.

En la tabla 14 se ilustran los valores de los rangos de ángulos en la posición inicial y final del ejercicio de Press militar.

Tabla 14

Rango de ángulos Press Militar

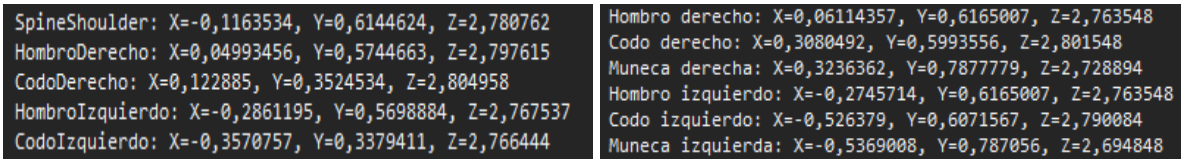
Posición	Rango total	Rango correcto	Límite inferior incorrecto	Límite superior incorrecto
Inicial	25° - 50°	33° - 43°	25° - 32°	44° - 50°
Final	2° - 22°	7° - 14°	2° - 6°	15° - 22°

Nota. Rango de ángulos establecidos para validaciones del ejercicio de press militar. Fuente: Elaborado por el autor

En la posición inicial del ejercicio, los valores de los ángulos límite inferior y superior determinan la proximidad de las manos a los hombros y el grado de extensión de los brazos. Definidos los ángulos con los cuales trabaja el sistema, en la Figura 71 se ilustra los valores capturados de las articulaciones en el ejercicio de levantaciones laterales y press militar.

Figura 71

Pruebas de valores capturados de las articulaciones en las dos variables de ejercicios.



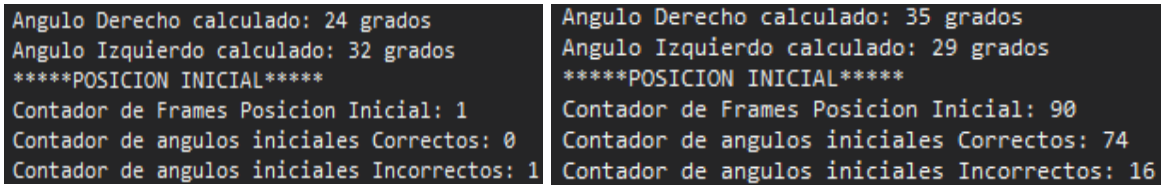
Fuente: Elaborado por el autor

Con los valores de las articulaciones, se calculan los vectores para el brazo derecho e izquierdo y, posteriormente, se inicia el cálculo de los ángulos. Si los ángulos calculados para ambos brazos están dentro del rango que valida las posiciones inicial o final, se activa un contador de frames que va del número 1 al 90. Durante estos 90 frames, se registra y muestra el conteo de los ángulos en la posición inicial, clasificándolos como correctos o incorrectos.

La Figura 72 muestra los valores de los ángulos calculados, junto con el inicio del contador de frames y los contadores de ángulos correctos e incorrectos, correspondientes a la posición inicial de una repetición de ejercicio de levantamientos laterales.

Figura 72

Pruebas valores de ángulos calculados y valor de contador



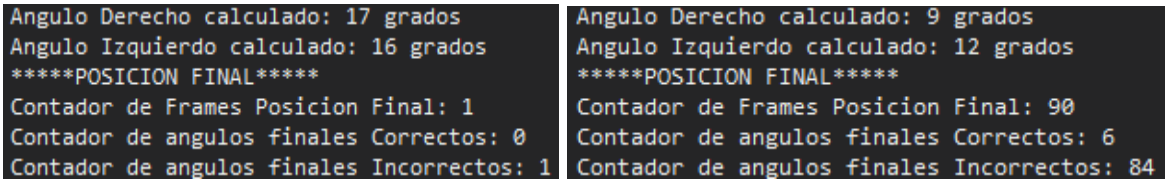
Fuente: Elaborado por el autor

Una vez finalizado el contador de frames en la posición inicial del ejercicio, se activa el contador de frames en la posición final.

La Figura 73 muestra los valores de los ángulos calculados, junto con el inicio del contador de frames y los contadores de ángulos correctos e incorrectos, correspondientes a la posición final de una repetición de ejercicio.

Figura 73

Pruebas de valores de ángulos calculados y contadores.



Fuente: Elaborado por el autor

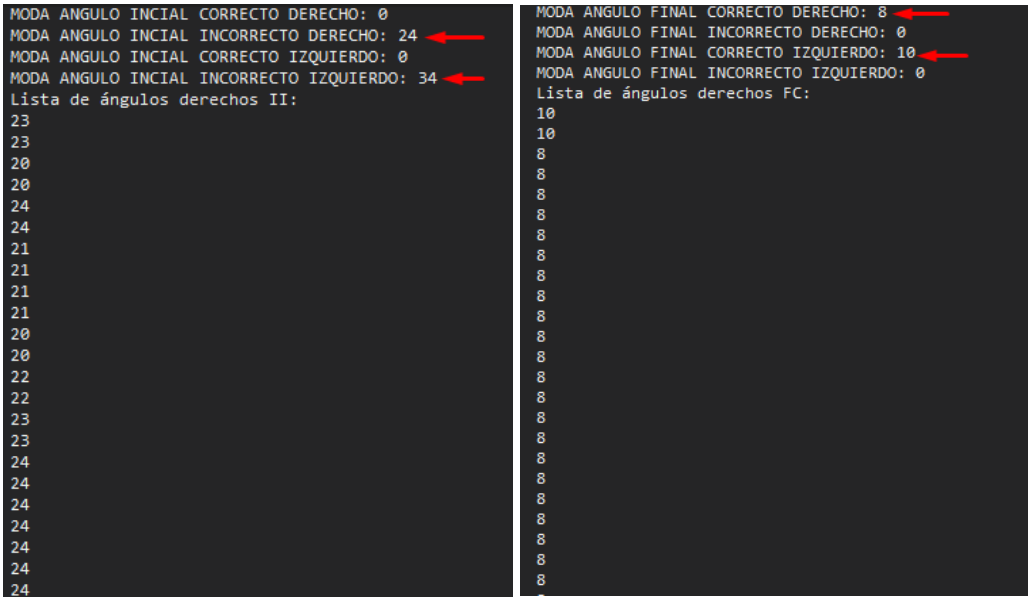
Al finalizar los contadores de frames de las posiciones inicial y final de una repetición del ejercicio, se verifican los valores capturados de los contadores y se determina si la repetición es correcta o incorrecta.

Paralelo a este proceso, en las listas de almacenamiento de ángulos se clasifican y guardan los valores de los ángulos de la posición inicial y final en los estados correcto o incorrecto. Al completar los 90 frames, se calcula la moda de los valores de las listas y ese valor se envía tanto al reporte pdf como a las tablas de control de los detalles de las posturas incorrectas.

En la Figura 74 se ilustra los valores almacenados en las listas y el valor de la moda calculada para la posición inicial y posición final de una repetición de ejercicio en levantamientos laterales.

Figura 74

Valores de listas y resultados de moda calculado



Fuente: Elaborado por el autor

4.1.3.3 Pruebas tratamiento de información.

Las pruebas realizadas para verificar el tratamiento eficaz de la información, inicia con la activación de la cámara RGB, para la captura de video en tiempo real. Seguidamente,

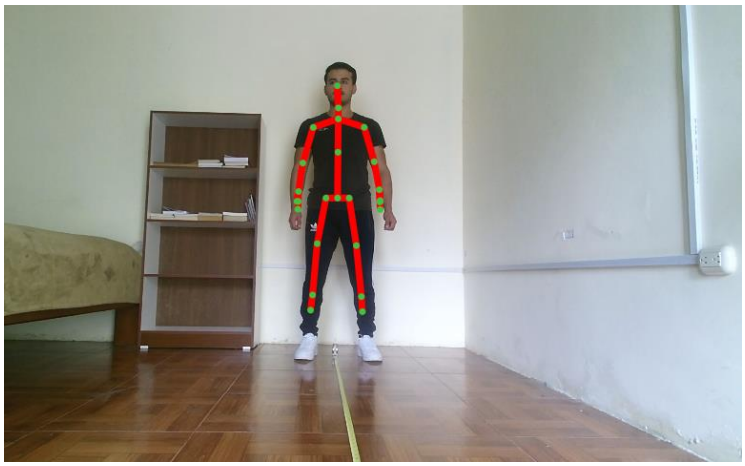
la activación del esqueleto prueba la detección y mapeo de la estructura ósea del usuario, lo que es esencial para el seguimiento preciso del ejercicio.

Las pruebas realizadas demostraron que el dispositivo Kinect mejora considerablemente la detección de las articulaciones cuando la persona se encuentra en un lugar bien iluminado.

En la Figura 75 se ilustra las pruebas de la imagen captada por el sensor Kinect y la detección del esqueleto humano.

Figura 75

Pruebas de activación de video y seguimiento de esqueleto



Fuente: Elaborado por el autor

El seguimiento de ejercicio evalúa la capacidad del sistema para monitorear los movimientos del usuario en las posturas inicial y final de cada ejercicio, esto garantiza que cada repetición se registre con exactitud. Se incluyen indicadores de movimientos de los brazos para llevar una ejecución de ejercicios controlada y detectada por el sistema. En la Figura 76 se ilustra los indicadores de movimientos para cada posición del ejercicio de levantamientos laterales

Figura 76

Pruebas de indicadores de movimientos Levantamientos Laterales

Fuente: Elaborado por el autor

En la Figura 77 se ilustra los indicadores de movimientos para cada posición del ejercicio de Press militar.

Figura 77

Pruebas de indicadores de movimientos Press Militar

Fuente: Elaborado por el autor

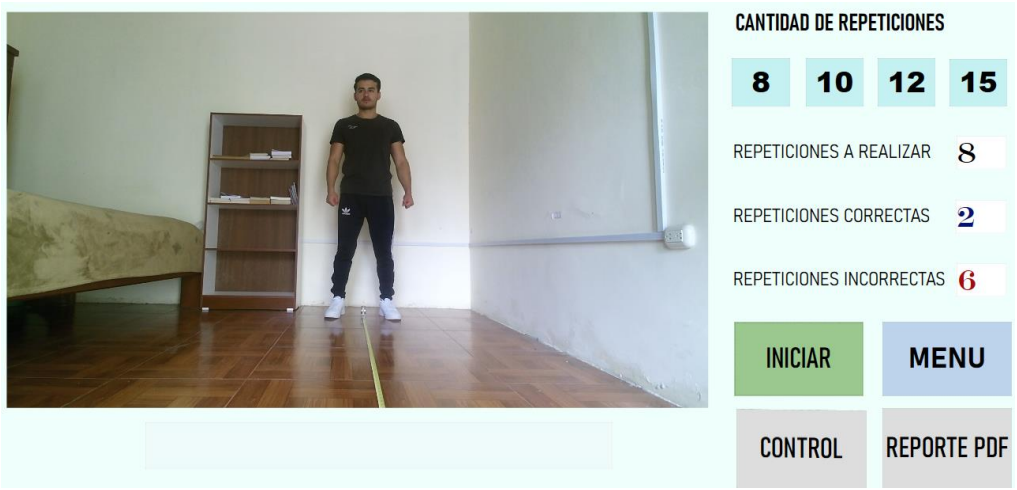
Los contadores de repeticiones proporcionan retroalimentación en tiempo real sobre el estado de cada ejercicio realizado, diferenciando entre repeticiones correctas e incorrectas. Adicionalmente, un contador total registra la cantidad de repeticiones seleccionadas por el usuario, conforme a la rutina de entrenamiento establecida.

Al finalizar la serie de repeticiones, el contador de repeticiones totales se restablece al valor inicial, y los totales de repeticiones correctas e incorrectas se mantienen registrados hasta que el usuario presione nuevamente el botón INICIAR.

En la Figura 78 se ilustra los valores de los contadores en una serie de repetición ejecutada.

Figura 78

Pruebas de monitoreo de repeticiones correctas e incorrectas



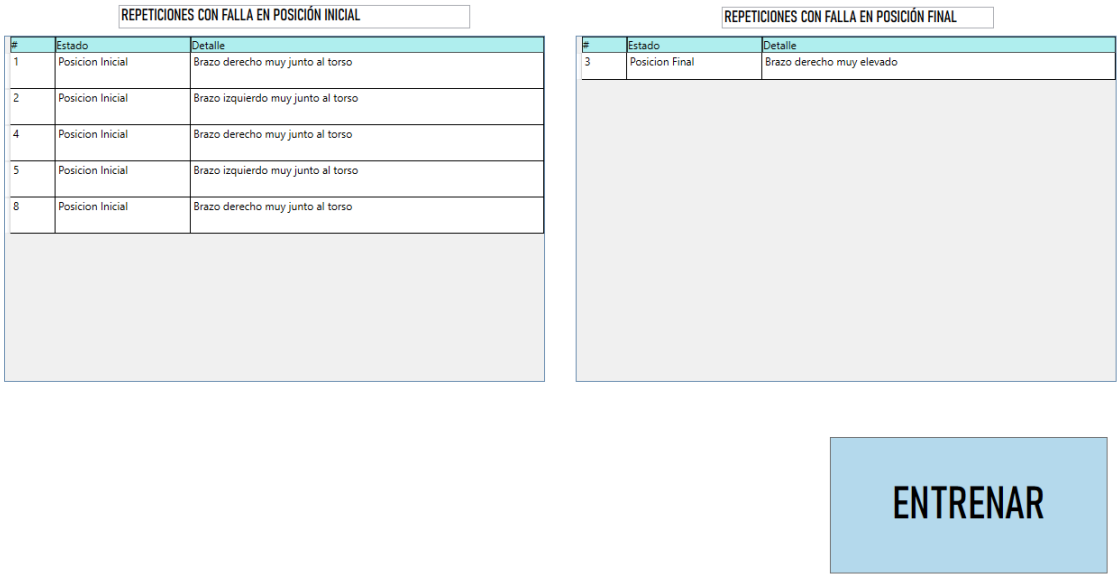
Fuente: Elaborado por el autor

El control de repeticiones incorrectas identifica los errores en la ejecución de los movimientos de los brazos en las posiciones inicial y final del ejercicio. Esto garantiza que el usuario pueda verificar el tipo de falla cometida y corregirla en sus nuevas ejecuciones de ejercicios.

En la Figura 79 se ilustra las pruebas de control de los detalles capturados en cada posición del ejercicio ejecutado.

Figura 79

Pruebas de integración control de repeticiones incorrectas



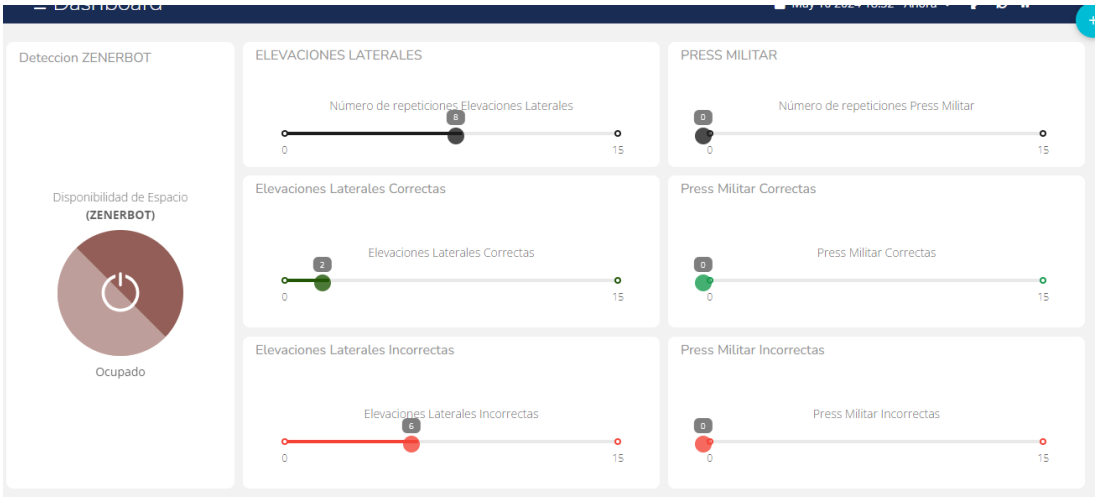
Fuente: Elaborado por el autor

Paralelo al proceso de ejecución de ejercicios, en la plataforma Ubidots se verifica la visualización de datos de las repeticiones ejecutadas por el usuario en tiempo real. Esto habilita al instructor para monitorear continuamente el rendimiento del usuario en cualquier lugar que se tenga acceso a internet. Además, ofrece la posibilidad de verificar la disponibilidad del espacio de entrenamiento, facilitando la dirección de los usuarios cuando este se encuentre libre.

En la Figura 80 se ilustra la interfaz gráfica dirigida al instructor para el monitoreo de ejercicios en tiempo real.

Figura 80

Pruebas de integración monitoreo de ejercicios en la nube



Fuente: Elaborado por el autor

4.1.3.4 Pruebas generación de alertas

Las alertas generadas son verificadas en la pantalla del dispositivo del instructor a través de la aplicación de Telegram.

Se verificó que las alertas en tiempo real sean visibles y comprensibles para el instructor, facilitando un seguimiento más controlado cuando el umbral de las repeticiones realizadas incorrectamente por el usuario supera el 50% del total de repeticiones realizadas. La notificación de alerta conforma el nombre del usuario que ha superado el umbral límite de repeticiones incorrectas, así como la cantidad de estas y el total de repeticiones realizadas. En la Figura 81 se muestra las pruebas de notificación de alertas dirigida hacia el instructor.

Figura 81

Pruebas de notificación de alertas

Fuente: Elaborado por el autor

4.1.3.5 Pruebas generación de reporte.

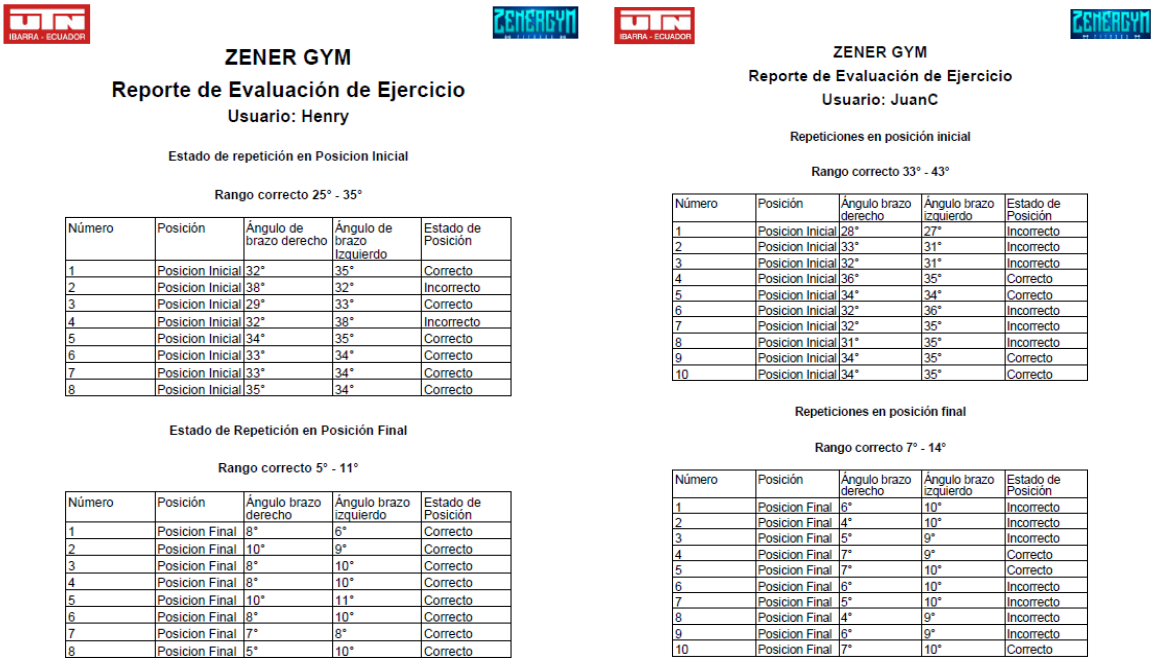
La generación de reportes en el formato PDF, incluyen el número total de repeticiones del ejercicio ejecutado, la posición inicial o final de cada repetición, el valor de la moda de la lista de ángulos en cada posición, y el estado de la posición como correcta o incorrecta.

Los reportes incluyen dos tablas informativas: la primera tabla contiene información sobre la posición inicial de cada repetición del ejercicio, mientras que la segunda tabla detalla la posición final. Esto proporciona una visión clara y precisa del rendimiento del usuario a lo largo de la sesión de entrenamiento, facilitando el análisis y seguimiento de la progresión del entrenamiento, basado en los ángulos de cada brazo.

En la Figura 82 se ilustra las pruebas de generación de reportes PDF, en los ejercicios de levantamientos laterales y Press militar.

Figura 82

Pruebas de generación de reportes PDF



ZENER GYM

Reporte de Evaluación de Ejercicio

Usuario: Henry

Estado de repetición en Posicion Inicial

Rango correcto 25° - 35°

Número	Posición	Ángulo de brazo derecho	Ángulo de brazo Izquierdo	Estado de Posición
1	Posicion Inicial	32°	35°	Correcto
2	Posicion Inicial	38°	32°	Incorrecto
3	Posicion Inicial	29°	33°	Correcto
4	Posicion Inicial	32°	38°	Incorrecto
5	Posicion Inicial	34°	35°	Correcto
6	Posicion Inicial	33°	34°	Correcto
7	Posicion Inicial	33°	34°	Correcto
8	Posicion Inicial	35°	34°	Correcto

Estado de Repetición en Posición Final

Rango correcto 5° - 11°

Número	Posición	Ángulo brazo derecho	Ángulo brazo izquierdo	Estado de Posición
1	Posicion Final	8°	6°	Correcto
2	Posicion Final	10°	9°	Correcto
3	Posicion Final	8°	10°	Correcto
4	Posicion Final	8°	10°	Correcto
5	Posicion Final	10°	11°	Correcto
6	Posicion Final	8°	10°	Correcto
7	Posicion Final	7°	8°	Correcto
8	Posicion Final	5°	10°	Correcto

ZENER GYM

Reporte de Evaluación de Ejercicio

Usuario: JuanC

Repeticiones en posición inicial

Rango correcto 33° - 43°

Número	Posición	Ángulo brazo derecho	Ángulo brazo izquierdo	Estado de Posición
1	Posicion Inicial	28°	27°	Incorrecto
2	Posicion Inicial	33°	31°	Incorrecto
3	Posicion Inicial	32°	31°	Incorrecto
4	Posicion Inicial	36°	35°	Correcto
5	Posicion Inicial	34°	34°	Correcto
6	Posicion Inicial	32°	36°	Incorrecto
7	Posicion Inicial	32°	35°	Incorrecto
8	Posicion Inicial	31°	35°	Incorrecto
9	Posicion Inicial	34°	35°	Correcto
10	Posicion Inicial	34°	35°	Correcto

Repeticiones en posición final

Rango correcto 7° - 14°

Número	Posición	Ángulo brazo derecho	Ángulo brazo izquierdo	Estado de Posición
1	Posicion Final	6°	10°	Incorrecto
2	Posicion Final	4°	10°	Incorrecto
3	Posicion Final	5°	9°	Incorrecto
4	Posicion Final	7°	9°	Correcto
5	Posicion Final	7°	10°	Correcto
6	Posicion Final	6°	10°	Incorrecto
7	Posicion Final	5°	10°	Incorrecto
8	Posicion Final	4°	9°	Incorrecto
9	Posicion Final	6°	9°	Incorrecto
10	Posicion Final	7°	10°	Correcto

Fuente: Elaborado por el autor

Los reportes PDF generados se guardan en el escritorio del ordenador, con el nombre de archivo que incluye: tipo de ejercicio realizado, nombre del usuario y un valor numérico que permite generar varios reportes por cada serie ejecutada.

4.1.4 Test de aplicación	**Sistema Prueba**	**Aplicación**
	¿Prueba de despliegue?	**Si/No**
Descripción: Pruebas de diseño del sistema basado en la aplicación del instructor del Gimnasio ZENER GYM.		
Prerrequisitos:		

1. **Pruebas eléctricas completas.** 2. **Pruebas de diseño de hardware completas.** 3. **Pruebas de diseño de software completas.** 4. **Ubicación del sistema dentro de un Gimnasio.**
Pasos: ✓ Visualización de usuarios en la base de datos. ✓ Visualización de espacio de disponibilidad para entrenamientos. ✓ Monitoreo de entrenamiento en tiempo real de plataforma en la nube Ubidots. ✓ Visualización de alertas generadas por usuarios. ✓ Visualización de reportes generados por usuarios.
Resultado esperado: El instructor tendrá acceso a la base de datos de registro de usuarios, de igual manera podrá monitorear los entrenamientos por medio de la plataforma en la nube Ubidots, verificando la disponibilidad del sistema y el seguimiento de cada repetición realizada por el usuario. Al finalizar la serie de repeticiones el instructor tendrá acceso al reporte pdf y recibirá una notificación de alerta en la aplicación de Telegram cuando el usuario active el umbral de repeticiones incorrectas.

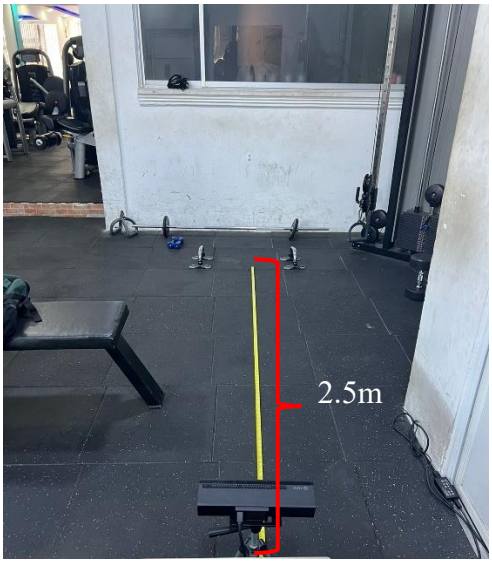
Resultado obtenido:

Se verificó el funcionamiento de la aplicación tomando en cuenta a 10 usuarios entre hombres y mujeres del gimnasio ZENERGYM, con los cuales la señorita instructora verificó la visualización de las diferentes funciones del sistema.

Inicialmente el dispositivo Kinect está ubicado sobre un trípode a una altura de 40 cm sobre el nivel del suelo, asegurando así la detección completa del cuerpo. La distancia establecida entre el Kinect y la persona es de 2.5 metros, lo cual garantiza la correcta obtención de datos.

Figura 83

Distancia entre dispositivo Kinect y usuarios



Fuente: Gimnasio ZENERGYM

La instructora verificó los 10 usuarios registrados en la base de datos MySQL, con la información de usuario, contraseña y nombre. En la Figura 84 se ilustra la base de datos con la información de los 10 usuarios registrados.

Figura 84

Base de datos de usuarios

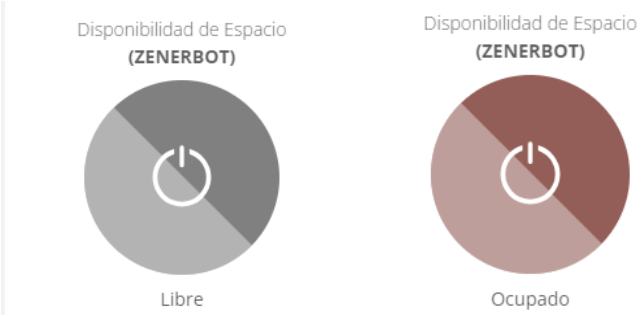
	id	usuario	password	nombre	id_tipo
	1	ErikF	ErikF12	Erik Flores	1
▶	2	Cesar	Cesar1998	Cesar Bautista	1
	3	Katta	Katta12	Katta	1
	4	Stefany	stefany2801	Stefany	1
	5	Marcelo Noboa	Marcelo Noboa2	Marcelo Noboa	1
	6	Sebas	sjce2007	sebas	1
	7	JuanC	JuanC12	Juan Carlos	1
	8	Alejandra	Ale123	Ale	1
	9	Gise	GiseC	Gisen Cardenas	1
	10	Roci	RociR	Rosi Ruiz	1
	11	Andres	AndresM	Andres Mediavilla	1

Fuente: [Base de datos-Elaborado por el autor]

Se realizó varias pruebas con los usuarios del gimnasio ZENER GYM, las cuales permitieron que la señorita instructora verifique el funcionamiento del monitoreo en tiempo real de la disponibilidad de uso del sistema, en la cual, si tiene dos estados, disponibilidad de espacio “libre” y disponibilidad de espacio “ocupado”. En la Figura 85 se ilustra los dos estados de disponibilidad del sistema, el lado izquierdo de la figura representa la disponibilidad del espacio libre y el lado derecho de la figura representa la disponibilidad del espacio ocupado.

Figura 85

Estados de disponibilidad de espacio

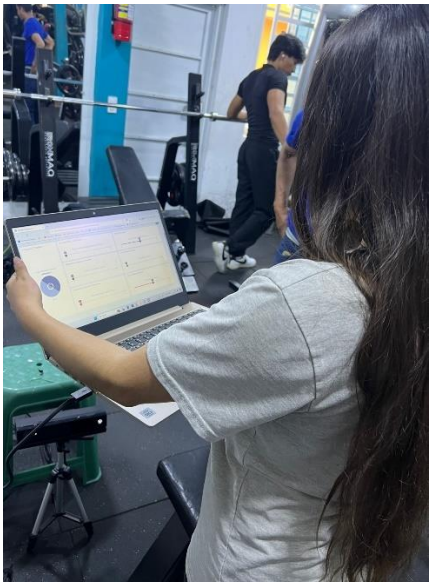


Fuente: [Ubidots-Elaborado por el autor]

Durante la ejecución de los ejercicios, la instructora verificó el monitoreo en tiempo real por cada repetición ejecutada en las dos variantes de ejercicios de hombros. En la Figura 86 se ilustra la verificación de monitoreo en tiempo real de ejecución de las repeticiones dirigida a la instructora.

Figura 86

Pruebas de monitoreo de entrenamiento mediante Ubidots



Fuente: [Monitoreo instructora-Elaborado por el autor]

Las pruebas realizadas a los diferentes usuarios confirmaron que la notificación de alerta dirigida al instructor se envió correctamente cada vez que el usuario finalizaba la serie de repeticiones y activaba el umbral de la cantidad de repeticiones incorrectas con relación al total de repeticiones realizadas. La Figura 87 ilustra la verificación y lectura de notificación de alertas dirigidas a la instructora.

Figura 87

Pruebas de notificaciones de alertas

Fuente: [Telegram-Elaborado por el autor]

Por medio de las pruebas realizadas, se generaron diferentes reportes PDF a los que tienen acceso tanto los usuarios como la instructora. Estos reportes muestran los valores de los ángulos en las posturas de entrenamiento tanto para el brazo izquierdo como para el derecho. El reporte contiene dos tablas correspondientes a las posiciones inicial y final de cada repetición. Los valores de los ángulos mostrados permiten determinar qué tan cerca o lejos se encuentran de los valores preestablecidos, que determinan si la postura es correcta o incorrecta.

En la Figura 88 se ilustra la verificación del reporte PDF generado por un usuario del gimnasio.

Figura 88

Verificación de reportes de entrenamiento

ZENER GYM

Reporte de Evaluación de Ejercicio

Usuario: Marcelo Noboa

Estado de repetición en Posicion Inicial

Rango correcto 25° - 35°

Número	Posición	Ángulo de brazo derecho	Ángulo de brazo Izquierdo	Estado de Posición
1	Posicion Inicial	35°	39°	Incorrecto
2	Posicion Inicial	34°	36°	Incorrecto
3	Posicion Inicial	29°	33°	Correcto
4	Posicion Inicial	33°	37°	Incorrecto
5	Posicion Inicial	32°	31°	Correcto
6	Posicion Inicial	29°	30°	Correcto
7	Posicion Inicial	34°	33°	Correcto
8	Posicion Inicial	30°	35°	Correcto

Estado de Repetición en Posición Final

Rango correcto 5° - 11°

Número	Posición	Ángulo brazo derecho	Ángulo brazo izquierdo	Estado de Posición
1	Posicion Final	13°	12°	Incorrecto
2	Posicion Final	8°	10°	Correcto
3	Posicion Final	2°	12°	Incorrecto
4	Posicion Final	3°	1°	Incorrecto
5	Posicion Final	2°	4°	Incorrecto
6	Posicion Final	3°	1°	Incorrecto
7	Posicion Final	4°	2°	Incorrecto
8	Posicion Final	15°	1°	Incorrecto

Fuente: [Reporte-Elaborado por el autor]

En la Figura se puede apreciar que se tiene dos tablas detalladas con el número de repetición, la posición inicial y final, los valores de los ángulos para cada brazo y el estado de posición “correcto o incorrecto”.

4.1.5 Test Funcional	**Sistema Prueba**	**Funcionalidad**
	¿Prueba de despliegue?	**Si/No**

Descripción:

Pruebas de funcionalidad del sistema con usuarios del gimnasio ZENERGYM.

Prerrequisitos:

1. **Pruebas globales del sistema completas.**
2. **Sistema probado por instructora.**
3. **Sistema implementado en un lugar específico para el usuario.**

Pasos:

- ✓ Pruebas de encendido del sistema
- ✓ Verificación de menú y guías de entrenamiento por parte del usuario.
- ✓ Verificación de monitoreo de entrenamiento por parte del usuario.
- ✓ Verificación de tablas de control de errores por parte del usuario.
- ✓ Verificación de reportes PDF por parte del usuario.

Resultado esperado:

El usuario iniciará el sistema sin ningún percance, verificando su acceso por medio del usuario y contraseña, el cual le permita abrir el menú de entrenamiento con las guías de entrenamiento textuales y gráficas, de igual manera se espera que el usuario verifique el monitoreo de entrenamiento, iniciando con la elección del ejercicio a realizar, la cantidad de repeticiones a ejecutar y el monitorio de repeticiones correctas e incorrectas

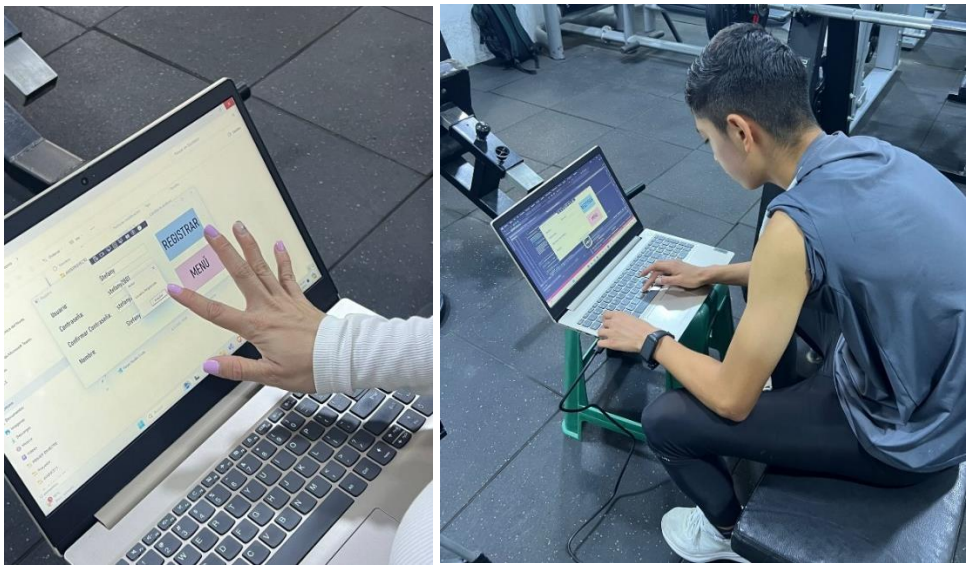
en tiempo real, finalmente el usuario verificará las tablas de control con los detalles de los errores realizados en las posturas de entrenamiento, paralelo a esto, el usuario puede verificar el valores de los ángulos en las posturas de entrenamiento y corregir su ejecución del ejercicio a más detalle.

Resultado obtenido:

Por medio de las pruebas realizadas a las 10 personas, incluidas las dos instructoras, se verificó que el sistema funcionó correctamente. Se comprobó que el sistema se encienda sin inconvenientes y dé acceso a la ventana inicial del sistema. En la Figura 89 se ilustra el registro correcto de un nuevo usuario.

Figura 89

Registro de usuarios ZENERGYM



Fuente: Elaborado por el autor

Con la información del usuario y contraseña, el usuario verificó que se despliegue la ventana de menú de entrenamiento. En la Figura 90 se ilustra dos usuarios iniciando el sistema e ingresando la información de usuario y contraseña.

Figura 90

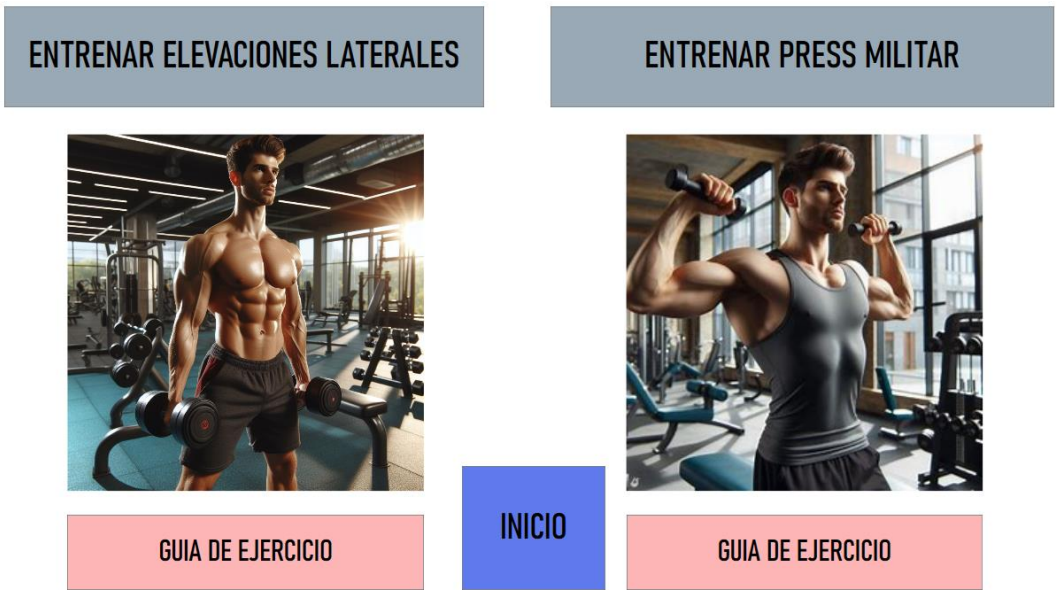
Inicio de sistema por usuarios

Fuente: Elaborado por el autor

Al iniciar el menú del sistema, el usuario tiene acceso a 4 opciones de ingreso, guías de entrenamiento y entrenamiento para cada variante de ejercicio. En la Figura 91 se ilustra el menú de opciones de entrenamiento del sistema.

Figura 91

Menú de sistema de entrenamiento

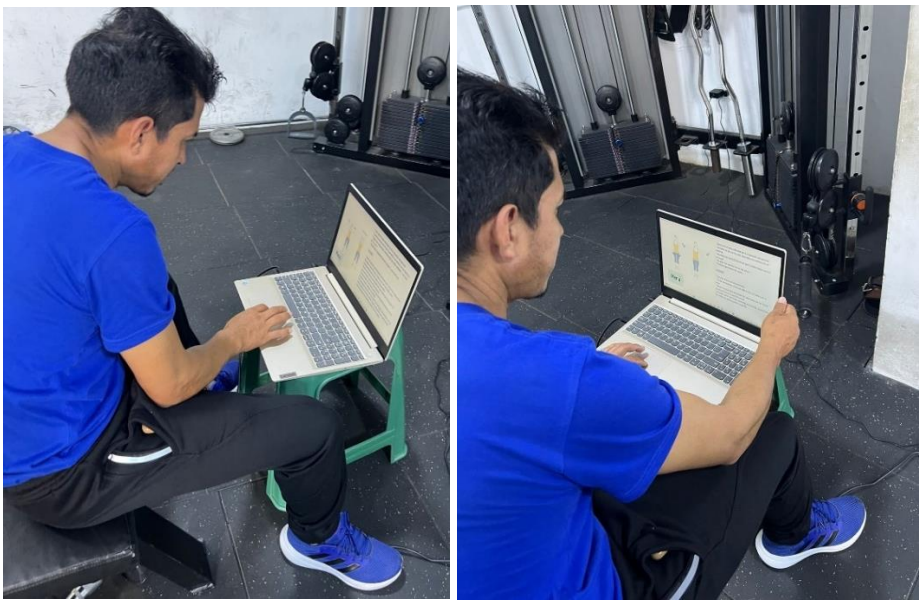


Fuente: [Sistema-Elaborado por el autor]

De las opciones a elegir, inicialmente el usuario verificó las guías de entrenamiento las cuales contienen la información de la ejecución de los ejercicios para la detección correcta por parte del sistema, el usuario verificó que sea una explicación clara y entendible. En la Figura 92 se ilustra al usuario verificando las guías de entrenamiento en las dos variantes de ejercicios de hombros.

Figura 92

Guías de entrenamiento con usuario



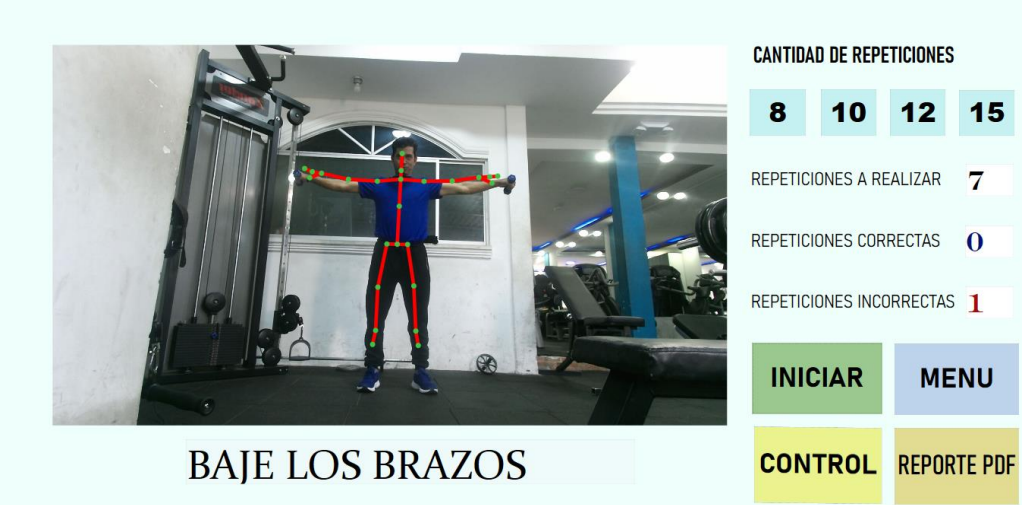
Fuente: Elaborado por el autor

Culminado el proceso de verificación de la guía de ejecución de las dos variantes de ejercicios, los usuarios iniciaron con el monitoreo en tiempo real de los ejercicios, verificando que se pueda elegir la cantidad de repetición entre 8,10,12 y 15.

En la Figura 93 se ilustra al usuario iniciando la serie de repeticiones y el seguimiento de las posturas de entrenamiento acompañado del monitoreo de las repeticiones correctas e incorrectas en la variante de Levantamientos Laterales.

Figura 93

Inicio de entrenamiento de Levantamientos Laterales con usuario



Fuente: [Sistema-Elaborado por el autor]

En la Figura 94 se ilustra al usuario iniciando la serie de repeticiones y el seguimiento de las posturas de entrenamiento acompañado del monitoreo de las repeticiones correctas e incorrectas en la variante de Press Militar.

Figura 94

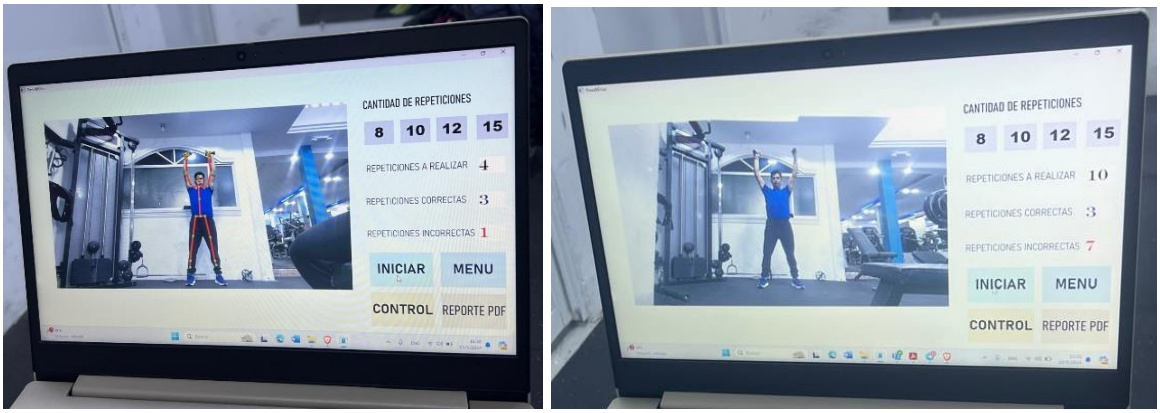
Inicio de entrenamiento de Press militar con usuario

Fuente: [Sistema-Elaborado por el autor]

Los usuarios verificaron el monitoreo de cada repetición ejecutada, aumentando los contadores en la detección de las repeticiones correctas e incorrectas, de igual manera al finalizar la cantidad de repeticiones seleccionadas, los usuarios verificaron que el seguimiento de esqueleto se detenga, permitiendo el acceso a verificar los errores ejecutados en las posturas de los ejercicios. En la Figura 95 se ilustra el monitoreo durante la ejecución de las repeticiones y la finalización de estas en la variante de Press militar.

Figura 95

Monitoreo de entrenamiento en tiempo real, variante Press militar

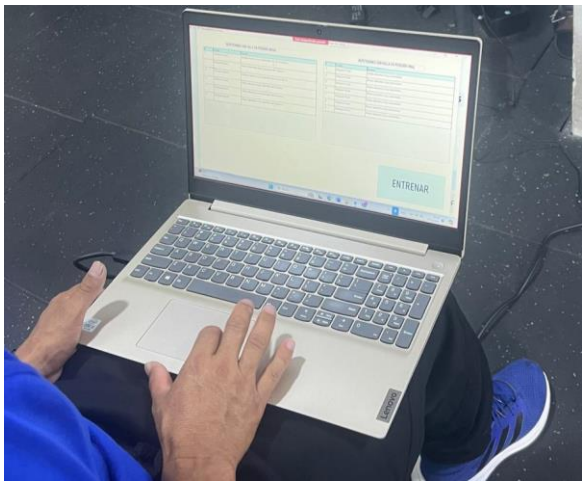


Fuente: [Sistema-Elaborado por el autor]

Durante las pruebas iniciales, los usuarios ejecutaron la mayoría de las repeticiones de ejercicios de manera incorrecta, por lo cual procedieron a verificar sus errores. En la Figura 96 se ilustra al usuario revisando los detalles de las repeticiones ejecutada de manera incorrecta.

Figura 96

Verificación de detalles de errores variante Press militar por el usuario

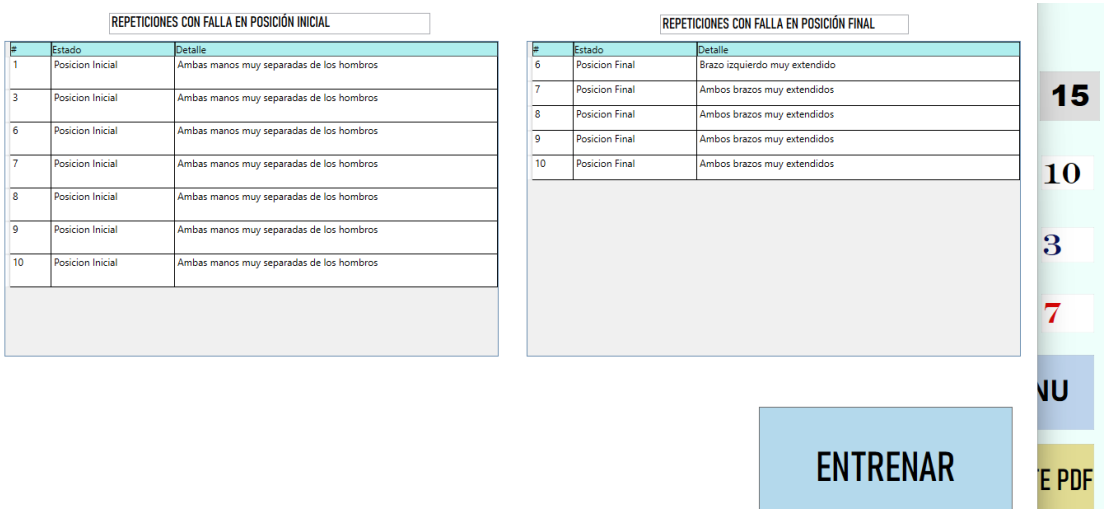


Fuente: Elaborado por el autor

En la Figura 97 se ilustra los detalles de los errores cometidos por parte del usuario en las posturas de posición inicial y posición final del ejercicio de Press militar, completando 7 repeticiones incorrectas y 3 repeticiones correctas.

Figura 97

Tablas de control de detalles variantes Press militar ilustradas al usuario

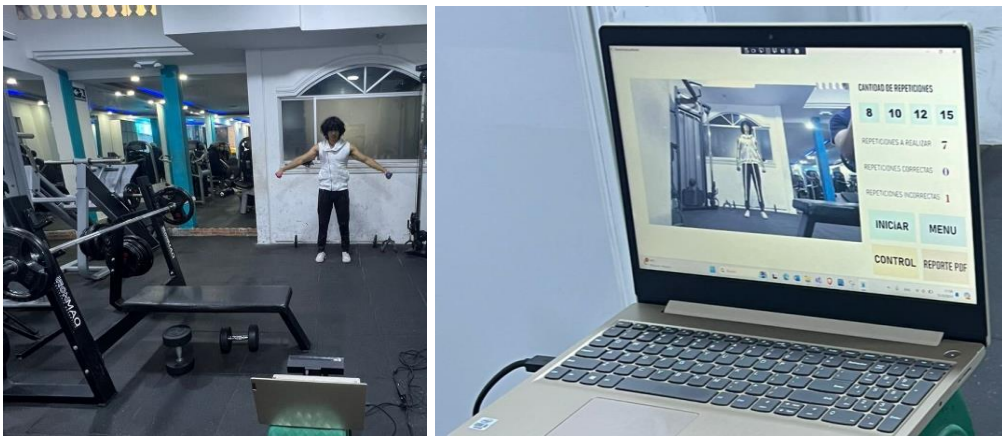


Fuente: [Sistema-Elaborado por el autor]

Para las pruebas de monitoreo del ejercicio de levantamientos laterales, se consideró a otro usuario, quien revisó las opciones para seleccionar la cantidad de repeticiones del ejercicio y dio inicio al monitoreo en esta variante de entrenamiento. La Figura 98 muestra al usuario realizando los ejercicios de levantamientos laterales y el monitoreo realizado por el sistema.

Figura 98

Monitoreo de entrenamiento en tiempo real variante levantamientos laterales

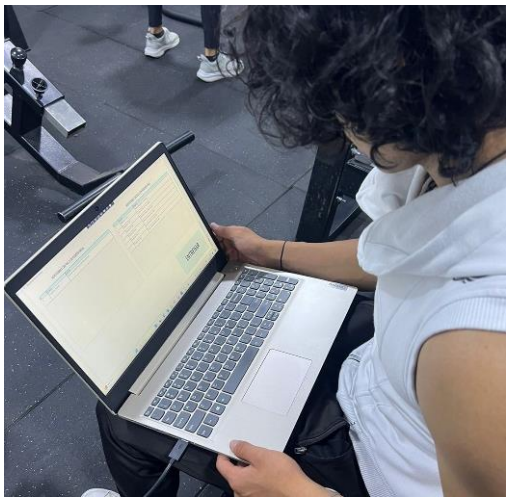


Fuente: Elaborado por el autor

Al finalizar la serie de repeticiones, el usuario verificó los detalles de los errores ejecutados durante la ejecución del ejercicio. En la Figura 99 se ilustra al usuario verificando los detalles de los errores en la ejecución de las repeticiones del ejercicio de levantamientos laterales.

Figura 99

Verificación de detalles de errores, variante levantamiento laterales.

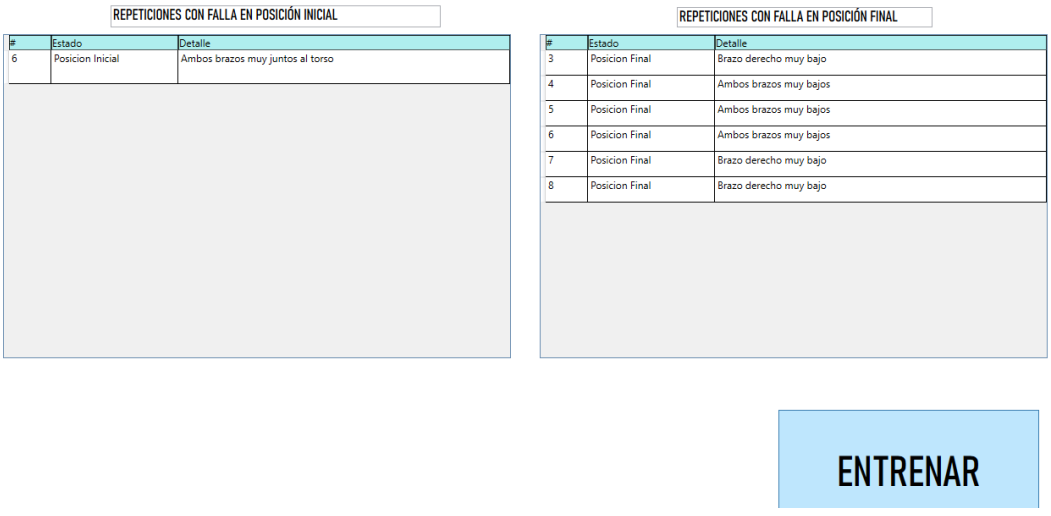


Fuente: Elaborado por el autor

En la Figura 100 se ilustra los detalles de los errores ejecutados por parte del usuario en las posturas de posición inicial y posición final del ejercicio de Levantamientos Laterales, completando seis repeticiones incorrectas y dos repeticiones correctas.

Figura 100

Tablas de control de detalles, variante levantamientos laterales ilustradas al usuario

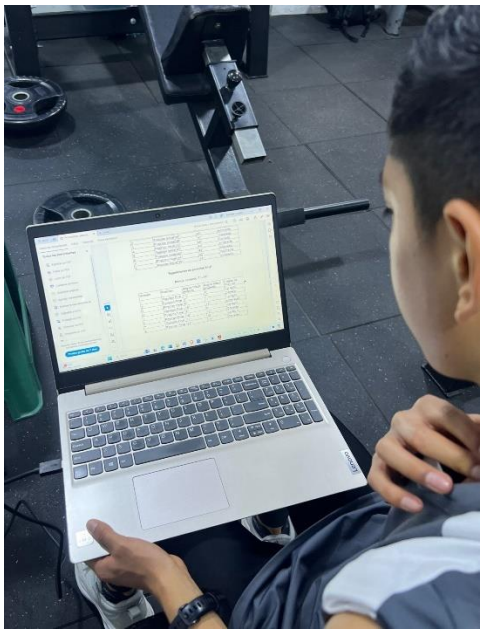


Fuente: [Sistema-Elaborado por el autor]

De igual manera los usuarios verificaron los archivos de reporte, en los cuales visualizaron el rango de ángulos que validan las posiciones correctas e incorrectas, comparando con los ángulos capturados por el sistema en la ejecución de las repeticiones de los ejercicios. En la Figura 101 se ilustra a un usuario verificando los ángulos en cada posición y brazo de los ejercicios ejecutados.

Figura 101

Verificación de reportes por el usuario



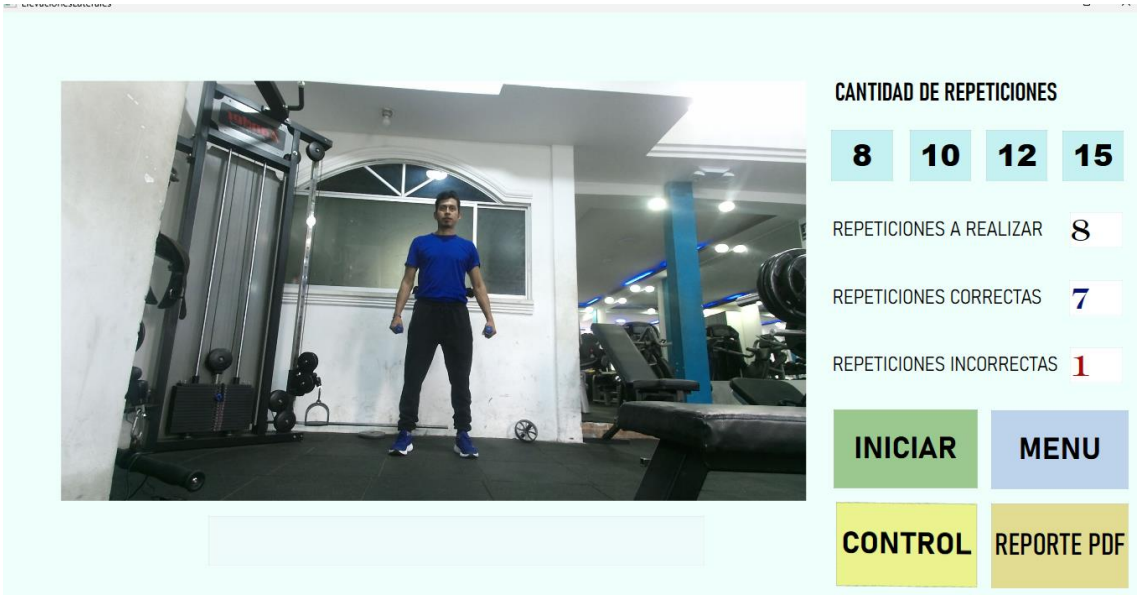
Fuente: Elaborado por el autor

Los usuarios comprobaron que el sistema opera de manera efectiva, siguiendo el monitoreo y control detallado en cada serie de entrenamiento. Se verificó que el sistema corrigió correctamente la mayoría de las repeticiones realizadas por los usuarios, y se obtuvieron resultados positivos en las nuevas series de repeticiones.

La Figura 102 ilustra el monitoreo del ejercicio de levantamientos laterales realizado por el usuario, después de varias pruebas con el sistema, se obtuvieron siete repeticiones correctas y una repetición incorrecta.

Figura 102

Monitoreo de entrenamiento, posturas corregidas por el usuario

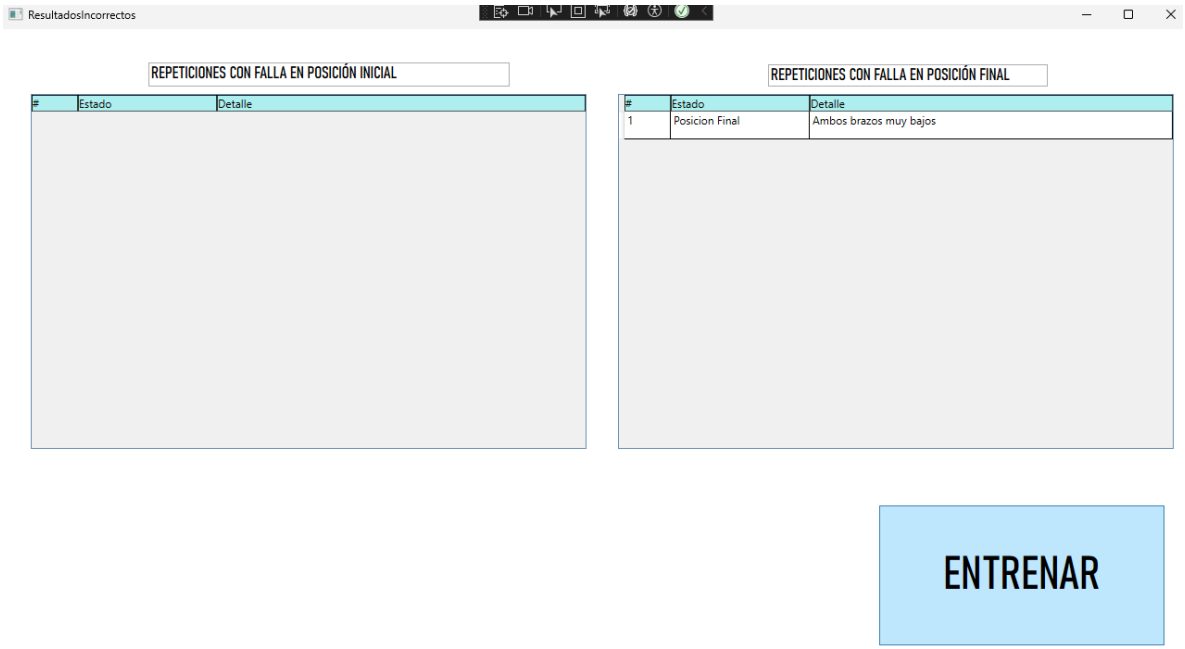


Fuente: [Sistema-Elaborado por el autor]

Por medio de las tablas de control, el usuario verificó el error detectado por el sistema, en las cuales se tiene un error en la repetición #1 específicamente en la posición final. En la Figura 103 se ilustra el detalle de la postura incorrecta detectada por el sistema.

Figura 103

Detalles de errores detectados por el sistema



Fuente: Elaborado por el autor

Las pruebas realizadas con los diferentes usuarios permitieron obtener los resultados esperados, ayudando a corregir las posturas en las posiciones de entrenamiento y proporcionando un monitoreo y seguimiento más detallado tanto para el usuario como para el instructor.

4.2 Encuetas dirigidas a instructoras y clientes

Para evaluar el nivel de aceptación del sistema, se plantean dos encuestas dirigidas a las instructoras y clientes del gimnasio ZENERGYM. Las encuestas incluyen preguntas relacionadas con el funcionamiento de la aplicación, basadas en las pruebas realizadas en el gimnasio. Las respuestas se completan en una escala lineal, donde "1" representa un nivel de aceptación bajo y "10" un nivel de aceptación alto.

4.2.1 Evaluación nivel de aceptación del sistema por parte de las instructoras

El gimnasio ZENERGYM cuenta con dos instructoras, a quienes se dirigió una encuesta con 6 preguntas. Inicialmente, las instructoras ingresaron sus nombres para verificar la información de la persona que completó la encuesta. En la Figura 104 se muestran los nombres de las dos instructoras que completaron la encuesta.

Figura 104

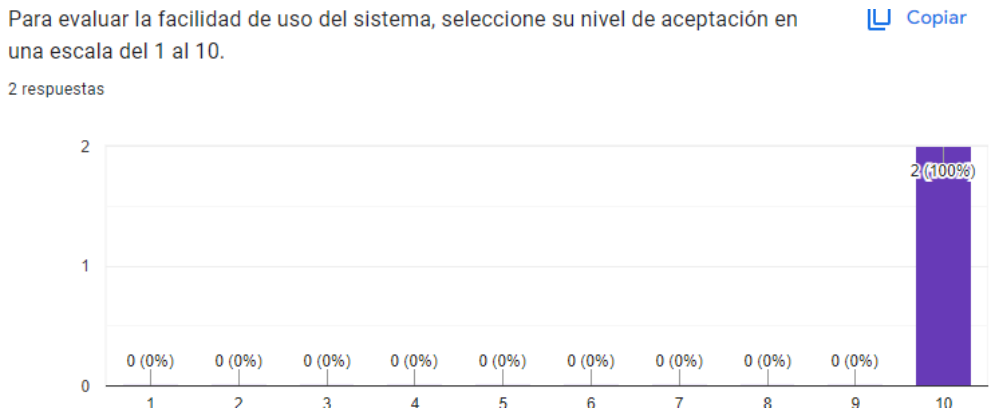
Nombres de instructoras para inicio de encuesta

Fuente: Elaborado por el autor

Pregunta 1: Se evalúa el nivel de aceptación de la facilidad de uso del sistema, obteniendo ambas respuestas un valor de 10, lo que representa un 100% de aceptación del sistema. En la Figura 105 se ilustra el valor de nivel de aceptación seleccionado por las instructoras.

Figura 105

Tabulación de pregunta 1 dirigida a instructoras

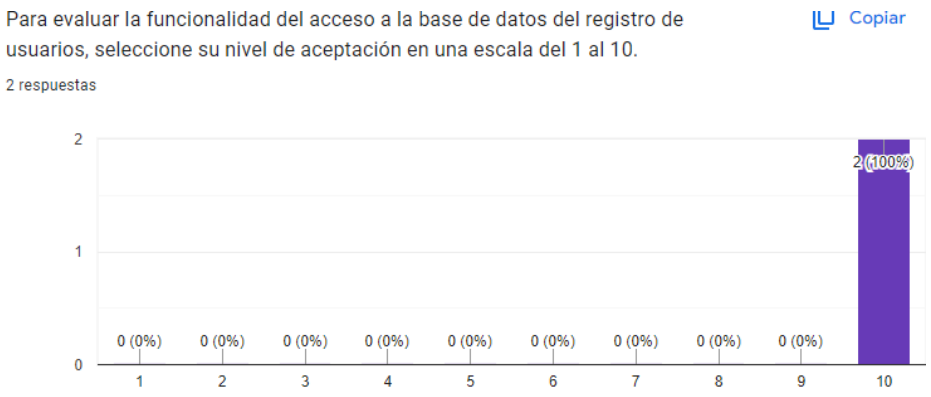


Fuente: Elaborado por el autor

Pregunta 2: Se evalúa el nivel de aceptación de la funcionalidad de acceso a la base de datos para la verificación de los usuarios registrados en el sistema. Esta información permite verificar los nombres de los clientes que ingresan al sistema y tienes acceso a la generación las alertas. Se obtuvo un valor de 10 en ambas respuestas, lo que representa un 100% de aceptación del sistema. En la Figura 106 se muestra el nivel de aceptación seleccionado por las instructoras.

Figura 106

Tabulación de pregunta 2 dirigida a instructoras

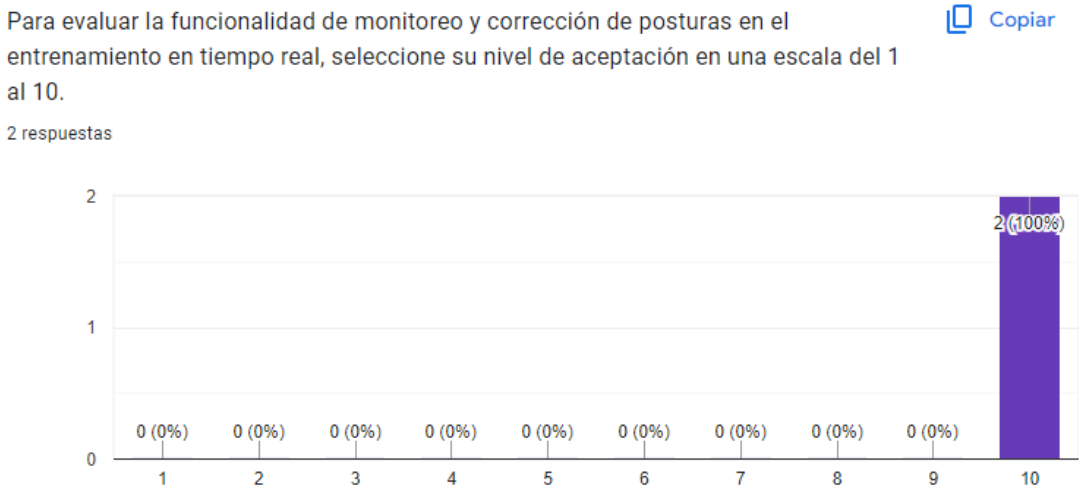


Fuente: Elaborado por el autor

Pregunta 3: Se evalúa el nivel de aceptación de la funcionalidad de monitoreo y corrección de posturas en el entrenamiento en tiempo real, bajo las pruebas realizadas por las instructoras el nivel de aceptación es de 10 en ambas respuestas, lo que representa un 100% de aceptación del sistema. En la Figura 107 se muestra el nivel de aceptación seleccionado por las instructoras.

Figura 107

Tabulación de pregunta 3 dirigida a instructoras

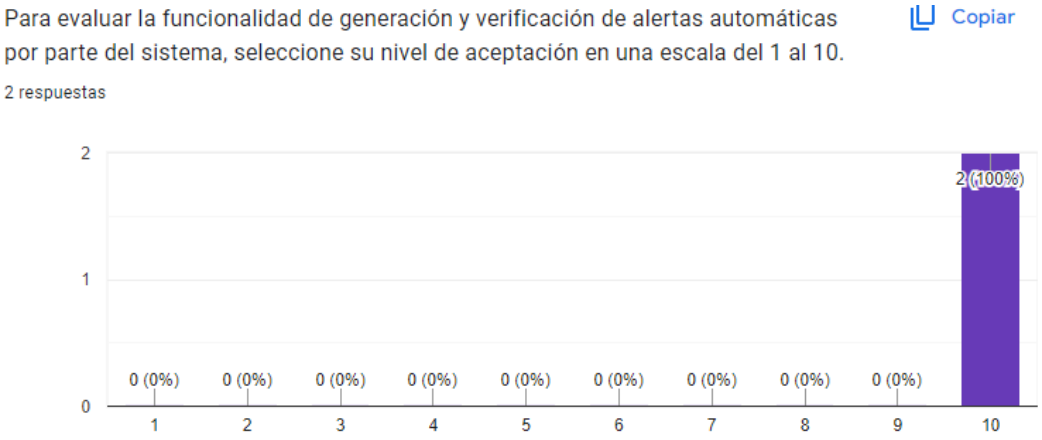


Fuente: Elaborado por el autor

Pregunta 4: Se evalúa el nivel de aceptación de la funcionalidad de generación y verificación de alertas automáticas por parte del sistema cuando el cliente activa el umbral, obteniendo un valor de 10 en ambas respuestas, lo que representa un 100% de aceptación del sistema. En la Figura 108 se muestra el nivel de aceptación seleccionado por las instructoras.

Figura 108

Tabulación de pregunta 4 dirigida a instructoras

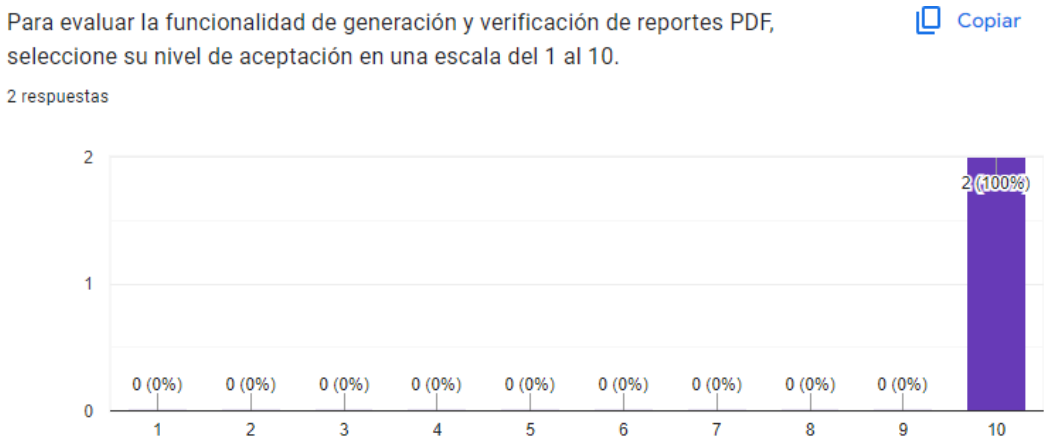


Fuente: Elaborado por el autor

Pregunta 5: Se evalúa el nivel de aceptación de la funcionalidad de generación y verificación de reportes PDF, obteniendo un valor de 10 en ambas respuestas, lo que representa un 100% de aceptación del sistema. En la Figura 109 se muestra el nivel de aceptación seleccionado por las instructoras.

Figura 109

Tabulación de pregunta 5 dirigida a instructoras

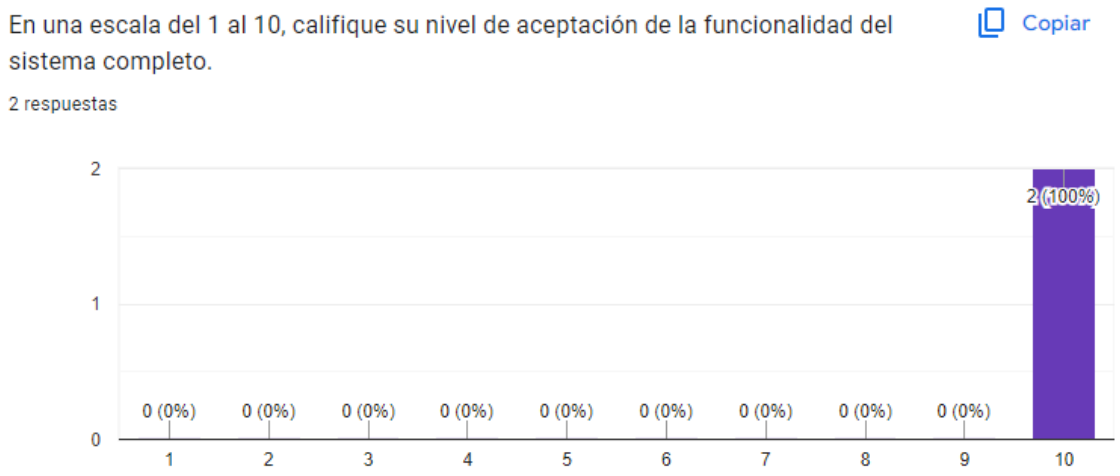


Fuente: Elaborado por el autor

Pregunta 6: Se evalúa el nivel de aceptación de la funcionalidad del sistema completo bajo las pruebas realizadas por las instructoras, quienes asignaron un valor de 10 en ambas respuestas, lo que representa un 100% de aceptación del sistema. En la Figura 110 se muestra el nivel de aceptación seleccionado por las instructoras.

Figura 110

Tabulación de pregunta 6 dirigida a instructoras



Fuente: Elaborado por el autor

4.2.2 Evaluación nivel de aceptación del sistema por parte de los clientes

Se realizó una encuesta con 8 preguntas sobre la funcionalidad del sistema a ocho clientes del gimnasio ZENERGYM. Estos clientes llevaron a cabo diversas pruebas del sistema, verificando cada una de las funcionalidades de la aplicación. Inicialmente, los clientes ingresaron sus nombres, como se muestra en la Figura 111.

Figura 111

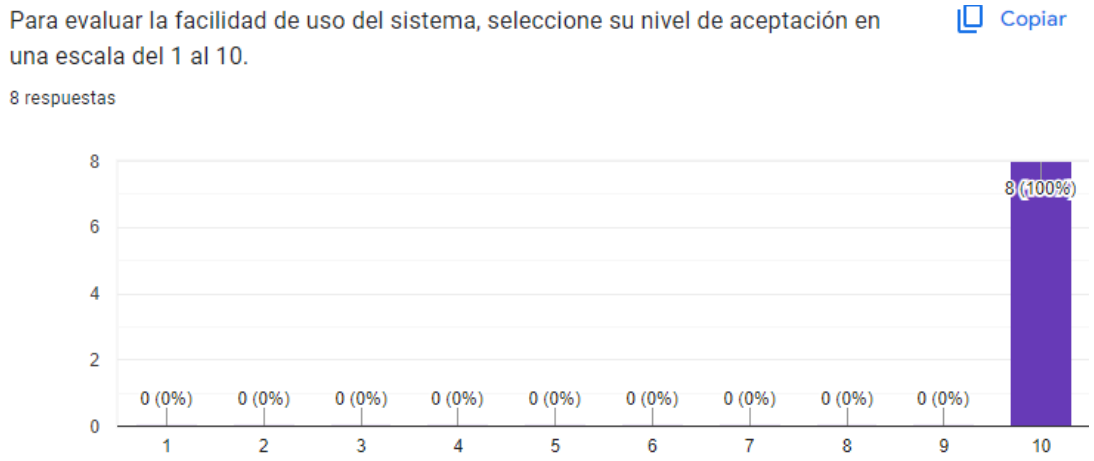
Nombre de clientes para inicio de encuesta

Fuente: Elaborado por el autor

Pregunta 1: Se evalúa el nivel de aceptación de la facilidad de uso del sistema, obteniendo un valor de 10 en las 8 respuestas generadas por los clientes, lo que representa un 100% de aceptación del sistema. En la Figura 112 se muestra el nivel de aceptación seleccionado por los clientes.

Figura 112

Tabulación de pregunta 1 dirigida a clientes

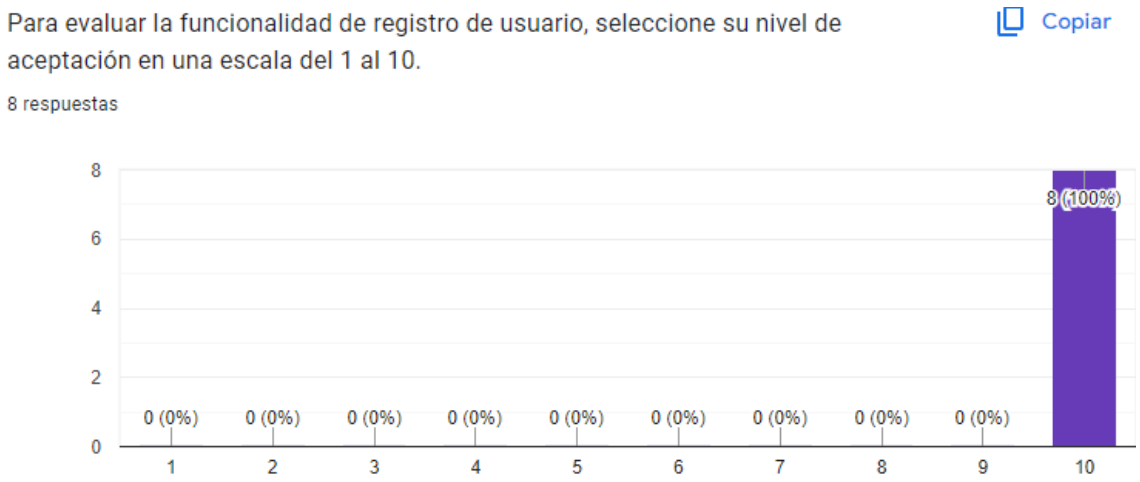


Fuente: Elaborado por el autor

Pregunta 2: Se evalúa el nivel de aceptación de la funcionalidad de registro de usuario, los resultados mostrados en la Figura 113 indican que todos los clientes otorgaron la calificación máxima de 10, lo que presenta un 100% de aceptación del sistema.

Figura 113

Tabulación de pregunta 2 dirigida a clientes

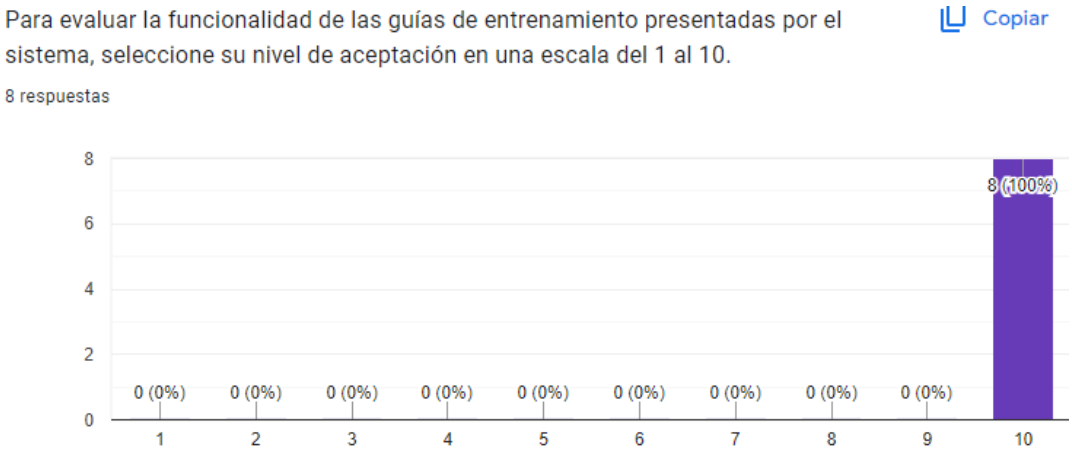


Fuente: Elaborado por el autor

Pregunta 3: Se evalúa el nivel de aceptación de la funcionalidad de las guías de entrenamiento presentadas por el sistema, los resultados mostrados en la Figura 114 indican que todos los clientes otorgaron la calificación máxima de 10, lo que presenta un 100% de aceptación del sistema.

Figura 114

Tabulación de pregunta 3 dirigida a clientes



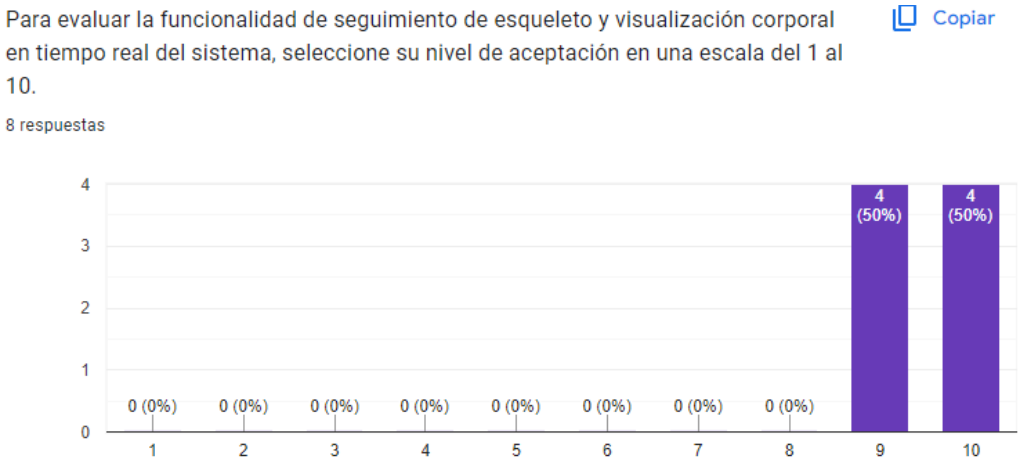
Fuente: Elaborado por el autor

Pregunta 4: Se evalúa el nivel de aceptación de la funcionalidad de seguimiento de esqueleto y visualización corporal en tiempo real, los resultados mostrados en la Figura 115 indican que el 50% de los clientes otorgaron una calificación de 9 y el otro 50% otorgó una calificación de 10, demostrando una alta aceptación de esta funcionalidad.

Se verificó que el sistema tiende a tener fallas menores por la presencia de la luz de las luminarias del gimnasio, por lo que cual, las pruebas realizadas con clientes que entrenan en horarios de la tarde y noche tuvieron inconvenientes menores con la detección y seguimiento del esqueleto en tiempo real.

Figura 115

Tabulación de pregunta 4 dirigida a clientes

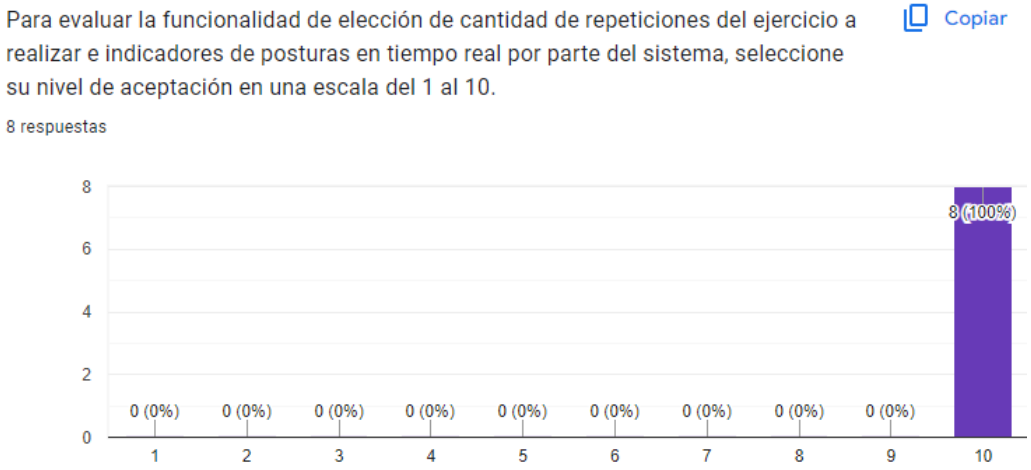


Fuente: Elaborado por el autor

Pregunta 5: Se evalúa el nivel de aceptación de la funcionalidad de elección de cantidad de repeticiones de ejercicios a realizar e indicadores de posturas en tiempo real por parte del sistema, los resultados mostrados en la Figura 116 indican que todos los clientes otorgaron la calificación máxima de 10, lo que presenta un 100% de aceptación del sistema.

Figura 116

Tabulación de pregunta 5 dirigida a clientes

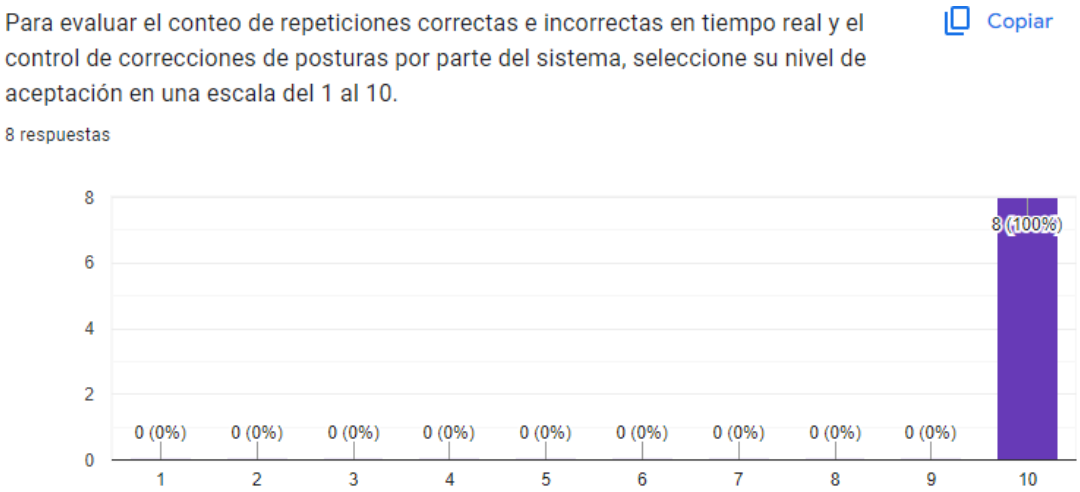


Fuente: Elaborado por el autor

Pregunta 6: Se evalúa el nivel de aceptación de la funcionalidad del conteo de repeticiones correctas e incorrectas en tiempo real y el control de correcciones de posturas por parte del sistema, los resultados mostrados en la Figura 117 indican que todos los clientes otorgaron la calificación máxima de 10, lo que presenta un 100% de aceptación del sistema.

Figura 117

Tabulación de pregunta 6 dirigida a clientes

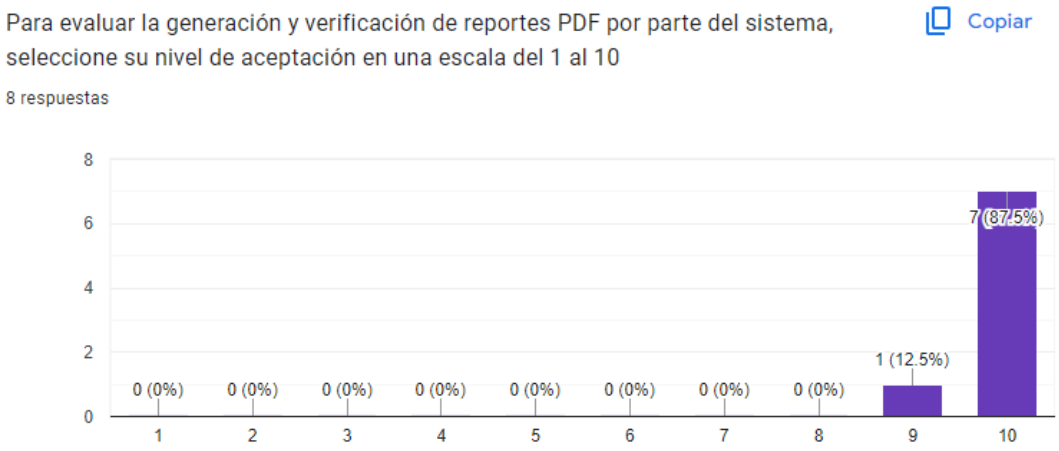


Fuente: Elaborado por el autor

Pregunta 7: Se evalúa el nivel de aceptación de la funcionalidad de generación de reportes PDF por parte del sistema, los resultados mostrados en la Figura 118 indican que el 12.5% de los clientes, que corresponde a un cliente otorgó una calificación de 9 y el otro 87.5%, que corresponde a siete personas, otorgaron una calificación de 10, demostrando una alta aceptación de esta funcionalidad.

Figura 118

Tabulación de pregunta 7 dirigida a clientes

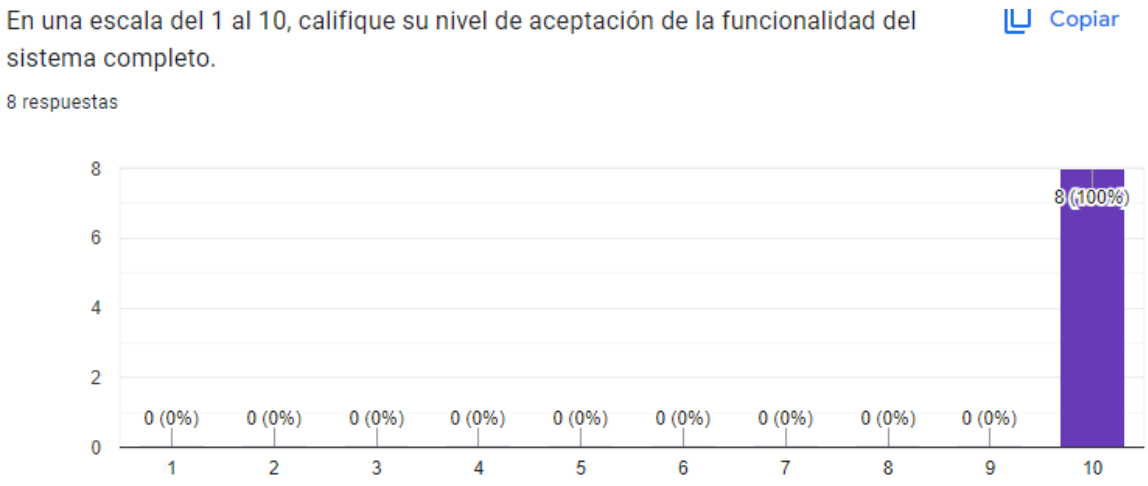


Fuente: Elaborado por el autor

Pregunta 8: Se evalúa el nivel de aceptación de la funcionalidad del sistema completo según las pruebas realizadas a los ocho clientes. La Figura 119 indica que el 100% de los clientes otorgaron la máxima calificación de 10, indicando una total satisfacción y aceptación del sistema.

Figura 119

Tabulación de pregunta 8 dirigida a clientes



Fuente: Elaborado por el autor

CONCLUSIONES

A través de la revisión bibliográfica e información brindada por la instructora del gimnasio ZENERGYM, se describió información relevante sobre los ejercicios de levantamientos laterales y press militar, destacando su importancia en el fisicoculturismo para el desarrollo de los músculos que conforman los hombros. Se describió las técnicas correctas en la ejecución de estos ejercicios, lo cual proporcionó una base teórica para el desarrollo del sistema de corrección de posturas de entrenamiento.

Se determinó los requerimientos de hardware y software, con el uso del dispositivo Kinect V2 para la captura de movimientos y una computadora con las características específicas para el procesamiento de los datos. En cuanto al software se determinó que visual studio 2022, con el uso del lenguaje de programación C# el cual sirvió para desarrollar una aplicación que integrara las funcionalidades necesarias del SDK del dispositivo Kinect y una interfaz gráfica interactiva para los clientes. Además, se determinó las plataformas de monitoreo como Ubidots y para la notificación de alertas a través de Telegram.

Se diseñó un sistema inteligente utilizando el dispositivo Kinect como herramienta principal para la captura de movimientos corporales de los clientes, aprovechando su alta precisión en la detección de articulaciones y movimientos tridimensionales. Además, se desarrolló un algoritmo que calcula los ángulos entre las articulaciones detectadas durante la ejecución de ejercicios en tiempo real, facilitando la identificación de posturas tanto correctas como incorrectas. Esta funcionalidad permitió que el sistema ofrezca retroalimentación inmediata al concluir cada serie de repeticiones, lo cual ayudó a los usuarios del gimnasio ZENERGYM a corregir sus posturas en las subsiguientes series.

Se realizó las pruebas de funcionamiento del sistema inteligente de control de posturas de entrenamiento, evidenciando su alto desempeño en la detección y corrección de posturas incorrectas durante los ejercicios de levantamientos laterales y press militar. En las pruebas realizadas participaron ocho clientes y dos instructoras del gimnasio ZENERGYM, cuya retroalimentación fue crucial para validar la funcionalidad y utilidad del sistema. Los resultados de las encuestas aplicadas confirmaron una excelente aceptación y satisfacción por parte de los usuarios e instructoras, obteniendo calificaciones entre 9 y 10 en cada pregunta planteada, destacando así su impacto y mejora significativa en las rutinas de entrenamiento.

RECOMENDACIONES

Para el uso del sistema de monitoreo de posturas, se recomienda al usuario entrenar con iluminación natural, evitando el uso de luminarias que puedan interferir con los valores capturados de las articulaciones. Además, es importante verificar que los usuarios no estén posicionados contraluz para evitar errores en la recepción de los datos.

Verificar la distancia del usuario en relación con el dispositivo Kinect antes de comenzar cualquier serie de entrenamiento monitoreada por el sistema. La precisión en la recolección de datos depende de la detección de las articulaciones del usuario, ya que el sistema calcula los ángulos en tiempo real a partir de las coordenadas X, Y, y Z de las articulaciones.

Al inicio de cada entrenamiento es importante verificar que el dispositivo Kinect se active correctamente, asegurándose que el led de color blanco del dispositivo Kinect esté encendido, lo cual indica una conexión correcta tanto al adaptador de voltaje como al ordenador. Esta verificación es crucial para garantizar el correcto funcionamiento del sistema durante los entrenamientos.

Para el desarrollo de proyectos similares, se sugiere añadir más ejercicios de entrenamiento y el uso de más de un dispositivo Kinect, donde el usuario tenga el acceso a más variantes de ejercicios y tenga un sistema que monitoree sus posturas de entrenamiento con un mejor rendimiento.

BIBLIOGRAFÍA

Organización Mundial de la Salud. (5 de Octubre de 2022). *Actividad física*. Obtenido de Actividad Física: https://www.who.int/es/news-room/fact-sheets/detail/physical-activity

Secretaría Nacional de Planificación. (2021). *Plan de Creación de Oportunidades 2021-2025 [versión PDF]*. Obtenido de Plan de Creación de Oportunidades 2021-2025 – Secretaría Nacional de Planificación: https://www.planificacion.gob.ec/wp-content/uploads/2021/09/Plan-de-Creacio%CC%81n-de-Oportunidades-2021-2025-Aprobado.pdf

Areba, J. B. de. (2001). *Metodología del análisis estructurado de sistemas*. Univ Pontifica Comillas.

Buenache, J. V. (2009). *In Corpore Sano: Principios Básicos del Entrenamiento con Pesas*. Vision Libros.

De la Fuente Garrido, D. (2012). *Aplicaciones de Kinect para neurohabilitación* [Bachelor thesis, Universitat Politècnica de Catalunya]. https://upcommons.upc.edu/handle/2099.1/15334

De La Guardia Gutiérrez, M. A., Ruvalcaba Ledezma, J. C., De La Guardia Gutiérrez, M. A., & Ruvalcaba Ledezma, J. C. (2020). La salud y sus determinantes, promoción de la salud y educación sanitaria. *Journal of Negative and No Positive Results*, *5*(1), 81-90. https://doi.org/10.19230/jonnpr.3215

Delavier, F. (2007). *GUÍA DE LOS MOVIMIENTOS DE MUSCULACIÓN. DESCRIPCIÓN ANATÓMICA (Color)*. Editorial Paidotribo.

Elevación lateral de hombros con mancuernas de pie brazos estirados. (s. f.). Recuperado 2 de mayo de 2024, de https://www.entrenamientos.com/ejercicios/elevacion-lateral-de-hombros-con-mancuernas-de-pie-brazos-estirados

Everett, G. (2020). *Halterofilia: Guía completa para deportistas y entrenadores*. Paidotribo.

Gómez Ayala, A.-E. (2007). Salud física y salud mental. Un binomio indisoluble. *Farmacia Profesional*, *21*(7), 53-56.

Guzm‡n, J., & Sierra, J. (s. f.). *Escuhando con los dedos. Reconociendo el LSC, un acercamiento con Kinect y Processing*. Lulu.com.

ISO/IEC/IEEE International Standard—Systems and software engineering – Life cycle processes – Requirements engineering. (2018). *ISO/IEC/IEEE 29148:2018(E)*, 1-104. https://doi.org/10.1109/IEEESTD.2018.8559686

Jamhoury, L. (2022, abril 21). Understanding Kinect V2 Joints and Coordinate System. *Medium*. https://lisajamhoury.medium.com/understanding-kinect-v2-joints-and-coordinate-system-4f4b90b9df16

La ergonomía en el deporte. (2014, noviembre 1). *Talent Pool Consulting*. https://www.talentpoolconsulting.com/la-ergonomia-en-el-deporte/

Linares, R. C. (2014). *Anatomía & Musculación: Guía visual completa*. Paidotribo.

Muñoz, J. E. (2016). *Ergonomía básica*. Ediciones de la U.

Navarro, F. M. (2007). *Auxiliares de Enfermeria de la Diputacion de Granada. Temario Especifico*. MAD-Eduforma.

Pantaleo, G., & Rinaudo, L. (2015). *Ingeniería de Software*. Alpha Editorial.

Paul, S., Basu, S., & Nasipuri, M. (2016). *Microsoft Kinect in Gesture Recognition: A Short Review*. https://www.semanticscholar.org/paper/Microsoft-Kinect-in-Gesture-

Recognition%3A-A-Short-Paul-Basu/bdbf4e2b14c7599600720410f48ae6d0b111e0f2

Pearl, B. (2008). *TRATADO GENERAL DE LA MUSCULACIÓN*. Editorial Paidotribo.

Perdomo Ogando, J. M., Pegudo Sánchez, A. G., & Capote Dominguez, T. E. (2018). Premisas para la investigación biomecánica en la cultura física. *Revista Cubana de Educación Superior*, *37*(2), 104-114.

Press militar o de hombros con mancuernas de pie. (s. f.). Recuperado 7 de septiembre de 2023, de https://www.entrenamientos.com/ejercicios/press-militar-o-de-hombros-con-mancuernas-de-pie

R, I. I. S. (2023). *Entrenamiento Científico Con Pesas: Fitness Inteligente*. Román.

Reconocimiento de posturas con Kinect I. (s. f.). Recuperado 1 de mayo de 2024, de http://software-tecnico-libre.es/es/articulo-por-tema/todas-las-secciones/todos-los-temas/todos-los-articulos/kinect-posturas-1

Remor, E., & Pérez-Llantada Rueda, M. C. (2007). La relación entre niveles de la actividad física y la experiencia de estrés y de síntomas de malestar físico. *Interamerican Journal of Psychology*, *41*(3), 313-322.

Rodríguez, A. S. (2019). *Fisicoculturismo.Orígenes antropológicos y connotaciones filosóficas*. Midac, SL.

Sanabria, N. S., & Patiño, A. M. O. (2013). Biomecánica del hombro y bases fisiológicas de los ejercicios de Codman. *CES Medicina*, *27*(2), Article 2.

Suarez, R. (2009). *Biomecánica deportiva y control del entrenamiento*. Funámbulos Editores.

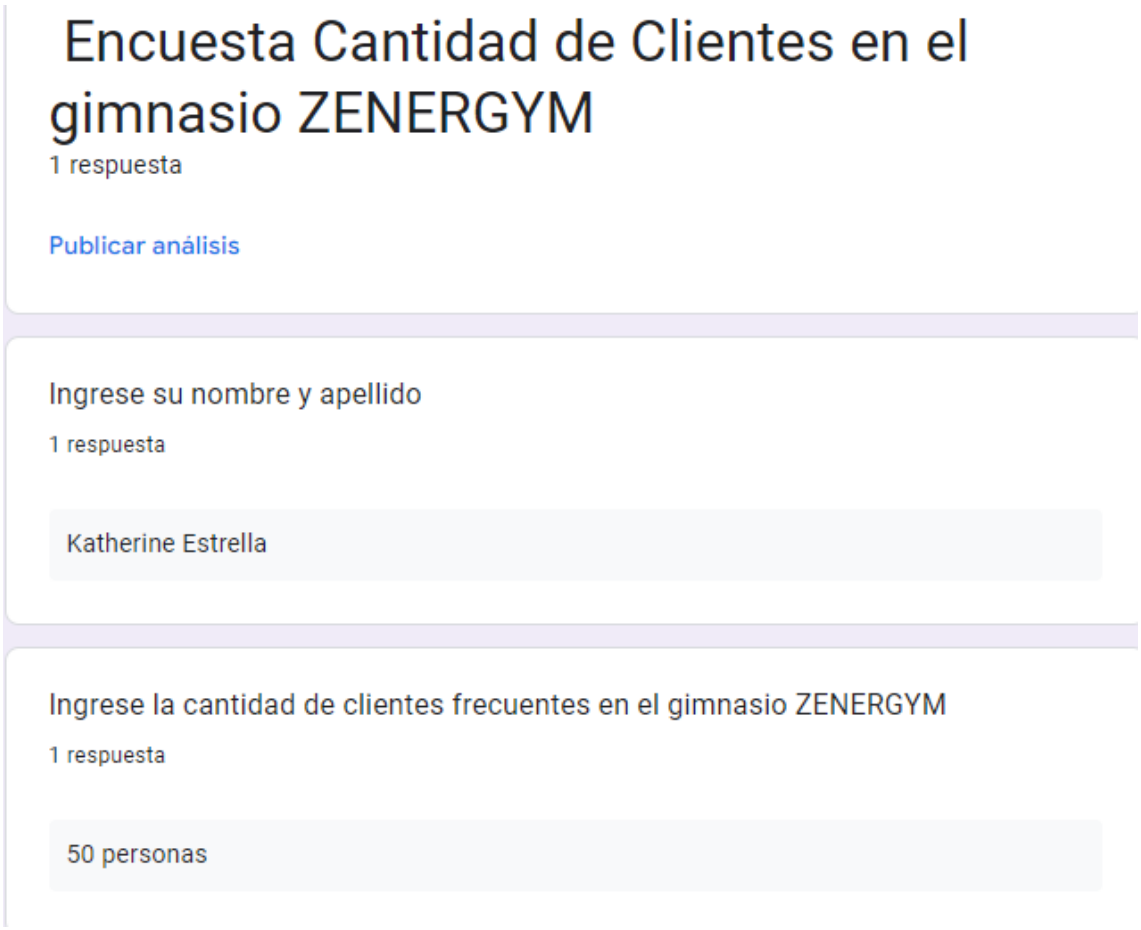
Voegeli, A. V. (2000). *Lecciones básicas de biomecánica del aparato locomotor*. Springer Science & Business Media.

Yang, L., Zhang, L., Dong, H., Alelaiwi, A., & El Saddik, A. (2015). Evaluating and Improving the Depth Accuracy of Kinect for Windows v2. *IEEE Sensors Journal*, *15*, 1-1. https://doi.org/10.1109/JSEN.2015.2416651

Zavala-Choez, F. N., & Vélez-Moreira, E. M. (2020). La gestión de la calidad y el servicio al cliente como factor de competitividad en las empresas de servicios—Ecuador. *Domino de las Ciencias*, *6*(3), Article 3. https://doi.org/10.23857/dc.v6i3.1284

ANEXOS

Anexo 1. (Encuesta dirigida a gerente del gimnasio ZENERGYM, cantidad de clientes en el gimnasio).



Encuesta Cantidad de Clientes en el gimnasio ZENERGYM

1 respuesta

Publicar análisis

Ingrese su nombre y apellido

1 respuesta

Katherine Estrella

Ingrese la cantidad de clientes frecuentes en el gimnasio ZENERGYM

1 respuesta

50 personas

Anexo2 (Encuesta dirigida Clientes del Gimnasio ZENERGYM)

Levantamiento de información sobre los requerimientos de los clientes.

1. ¿Estaría usted de acuerdo de utilizar una herramienta la cual le ayude con la corrección de posturas corporales en sus entrenamientos?

a. si

b. no

2. ¿Le resultaría cómodo tener la posibilidad de visualizar en tiempo real su ejecución de ejercicios por medio de una pantalla?

a. si

b. no

3. ¿Le gustaría observar la cantidad de repeticiones realizadas de manera correcta e incorrecta?

a. si

b. no

4. Le gustaría tener acceso a información de la explicación de ejecución de los ejercicios de press militar y levantamientos laterales?

a. si

b. no

5. ¿Estaría usted de acuerdo que su instructor/a del gimnasio reciba alertas cuando realice cierta cantidad de repeticiones de manera errónea en los ejercicios de elevaciones laterales y press militar?

a. si

b. no

6. ¿Estaría de acuerdo en recibir una explicación adicional por parte del instructor/a cuando los ejercicios de hombros en sus dos variantes no se han ejecutado de forma correcta?

a. si

b. no

7. ¿Le gustaría que su instructor verifique la disponibilidad del área de entrenamiento para los ejercicios de hombros antes de comenzar su rutina?

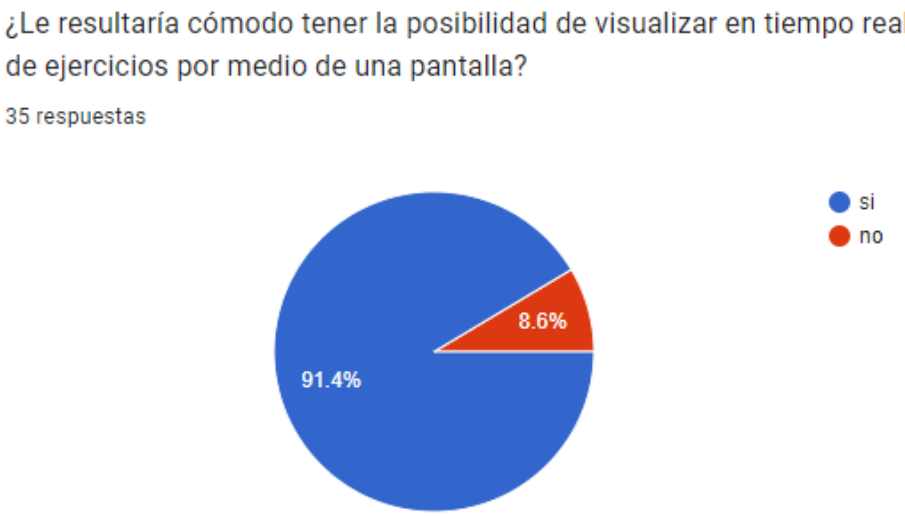
a. si

b. no

Anexo3 (Resultados y análisis de encuesta realizada a clientes del gimnasio ZENERGYM)

Pregunta 1: De acuerdo con la pregunta 1 de la encuesta realizada, se obtuvo un porcentaje de 94.3% de aprobación por parte de los usuarios al contar con una herramienta que permita la corrección de posturas corporales en el entrenamiento, % que demuestra la aceptación por parte de los clientes del sistema a diseñar. En la siguiente figura se muestran los resultados.

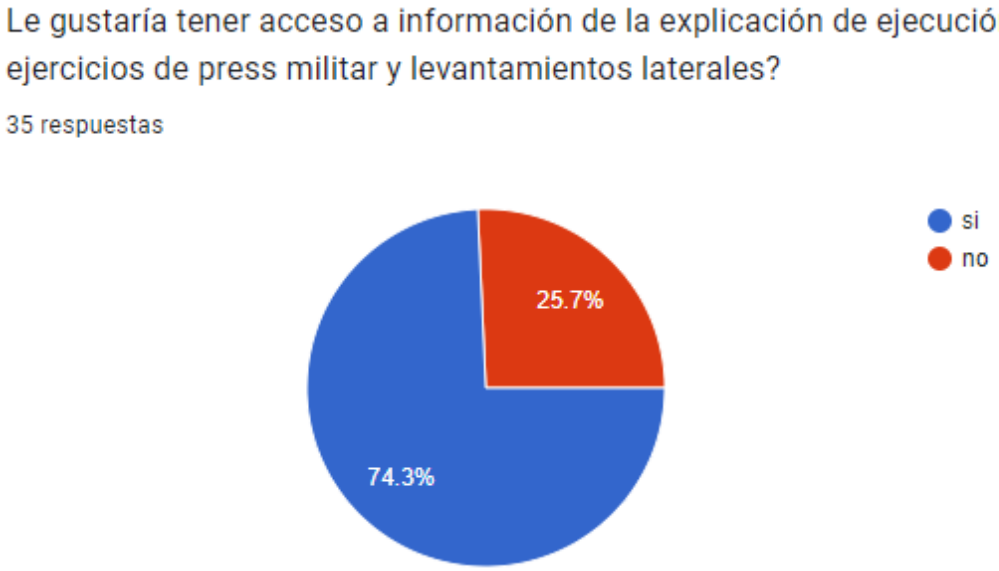
Pregunta 2: En la pregunta 2 los usuarios del gimnasio mostraron una aceptación del 91.4% en la cual, por medio de una pantalla se pueda observar la ejecución de los ejercicios realizados en tiempo real. En la siguiente figura se muestran los resultados obtenidos.



Pregunta 3: La pregunta 3 realizada a los usuarios del gimnasio se pudo determinar que la observación de la cantidad de repeticiones realizadas de manera correcta e incorrecta presenta una gran aprobación por parte de los clientes con un total del 88.6% de resultados del "Si". En la siguiente figura se muestra el porcentaje de aceptación obtenidos.

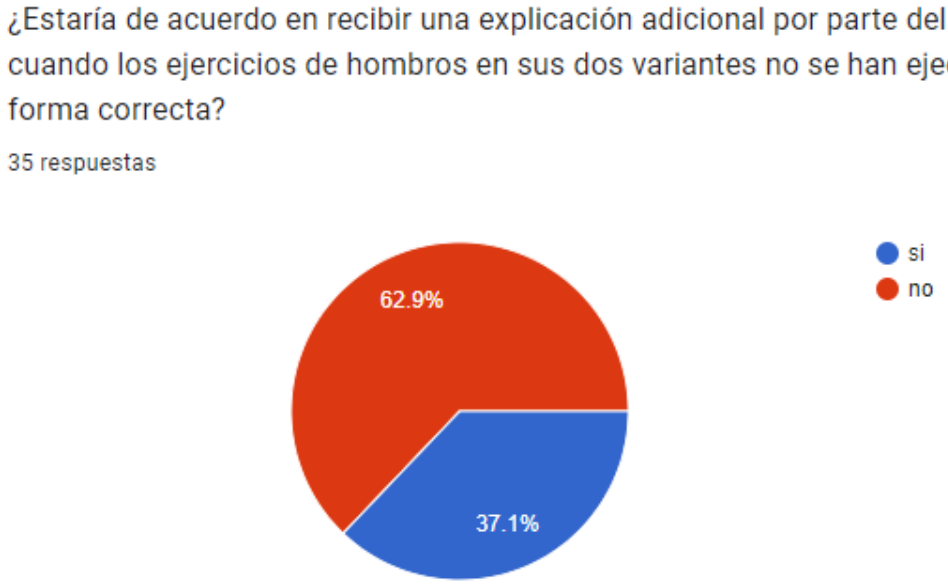
Pregunta 4: El acceso a una guía de explicación de los ejercicios de press militar y levantamientos laterales es un punto clave para los usuarios, obteniendo un 74.3% de

aceptación en la totalidad de las encuestas realizadas, el 25.7% de las personas no le gustaría tener acceso a la información con relación a la guía de los ejercicios a realizar. La siguiente figura muestra la tabulación de los resultados.

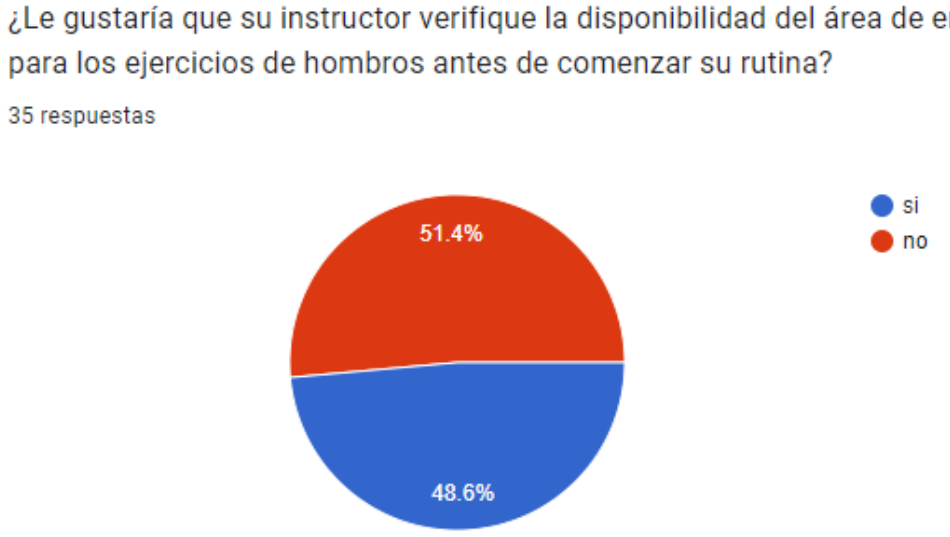


Pregunta5: Los resultados obtenidos de la quinta pregunta realizada indican que hay un 42.9% de aceptación con respecto al envío de alertas cuando se ejecuten una cierta cantidad de repeticiones erróneas de los ejercicios. Esto muestra un porcentaje relativamente bajo de interés en dicho requisito. Los resultados se presentan en la figura siguiente.

Pregunta6: Las respuestas a la pregunta 6 revelan que los usuarios del gimnasio no consideran de suma importancia recibir guía adicional por parte del instructor o instructora cuando cometen errores de postura de manera constante en los ejercicios. El porcentaje de aceptación de esta idea es solo del 37.1% del total de respuestas obtenidas. Los resultados detallados se presentan en la figura siguiente.



Pregunta7: Las respuestas de la pregunta 7 demuestran que la verificación de disponibilidad del área de entrenamiento para los ejercicios de hombros antes de comenzar una rutina de entrenamiento es de importancia neutral para los clientes con una aceptación del 48.6% de la encuesta realizada. En la siguiente figura se muestran los resultados.



Anexo4 (Encuesta dirigida a señorita gerente del gimnasio ZENERGYM)

Levantamiento de información sobre los requerimientos de la gerente del gimnasio ZENERGYM.

1. ¿Estaría usted de acuerdo en proporcionar un seguimiento adicional en caso de que el sistema detecte que el cliente enfrenta dificultades continuas durante la ejecución de los ejercicios?

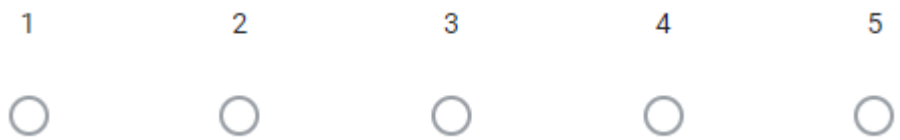
a) Estoy de acuerdo

b) No estoy de acuerdo

2. ¿Qué requisitos funcionales mínimos usted cree necesarios para que el sistema inteligente de corrección de posturas en los ejercicios de hombros opere efectivamente en un gimnasio?

3. Seleccione que tan reducido o compacto debería ser el sistema de corrección de posturas ("1" menos importancia y "5" mayor importancia).

4. Que tan importante usted cree que es tener una base de datos para el registro de los usuarios que utilicen el sistema de corrección de posturas



5. En cuanto al uso del sistema de corrección de ejercicios ¿Cuál de las siguientes opciones considera usted la más adecuada?

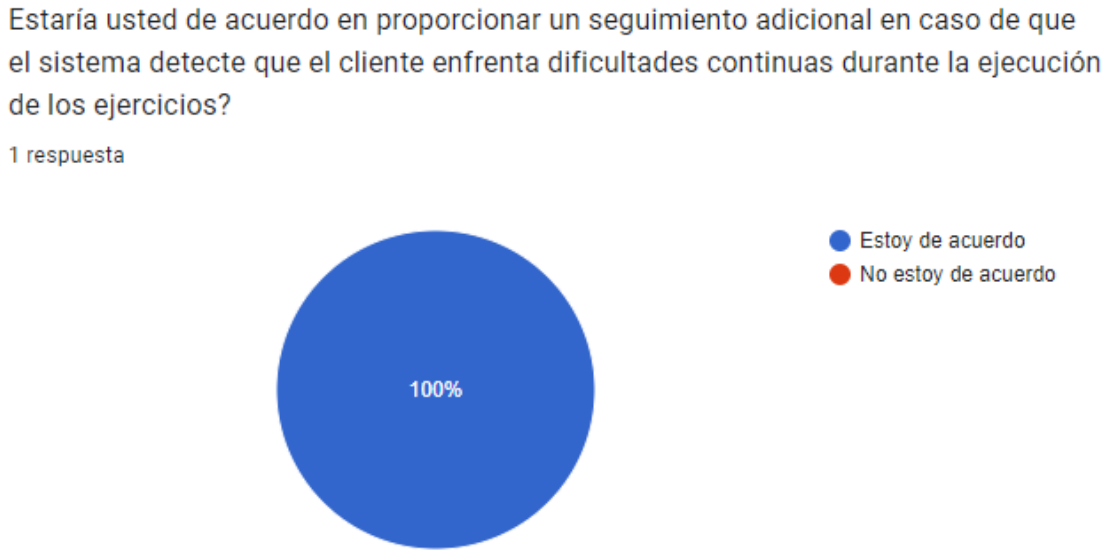
a) Sistema de uso con complejidad baja

b) Sistema de uso con complejidad media

c) Sistema de uso con complejidad alta

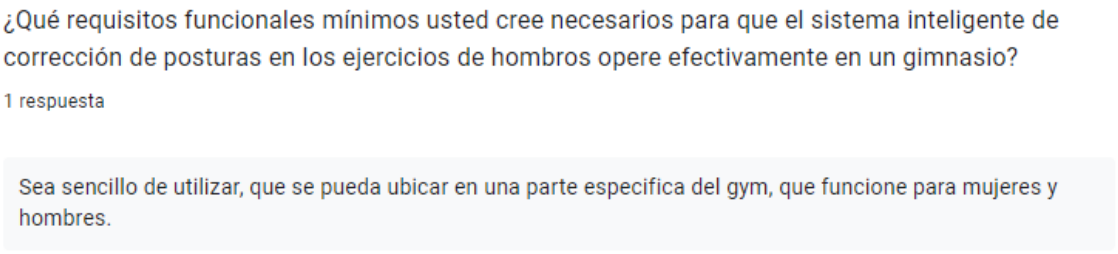
Anexo5 (Resultados y análisis de encuesta (Anexo4))

Pregunta1: Los resultados obtenidos en la primera pregunta planteada demuestra que la señorita gerente está totalmente de acuerdo en proporcionar un seguimiento adicional cuando el sistema detecta errores continuos en la ejecución de los ejercicios por parte de los clientes. En la siguiente Figura se muestran los resultados obtenidos.

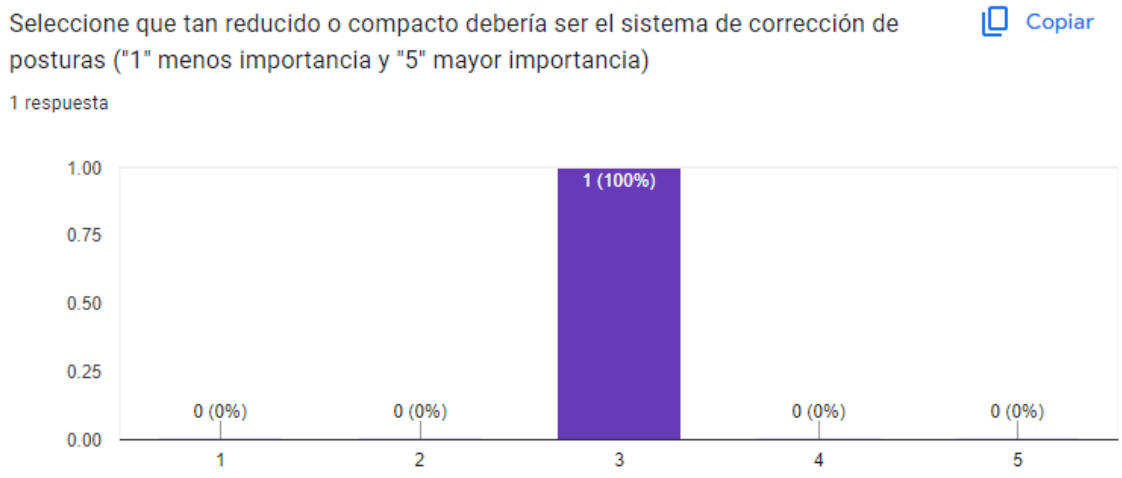


Pregunta2: El resultado de la segunda pregunta planteada es de respuesta abierta por lo cual se puede apreciar que los requisitos mínimos por parte de la instructora es tener un

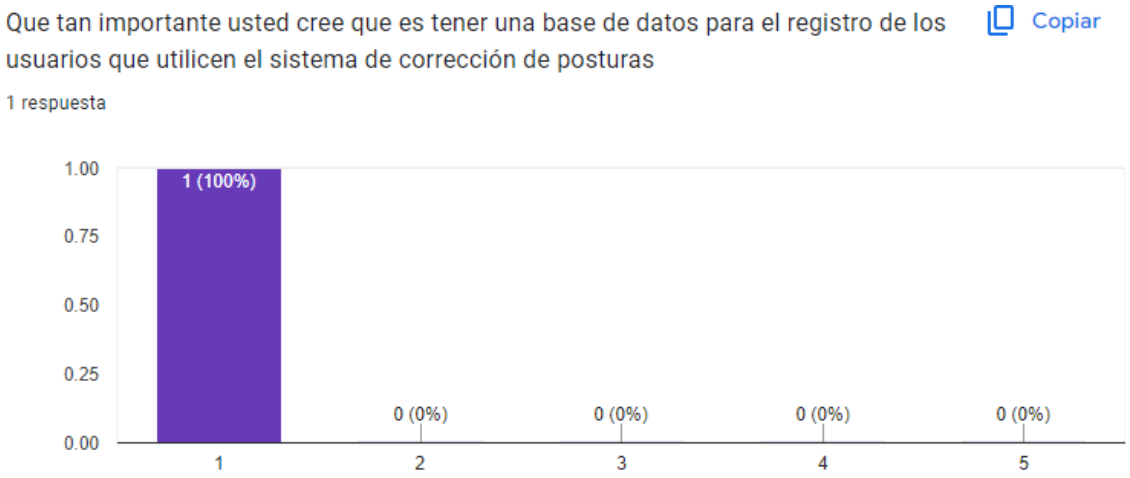
sistema sencillo, que se pueda ubicar en un sitio del gimnasio y sea funcional para hombres y mujeres. En la siguiente figura se demuestra la tabulación de la respuesta obtenida.



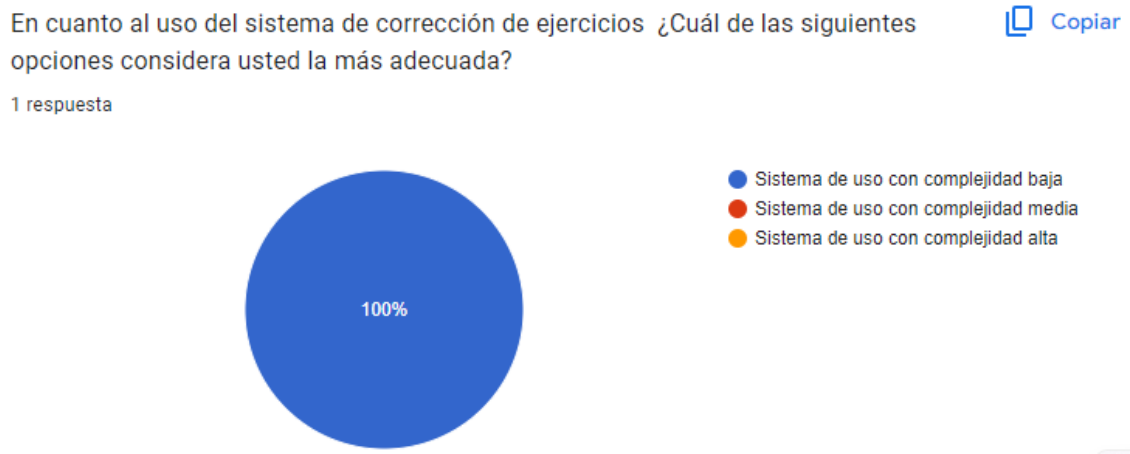
Pregunta3: Los resultados de la tercera pregunta, demuestran la importancia que se le atribuye a que el sistema sea de tamaño reducido o compacto. Esto se refleja en la calificación obtenida, que es de 3 sobre 5 en la escala de importancia. En la siguiente figura se demuestra la tabulación de los resultados obtenidos.



Pregunta4: Los resultados obtenidos de la pregunta 4 indican que, desde el punto de vista de la gerente e instructora, la base de datos no se considera muy importante, lo cual se refleja en su valoración de 1 en una escala de importancia de 1 a 5. En la siguiente figura se tabula los resultados obtenidos.



Pregunta5: Los resultados de la pregunta 5 indican que el sistema debe ser de uso sencillo y presentar una baja complejidad. Los resultados obtenidos se tabulan en la figura siguiente



Anexo 6. MANUAL DE INSTALACIÓN SDK KINECT

Introducción

El manual de instalación del SDK Kinect provee información necesaria para la instalación del Kit de desarrollo de software, permitiendo tener acceso a las funciones del dispositivo Kinect junto con el sistema de correcciones de posturas para los ejercicios de levantamientos laterales y Press militar.

Propósito

El manual desarrollado tiene el propósito de dar una guía detallada de la instalación del SDK de Kinect, permitiendo dar acceso a las funciones y activación de sensores del dispositivo Kinect.

Requerimientos

Para la instalación del SDK del Kinect es necesario contar con los recursos necesarios del ordenador que se listan a continuación.

- Procesador de 64 bits.
- Puerto USB 3.0
- 4GB de RAM
- Tarjeta gráfica que admita DirectX 11.
- Windows 8 o superior
- Adaptador Kinect

Instalación

A. El proceso de instalación inicia con la descarga del controlador, el proceso se realiza accediendo al siguiente enlace de Microsoft:

https://www.microsoft.com/en-us/download/details.aspx?id=44561

Al abrir el enlace, diríjase a la sección Kinect para Windows, seleccione el idioma y descargue el controlador.

Ilustración 1. Descarga SDK 2.0

B. Desde la ubicación de descargas, haga doble click en el instalador, acepte los términos y condiciones y haga click en instalar.

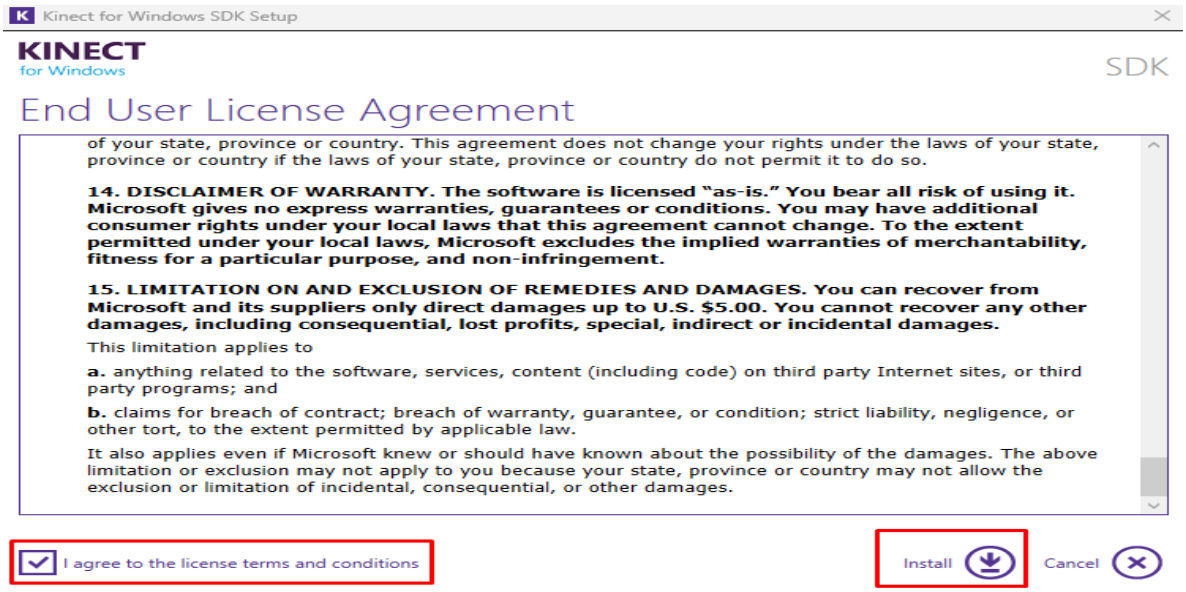


Ilustración 2. Inicio de instalación de controlador

C. Verifique que la instalación se ha completado con éxito y conecte el dispositivo Kinect al puerto USB 3.0 de su ordenador. Puede verificar que la instalación se ha realizado con éxito cuando se encuentre el dispositivo Kinect.

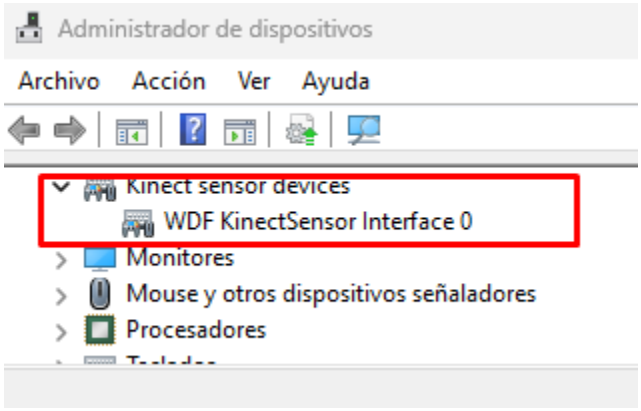


Ilustración 3. Verificación de controlador

D. El proceso de instalación está completo, en la barra de búsqueda inserte "SDK Browser" y tendrá acceso a probar los sensores y funciones del dispositivo Kinect.

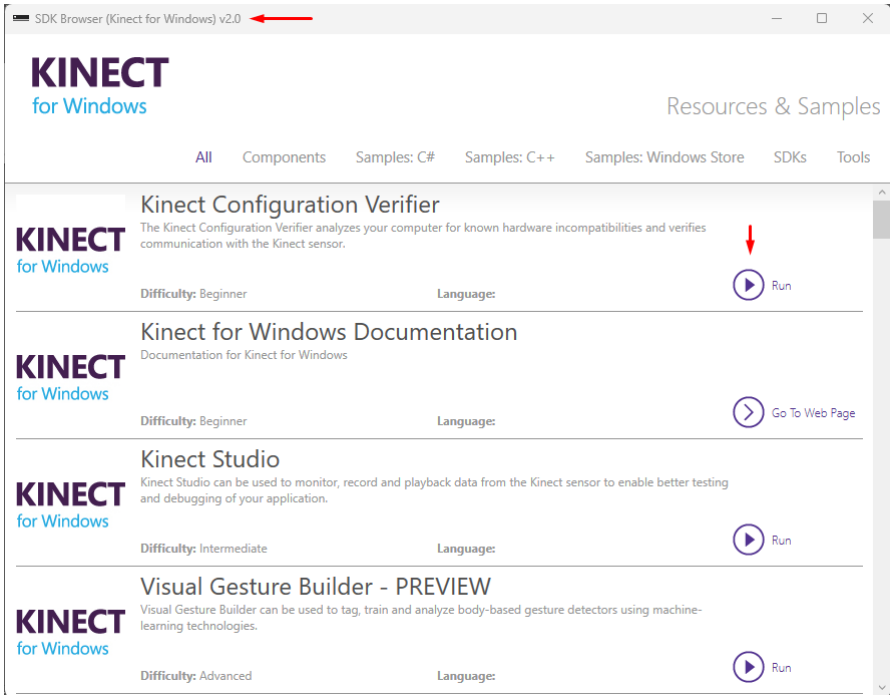


Ilustración 4. Verificación de funciones Kinect

Anexo 7. MANUAL DE INSTRUCTOR

Introducción

El manual de instructor de la aplicación está diseñado para ayudar a los instructores a utilizar el sistema de monitoreo de correcciones de posturas en las dos variantes de ejercicios de hombros, levantamientos laterales y Press militar. El sistema permite a los instructores supervisar en tiempo real la ejecución de las repeticiones de los ejercicios y brindar un seguimiento más a detalle. De igual manera, el instructor, tiene acceso a una base de datos en MySQL para verificar los usuarios que han utilizado el sistema, los mismos que generan notificaciones de alerta cuando realizan cierta cantidad de repeticiones de manera incorrecta, al finalizar las rutinas de entrenamiento el instructor tiene acceso a reportes PDF, los cuales permiten verificar el nivel de error ejecutado por medio de los ángulos calculados en la posición inicial y final del ejercicio. Además, el instructor recibirá una notificación de alerta vía Telegram cuando el usuario active un umbral de repeticiones ejecutadas incorrectamente.

Propósito

El propósito de este manual es proporcionar una guía detallada sobre el uso de las herramientas y funcionalidades disponibles para el monitoreo y corrección de posturas de los clientes del gimnasio. Esto incluye acceder y manejar la base de datos de MySQL, monitorear la disponibilidad del espacio de entrenamiento junto con el dispositivo Kinect y la supervisión en tiempo real de las repeticiones de los ejercicios.

Requerimientos

Acceso a una computadora o dispositivo móvil con acceso a internet.

Instalación y credenciales de acceso a la base de datos MySQL

Cuenta y credenciales de acceso a Ubidots.

Aplicación de Telegram instalada y configurada con el sistema.

Manual de Uso

1. Ingrese a la base de datos en MySQL ingresando las credenciales de acceso proporcionadas.

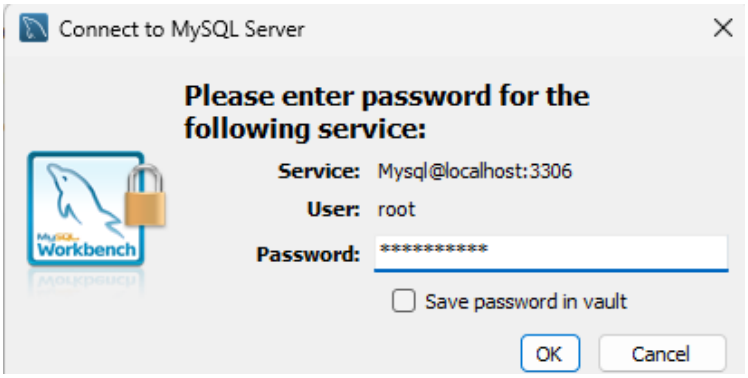


Ilustración 5. Ingreso a base de datos

Dentro de la base de datos diríjase a la base de datos "sistema_usuarios" e ingrese a la imagen de la tabla de usuarios.

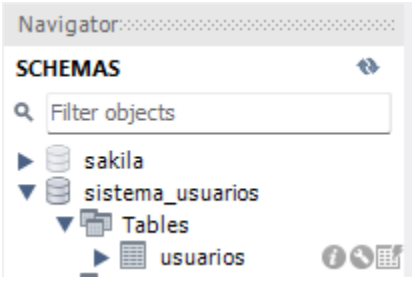


Ilustración 6. Tabla usuarios

Mediante el acceso a la base de datos el instructor tiene acceso a verificar los usuarios que utilizan el sistema, los usuarios registrados podrán generar alertas y reportes.

2. El seguimiento en tiempo real en Ubidots inicia con el ingreso a la página web y se da click en INICIAR SESION.

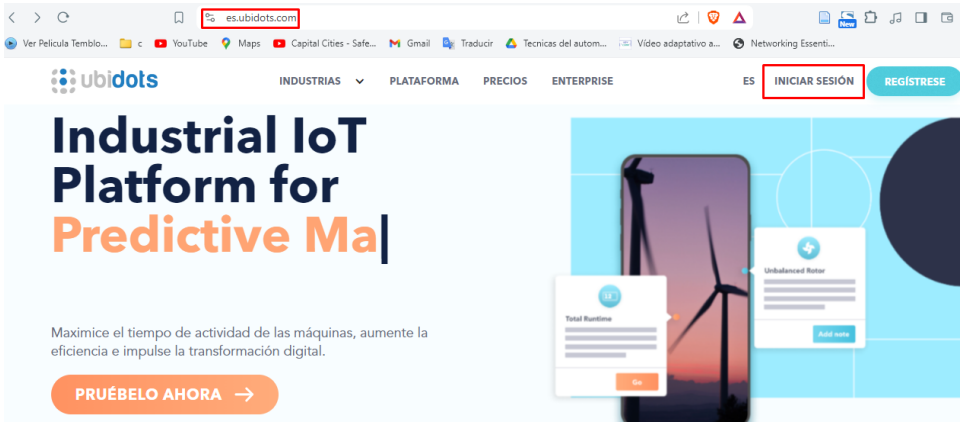


Ilustración 7. Página web Ubitos

El instructor ingresa con las credenciales de usuario y contraseña.

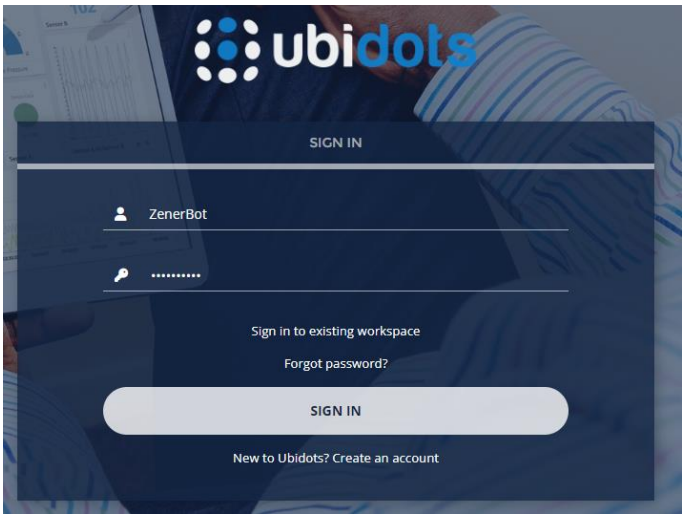


Ilustración 8. Inicio de sesión Ubidots

En la sección de Datos, se ingresa a la opcion de Tableros y el instructor tiene acceso al monitoreo en tiempo real. En esta ventana se visualiza la disponibilidad de espacio de entrenamiento y el número de repeticiones ejecutadas que se actualizan en tiempo real.

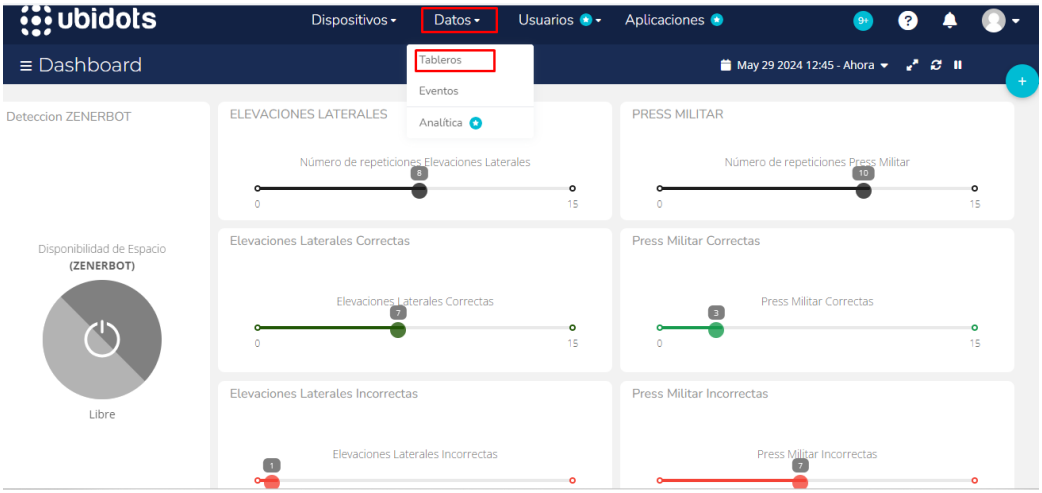


Ilustración 9. Tableros de monitoreo de entrenamiento

El instructor supervisa las repeticiones ejecutadas por los usuarios y puede utilizar los datos para intervenir en el caso que vea necesario.

3. La gestión de alertas en Telegram dirigidas al instructor se enviarán automáticamente cuando las repeticiones ejecutadas de manera incorrecta por parte del usuario superen al 50% del total de las realizadas. Para lo cual se debe tener instalada la aplicación de Telegram y configurada el chatbot.

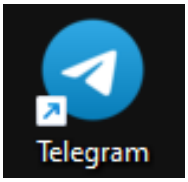


Ilustración 10. Aplicación de Telegram

Al recibir la notificación de alerta el instructor abre el chat "ZENERBOT" y verifica las notificaciones de alertas. Las alertas se conforman por el nombre del usuario, cantidad de repeticiones, y el tipo de ejercicio ejecutado.

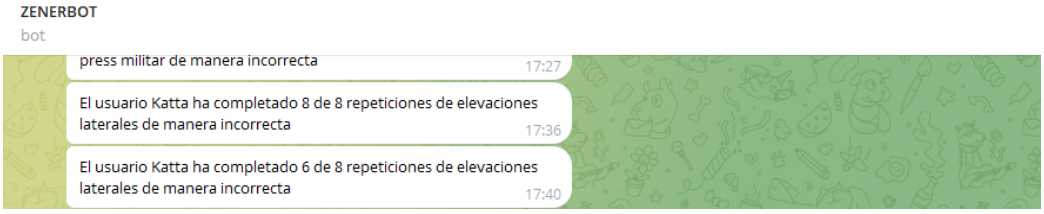


Ilustración 11. Notificaciones de alertas generadas por el usuario

Con la información de las alertas el instructor puede dar una explicación adicional del ejercicio ejecutado, evitando lesiones y estrés en los usuarios.

4. Los reportes PDF se generan cuando el usuario hace click en la opción de reporte PDF, estos reportes se guardan en la carpeta del escritorio del ordenador, para lo cual el instructor debe dirigirse a la carpeta designada y abrir los diferentes reportes.

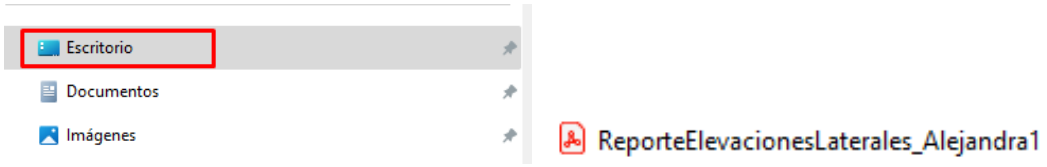


Ilustración 12. Acceso a reporte

La revisión de reportes se inicia abriendo el archivo, en el cual el instructor puede verificar las tablas que muestran los valores de los ángulos de las posturas iniciales y finales de cada repetición ejecutada.

Estado de repetición en Posicion Inicial

Rango correcto 25° - 35°

Número	Posición	Ángulo de brazo derecho	Ángulo de brazo Izquierdo	Estado de Posición
1	Posicion Inicial	33°	31°	Correcto
2	Posicion Inicial	32°	29°	Correcto
3	Posicion Inicial	32°	30°	Correcto
4	Posicion Inicial	32°	31°	Correcto
5	Posicion Inicial	31°	31°	Correcto
6	Posicion Inicial	32°	30°	Correcto
7	Posicion Inicial	31°	31°	Correcto
8	Posicion Inicial	33°	30°	Correcto

Estado de Repetición en Posición Final

Rango correcto 5° - 11°

Número	Posición	Ángulo brazo derecho	Ángulo brazo izquierdo	Estado de Posición
1	Posicion Final	15°	14°	Incorrecto
2	Posicion Final	8°	12°	Incorrecto
3	Posicion Final	12°	20°	Incorrecto
4	Posicion Final	13°	20°	Incorrecto
5	Posicion Final	18°	19°	Incorrecto
6	Posicion Final	12°	16°	Incorrecto
7	Posicion Final	14°	20°	Incorrecto
8	Posicion Final	7°	20°	Incorrecto

Ilustración 13. Revisión de reportes

La información de las tablas permite que el instructor analice los valores de los ángulos capturados por la aplicación y de un seguimiento adecuado del avance y correcciones de los ejercicios ejecutados por los usuarios.

Anexo 8. MANUAL DE USUARIO

Introducción

ZENERBOT es un sistema de monitoreo en tiempo real de posturas corporales para los ejercicios de levantamientos laterales y Press militar aplicado al gimnasio ZENERGYM.

El sistema cuenta con un dispositivo Kinect v2, el cual es el encargado de detectar a la persona ubicada a 2.5 metros frontalmente al dispositivo. El sistema cuenta con varias ventanas interactivas que permiten tener acceso al sistema. La detección de las posturas se basa en las posturas de las posiciones de ejecución de los ejercicios, las posiciones detectadas por el sistema son la posición inicial y la posición final, las cuales permiten detectar si se realizaron de forma correcta o incorrecta. Toda repetición ejecutada de forma incorrecta es detallada por medio de tablas de seguimiento de entrenamiento, las cuales permiten dar un entrenamiento más satisfactorio para el usuario.

Propósito

El propósito de este manual es proporcionar una guía completa sobre el uso del sistema ZENERBOT. Se cubren los requerimientos del sistema, se describen cada una de las ventanas de usuario y se proporcionan instrucciones detalladas para la ejecución de los ejercicios. Además, se incluyen tablas de control que detallan las repeticiones ejecutadas incorrectamente, con el fin de mejorar la precisión en repeticiones posteriores.

Requerimientos

Kinect v2

Aplicación ZENERBOT

Conexión a internet

Manual de Uso

1. La ejecución del sistema para el usuario inicia al abrir la aplicación, en la cual se mostrará la ventana principal para el registro de usuario o ingreso al menú de entrenamiento por medio de las credenciales de usuario y contraseña.

Los botones iniciales permiten al usuario, prender o apagar el dispositivo Kinect, salir del sistema, crear un nuevo usuario, e iniciar el sistema.

Ilustración 14. Ventana inicial del sistema

2. Al ingresar al sistema se abre una ventana del menú de entrenamiento, con guías de entrenamiento y el entrenamiento en tiempo real.

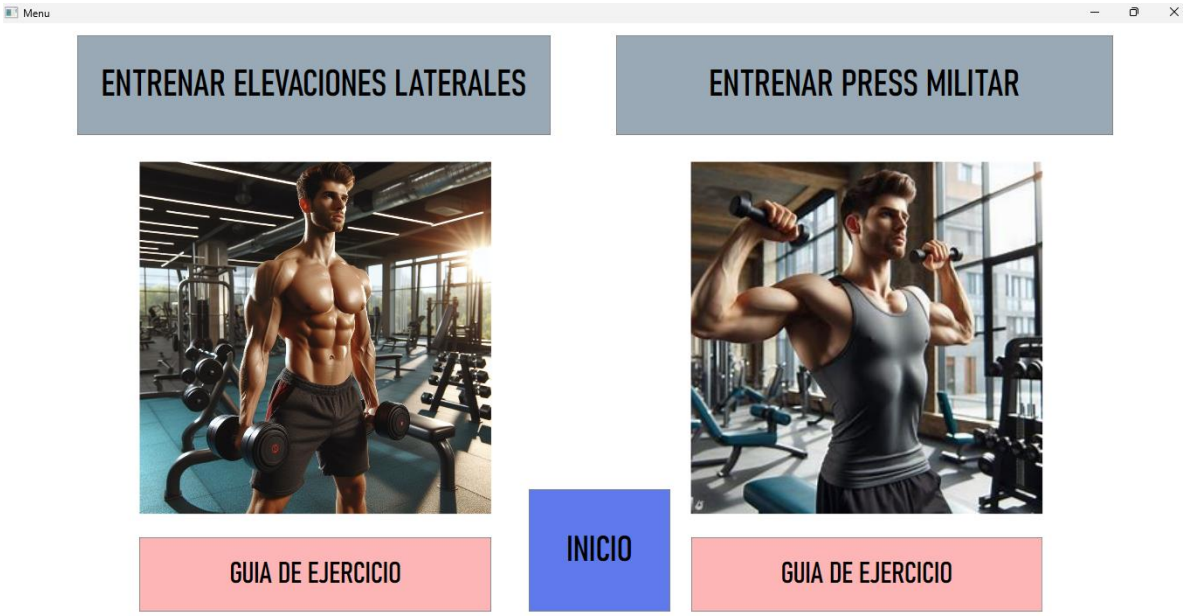


Ilustración 15. Ventana menú del sistema

3. El usuario accede a las guías de entrenamiento, en la cual se muestra las instrucciones detalladas de la ejecución de cada ejercicio.

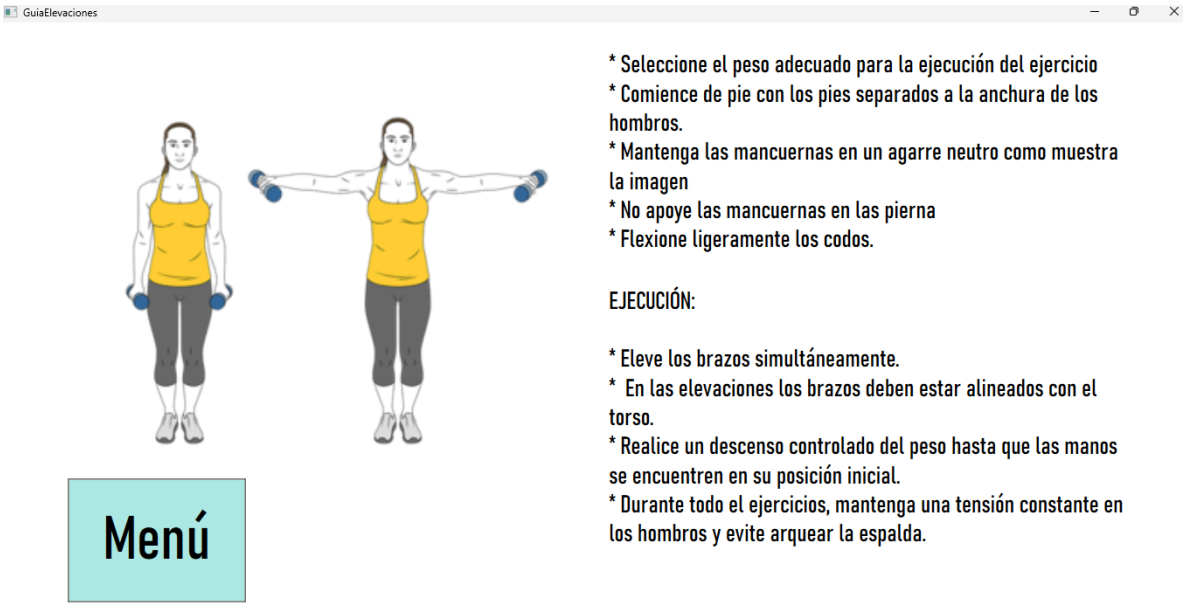


Ilustración 16. Ventana guía levantamientos laterales

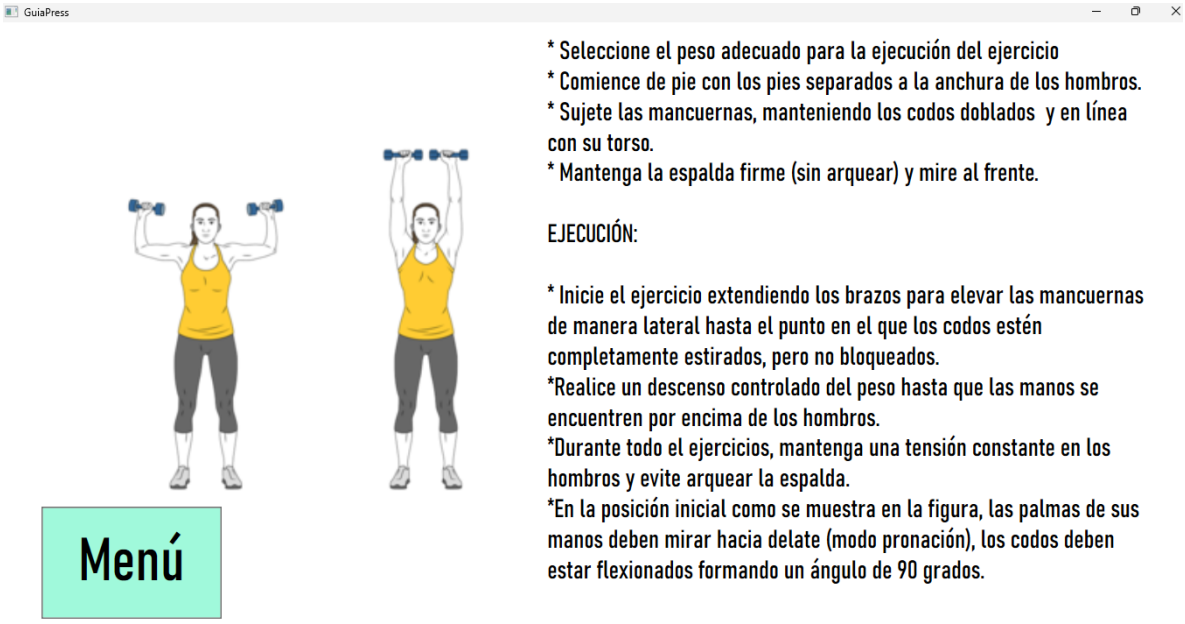


Ilustración 17. Ventana de guía press militar

4. Para el monitoreo en tiempo real, el usuario debe ubicarse a una distancia de 2.5m del dispositivo Kinect.

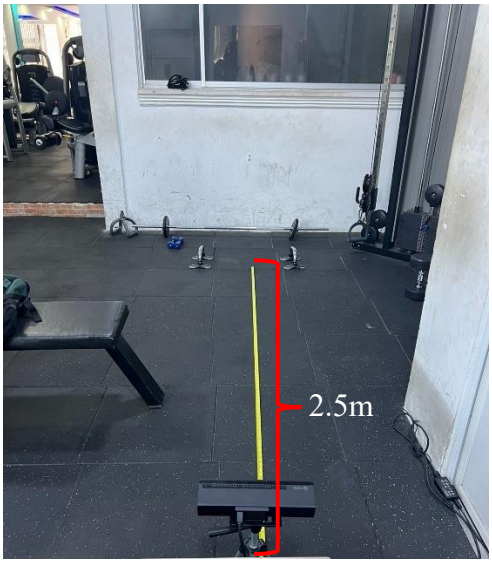


Ilustración 18. Distancia entre Kinect y usuario

En las pantallas de entrenamiento el usuario tiene el acceso a elegir la cantidad de repeticiones a realizar

s

Ilustración 19. Cantidad de repeticiones a realizar

Seleccionada la cantidad de repeticiones a realizar, el usuario debe presionar en el botón iniciar y la detección del esqueleto se activará.

Ilustración 20. Seguimiento de esqueleto

Bajo la detección del cuerpo los indicadores de las posturas de los brazos se activarán, el usuario debe seguir las indicaciones del sistema.

Ilustración 21. Indicadores de posturas levantamientos laterales

Ilustración 22. Indicadores de posturas press militar

Paralelo a este proceso el usuario puede verificar la cantidad de repeticiones faltantes por completar y el estado de repetición ejecutada entre correctas o incorrectas.

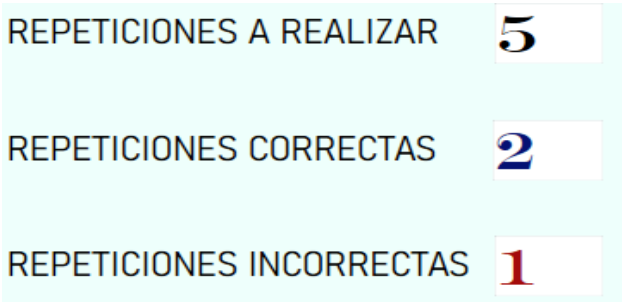


Ilustración 23. Monitoreo de repeticiones

5. Para acceder a las tablas de control, el usuario debe ingresar en la opción de control durante la serie de repeticiones o al finalizarla. Esto le permitirá verificar los errores cometidos y corregirlos en las nuevas series de entrenamiento.

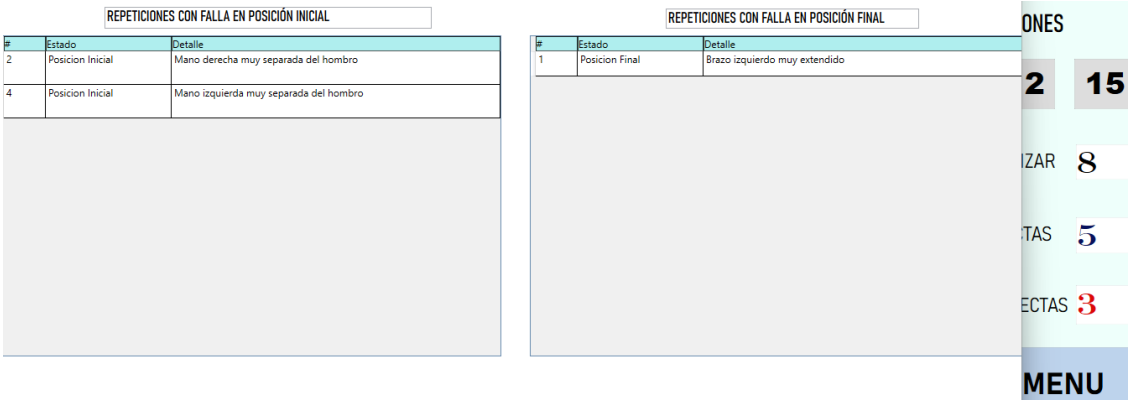


Ilustración 24. Control de repeticiones incorrectas

6. El acceso a los reportes PDF se genera cuando el usuario hace clic en la opción "Generar reporte". A través del reporte, el usuario puede verificar el rango de ángulos necesarios para ejecutar una repetición correctamente.

Ilustración 25. Archivo de reporte

Anexo 9. PLAN DE CONTINGENCIA

El plan de contingencia del sistema de corrección de posturas es de suma importancia para minimizar el impacto negativo de interrupción inesperada por alguna falla del sistema, asegurando que el sistema pueda continuar operando de manera efectiva.

Identificación de fallos en el sistema

Fallo del dispositivo Kinect: Problemas técnicos que impiden el inicio y operación del dispositivo Kinect.

Fallo del sistema de software: Errores en la funcionalidad de la aplicación de corrección de posturas corporales.

Interrupciones de la red: Pérdida de conectividad a internet del sistema que impide el monitoreo en la nube y el envío de alertas dirigidas al instructor.

Fuente de energía: Interrupción de la fuente de energía que afecta el encendido del sistema.

Respuesta de fallos en el sistema

Fallo del dispositivo Kinect:

Cuando el dispositivo Kinect presenta alguna falla, su LED de color blanco no se activará, indicando la existencia de un problema. En este caso, se debe revisar las conexiones y el funcionamiento mediante el SDK de Kinect. Además, es necesario verificar en la opción de Administrador de dispositivos que el controlador del dispositivo Kinect esté instalado correctamente y, posteriormente, comprobar nuevamente el funcionamiento del dispositivo.

Fallo del sistema de software:

Cuando se presenten fallos en el sistema de software, es importante reiniciar la aplicación y verificar que el dispositivo Kinect se pueda activar mediante los botones de la ventana de inicio. Si los errores persisten, se debe notificar a la persona encargada del soporte técnico del sistema.

Interrupciones de la red:

Cuando existan problemas con el monitoreo en tiempo real de los entrenamientos de los usuarios o con el envío de alertas automáticas, se debe verificar la conexión de red del ordenador en el que se está utilizando la aplicación de corrección de posturas, el ordenador debe estar conectado a una red con acceso a internet.

Fuente de energía:

Cuando el sistema no active las funciones del dispositivo Kinect, se debe verificar las conexiones del adaptador de voltaje y del adaptador Kinect. Si el dispositivo Kinect solo está conectado al ordenador y no a la fuente de energía principal, se encenderá un LED rojo en el adaptador Kinect. Al conectar la fuente de energía principal, se debería encender un LED verde, indicando que el dispositivo Kinect tiene el suministro de energía correcto.

Si hay fallas en el adaptador de voltaje, se debe utilizar un multímetro para medir la salida de voltaje, verificando que el valor obtenido esté entre 12V y 15V.

Anexo 10. Código de Programación

Anexo 10A. Ventana Principal

```
using System;
using System.Windows;
using System.Windows.Controls;
using Microsoft.Kinect;

namespace Proyecto
{

    public partial class MainWindow : Window
    {
        public string usuarioGuardado = "";
        /// Activacion de Sensor
        private KinectSensor SensorKinect;
        //Variable para seguimiento de cuerpo
        private BodyFrameReader bodyFrameReader;
        //Variables para el telegram
        /// Initializes a new instance of the MainWindow class.
        public MainWindow()
        {
            InitializeComponent();
            Inicializarkinect();
            Closed += MainWindow_Closed;
        }
        private async void Inicializarkinect()
        {
            //Obtenemos la conexion del kinect al sistema
            this.SensorKinect = KinectSensor.GetDefault();
            if (this.SensorKinect != null)
            {
                this.SensorKinect.Open();
                this.bodyFrameReader =
this.SensorKinect.BodyFrameSource.OpenReader();
                //Se abre el sensor kinect para usarlo
                this.DataContext = this;
            }
            else
            {
                MessageBox.Show("No se ha detectado el sensor Kinect. Verifique
la conexión del Kinect.", "Aviso", MessageBoxButton.OK,
MessageBoxImage.Information);
            }
        }
        private void MainWindow_Closed(object sender, EventArgs e)
        {
            if (this.bodyFrameReader != null)
            {
                this.bodyFrameReader.Dispose();
                this.bodyFrameReader = null;
            }
            if (this.SensorKinect != null)
            {
                // this.Sensorkinect.Close();
```

```
                this.SensorKinect = null;
            }
        }
        public void Button_Click(object sender, RoutedEventArgs e)
//Boton INICIO
        {
            string usuario  = txtUser.Text;
            string usuarioGuardado = "";
            string password = txtPass.Text;
            try
            {
                Control ctrl = new Control();
                string respuesta = ctrl.ctrlLogin(usuario, password);
                if (respuesta.Length > 0)
                {
                    MessageBox.Show(respuesta, "Aviso", MessageBoxButton.OK,
MessageBoxImage.Information);
                }
                else
                {
                    AgregarUsuario.CurrentUsername = usuario;
                    Menu menu = new Menu();
//Creador de objeto a Menu
                    this.Close();
                    menu.Show();
                    usuarioGuardado = usuario;
                }

            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }

        private void btnNuevoU_Click(object sender, RoutedEventArgs e)
        {
            Registro registro = new Registro();
            this.Close();
            registro.Show();
        }

        private void TextBox_TextChanged(object sender, TextChangedEventArgs e)
        {

        }

        private void Button_Click_1(object sender, RoutedEventArgs e)
//boton salir
        {
                SensorKinect.Close();
//Cierra el sensor Kinect
                this.Close();
        }

        private void Button_Click_2(object sender, RoutedEventArgs e)
//Boton prender Kinect
        {
```

```
            if (SensorKinect.IsAvailable)
            {
                this.SensorKinect.Open();
                MessageBox.Show("Sensor Kinect prendido", "Aviso",
MessageBoxButton.OK, MessageBoxImage.Information);
            }
            else
            {
                MessageBox.Show("Verifique la conexion del Kinect", "Aviso",
MessageBoxButton.OK, MessageBoxImage.Information);
            }
        }

        private void Button_Click_3(object sender, RoutedEventArgs e)
//Boton Apagar Kinect
        {
            if (SensorKinect.IsAvailable)
            {
                this.SensorKinect.Close();
                MessageBox.Show("Sensor Kinect apagado", "Aviso",
MessageBoxButton.OK, MessageBoxImage.Information);
            }
            else
            {
                MessageBox.Show("Verifique la conexion del Kinect", "Aviso",
MessageBoxButton.OK, MessageBoxImage.Information);
            }
        }
        private void txtPass_TextChanged(object sender, TextChangedEventArgs e)
        {

        }
    }
}
```

Anexo 10B. Menú de entrenamiento

```
using System;
using System.Timers;
using System.Windows;
using Microsoft.Kinect;
using Ubidots;

namespace Proyecto
//Espacio de lineas para organizar clases y contenedores
{
    public partial class Menu : Window
//Se define una clase menu como parte de la clase principal Window
    {
        /// Activacion de Sensor
        private KinectSensor sensor;
//Se declara una variable privada para el sensor Kinect
        private BodyFrameReader bodyFrameReader;
//variable para la lectura de los marcos del cuerpo
        //Variables y campos para la conexion Ubidots disponibilidad de espacio
        string apiKey = "BBUS-41a2f5af076c246639f66d4be1b76e40ea9";
        string variableId = "65dce502842f44000d0a13ae";
        ApiClient api;
```

```
        Variable exampleVariable;
        Timer timer;
        public Menu()
        {
            //Creacion de tres metodos para inicializar componentes, kinect y
el cursor de la mano
            InitializeComponent();
            InicializarKinect();
            InicializarTimer();
            Closed += Menu_Closed;

        }

        private void InicializarTimer()
// Método para inicializar el temporizador
        {
            timer = new Timer(30000);
//Creaci[on de instancia para 30 segundos de temporizador
            timer.Elapsed += Timer_Elapsed;
//Se suscribe el método Timer_Elapsed al evento Elapsed del temporizador
            timer.AutoReset = true;
//Propiedad para reiniciar automaticamente el temporizador
            timer.Start();
//Inicia el temporizador
        }

        private void Timer_Elapsed(object sender, ElapsedEventArgs e)
//DEclaracion de metodo, envio de valor ubidots
        {
            EnviarValorUbidots();
//Llama al metodo enviarvalorubidots
        }

        private void InicializarKinect()
        {
            // Inicializar sensor Kinect
            this.sensor = KinectSensor.GetDefault();
//Se obtiene la instancia predeterminada del sensor
            if (this.sensor != null)
            {
                this.sensor.Open();
//Se abre el sensor
                this.bodyFrameReader =
this.sensor.BodyFrameSource.OpenReader();               //Se iniciliza el
lector de marcos de cuerpO
            }
            // Se inicializa el cliente de API de Ubidots utilizando la clave
apikey
            api = new ApiClient(apiKey);
            // Se btiene una referencia a la variable específica en Ubidots
utilizando su variableID
            exampleVariable = api.GetVariable(variableId);
        }

        private void EnviarValorUbidots()
        {
            // Obtener el valor actual de Kinect y enviarlo a Ubidots
            using (BodyFrame bodyFrame = bodyFrameReader.AcquireLatestFrame())
```

```
            {
                if (bodyFrame != null)
                {
                    Body[] bodies = new Body[bodyFrame.BodyCount];
                    bodyFrame.GetAndRefreshBodyData(bodies);
                    foreach (var body in bodies)
                    {
                        if (body.IsTracked)
                        {
                            // Si se detecta un cuerpo, enviar un 1 a Ubidots
                            exampleVariable.SaveValue(1);
                            Console.WriteLine($"Valor 1 enviado a la variable
{exampleVariable.GetName()}");
                            return; // Sale del bucle al detectar una persona
                        }
                    }

                    // Si no se detecta ningún cuerpo, enviar un 0 a Ubidots
                    exampleVariable.SaveValue(0);
                    Console.WriteLine($"Valor 0 enviado a la variable
{exampleVariable.GetName()}");
                }
            }
        }
        private void Menu_Closed(object sender, EventArgs e)
//Método para cerrar el menu
        {
            if (this.bodyFrameReader != null)
//Se verifica si el lector de frames de cuerpo no es nulo
            {
                this.bodyFrameReader.Dispose();
//Se libera los recursos utilizados por el lector de frames de cuerpo
                this.bodyFrameReader = null;
//Se establece el lector de frames de cuerpo a nulo para indicar no está en uso
            }
            if (this.sensor != null)
// Verifica si el sensor Kinect no es nulo
            {
                // this.kinect.Close();
                this.sensor = null;
            }
        }
        private void Button_Click(object sender, RoutedEventArgs e)
//Metodo para el boton de guia Press
        {
            GuiaPress guiaPress = new GuiaPress();
//Creacion de instancia de la ventana GuiaPress
            this.Close();
//La ventana actual se cierra
            guiaPress.Show();
//Se muestra la ventana de guia Press
        }

        private void Button_Click_1(object sender, RoutedEventArgs e)
//Metodo para el boton de entre. press militar
        {
            PressMilitar pressMilitar = new PressMilitar();
// Instancia de la ventana PressMilitar
```

```
            this.Close();
//La ventana actual se cierra
            pressMilitar.Show();
//Se muestra la ventana de entre. Press Militar
        }
        private void Button_Click_2(object sender, RoutedEventArgs e)
//Metodo para el boton de menu principal
        {
            MainWindow main = new MainWindow();
//Instancia de la ventana de menu principal
            this.Close ();
//La ventana actual se cierra
            main.Show ();
//Se muestra la ventana principal
        }
        private void Button_Click_3(object sender, RoutedEventArgs e)
//Metodo para el boton guia Levantamientos laterales
        {
            GuiaElevaciones guiaElevaciones = new GuiaElevaciones();
//Instancia de la ventana guia levantamientos
            this.Close ();
//La ventana actual se cierra
            guiaElevaciones.Show();
//Se muestra la ventana guia levantamientos

        }
        private void btn_Entrenar_EL_Click(object sender, RoutedEventArgs e)
//Metodo para el boton entre. elevaciones laterales
        {
            ElevacionesLaterales elevacionesLaterales = new
ElevacionesLaterales(); //Instancia de la ventana elevaciones laterales
            this.Close ();
//Se cierra la ventana actual
            elevacionesLaterales.Show();
//Se muestra la ventana de entren. elevaciones laterales
        }
    }
}
```

Anexo 10C. Guía Levantamientos Laterales

```
using Microsoft.Kinect;
using System;
using System.Timers;
using System.Windows;
using Ubidots;

namespace Proyecto
{
    public partial class GuiaElevaciones : Window
    {
        private KinectSensor kinect;
        private BodyFrameReader bodyFrameReader;
//Creacion de variable tipo lectura de frame de cuerpo
        //Variables y campos para la conexion Ubidots disponibilidad de espacio
        string apiKey = "BBUS-41a2f5af076c246639f66d4be1b76e40ea9";
```

```
        string variableId = "65dce502842f44000d0a13ae";
        ApiClient api;
        Variable exampleVariable;
        Timer timer;


        public GuiaElevaciones()                //Se crea el constructor de la
clase GuiaElevaciones
        {
            InitializeComponent();              //Se inicializa los componentes
de la interfaz de usuarios
            Inicializarkinect();                //Se llama al metodo
InicializarKinect.
            InicializarTimer();                 //Se llama al metodo del
temporizador
            Closed += GuiaElevaciones_Closed;   // Se suscribe al evento Closed
para ejecutar el método cuando la ventana se cierra
        }

        private void InicializarTimer()
        {
            // Inicializar el temporizador para enviar valores cada 30 segundos
            timer = new Timer(30000); // 30,000 milisegundos = 30 segundos
            timer.Elapsed += Timer_Elapsed;
            timer.AutoReset = true;
            timer.Start();
        }
        private void Timer_Elapsed(object sender, ElapsedEventArgs e)
        {
            // Al dispararse el temporizador, enviar el valor actual a Ubidots
            EnviarValorUbidots();
        }
        private void Inicializarkinect()
        {
            // Inicializar sensor Kinect
            this.kinect = KinectSensor.GetDefault();
//Se obtiene la instancia predeterminada del sensor
            if (this.kinect != null)
//Se verifica si el sensor Kinect no es nulo
            {
                kinect.Open();
//Se abre el sensor
                this.bodyFrameReader =
this.kinect.BodyFrameSource.OpenReader(); // Se inicializa el lector de frames
de cuerpo
            }
            // Inicializa el cliente de API de Ubidots utilizando la clave
apiKey
            api = new ApiClient(apiKey);
            // Se btiene una referencia a la variable específica en Ubidots
utilizando su variableID
            exampleVariable = api.GetVariable(variableId);
        }
        private void EnviarValorUbidots()
        {
            // Obtener el valor actual de Kinect y enviarlo a Ubidots
            using (BodyFrame bodyFrame = bodyFrameReader.AcquireLatestFrame())
            {
```

```
                if (bodyFrame != null)
                {
                    Body[] bodies = new Body[bodyFrame.BodyCount];
                    bodyFrame.GetAndRefreshBodyData(bodies);
                    foreach (var body in bodies)
                    {
                        if (body.IsTracked)
                        {
                            // Si se detecta un cuerpo, enviar un 1 a Ubidots
                            exampleVariable.SaveValue(1);
                            return; // Sale del bucle al detectar una persona
                        }
                    }

                    // Si no se detecta ningún cuerpo, enviar un 0 a Ubidots
                    exampleVariable.SaveValue(0);
                }
            }
        }
        // Método que se ejecuta para cerrar la ventana
        private void GuiaElevaciones_Closed(object sender, EventArgs e)
        {
            if (this.bodyFrameReader != null)      //Verificacion si el lector
de frames del cuerpo es nulo
            {
                this.bodyFrameReader.Dispose();   // Se libera los recursos
utilizados por el lector de frames de cuerpo
                this.bodyFrameReader = null;      // Se establece el lector de
frames de cuerpo a nulo, indicando que no se esta utilizando
            }
            if (this.kinect != null)              //Se verifica si el sensor
Kinect es nulo
            {
                // this.Sensorkinect.Close();
                this.kinect = null;               // Se establece al sensor
kinect como nulo, indicando que no se esta utilizando
            }
        }
        private void Button_Click(object sender, RoutedEventArgs e)  //Metodo
del boton regresar a menu
        {
            Menu menu = new Menu();
//Instancia de la ventana Menu
            this.Close();                                             //Se
cierra la ventana actual
            menu.Show();                                              //Se
muestra la ventana de menu
        }
        private void Window_Loaded(object sender, RoutedEventArgs e)
        {
        }
    }
}
```

Anexo 10D. Guía Press Militar.

```
using System;
using System.Windows;
using System.Timers;
using Microsoft.Kinect;
using Ubidots;

namespace Proyecto
{
    public partial class GuiaPress : Window
    {
        private KinectSensor kinect;
        private BodyFrameReader bodyFrameReader;                    //Creacion
de variable tipo lectura de frame de cuerpo
        //Variables y campos para la conexion Ubidots disponibilidad de espacio
        string apiKey = "BBUS-41a2f5af076c246639f66d4be1b76e40ea9";
        string variableId = "65dce502842f44000d0a13ae";
        ApiClient api;
        Variable exampleVariable;
        Timer timer;
        public GuiaPress()                    //Se crea el constructor de la
clase GuiaElevaciones
        {
            InitializeComponent();           //Se inicializa los componentes de
la interfaz de usuarios
            Inicializarkinect();             //Se llama al metodo
InicializarKinect.
            InicializarTimer();              //Se llama al metodo del
temporizador
            Closed += GuiaPress_Closed;      //Se suscribe al evento Closed
para ejecutar el método cuando la ventana se cierra
        }

        private void InicializarTimer()     // Se crea el método para
inicializar el temporizador
        {
            // Inicializar el temporizador para enviar valores cada 30 segundos
            timer = new Timer(30000);       //Se crea una nueva instancia del
temporizador con el intervalo de 30s
            timer.Elapsed += Timer_Elapsed; //se suscribe el evento Elapsed del
temporizador al método Timer_Elapsed
            timer.AutoReset = true;         //Se configura el temporizador para
que se reinicie automáticamente depsues de cada 30s
            timer.Start();                  //Inicia el temporizador
        }
        // Método que se ejecuta cuando el temporizador alcanza el tiempo
especifico
        private void Timer_Elapsed(object sender, ElapsedEventArgs e)
        {
            // Llama al método EnviarValorres para enviar el dato a ubidots
            EnviarValorUbidots();
        }

        //Metodo para inicial kinect
        private void Inicializarkinect()
        {
```

```
            this.kinect = KinectSensor.GetDefault();
//Se obtiene la instancia predeterminada del sensor
            if (this.kinect != null)
//Se verifica si el sensor Kinect no es nulo
            {
                kinect.Open();
//Se abre el sensor
                this.bodyFrameReader =
this.kinect.BodyFrameSource.OpenReader();  // Se inicializa el lector de frames
de cuerpo
            }
            api = new ApiClient(apiKey); // Se nicializa el cliente de API de
Ubidots utilizando la clave apiKey
            exampleVariable = api.GetVariable(variableId); // Se btiene una
referencia a la variable específica en Ubidots utilizando su variableID
        }

        private void EnviarValorUbidots()
//Metodo para enviar el valor a ubidots
        {
            using (BodyFrame bodyFrame = bodyFrameReader.AcquireLatestFrame())
//Se obtiene el último frame de cuerpocapturado por el sensor
            {
                if (bodyFrame != null)                                  // Se
verifica si el frame de cuerpo no es nulo
                {
                    Body [] bodies = new Body[bodyFrame.BodyCount]; //Se crea
un arreglo para almacenar los datos de los cuerpos detectados en el frame
                    bodyFrame.GetAndRefreshBodyData(bodies);        //Se
actualiza el arreglo con los datos de los cuerpos detectados en el frame
                    foreach (Body body in bodies)                   //Recorre a
través de cada cuerpo detectado en el frame
                    {
                        if (body.IsTracked)                         //Se
verifica si el cuerpo está siendo rastreado
                        {
                            exampleVariable.SaveValue(1);          // Si se
detecta un cuerpo envia un 1 a Ubidots
                            return;
                        }
                    }
                    exampleVariable.SaveValue(0);                  // Si no se
detecta ningún cuerpo envia un 0 a Ubidots
                }
            }
        }
        // Método que se ejecuta para cerrar la ventana
        private void GuiaPress_Closed(object sender, EventArgs e)
        {
            if (this.bodyFrameReader != null)
//Verificacion si el lector de frames del cuerpo es nulo
            {
                this.bodyFrameReader.Dispose();                    // Se libera
los recursos utilizados por el lector de frames de cuerpo
                this.bodyFrameReader = null;                       // Se
establece el lector de frames de cuerpo a nulo, indicando que no se esta
utilizando
            }
```

```
            if (this.kinect != null)                                //Se verifica
si el sensor Kinect es nulo
            {
                //this.kinect.Close();
                this.kinect = null;                                 // Se
establece al sensor kinect como nulo, indicando que no se esta utilizando
            }
        }
        private void Button_Click(object sender, RoutedEventArgs e)  //Metodo
del boton regresar a menu
        {
            Menu menu = new Menu();
//Instancia de la ventana Menu
            this.Close();                                                //Se
cierra la ventana actual
            menu.Show();                                                 //Se
muestra la ventana de menu
        }
    }
}
```

Anexo 10E. Entrenamiento Levantamientos Laterales

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Media;
using System.Timers;
using System.Windows.Media.Imaging;
using Microsoft.Kinect;
using Ubidots;
using System.Windows.Media.Media3D;
using Telegram.Bot;
using System.Collections.ObjectModel;
using static Proyecto.ResultadosIncorrectos;              //Declaracion de
directiva para acceder a las propiedades de ResultadosIncorrectos
using iTextSharp.text.pdf;
using iTextSharp.text;
using System.IO;
using Image = iTextSharp.text.Image;                     //Declaracion de alias
para el uso de Image
using System.Linq;                                       //Creacion de
directiva para manejo de las listas.
using System.Windows.Threading;
using System.Media;
namespace Proyecto
{
    public partial class ElevacionesLaterales : Window
    {
        private bool activacionCuerpo = false;              //Variable
descativacion de cuerpo
        //Creacion de propiedad para almacenar colecciones tabla control
        public ObservableCollection<ControlRepeticionesI>
ControlRepeticionesIncorrectasI { get; set; } = new
ObservableCollection<ControlRepeticionesI>();
```

```
        public ObservableCollection<ControlRepeticionesF>
ControlRepeticionesIncorrectasF { get; set; } = new
ObservableCollection<ControlRepeticionesF>();
        //Creacion de propiedad para almacenar colecciones pdf
        public ObservableCollection<ControlReportePosicionInicial>
ControlRepeticionesIniciales { get; set; } = new
ObservableCollection<ControlReportePosicionInicial>();
        public ObservableCollection<ControlReportePosicionFinal>
ControlRepeticionesFinales { get; set; } = new
ObservableCollection<ControlReportePosicionFinal>();

        string nombreUsuario = AgregarUsuario.CurrentUsername;
        //Variables para el seguimiento de esqueleto
//Radio de dibujo de circulos en las manos
        public const double JointThickness = 3;
//Grosor de las líneas de unión dibujadas
        public const double ClipBoundsThickness = 10;
//Grosor de los rectángulos del borde del clip
        public const float InferredZPositionClamp = 0.1f;
//Constante para evitar que los valores Z de los puntos del espacio de la
cámara sean negativo
        public readonly Brush trackedJointBrush = new
SolidColorBrush(Color.FromArgb(255, 68, 192, 68)); //Pincel utilizado para
dibujar juntas que actualmente están rastreadas
        public readonly Brush inferredJointBrush = Brushes.Yellow;
//Pincel utilizado para dibujar juntas que se infieren actualmente.
        public readonly Pen inferredBonePen = new Pen(Brushes.Gray, 1);
//Bolígrafo utilizado para dibujar huesos que actualmente se infieren.
        public DrawingGroup drawingGroup;
//Grupo de dibujo para salida de renderizado corporal
        public DrawingImage imageSource;
//Imagen de dibujo que mostraremos.
        public CoordinateMapper coordinateMapper = null;
//Mapeador de coordenadas para asignar un tipo de punto a otro
        public BodyFrameReader bodyFrameReader = null;
//Lector de estructuras corporales
        public Body[] bodies = null;
//Creacion de array para guardar los cuerpos>
        public List<Tuple<JointType, JointType>> bones;
//Definicion de huesos
        public int displayWidth;
//Ancho de visualización (espacio de profundidad)
        public int displayHeight;
//Altura de la pantalla (espacio de profundidad)
        public List<Pen> bodyColors;
//Lista de colores para el seguimiento de cuerpo

        //Creacion de campos
        public KinectSensor kinect;
// Campo que almacena una referencia a un objeto KinectSensor
        public ColorFrameReader LecturaFrameColor;
// Campo que almacena una referencia a un objeto LecturaFrameColor
        public WriteableBitmap MapaBits;
// Campo que almacena una referencia a un objeto MapaBits

        //Variables y campos para la conexion Ubidots disponibilidad de espacio
        string apiKey = "BBUS-41a2f5af076c246639f66d4be1b76e40ea9";
        string variableDisponibilidadId = "65dce502842f44000d0a13ae";
```

```
        ApiClient api;
        Variable variableMonitoreo;
        Timer timer;

        //Variable y campos ubidots elevaciones laterales
        string variableIDCantidadRepeticiones = "6626f44eebfad903d6091080";
        Variable rElevaciones;
        string variableIDElevacionesC = "66157844c4e6c45c7287448a";
        Variable rCorrectasElevaciones;
        string variableIDElevacionesI = "6615d158e15972088231a60f";
        Variable rIncorrectasElevaciones;

        //Campo para el telegram
        private readonly TelegramBotClient botClient;
//Inicializacion de cliente bot


        //Variables control de angulos
        // Ángulos para la posición inicial y final
        private const int AnguloInicialMin = 20;                    // Angulo
minimo aceptable en posición inicial
        private const int AnguloInicialMax = 40;                    // Angulo
maximo aceptable en posición inicial
        private const int AnguloFinalMin = 1;                       // Angulo
minimo aceptable en posición final
        private const int AnguloFinalMax = 20;                      // Angulo
maximo aceptable en posición final

        //Angulos de posiciones correctas
        private const int AnguloInicialCorrectoMin = 25;            // Angulo
minimo aceptable posicion inicial
        private const int AnguloInicialCorrectoMax = 35;            // Angulo
maximo aceptable posicion inicial
        private const int AnguloFinalCorrectoMin = 5;               // Angulo
minimo aceptable posicion final
        private const int AnguloFinalCorrectoMax = 11;              // Angulo
maximo aceptable posicion inicial
        //Angulos de posiciones incorrectas
        private const int AnguloInicialBajoIncorrectoMin = 20;
//Separado al torso
        private const int AnguloInicialBajoIncorrectoMax = 24;
//Separado al torso
        private const int AnguloInicialAltoIncorrectoMin = 36;        //Junto
al torso
        private const int AnguloInicialAltoIncorrectoMax = 40;       //Junto
al torso

        private const int AnguloFinalBajoIncorrectoMin = 1;
//Posicion muy baja de lo brazos
        private const int AnguloFinalBajoIncorrectoMax = 4;
//Posicion muy baja de los brazos
        private const int AnguloFinalAltoIncorrectoMin = 12;
//Posicion muy alta de los brazos
        private const int AnguloFinalAltoIncorrectoMax = 20;
//Posicion muy alta de los brazos

        //Variables para angulos calculados
        private int anguloDerecho;
```

```
        private int anguloIzquierdo;
        private int anguloDerechoC;
        private int anguloIzquierdoC;
        private int anguloDerechoI;
        private int anguloIzquierdoI;

        //Variables para alamcenar los valores del calculo de la moda
        private int modaAnguloDerechoIC;
        private int modaAnguloIzquierdoIC;
        private int modaAnguloDerechoII;
        private int modaAnguloIzquierdoII;
        private int modaAnguloDerechoFC;
        private int modaAnguloIzquierdoFC;
        private int modaAnguloDerechoFI;
        private int modaAnguloIzquierdoFI;


        //Variables para validar los ejercicios de forma correcta e incorrecta
        private int totalRepeticionesRealizadas = 0;
        private int repeticionesRestantes;
//Un valor predeterminado
        private int valorDefecto;
        private int repeticionesCorrectas;
        private int repeticionesIncorrectas;
        public int correctas;
        public int incorrectasI;
        public int incorrectasF;
        //Contador para detalles de errores
        private int numeroTotalRepeticiones = 1;
        //Contador para el nombre del pdf
        private static int contadorReporte = 1;
        //string
        string estadoEvaluacionI = ""; // Estado por defecto
        string estadoEvaluacionF = ""; //Estado por defecto

        enum EstadoEjercicio           //Declaración de enumeraciones
        {
            EnPosicionInicial,
            EnPosicionFinal
        }
        private EstadoEjercicio estadoEjercicio =
EstadoEjercicio.EnPosicionInicial;    //Inicio de estado en posicion inicial
        private int contadorFramesI;
        private int contadorFramesF;
        private int contadorAnguloInicialCorrectos;
        private int contadorAnguloFinalCorrectos;
        private int contadorAnguloInicialIncorrectos;
        private int contadorAnguloFinalIncorrectos;
        private const int FramesParaTresSegundos = 90;

        //

        public ElevacionesLaterales()
//Creacion de constructor de esta clase
        {
            InitializeComponent();
//Metodo para iniciar componentes de IU
```

```
            InicializarKinect();
//Llama al metodo para iniciar el sensor kinect
            InicializarTimer();
//Llama al metodo para iniciar un time de ubidots
            Closed += ElevacionesLaterales_Closed;
//Se asocia el metodo elevacionesLaterales al evento Closed
            ActualizarContadoresUI();
//Llama a un metodo para actualizar los contadores en la IU
            botClient = new TelegramBotClient("6722450347:AAFR9q-
qn_eJ38U9yk4pHx6UNnSC7_lTIGM");                  //Inicializa un cliente de bot
de Telegram con el token
// Inicializa el time
        }
        private void InicializarTimer()
//Metodo para inicializar temporizador
        {
            timer = new Timer(30000);
//Creaci[on de instancia para 30 segundos de temporizador
            timer.Elapsed += Timer_Elapsed;
//Se suscribe el método Timer_Elapsed al evento Elapsed del temporizador
            timer.AutoReset = true;
//Propiedad para reiniciar automaticamente el temporizador
            timer.Start();
//Inicia el temporizador
        }
        public void Timer_Elapsed(object sender, ElapsedEventArgs e)
//DEclaracion de metodo
        {
            EnviarValorUbidots();
//Llama al metodo enviarvalorubidots
        }

        public void InicializarKinect()
        {
            this.kinect = KinectSensor.GetDefault();
//Inicializa el sensor kinect
            this.kinect.Open();
//Abre el sensor kinect
            this.LecturaFrameColor = this.kinect.ColorFrameSource.OpenReader();
//Inicialización del lector de frames de color
            this.LecturaFrameColor.FrameArrived +=
this.LecturaFrameColor_FrameArrived; //Suscripción al evento FrameArrived
            FrameDescription colorFrameDescription =
this.kinect.ColorFrameSource.CreateFrameDescription(ColorImageFormat.Bgra);
//Se obtiene una descripcion del frame de color
            this.MapaBits = new WriteableBitmap(colorFrameDescription.Width,
colorFrameDescription.Height, 96.0, 96.0, PixelFormats.Bgr32, null); //Se
inicializa WriteableBitMap
            this.bodyFrameReader = this.kinect.BodyFrameSource.OpenReader();
//Lectura de los marcos del cuerpo
            this.DataContext = this;
            //Seguimiento de esqueleto
            this.coordinateMapper = this.kinect.CoordinateMapper;
//Se obtiene el mapeador de coordenadas
            // get the depth (display) extents
            FrameDescription frameDescription =
this.kinect.DepthFrameSource.FrameDescription;              //Se obtiene las
extensiones de profundidad
```

```
            // Obtiene el tamaño del espacio articular alto y ancho
            this.displayWidth = frameDescription.Width;
            this.displayHeight = frameDescription.Height;
            //Se define un hueso como una linea entre dos articulaciones
            this.bones = new List<Tuple<JointType, JointType>>();
            //Se crea una lista de huesos
            // Torso
            this.bones.Add(new Tuple<JointType, JointType>(JointType.Head,
JointType.Neck));
            this.bones.Add(new Tuple<JointType, JointType>(JointType.Neck,
JointType.SpineShoulder));
            this.bones.Add(new Tuple<JointType,
JointType>(JointType.SpineShoulder, JointType.SpineMid));
            this.bones.Add(new Tuple<JointType, JointType>(JointType.SpineMid,
JointType.SpineBase));
            this.bones.Add(new Tuple<JointType,
JointType>(JointType.SpineShoulder, JointType.ShoulderRight));
            this.bones.Add(new Tuple<JointType,
JointType>(JointType.SpineShoulder, JointType.ShoulderLeft));
            this.bones.Add(new Tuple<JointType, JointType>(JointType.SpineBase,
JointType.HipRight));
            this.bones.Add(new Tuple<JointType, JointType>(JointType.SpineBase,
JointType.HipLeft));

            // Brazo derecho
            this.bones.Add(new Tuple<JointType,
JointType>(JointType.ShoulderRight, JointType.ElbowRight));
            this.bones.Add(new Tuple<JointType,
JointType>(JointType.ElbowRight, JointType.WristRight));
            this.bones.Add(new Tuple<JointType,
JointType>(JointType.WristRight, JointType.HandRight));
            this.bones.Add(new Tuple<JointType, JointType>(JointType.HandRight,
JointType.HandTipRight));
            this.bones.Add(new Tuple<JointType,
JointType>(JointType.WristRight, JointType.ThumbRight));

            // Brazo izquierdo
            this.bones.Add(new Tuple<JointType,
JointType>(JointType.ShoulderLeft, JointType.ElbowLeft));
            this.bones.Add(new Tuple<JointType, JointType>(JointType.ElbowLeft,
JointType.WristLeft));
            this.bones.Add(new Tuple<JointType, JointType>(JointType.WristLeft,
JointType.HandLeft));
            this.bones.Add(new Tuple<JointType, JointType>(JointType.HandLeft,
JointType.HandTipLeft));
            this.bones.Add(new Tuple<JointType, JointType>(JointType.WristLeft,
JointType.ThumbLeft));

            // Pierna derecha
            this.bones.Add(new Tuple<JointType, JointType>(JointType.HipRight,
JointType.KneeRight));
            this.bones.Add(new Tuple<JointType, JointType>(JointType.KneeRight,
JointType.AnkleRight));
            this.bones.Add(new Tuple<JointType,
JointType>(JointType.AnkleRight, JointType.FootRight));

            // Pierna izquierda
```

```
            this.bones.Add(new Tuple<JointType, JointType>(JointType.HipLeft,
JointType.KneeLeft));
            this.bones.Add(new Tuple<JointType, JointType>(JointType.KneeLeft,
JointType.AnkleLeft));
            this.bones.Add(new Tuple<JointType, JointType>(JointType.AnkleLeft,
JointType.FootLeft));

            //Colores para los cuerpos
            this.bodyColors = new List<Pen>();
            this.bodyColors.Add(new Pen(Brushes.Red, 4));
            this.bodyColors.Add(new Pen(Brushes.Red, 4));
            this.bodyColors.Add(new Pen(Brushes.Red, 4));
            this.bodyColors.Add(new Pen(Brushes.Red, 4));
            this.bodyColors.Add(new Pen(Brushes.Red, 4));
            this.bodyColors.Add(new Pen(Brushes.Red, 4));

            // Crea el grupo de dibujo que se usa para dibujar.
            this.drawingGroup = new DrawingGroup();

            // Crea una fuente de imagen que se usa en el control de imagen
            this.imageSource = new DrawingImage(this.drawingGroup);
            this.Loaded += ElevacionesLaterales_Loaded;

            //Ubidots
            api = new ApiClient(apiKey);
// Inicializa el cliente de API de Ubidots.
            variableMonitoreo = api.GetVariable(variableDisponibilidadId);
// Obtiene la inf de la variable de monitoreo en Ubidots
            rCorrectasElevaciones = api.GetVariable(variableIDElevacionesC);
// Obtiene la variable que almacena el número de repeticiones correctas
            rIncorrectasElevaciones = api.GetVariable(variableIDElevacionesI);
// Obtiene la variable que almacena el número de repeticiones incorrectas
            rElevaciones = api.GetVariable(variableIDCantidadRepeticiones);
// Obtiene la referencia a la variable que almacena el número total de
repeticiones
        }

        public void EnviarValorUbidots()
//Metodo para enviar el valor a ubidots
        {
            // Obtener el valor actual de Kinect y enviarlo a Ubidots
            using (BodyFrame bodyFrame = bodyFrameReader.AcquireLatestFrame())
//Se obtiene el último frame de cuerpocapturado por el sensor
            {
                if (bodyFrame != null)
// Se verifica si el frame de cuerpo no es nulo
                {
                    Body[] bodies = new Body[bodyFrame.BodyCount];
//Se crea un arreglo para almacenar los datos de los cuerpos detectados en el
frame
                    bodyFrame.GetAndRefreshBodyData(bodies);
//Se actualiza el arreglo con los datos de los cuerpos detectados en el frame
                    foreach (var body in bodies)
//Recorre a través de cada cuerpo detectado en el frame
                    {
                        if (body.IsTracked)
//Se verifica si el cuerpo está siendo rastreado
                        {
```

```
                            // Si se detecta un cuerpo envia un 1 a Ubidots
                            variableMonitoreo.SaveValue(1);
                            return; // Sale del ciclo al detectar una persona
                        }
                    }
                    // Si no se detecta ningún cuerpo envia un 0 a Ubidots
                    variableMonitoreo.SaveValue(0);
                }
            }
        }
        public ImageSource ImageSource1                         //Declaracion de
propiedad
        {
            get
            {
                return this.MapaBits;                          //Permite acceder
a los frames a color y actualizar con los movimientos
            }
        }
        public ImageSource ImageSource                          //Declaracion de
propiedad
        {
            get
            {
                return this.imageSource;                      //Permite acceder
a los frames a color y actualizar con los movimientos
            }
        }

        public void ElevacionesLaterales_Loaded(object sender, RoutedEventArgs
e)   //Inicio de metodo
        {
            if (this.bodyFrameReader != null)
//Verifica que la entrada de frames no sea nula
            {
                bodyFrameReader.FrameArrived += BodyFrameReader_FrameArrived;
// Suscribe el método BodyFrameReader_FrameArrived al evento FrameArrived del
lector de frames de cuerpo
            }
        }

        public void ElevacionesLaterales_Closed(object sender, EventArgs e)
// Método que se llama cuando se cierra la ventana.
        {
            if (this.LecturaFrameColor != null & this.bodyFrameReader != null)
// Verifica si los lectores de frames de color y cuerpo no son nulos
            {
                // Llama a Dispose para liberar los recursos del lector de
frames de cuerpo.
                this.bodyFrameReader.Dispose();
                this.bodyFrameReader = null;
                // Llama a Dispose para liberar los recursos del lector de
frames de color
                this.LecturaFrameColor.Dispose();
                this.LecturaFrameColor = null;
            }
            if (this.kinect != null)             // Verifica si el sensor
Kinect no es nulo
```

```
            {
                // this.Sensorkinect.Close();
                this.kinect = null;              // Establece el sensor Kinect
a null
            }
        }
        public void LecturaFrameColor_FrameArrived(object sender,
ColorFrameArrivedEventArgs e)                      //Metodo para frames de color
        {
            using (ColorFrame colorFrame = e.FrameReference.AcquireFrame())
//Se adquiere el frame mas reciente
            {
                if (colorFrame != null)
//Condicion cuando se obtiene un frame
                {
                    FrameDescription colorFrameDescription =
colorFrame.FrameDescription;  //Se obtiene la descripcion del frame
                    using (KinectBuffer colorBuffer =
colorFrame.LockRawImageBuffer()) ;   //bloqueo de bufer cuando se esta
procesando un frame
                    {
                        this.MapaBits.Lock();
//Bloqueo de bitmapa
                        if ((colorFrameDescription.Width ==
this.MapaBits.PixelWidth) && (colorFrameDescription.Height ==
this.MapaBits.PixelHeight)) //Verificacion de dimesiones del frame
                        {
                            //Copia de datos al bitmapa
                            colorFrame.CopyConvertedFrameDataToIntPtr(
//Llamado de metodo para copiar los datos de la imagen al bufer
                                        this.MapaBits.BackBuffer,
//Buffer para dibujar los datos de imagen
                                        (uint)(colorFrameDescription.Width *
colorFrameDescription.Height * 4), //Cantidad de datos a copiar
                                        ColorImageFormat.Bgra);
//Se especifica el formato de color a usar
                            this.MapaBits.AddDirtyRect(new Int32Rect(0, 0,
this.MapaBits.PixelWidth, this.MapaBits.PixelHeight)); //Actualizacion de
interfaz al usuario
                        }
                        this.MapaBits.Unlock();
//Desbloqueo de bitmapa
                    }
                }
            }
        }
        public static int Moda(List<int> values)
// Método para calcular la moda de la lista de angulos
        {
            return values.GroupBy(v => v)
// Agrupa los valores de la lista por su valor
                         .OrderByDescending(g => g.Count())
// Ordena los grupos en orden descendente por el número de elementos en cada
grupo
                         .First()
// Toma el primer grupo (el que tiene más elementos)
                         .Key;
// Devuelve el valor que más se repite en la lista
```

```
        }

        public async void BodyFrameReader_FrameArrived(object sender,
BodyFrameArrivedEventArgs e)                            //Metodo para lectura de
frames de color
        {
            if (!activacionCuerpo)
//Verificacion para activacion de lectura de datos.
                return;
            bool datosrecibidos = false;
// Indica si se han recibido datos de cuerpo
            using (BodyFrame bodyFrame = e.FrameReference.AcquireFrame())
// Se adquiere el frame de cuerpo más reciente
            {
                if (bodyFrame != null)
// Se comprueba si el frame de cuerpo adquirido no es nulo
                {
                    if (this.bodies == null)
// Inicializa el array de cuerpos si es nulo
                    {
                        this.bodies = new Body[bodyFrame.BodyCount];
                    }
                    //Se llena el array de cuerpos con los datos del frame de
cuerpo
                    bodyFrame.GetAndRefreshBodyData(this.bodies);
                    datosrecibidos = true;
                }
            }
            if (datosrecibidos)
//Si lectura de frames de cuerpo está activa
            {
                using (DrawingContext dc = drawingGroup.Open())
//Creacion de bloque para dibujar cada cuerpo
                {
                    dc.DrawRectangle(Brushes.Transparent, null, new Rect(0.0,
0.0, this.displayWidth, this.displayHeight));   //Dibujar el rectangulo en la
interfaz
                    int colorLapiz = 0;
                    foreach (Body body in this.bodies)
//Iteracion de bucle para cada cuerpo
                    {
                        Pen dibujarEsfero = this.bodyColors[colorLapiz++];
//Recorrido de colores para dibujar cada cuerpo
                        if (body.IsTracked)
//Condicion si el cuerpo es detectado
                        {
                            //Calculo de angulos para reporte
                            // Calcula los ángulos
                            this.DrawClippedEdges(body, dc);
//Ajusta bordes del cuerpo para establecer los limites de deteccion
                            IReadOnlyDictionary<JointType, Joint> joints =
body.Joints;                                    //Declaracion de variable
"joints" para acceder a los valores de las articulaciones
                            Dictionary<JointType, Point> jointPoints = new
Dictionary<JointType, Point>();                 //Declaracion de diccionario
para acceder a los valores de las articulaciones
```

```
                            //Coversion de las articulaciones de un espacio de
3D a 2D para visualizacion del usuario
                            foreach (JointType jointType in joints.Keys)
//Iteracion de cada articulacion en el diccionario
                            {
                                CameraSpacePoint position =
joints[jointType].Position;                                         //Variable
para representar una posicion tridimencional
                                if (position.Z < 0)
//Verificacion que el valor de la componente z menor a 0
                                {
                                    position.Z = InferredZPositionClamp;
//Ajuste de valor predeterminado, para que el valor z no sea confiable
                                }

                                DepthSpacePoint depthSpacePoint =
this.coordinateMapper.MapCameraPointToDepthSpace(position); //Estructura para
representar la posicion en tridimensional en bidimensional
                                jointPoints[jointType] = new
Point(depthSpacePoint.X, depthSpacePoint.Y);
//Almacenamiento de las posiciones convertidas y poder accerde mediante Point
                            }
                            this.DibujarCuerpo(joints, jointPoints, dc,
dibujarEsfero);

//************************************************************Procesamiento
validacion de ejercicio y tipo de
repeticiones*************************************************************
**//

                            Joint SpineShoulder =
body.Joints[JointType.SpineShoulder];     //Obetener articulaciones del
SpineShoulder
                            Joint hombroDerecho =
body.Joints[JointType.ShoulderRight];     //Obtener las articulaciones del
hombro derecho
                            Joint codoDerecho =
body.Joints[JointType.ElbowRight];          //Obtener las articulaciones del
hombro codo derecho
                            Joint hombroIzquierdo =
body.Joints[JointType.ShoulderLeft];    //Obtener las articulaciones del hombro
izquierdo
                            Joint codoIzquierdo =
body.Joints[JointType.ElbowLeft];         //Obtener las articulaciones del
hombro codo izquierdo

                            // Imprimir las coordenadas de SpineShoulder
                            Console.WriteLine($"SpineShoulder:
X={SpineShoulder.Position.X}, Y={SpineShoulder.Position.Y},
Z={SpineShoulder.Position.Z}");
                            // Imprimir las coordenadas del hombro derecho
                            Console.WriteLine($"HombroDerecho:
X={hombroDerecho.Position.X}, Y={hombroDerecho.Position.Y},
Z={hombroDerecho.Position.Z}");
                            // Imprimir las coordenadas del codo derecho
                            Console.WriteLine($"CodoDerecho:
X={codoDerecho.Position.X}, Y={codoDerecho.Position.Y},
Z={codoDerecho.Position.Z}");
```

```
                            // Imprimir las coordenadas del hombro izquierdo
                            Console.WriteLine($"HombroIzquierdo:
X={hombroIzquierdo.Position.X}, Y={hombroIzquierdo.Position.Y},
Z={hombroIzquierdo.Position.Z}");
                            // Imprimir las coordenadas del codo izquierdo
                            Console.WriteLine($"CodoIzquierdo:
X={codoIzquierdo.Position.X}, Y={codoIzquierdo.Position.Y},
Z={codoIzquierdo.Position.Z}");

                            // Calcula el ángulo
                            anguloDerecho =
CalcularAngulo(SpineShoulder.Position, hombroDerecho.Position,
codoDerecho.Position);
                            anguloIzquierdo =
CalcularAngulo(SpineShoulder.Position, hombroIzquierdo.Position,
codoIzquierdo.Position);
                            // Imprime el ángulo calculado en la consola
                            Console.WriteLine($"Angulo Derecho calculado:
{anguloDerecho} grados");
                            Console.WriteLine($"Angulo Izquierdo calculado:
{anguloIzquierdo} grados");

                            //Estados de Ejercicios
                            //Posicion Inicial
                            bool posicionInicialCorrecta =
EstaEnRango(anguloDerecho, AnguloInicialCorrectoMin, AnguloInicialCorrectoMax)
&&                  //Variable para verificar la posicion inicial correcta

EstaEnRango(anguloIzquierdo, AnguloInicialCorrectoMin,
AnguloInicialCorrectoMax);

                            bool posicionInicialIncorrecta =
EstaEnRango(anguloDerecho, AnguloInicialBajoIncorrectoMin,
AnguloInicialBajoIncorrectoMax) ||  //Variable para verificar la posicion
inicial incorrecta

EstaEnRango(anguloDerecho, AnguloInicialAltoIncorrectoMin,
AnguloInicialAltoIncorrectoMax) ||

EstaEnRango(anguloIzquierdo, AnguloInicialBajoIncorrectoMin,
AnguloInicialBajoIncorrectoMax) ||

EstaEnRango(anguloIzquierdo, AnguloInicialAltoIncorrectoMin,
AnguloInicialAltoIncorrectoMax);
                            //Posicion Final

                            bool posicionFinalCorrecta =
EstaEnRango(anguloDerecho, AnguloFinalCorrectoMin, AnguloFinalCorrectoMax) &&
//Variable para verificar la posicion final correcta

EstaEnRango(anguloIzquierdo, AnguloFinalCorrectoMin, AnguloFinalCorrectoMax);

                            bool posicionFinalIncorrecta =
EstaEnRango(anguloDerecho, AnguloFinalBajoIncorrectoMin,
AnguloFinalBajoIncorrectoMax) ||       //Variable para verificar la posicion
final incorrecta
```

```
EstaEnRango(anguloDerecho, AnguloFinalAltoIncorrectoMin,
AnguloFinalAltoIncorrectoMax) ||

EstaEnRango(anguloIzquierdo, AnguloFinalBajoIncorrectoMin,
AnguloFinalBajoIncorrectoMax) ||

EstaEnRango(anguloIzquierdo, AnguloFinalAltoIncorrectoMin,
AnguloFinalAltoIncorrectoMax);

                            //controlador de angulos
                            // Proceso para contar una repetición
                            if (estadoEjercicio ==
EstadoEjercicio.EnPosicionInicial && EstaEnRango(anguloDerecho,
AnguloInicialMin, AnguloInicialMax) && //Condición para validar posicion
inicial
                                EstaEnRango(anguloIzquierdo, AnguloInicialMin,
AnguloInicialMax))
                            {
                                contadorFramesF = 0;
//Resetea el contador de frames al cambiar de estado
                                contadorAnguloFinalIncorrectos = 0;
//Reseta los contadores de posicion final
                                contadorAnguloFinalCorrectos = 0;
//Resetea los contadores de posicion final
                                if (contadorFramesI == 0)
//Condición para reiniciar los valores de ángulos y moda calculados.
                                {
                                    modaAnguloDerechoIC = 0;
                                    modaAnguloIzquierdoIC = 0;
                                    modaAnguloDerechoII = 0;
                                    modaAnguloIzquierdoII = 0;
                                    anguloDerechoC = 0;
                                    anguloIzquierdoC = 0;
                                    anguloDerechoI = 0;
                                    anguloIzquierdoI = 0;
                                    var ultimoControlInicial =
ControlRepeticionesIniciales.LastOrDefault();   //Se obtiene el último control
de repeticiones iniciales de la colección
                                    if (ultimoControlInicial != null)
//Se verifica si se encontró un último control de repeticiones iniciales
                                    {

ultimoControlInicial.AngulosDerechosIC.Clear();                         //Se
limpia la colección de ángulos derechos IC

ultimoControlInicial.AngulosIzquierdosIC.Clear();                       // Se
limpia la colección de ángulos izquierdos IC

ultimoControlInicial.AngulosDerechosII.Clear();                         // Se
limpia la colección de ángulos derechos II

ultimoControlInicial.AngulosIzquierdosII.Clear();                       // Se
limpia la colección de ángulos izquierdos II
                                    }
                                    var nuevaPosicionInicial = new
ControlReportePosicionInicial               // Se crea una nueva instancia de
ControlReportePosicionInicial e inicializa sus propiedades
```

```
                                    {
                                        NumeroRepeticionT =
numeroTotalRepeticiones,                          // Se establece el número
total de repeticiones
                                        EstadoPosicionInicial = "Posicion
Inicial",                            // Se establece el estado de la posicion
                                        EstadoEvaluacionI = "",
// Se establece el estado de evaluacion por defecto
                                        ModaAnguloDerechoIC = 0,
// Se reinicia el valor de moda
                                        ModaAnguloIzquierdoIC = 0,
// Se reinicia el valor de moda
                                        ModaAnguloDerechoII = 0,
// Se reinicia el valor de moda
                                        ModaAnguloIzquierdoII = 0,
// Se reinicia el valor de moda
                                    };
                                    Dispatcher.Invoke(() =>
                                    {

ControlRepeticionesIniciales.Add(nuevaPosicionInicial);
                                    });
                                }
                                contadorFramesI++;
//Aumento de contador de Frames
                                if (contadorFramesI >=1 & contadorFramesI <=
90)                                     //Condicion para contar 90 frames
                                {
                                    if (posicionInicialCorrecta)
//condición de posicion correcta
                                    {
                                        // Calcula el ángulo
                                        anguloDerechoC =
CalcularAngulo(SpineShoulder.Position, hombroDerecho.Position,
codoDerecho.Position); //Calcula angulos correctos
                                        anguloIzquierdoC =
CalcularAngulo(SpineShoulder.Position, hombroIzquierdo.Position,
codoIzquierdo.Position);//Calcula angulos incorrectos
                                        contadorAnguloInicialCorrectos++;
//Aumento de contador de ángulos correctos

ControlRepeticionesIniciales.LastOrDefault()?.AngulosDerechosIC.Add(anguloDerec
hoC);    //Agregacion de angulos a la lista

ControlRepeticionesIniciales.LastOrDefault()?.AngulosIzquierdosIC.Add(anguloIzq
uierdoC);//Agragacion de ángulos a la lista
                                    }
                                    //Condicion de posicion incorrecta
                                    if (posicionInicialIncorrecta)
                                    {
                                        // Calcula el ángulo
                                        anguloDerechoI =
CalcularAngulo(SpineShoulder.Position, hombroDerecho.Position,
codoDerecho.Position); //Calcula angulos correctos
                                        anguloIzquierdoI =
CalcularAngulo(SpineShoulder.Position, hombroIzquierdo.Position,
codoIzquierdo.Position);//Calcula angulos incorrectos
```

```
                                        contadorAnguloInicialIncorrectos++;
//Aumento de contador de ángulos incorrectos

ControlRepeticionesIniciales.LastOrDefault()?.AngulosDerechosII.Add(anguloDerec
hoI);         //Agregacion de angulos a la lista

ControlRepeticionesIniciales.LastOrDefault()?.AngulosIzquierdosII.Add(anguloIzq
uierdoI);     //Agregacion de angulos a la lista
                                    }
                                    if (contadorFramesI ==
FramesParaTresSegundos)                                          //Condicion
cuando los frames son igual a 90
                                    {
                                        anguloDerecho = 0;
                                        anguloIzquierdo = 0;
                                        if (contadorAnguloInicialCorrectos >=
46)                                                  //Condición
para activar el contador de posicion inicial correcta
                                        {
                                            modaAnguloDerechoIC =
Moda(ControlRepeticionesIniciales.LastOrDefault().AngulosDerechosIC);
//Cálculo de moda angulosderechosIC
                                            modaAnguloIzquierdoIC =
Moda(ControlRepeticionesIniciales.LastOrDefault().AngulosIzquierdosIC);
//Cálculo de moda angulosizquierdosIC
                                            estadoEvaluacionI = "Correcto";
                                            correctas++;//Aumento de contador
para la posicion correcta
                                            Dispatcher.Invoke(() =>
                                            {   //Actualización de valores en
PDF

ControlRepeticionesIniciales.LastOrDefault().EstadoEvaluacionI =
estadoEvaluacionI;

ControlRepeticionesIniciales.LastOrDefault().ModaAnguloDerechoIC =
modaAnguloDerechoIC;

ControlRepeticionesIniciales.LastOrDefault().ModaAnguloIzquierdoIC =
modaAnguloIzquierdoIC;
                                            });
                                        }
                                        if (contadorAnguloInicialIncorrectos >
44)//Condición para activar posicion incorrecta
                                        {
                                            modaAnguloDerechoII =
Moda(ControlRepeticionesIniciales.LastOrDefault().AngulosDerechosII);
//Cálculo de moda angulosderechosII
                                            modaAnguloIzquierdoII =
Moda(ControlRepeticionesIniciales.LastOrDefault().AngulosIzquierdosII);
//Cálculo de moda angulosizquierdosII
                                            estadoEvaluacionI = "Incorrecto";
                                            incorrectasI++;//Aumento de
contador para la posicion incorrecta
                                            string detalle =
DetalleErrorPosicionInicial(modaAnguloDerechoII, modaAnguloIzquierdoII);
//Creación de variable, almacena el detalle de Pos.I
```

```
                                            var estadoIncorrectoI = new
ControlRepeticionesI                                                    //Creación
de instancia para asignar informacion
                                            {
                                                NumeroRepeticionI =
numeroTotalRepeticiones,                                                    //Se
agrega el valor de repeticion tabla control
                                                EstadoPosicionI = "Posicion
Inicial",                                                  //Se agrega el
valor de posicion tabla control
                                                DetalleAnguloI = detalle
//Se agrga el detalle de posicion tabla control
                                            };
                                            Dispatcher.Invoke(() =>
//Actualización UI
                                            {

ControlRepeticionesIncorrectasI.Add(estadoIncorrectoI);
                                            });
                                            Dispatcher.Invoke(() =>
                                            {

ControlRepeticionesIniciales.LastOrDefault().EstadoEvaluacionI =
estadoEvaluacionI;

ControlRepeticionesIniciales.LastOrDefault().ModaAnguloDerechoII =
modaAnguloDerechoII;

ControlRepeticionesIniciales.LastOrDefault().ModaAnguloIzquierdoII =
modaAnguloIzquierdoII;
                                            });
                                        }//Fin condicional estado inicial
incorrecto
                                        estadoEjercicio =
EstadoEjercicio.EnPosicionFinal; // Cambia el estado
                                        //Aviso cambio de posicion UI
                                        Dispatcher.Invoke(() =>
                                        {
                                            txtIndicacion.Text = "ELEVE LOS
BRAZOS";       //Mensaje indicador
                                            Task.Delay(2000).ContinueWith(t =>
Dispatcher.Invoke(() => txtIndicacion.Text = ""));  //Se indica el mensaje
durante dos segundos
                                        });
                                    }
                                    Console.WriteLine($"*****POSICION
INICIAL*****");
                                    Console.WriteLine($"Contador de Frames
Posicion Inicial: {contadorFramesI}");
                                    Console.WriteLine($"Contador de angulos
iniciales Correctos: {contadorAnguloInicialCorrectos}");
                                    Console.WriteLine($"Contador de angulos
iniciales Incorrectos: {contadorAnguloInicialIncorrectos}");
                                    Console.WriteLine($"Angulo DERECHO
CORRECTO: {anguloDerechoC}");
                                    Console.WriteLine($"Angulo DERECHO
INCORRECTO: {anguloDerechoI}");
```

```
                                    Console.WriteLine($"Angulo IZQUIERDO
CORRECTO: {anguloIzquierdoC}");
                                    Console.WriteLine($"Angulo IZQUIERDO
INCORRECTO: {anguloIzquierdoI}");
                                    Console.WriteLine($"MODA ANGULO INCIAL
CORRECTO DERECHO: {modaAnguloDerechoIC}");
                                    Console.WriteLine($"MODA ANGULO INCIAL
INCORRECTO DERECHO: {modaAnguloDerechoII}");
                                    Console.WriteLine($"MODA ANGULO INCIAL
CORRECTO IZQUIERDO: {modaAnguloIzquierdoIC}");
                                    Console.WriteLine($"MODA ANGULO INCIAL
INCORRECTO IZQUIERDO: {modaAnguloIzquierdoII}");
                                    Console.WriteLine("Lista de ángulos
derechos II:");
                                    var listaAngulosDerechosII =
ControlRepeticionesIniciales.LastOrDefault()?.AngulosDerechosII;
                                    foreach (var anguloII in
listaAngulosDerechosII)
                                    {
                                        Console.WriteLine(anguloII);
                                    }
                                }
                            }
                            //Inicio estado final
                            if (estadoEjercicio ==
EstadoEjercicio.EnPosicionFinal && EstaEnRango(anguloDerecho, AnguloFinalMin,
AnguloFinalMax) && //Activación de posición final
                                EstaEnRango(anguloIzquierdo, AnguloFinalMin,
AnguloFinalMax))
                            {
                                contadorFramesI = 0;
// Resetea el contador de frames al cambiar de estado
                                contadorAnguloInicialIncorrectos = 0;
//Resetea loa contadores de angulos iniciales
                                contadorAnguloInicialCorrectos = 0;
//Resetea loa contadores de angulos iniciales
                                if (contadorFramesF == 0)
// Solo inicializar en el primer frame
                                {
                                    anguloDerechoC = 0;
//Valor de angulo derecho correcto a 0
                                    anguloIzquierdoC = 0;
//Valor de angulo izquierdo correcto a 0
                                    anguloDerechoI = 0;
//Valor de angulo derecho incorrecto a 0
                                    anguloIzquierdoI = 0;
//Valor de angulo izquierdo incorrecto a 0
                                    var ultimoControlFinal =
ControlRepeticionesFinales.LastOrDefault();   //Variable para limpiar los
valores del pdf
                                    //Actualizacion de valores a 0
                                    if (ultimoControlFinal != null)
                                    {

ultimoControlFinal.AngulosDerechosFC.Clear();

ultimoControlFinal.AngulosIzquierdosFC.Clear();
```

```
ultimoControlFinal.AngulosDerechosFI.Clear();

ultimoControlFinal.AngulosIzquierdosFI.Clear();
                                    }
                                    //Actualizacion de listas a valores por
defecto
                                    var nuevaPosicionFinal = new
ControlReportePosicionFinal
                                    {
                                        NumeroRepeticionT =
numeroTotalRepeticiones,
                                        EstadoPosicionFinal = "Posicion Final",
                                        EstadoEvaluacionF = "",
                                        ModaAnguloDerechoFC = 0,
                                        ModaAnguloIzquierdoFC = 0,
                                        ModaAnguloDerechoFI = 0,
                                        ModaAnguloIzquierdoFI = 0
                                    };
                                    //Actualizacion el UI
                                    Dispatcher.Invoke(() =>
                                    {

ControlRepeticionesFinales.Add(nuevaPosicionFinal);
                                    });
                                }
                                contadorFramesF++;
//Aumento de contador de Frames
                                if (contadorFramesF >= 1 && contadorFramesF <=
90)                                     //Recorrido de los 90 frames
                                {
                                    if (posicionFinalCorrecta)
//Condición de posicion correcta
                                    {
                                        anguloDerechoC =
CalcularAngulo(SpineShoulder.Position, hombroDerecho.Position,
codoDerecho.Position);       //Valores para calcular angulo derecho C
                                        anguloIzquierdoC =
CalcularAngulo(SpineShoulder.Position, hombroIzquierdo.Position,
codoIzquierdo.Position); //Valores para calcular angulo izquierdo C
                                        contadorAnguloFinalCorrectos++;
//Aumento de contador de ángulos correctos

ControlRepeticionesFinales.LastOrDefault()?.AngulosDerechosFC.Add(anguloDerecho
C);                          //Agregacion de angulos derechosC a la lista

ControlRepeticionesFinales.LastOrDefault()?.AngulosIzquierdosFC.Add(anguloIzqui
erdoC);                      //Agregacion de angulos izquierdosC a la lista
                                    }
                                    if (posicionFinalIncorrecta)  //Condición
de posición incorrecta para activar contador y calculo de angulos
                                    {
                                        anguloDerechoI =
CalcularAngulo(SpineShoulder.Position, hombroDerecho.Position,
codoDerecho.Position);       //Valores para calcular angulo derecho In
                                        anguloIzquierdoI =
CalcularAngulo(SpineShoulder.Position, hombroIzquierdo.Position,
codoIzquierdo.Position); //Valores para calcular angulo izquierdo In
```

```
                                        contadorAnguloFinalIncorrectos++;
//Aumento de contador de ángulos incorrectos

ControlRepeticionesFinales.LastOrDefault()?.AngulosDerechosFI.Add(anguloDerecho
I);                          //Agregacion de angulos derechosIn a la lista

ControlRepeticionesFinales.LastOrDefault()?.AngulosIzquierdosFI.Add(anguloIzqui
erdoI);                      //Agregacion de angulos izquierdosIn a la lista
                                    }
                                    if (contadorFramesF ==
FramesParaTresSegundos) //Condicion cuando los frames son igual a 90
                                    {
                                        anguloDerecho = 0;
//Se reinicia los valores de angulo derecho
                                        anguloIzquierdo = 0;
//Se reinicia los valores de angulo izquierdo
                                        if (contadorAnguloFinalCorrectos >= 46)
//Condición que ctiva la posicion final como correcta
                                        {
                                            modaAnguloDerechoFC =
Moda(ControlRepeticionesFinales.LastOrDefault().AngulosDerechosFC);
//Calculo de moda de la lista de valores de angulos derechos FC
                                            modaAnguloIzquierdoFC =
Moda(ControlRepeticionesFinales.LastOrDefault().AngulosIzquierdosFC); //Calculo
de moda de la lista de valores de angulos izquierdos FC
                                            estadoEvaluacionF = "Correcto";
//Actualizacion de estado final correto
                                            correctas++;
//Aumento de contador para la posicion final correcta
                                            //Actualizacion de informacion en
el reporte pdf
                                            Dispatcher.Invoke(() =>
                                            {

ControlRepeticionesFinales.LastOrDefault().EstadoEvaluacionF =
estadoEvaluacionF;

ControlRepeticionesFinales.LastOrDefault().ModaAnguloDerechoFC =
modaAnguloDerechoFC;

ControlRepeticionesFinales.LastOrDefault().ModaAnguloIzquierdoFC =
modaAnguloIzquierdoFC;
                                            });
                                        }
                                        if (contadorAnguloFinalIncorrectos >
44)    //Condición que activa la posicion final como incorrecta
                                        {
                                            modaAnguloDerechoFI =
Moda(ControlRepeticionesFinales.LastOrDefault().AngulosDerechosFI);
//Calculo de moda de la lista de valores de angulos derechos FC
                                            modaAnguloIzquierdoFI =
Moda(ControlRepeticionesFinales.LastOrDefault().AngulosIzquierdosFI); //Calculo
de moda de la lista de valores de angulos izquierdos FC
                                            estadoEvaluacionF = "Incorrecto";
//Actualizacion de estado final incorrecto
                                            incorrectasF++;
//Aumento de contador para la posicion final  incorrecta
```

```
                                            string detalle =
DetalleErrorPosicionFinal(modaAnguloDerechoFI, modaAnguloIzquierdoFI);
//Variable para verificar el detalle de la posicion FIn
                                            //Actualizacion de informacion en
la tabla de control
                                            var estadoIncorrectoF = new
ControlRepeticionesF
                                            {
                                                NumeroRepeticionF =
numeroTotalRepeticiones,
                                                EstadoPosicionF = "Posicion
Final",
                                                DetalleAnguloF = detalle
                                            };
                                            // Se agrega la instancia a la
colección que se muestra en la UI control
                                            Dispatcher.Invoke(() =>
                                            {

ControlRepeticionesIncorrectasF.Add(estadoIncorrectoF);

                                            });
                                            // Se agrega la instancia a la
colección que se muestra en la UI reporte
                                            Dispatcher.Invoke(() =>
                                            {

ControlRepeticionesFinales.LastOrDefault().EstadoEvaluacionF =
estadoEvaluacionF;

ControlRepeticionesFinales.LastOrDefault().ModaAnguloDerechoFI =
modaAnguloDerechoFI;

ControlRepeticionesFinales.LastOrDefault().ModaAnguloIzquierdoFI =
modaAnguloIzquierdoFI;
                                                });
                                        }  //Fin condicion incorrecta.
                                        if (correctas == 2)     //Condicion
para validar una repeticion completa como correcta
                                        {
                                            ReproducirSonidoE(true);
                                            repeticionesCorrectas++;  //Se
aumenta el contador de repeticiones correctas

rCorrectasElevaciones.SaveValue(repeticionesCorrectas); //Se guarda el valor en
la plataforma en la nube Ubidots.
                                            correctas = 0;            //al
finalizar los 90 frames el contador para validar una repeticion correcta se
reinicia.
                                        }
                                        //Para validar una repeticion como
incorrecta, se toma en cuenta si una de las posiciones tanto en inicial como
final es incorrecta, toda una repeticion se valida como incorrecta.
                                        if (incorrectasI == 1 || incorrectasF
== 1 || correctas == 1 & incorrectasI == 1 || correctas == 1 & incorrectasF ==
1) //Validacion para una repeticion completa como incorrecta
                                        {
                                            ReproducirSonidoE(false);
```

```
                                            repeticionesIncorrectas++;  //Se
aumenta el contador de repeticiones incorrectas

rIncorrectasElevaciones.SaveValue(repeticionesIncorrectas); //Se guarda el
valor en la platafome en la nube Ubidots
                                            incorrectasF = 0;           //al
finalizar los 90 frames el contador para validar una repeticion incorrectaF se
reinicia.
                                            incorrectasI = 0;           //al
finalizar los 90 frames el contador para validar una repeticion incorrectaI se
reinicia.
                                            correctas = 0;              //al
finalizar los 90 frames el contador para validar una repeticion correcta se
reinicia.
                                        }
                                        numeroTotalRepeticiones++;      //Se
aumenta el contador de repeticiones totales
                                        repeticionesRestantes--;        //Se
disminuye el valor de repeticiones totales a realizar
                                        totalRepeticionesRealizadas =
repeticionesCorrectas + repeticionesIncorrectas; //Se suma los valores de
repeticones correctas e incorrectas
                                        ActualizarContadoresUI();       //
Actualizar valores en la paltalla de UI
                                        estadoEjercicio =
EstadoEjercicio.EnPosicionInicial; //al finalizar las repeticiones se vuelve al
estado inicial
                                        //Aviso cambio de posicion UI
                                        Dispatcher.Invoke(() =>
                                        {
                                            txtIndicacion.Text = "BAJE LOS
BRAZOS";          //Mensaje indicador de cambio de posicion
                                            Task.Delay(2000).ContinueWith(t =>
Dispatcher.Invoke(() => txtIndicacion.Text = "")); //Se indica el mensaje
durantos dos segundos.
                                        });
                                    }
                                    Console.WriteLine($"*****POSICION
FINAL*****");
                                    Console.WriteLine($"Contador de Frames
Posicion Inicial: {contadorFramesF}");
                                    Console.WriteLine($"Contador de angulos
iniciales Correctos: {contadorAnguloFinalCorrectos}");
                                    Console.WriteLine($"Contador de angulos
iniciales Incorrectos: {contadorAnguloFinalIncorrectos}");
                                    Console.WriteLine($"Angulo DERECHO
CORRECTO: {anguloDerechoC}");
                                    Console.WriteLine($"Angulo DERECHO
INCORRECTO: {anguloDerechoI}");
                                    Console.WriteLine($"Angulo IZQUIERDO
CORRECTO: {anguloIzquierdoC}");
                                    Console.WriteLine($"Angulo IZQUIERDO
INCORRECTO: {anguloIzquierdoI}");
                                    Console.WriteLine($"MODA ANGULO INCIAL
CORRECTO DERECHO: {modaAnguloDerechoFC}");
                                    Console.WriteLine($"MODA ANGULO INCIAL
INCORRECTO DERECHO: {modaAnguloDerechoFI}");
```

```
                                    Console.WriteLine($"MODA ANGULO INCIAL
CORRECTO IZQUIERDO: {modaAnguloIzquierdoFC}");
                                    Console.WriteLine($"MODA ANGULO INCIAL
INCORRECTO IZQUIERDO: {modaAnguloIzquierdoFI}");
                                    Console.WriteLine("Lista de ángulos
derechos FC:");
                                    var listaAngulosDerechosFC =
ControlRepeticionesFinales.LastOrDefault()?.AngulosDerechosFC;
                                    foreach (var anguloFC in
listaAngulosDerechosFC)
                                    {
                                        Console.WriteLine(anguloFC);
                                    }
                                }
                            }
                        }
                    }
                    this.drawingGroup.ClipGeometry = new RectangleGeometry(new
Rect(0.0, 0.0, this.displayWidth, this.displayHeight));   //Rectangulo del
limite de deteccion del cuerpo
                }
                if (repeticionesRestantes == 0)            //Condicion cuando
las repeticiones llegan a 0
                {
                    repeticionesRestantes = valorDefecto;  //El valor de
repetciones vuelve al valor seleccionado por el usuario
                    ActualizarContadoresUI();              //Se actualiza el
valor en la pantalla
                    double porcentajeIncorrectas =
(double)repeticionesIncorrectas / totalRepeticionesRealizadas;  //Se calcula el
porcenta para enviar el valor de alerta a telegram
                    if (porcentajeIncorrectas > 0.5)    //Se verifica que el
valor calculado supere el 50%
                    {
                        //Envio de alerta con el nombre y la cantidad de
repeticiones incorrectas ejecutadas.
                        await botClient.SendTextMessageAsync(chatId:
"5599983300", text: $"El usuario {nombreUsuario} ha completado
{repeticionesIncorrectas} de {totalRepeticionesRealizadas} repeticiones de
elevaciones laterales de manera incorrecta ");
                    }
                    numeroTotalRepeticiones = 1;         // Se reinicia el
valor a mostrar en reporte y control
                    totalRepeticionesRealizadas = 0;     //Se reinica el valor
de las repeticiones totales
                    porcentajeIncorrectas = 0;           //Se reinicia el valor
de porcentaje de repeticiones incorrectas
                    ActualizarContadoresUI();            //Se actualiza los
valorres en los campos de UI
                    BorrarCuerpo();                      //Se llama al metodo
para desaactivar la deteccion del esqueleto
                    activacionCuerpo = false;            // Detiene el
seguimiento y procesamiento de frames
                }
            }
        }
```

```
        //Metodo para dibujar cuerpo, recibe 4 parámetros, articulaciones del
cuerpo, articulaciones en dos dimensiones, el cuadro de dibujo,boligrafo para
dibujar el cuerpo
        private void DibujarCuerpo(IReadOnlyDictionary<JointType, Joint>
joints, IDictionary<JointType, Point> jointPoints, DrawingContext
drawingContext, Pen drawingPen)
        {
            // Draw the bones
            foreach (var bone in this.bones)
//Se itera sobre cada hueso en la colección de huesos
            {
                this.DibujarHuesos(joints, jointPoints, bone.Item1, bone.Item2,
drawingContext, drawingPen);  //Se llama al método DibujarHuesos para dibujar
cada hueso con las articulaciones y puntos capturados.
            }
            // Draw the joints
            foreach (JointType jointType in joints.Keys)
// Se recorre sobre cada tipo de articulación en el diccionario de
articulaciones
            {
                Brush drawBrush = null;
//Inicializa el esfero como nulo
                TrackingState trackingState = joints[jointType].TrackingState;
//Se obtiene el estado de seguimiento de la articulacion actual
                if (trackingState == TrackingState.Tracked)
//Condicion si la articulacion es rastreada
                {
                    drawBrush = this.trackedJointBrush;
//Se asigna un esfero para la articulacion rastreada
                }
                else if (trackingState == TrackingState.Inferred)
//Condicion si la articulacion rastreada es inferida
                {
                    drawBrush = this.inferredJointBrush;
//Se asigna un esfero para la articulacion inferida
                }
                if (drawBrush != null)
//Condicion si hay un esfero asignado
                {
                    drawingContext.DrawEllipse(drawBrush, null,
jointPoints[jointType], JointThickness, JointThickness); //Se dibuja una punto
en la posición de la articulación con el pincel seleccionado
                }
            }
        }
        //Metodo borrar cuerpo
        private void BorrarCuerpo()
        {
            using (DrawingContext dc = this.drawingGroup.Open())
            {
                // Limpiar el área dibujando un rectángulo del tamaño completo
del área de dibujo con un esfero transparente
                dc.DrawRectangle(Brushes.Transparent, null, new Rect(0.0, 0.0,
this.displayWidth, this.displayHeight));
            }
        }

        // Método para dibujar los huesos entre dos articulaciones
```

```
        private void DibujarHuesos(IReadOnlyDictionary<JointType, Joint>
joints, IDictionary<JointType, Point> jointPoints, JointType jointType0,
JointType jointType1, DrawingContext drawingContext, Pen drawingPen)
        {
            Joint joint0 = joints[jointType0];           //Se obtiene la
primera articulación del diccionario usando jointType0
            Joint joint1 = joints[jointType1];           //Se obbtiene la
segunda articulación del diccionario usando jointType1
            if (joint0.TrackingState == TrackingState.NotTracked ||
joint1.TrackingState == TrackingState.NotTracked) //Condicion que verifica si
alguna de las articulaciones no es rastreada
            {
                return;                                  //Si alguna de las
articulaciones no está siendo rastreada se sale del metodo
            }
            Pen drawPen = this.inferredBonePen;        //Se usa el bolígrafo
para huesos inferidos
            if ((joint0.TrackingState == TrackingState.Tracked) &&
(joint1.TrackingState == TrackingState.Tracked)) //Condicion cuando ambas
articulaciones están siendo rastreadas
            {
                drawPen = drawingPen;                //Se cambia el bolígrafo
al especificado en el parámetro si ambas articulaciones están rastreadas
            }
            drawingContext.DrawLine(drawPen, jointPoints[jointType0],
jointPoints[jointType1]); //Se dibuja una línea entre las dos articulaciones.
        }

        // Método para dibujar bordes recortados alrededor del cuerpo rastreado
        private void DrawClippedEdges(Body body, DrawingContext drawingContext)
        {
            FrameEdges clippedEdges = body.ClippedEdges;               //Se
obtiene los bordes recortados del cuerpo
            if (clippedEdges.HasFlag(FrameEdges.Bottom))
//Condicion que verifica si el borde inferior está recortado
            {
                drawingContext.DrawRectangle(                          //Se
dibuja un rectángulo rojo en el borde inferior
                    Brushes.Red,                                       //Se
define el color de la linea
                    null,                                              //No hay
borde
                    new Rect(0, this.displayHeight - ClipBoundsThickness,
this.displayWidth, ClipBoundsThickness)); //Se define la posición y tamaño del
rectángulo
            }
            if (clippedEdges.HasFlag(FrameEdges.Top))
//Condicion que verifica si el borde superior está recortado
            {
                drawingContext.DrawRectangle(                          //Se
dibuja un rectángulo rojo en el borde superior
                    Brushes.Red,                                       //Se
define el color de la linea
                    null,                                              //No hay
borde
                    new Rect(0, 0, this.displayWidth, ClipBoundsThickness));
//Se define la posición y tamaño del rectángulo
            }
```

```
            if (clippedEdges.HasFlag(FrameEdges.Left))
//Condicion que verifica si el borde izquierdo está recortado
            {
                drawingContext.DrawRectangle(                           //Se
dibuja un rectángulo rojo en el borde zquierdo
                    Brushes.Red,                                        //Se
define el color de la linea
                    null,                                               //No hay
borde
                    new Rect(0, 0, ClipBoundsThickness, this.displayHeight));
//Se define la posición y tamaño del rectángulo
            }
            if (clippedEdges.HasFlag(FrameEdges.Right))
//Condicion que verifica si el borde derecho está recortado
            {
                drawingContext.DrawRectangle(
//Se dibuja un rectángulo rojo en el borde derecho
                    Brushes.Red,
//Se define el color de la linea
                    null,
//No hay borde
                    new Rect(this.displayWidth - ClipBoundsThickness, 0,
ClipBoundsThickness, this.displayHeight)); //Se define la posición y tamaño del
rectángulo
            }
        }

        private void ReproducirSonidoE(bool esCorrecto)
        {
            try
            {
                string rutaSonido = esCorrecto ?
@"C:\Users\Lenovo\Documents\DOCUMENTOS ERIK\ESTUDIO\OCTAVO II\PRIMER
BIMESTRE\AVANCE11\Proyecto\imagenes\Acierto.wav"
                                               :
@"C:\Users\Lenovo\Documents\DOCUMENTOS ERIK\ESTUDIO\OCTAVO II\PRIMER
BIMESTRE\AVANCE11\Proyecto\imagenes\Error.wav";
                SoundPlayer player = new SoundPlayer(rutaSonido);
                player.Play();
            }
            catch (Exception ex)
            {
                Console.WriteLine("Error al reproducir el sonido: " +
ex.Message);
            }
        }


        //Metodo para calcular angulos
        public int CalcularAngulo(CameraSpacePoint puntoA, CameraSpacePoint
puntoB, CameraSpacePoint puntoC)     //Hombro "A",codo "B", muñeca "C"
        {
            // Calcula los vectores a partir de los puntos
            Vector3D vectorAB = new Vector3D(puntoB.X - puntoA.X, puntoB.Y -
puntoA.Y, puntoB.Z - puntoA.Z);     //Vector hacia el hombro
            Vector3D vectorAC = new Vector3D(puntoC.X - puntoA.X, puntoC.Y -
puntoA.Y, puntoC.Z - puntoA.Z);     //Vector hacia el codo
```

```
            // Calcula de ángulo
            double productoPunto = Vector3D.DotProduct(vectorAB, vectorAC);
//Operación del producto punto entre los dos vectores
            double magnitudAB = vectorAB.Length;
//Calculo de la magnitud del vector hacia el hombro
            double magnitudAC = vectorAC.Length;
//Calculo de la magnitud del vector hacia el codo
            double cosenoAngulo = productoPunto / (magnitudAB * magnitudAC);
//Calculo del coseno del ángulo
            double anguloRadianes = Math.Acos(cosenoAngulo);
//Calculo del coseno en radianes
            // Convierte el ángulo a grados
            double anguloGrados = anguloRadianes * (180.0 / Math.PI);
//Conversion de radianes a grados
            return (int)Math.Truncate(anguloGrados);
// Redondeo y converción a entero
        }

        // Método  que verifica si un valor está dentro del rango para validar
una repeticion correcta o incorrecta, posicion correcta o incorrecta
        public bool EstaEnRango(int valor, int minimo, int maximo)  //Toma tres
valores para compararlos
        {
            return valor >= minimo && valor <= maximo;              // Devuelve
true si el valor del angulo está entre el mínimo y el máximo, de lo contrario
devuelve false.
        }

        //Método para detalles de posicion inicial incorrectas
        private string DetalleErrorPosicionInicial(int promedioAnguloDerechoII,
int promedioAnguloIzquierdoII)
        {
            bool bajoDerechoI = EstaEnRango(promedioAnguloDerechoII,
AnguloInicialBajoIncorrectoMin, AnguloInicialBajoIncorrectoMax);
//Verificacion de condicion cuando el brazo derecho está muy separado al torso
PI
            bool altoDerechoI = EstaEnRango(promedioAnguloDerechoII,
AnguloInicialAltoIncorrectoMin, AnguloInicialAltoIncorrectoMax);
//Verificacion de condicion cuando el brazo derecho está muy junto al tordo PI
            bool bajoIzquierdoI = EstaEnRango(promedioAnguloIzquierdoII,
AnguloInicialBajoIncorrectoMin, AnguloInicialBajoIncorrectoMax); //Verificacion
de condicion cuando el brazo izquierdo está muy separado al torso PI
            bool altoIzquierdoI = EstaEnRango(promedioAnguloIzquierdoII,
AnguloInicialAltoIncorrectoMin, AnguloInicialAltoIncorrectoMax); //Verificacion
de condicion cuando el brazo izquierdo está muy junto al torso PI

            List<string> detalles = new List<string>();
//Lista para almacenar los detalles.
            if (bajoDerechoI || bajoIzquierdoI)
            {
                if (bajoDerechoI && bajoIzquierdoI)
//Condicion cuando ambos brazos están muy separados al torso
                {
                    detalles.Add("Ambos brazos muy separados del torso");
//Se agrega el detalle a la tabla de control
                }
                else
                {
```

```
                    if (bajoDerechoI)
//Condicion cuando brazo derecho está muy separado del torso
                    {
                        detalles.Add("Brazo derecho muy separado del torso");
//Se agrega el detalle a la tabla de control
                    }
                    if (bajoIzquierdoI)
//Condicion cuando el brazo izquierdo está muy seoarado del torso
                    {
                        detalles.Add("Brazo izquierdo muy separado del torso");
//Se agrega el detalle a la tabla de control
                    }
                }
            }
            if (altoDerechoI || altoIzquierdoI)
            {
                if (altoDerechoI && altoIzquierdoI)
//Condicion cuando ambos brazos están muy juntos al torso
                {
                    detalles.Add("Ambos brazos muy juntos al torso");
//Se agrega el detalle a la tabla de control
                }
                else
                {
                    if (altoDerechoI)
//Condicion cuando el brazo derecho está muy junto al torso
                    {
                        detalles.Add("Brazo derecho muy junto al torso");
//Se agrega el detalle a la tabla de control
                    }
                    if (altoIzquierdoI)
//Condicion cuando el brazo inzquierdo está muy junto al torso
                    {
                        detalles.Add("Brazo izquierdo muy junto al torso");
//Se agrega el detalle a la tabla de control
                    }
                }
            }
            // Devolver mensaje si no hay errores
            if (detalles.Count == 0)
            {
                return "Ángulo fuera de rango de detección";
//Se agrega el detalle a la tabla de control
            }
            return string.Join(" Y ", detalles);
        }

        //Método para detalles de posicion final incorrectas
        private string DetalleErrorPosicionFinal(int promedioAnguloDerechoFI,
int promedioAnguloIzquierdoFI)
        {
            bool bajoDerechoF = EstaEnRango(promedioAnguloDerechoFI,
AnguloFinalBajoIncorrectoMin, AnguloFinalBajoIncorrectoMax);     //Verificación
de condicion cuando el brazo derecho está muy bajo a nivel de los hombros PF.
            bool altoDerechoF = EstaEnRango(promedioAnguloDerechoFI,
AnguloFinalAltoIncorrectoMin, AnguloFinalAltoIncorrectoMax);     //Verificación
de condición cuando el brazo derecho está muy alto a nivel de los hombros PF.
```

```
            bool bajoIzquierdoF = EstaEnRango(promedioAnguloIzquierdoFI,
AnguloFinalBajoIncorrectoMin, AnguloFinalBajoIncorrectoMax); //Verificación de
condicion cuando el brazo izquierdo está muy bajo a nivel de los hombros PF.
            bool altoIzquierdoF = EstaEnRango(promedioAnguloIzquierdoFI,
AnguloFinalAltoIncorrectoMin, AnguloFinalAltoIncorrectoMax); //Verificación de
condición cuando el brazo izquierdo está muy alto a nivel de los hombros PF.
            List<string> detalles = new List<string>();
            // Evaluar combinaciones de brazos bajos
            if (bajoDerechoF || bajoIzquierdoF)
            {
                if (bajoDerechoF && bajoIzquierdoF)
//Condicion cuando ambos brazos están muy bajos a nivel de los hombros.
                {
                    detalles.Add("Ambos brazos muy bajos");             //Se
agrega el detalle a la tabla de control
                }
                else
                {
                    if (bajoDerechoF)
//Condicion cuando el brazo derecho está muy bajo a nivel del hombro.
                    {
                        detalles.Add("Brazo derecho muy bajo");        //Se
agrega el detalle a la tabla de control
                    }
                    if (bajoIzquierdoF)
//Condicion cuando el brazo izquierdo está muy bajo a nivel del hombro.
                    {
                        detalles.Add("Brazo izquierdo muy bajo");      //Se
agrega el detalle a la tabla de control
                    }
                }
            }
            // Evaluar combinaciones de brazos altos
            if (altoDerechoF || altoIzquierdoF)
            {
                if (altoDerechoF && altoIzquierdoF)
//Condicion cuando ambos brazos están muy elevados a nivel de los hombros.
                {
                    detalles.Add("Ambos brazos muy elevados");        //Se
agrega el detalle a la tabla de control
                }
                else
                {
                    if (altoDerechoF)
//Condicio cuando el brazo derecho está muy elevado a nivel de los hombros
                    {
                        detalles.Add("Brazo derecho muy elevado");    //Se
agrega el detalle a la tabla de control
                    }
                    if (altoIzquierdoF)
//Condicion cuando el brazo izquierdo está muy elevado a nivel de los hombros.
                    {
                        detalles.Add("Brazo izquierdo muy elevado"); //Se
agrega el detalle a la tabla de control
                    }
                }
            }
            // Devolver mensaje si no hay errores
```

```
            if (detalles.Count == 0)
            {
                return "Ángulo fuera de rango de detección";        //Se agrega
el detalle a la tabla de control
            }

            return string.Join(" Y ", detalles);
        }
        private void TextBox_TextChanged(object sender, TextChangedEventArgs e)
//TextRepeticiones Correctas
        {

        }

        private void TextBox_TextChanged_1(object sender, TextChangedEventArgs
e)  //TextRepeticiones Incorrectas
        {

        }


        private void ActualizarContadoresUI()
        {
            Dispatcher.Invoke(() =>
            {
                TextCorretas.Text = repeticionesCorrectas.ToString();
                TextIncorrectas.Text = repeticionesIncorrectas.ToString();
                TextRaRealizar.Text = repeticionesRestantes.ToString();
            });
        }

        // Metodo que se ejecuta cuando el texto en el TextBox 'TextRaRealizar'
cambia
        private void TextRaRealizar_TextChanged(object sender,
TextChangedEventArgs e)    //TextRepeticiones a Realizar
        {
            if (int.TryParse(TextRaRealizar.Text, out int numRepeticion))
//Se convierte el valor ingresado a un valor entero
            {
                repeticionesRestantes = numRepeticion;
//Si la conversión es exitosa, asigna el valor a la variable
repeticionesRestantes
                ActualizarContadoresUI();
//Actualizacion valores de UI
            }
            else
            {
                repeticionesRestantes = 0;
//Manejo en caso de que el texto no sea un número
            }
        }
        private void Button_Click(object sender, RoutedEventArgs e)
//Boton Menu
        {
            //Se inicializa las variables
            contadorFramesF = 0;
            contadorFramesI = 0;
            repeticionesCorrectas = 0;
```

```
            repeticionesIncorrectas = 0;
            repeticionesRestantes = valorDefecto;
            estadoEjercicio = EstadoEjercicio.EnPosicionInicial;
            Menu menu = new Menu();
            this.Close();
            menu.Show();
        }
        private void BtnIniciar_Click(object sender, RoutedEventArgs e)
//Boton INICIAR
        {
            //Se inicializa las variables
            contadorFramesF = 0;
            contadorFramesI = 0;
            contadorAnguloFinalIncorrectos = 0;
            contadorAnguloFinalCorrectos = 0;
            contadorAnguloInicialIncorrectos = 0;
            contadorAnguloInicialCorrectos = 0;
            activacionCuerpo = true;  // Activa el seguimiento y procesamiento
            repeticionesCorrectas = 0;
            repeticionesIncorrectas = 0;
            correctas = 0;
            incorrectasI = 0;
            incorrectasF = 0;
            numeroTotalRepeticiones = 1;
            repeticionesRestantes = valorDefecto;
            estadoEjercicio = EstadoEjercicio.EnPosicionInicial;
            ActualizarContadoresUI();
            //Limpieza de campos de tablas de control y reporte pdf.
            ControlRepeticionesIncorrectasI.Clear();
            ControlRepeticionesIncorrectasF.Clear();
            ControlRepeticionesIniciales.Clear();
            ControlRepeticionesFinales.Clear();
        }

        private void txtIndicacion_TextChanged(object sender,
TextChangedEventArgs e)
        {

        }

        private void Button_Click_1(object sender, RoutedEventArgs e)
//Boton 15 Repeticiones
        {
            //Se inicializa las variable de repeticiones restantes con el valor
de 15
            repeticionesRestantes = 15;
            valorDefecto = 15;
            ActualizarContadoresUI();
            rElevaciones.SaveValue(valorDefecto);  //Envio de valor a ubidots
        }

        private void btn8_Click(object sender, RoutedEventArgs e)
//Boton 8 repetciones
        {
            //Se inicializa las variable de repeticiones restantes con el valor
de 15
            repeticionesRestantes = 8;
            valorDefecto = 8;
```

```
            ActualizarContadoresUI();
            rElevaciones.SaveValue(valorDefecto); //Envio de valor a ubidots
        }

        private void btn10_Click(object sender, RoutedEventArgs e)
//Boton 10 repeticiones
        {
            //Se inicializa las variable de repeticiones restantes con el valor
de 10
            repeticionesRestantes = 10;
            valorDefecto = 10;
            ActualizarContadoresUI();
            rElevaciones.SaveValue(valorDefecto); //Envio de valor a ubidots
        }

        private void btn12_Click(object sender, RoutedEventArgs e)
//Boton 12 repeticiones
        {
            //Se inicializa las variable de repeticiones restantes con el valor
de 8
            repeticionesRestantes = 12;
            valorDefecto = 12;
            ActualizarContadoresUI();
            rElevaciones.SaveValue(valorDefecto); //Envio de valor a ubidots
        }

        private void TextIncorrectas_TextChanged(object sender,
TextChangedEventArgs e)
        {

        }

        private void btnDetalles_Click(object sender, RoutedEventArgs e)
        {
            var ventanaResultados = new ResultadosIncorrectos();
            ventanaResultados.DataContext = this; // Asegúrate de que
ResultadosIncorrectos tenga el DataContext adecuado para el binding
            ventanaResultados.Show();
        }

        //Metodo para generar reporte
        private void GenerarReportePdf()
        {
            string desktopPath =
Environment.GetFolderPath(Environment.SpecialFolder.Desktop);
//Ubicacion donde se guarda el pdf.
            string filePath = Path.Combine(desktopPath,
$"ReporteElevacionesLaterales_{nombreUsuario}{contadorReporte}.pdf");//Nombre
para guardar el reporte pdf
            //Imagenes para el reporte pdf
            string imagePathLeft = Path.Combine(desktopPath,
"C:\\Users\\Lenovo\\Documents\\DOCUMENTOS ERIK\\ESTUDIO\\OCTAVO II\\PRIMER
BIMESTRE\\AVANCE 05\\Proyecto\\imagenes\\utn.jpg");
            string imagePathRight = Path.Combine(desktopPath,
"C:\\Users\\Lenovo\\Documents\\DOCUMENTOS ERIK\\ESTUDIO\\OCTAVO II\\PRIMER
BIMESTRE\\AVANCE 05\\Proyecto\\imagenes\\zener.jpg");
            //Instancia para generar el tamaño del reporte
            Document document = new Document(PageSize.A4);
```

```
            try
            {
                PdfWriter.GetInstance(document, new FileStream(filePath,
FileMode.Create));
                document.Open();

                // Añadir imágenes
                Image imgLeft = Image.GetInstance(imagePathLeft);
                imgLeft.SetAbsolutePosition(30, document.Top - 10);
// Ajusta posición y tamaño
                imgLeft.ScaleToFit(80, 80);
                document.Add(imgLeft);
                Image imgRight = Image.GetInstance(imagePathRight);
                imgRight.SetAbsolutePosition(document.PageSize.Width - 105,
document.Top - 5); // Ajusta posición y tamaño
                imgRight.ScaleToFit(80, 80);
                document.Add(imgRight);
                // Añadir título
                Paragraph titulo = new Paragraph("ZENER GYM\nReporte de
Evaluación de Ejercicio", FontFactory.GetFont("Times New Roman", 20,
Font.BOLD));
                titulo.Alignment = Element.ALIGN_CENTER;
                document.Add(titulo);
                Paragraph nombre = new Paragraph($"Usuario: {nombreUsuario}",
FontFactory.GetFont("Times New Roman", 16, Font.BOLD));
                nombre.Alignment = Element.ALIGN_CENTER;
                document.Add(nombre);
                document.Add(new Paragraph("\n")); // Espacio

                // Se añade subtitulo para la primera tabla
                Paragraph subPosicionInicial = new Paragraph("Estado de
repetición en Posicion Inicial", FontFactory.GetFont("Times New Roman", 12,
Font.BOLD));
                subPosicionInicial.Alignment = Element.ALIGN_CENTER;
                document.Add(subPosicionInicial);
                document.Add(new Paragraph("\n"));//Espacio
                Paragraph rangocorrectoI = new Paragraph("Rango correcto 25° -
35°", FontFactory.GetFont("Times New Roman", 12, Font.BOLD));
                rangocorrectoI.Alignment = Element.ALIGN_CENTER;
                document.Add(rangocorrectoI);
                document.Add(new Paragraph("\n"));
                // Numero de columnas y nombres
                PdfPTable tableI = new PdfPTable(5);
                tableI.AddCell("Número");
// Columna número de repeticiones
                tableI.AddCell("Posición");
// Columna de posicion inicial o final
                tableI.AddCell("Ángulo de brazo derecho");
// Columna para ángulos c/r brazo derecho PI
                tableI.AddCell("Ángulo de brazo Izquierdo");
// Columna para ángulos c/r brazo izquierdo PI
                tableI.AddCell("Estado de Posición");
// Columna para posicion I correcta o incorrecta
                foreach (var item in ControlRepeticionesIniciales)
// Inicio de ciclo
                {
                    tableI.AddCell(item.NumeroRepeticionT.ToString());
// Se añade los valores de numero de repeticion PI
```

```
                    tableI.AddCell(item.EstadoPosicionInicial);
// Se añade el estado de posicion PI
                    if (item.EstadoEvaluacionI == "Correcto")
// Condicion si la posicion es correcta PI
                    {
                        tableI.AddCell(item.ModaAnguloDerechoIC.ToString() +
"°");       // Se añade ángulos derechos correctas PI
                        tableI.AddCell(item.ModaAnguloIzquierdoIC.ToString() +
"°");     // Se añade ángulos izquierdos correctos PI
                    }
                    else if (item.EstadoEvaluacionI == "Incorrecto")
// Condicion si la posicion es incorrecta PI
                    {
                        tableI.AddCell(item.ModaAnguloDerechoII.ToString() +
"°");       //Se añade ángulos izquierdos incorrectos PI
                        tableI.AddCell(item.ModaAnguloIzquierdoII.ToString() +
"°");     //Se añade ángulos izquierdos incorrectos PI
                    }
                    tableI.AddCell(item.EstadoEvaluacionI);
// Se añade el estado de posicion PI
                }
                document.Add(tableI);
// Se agrega la tabla al documento pdf
                //Espacio entre tablas
                document.Add(new Chunk("\n"));
                // Añadir subtitulo para la segunda tabla
                Paragraph subTitleF = new Paragraph("Estado de Repetición en
Posición Final", FontFactory.GetFont("Times New Roman", 12, Font.BOLD));
                subTitleF.Alignment = Element.ALIGN_CENTER;
                document.Add(subTitleF);
                document.Add(new Chunk("\n"));
                Paragraph rangocorrectoF = new Paragraph("Rango correcto 5° -
11°", FontFactory.GetFont("Times New Roman", 12, Font.BOLD));
                rangocorrectoF.Alignment = Element.ALIGN_CENTER;
                document.Add(rangocorrectoF);
                document.Add(new Paragraph("\n"));

                PdfPTable tableF = new PdfPTable(5);
// Ajusta número de columnas
                tableF.AddCell("Número");
// Columna número de repeticiones Pf
                tableF.AddCell("Posición");
// Columna de posicion inicial o final
                tableF.AddCell("Ángulo brazo derecho");
// Columna para ángulos c/r brazo derecho PF
                tableF.AddCell("Ángulo brazo izquierdo");
// Columna para ángulos c/r brazo izquierdo PF
                tableF.AddCell("Estado de Posición");
// Columna para posicion correcta o incorrecta PF
                foreach (var item in ControlRepeticionesFinales)
//Inicio de ciclo
                {
                    tableF.AddCell(item.NumeroRepeticionT.ToString());
// Se añade los valores de numero de repeticion PF
                    tableF.AddCell(item.EstadoPosicionFinal);
// Se añade el estado de posicion PF
                    if (item.EstadoEvaluacionF == "Correcto")
// Condicion si la posicion es correcta PF
```

```
                    {
                        tableF.AddCell(item.ModaAnguloDerechoFC.ToString() +
"°");   // Se añade ángulos derechos correctas PF
                        tableF.AddCell(item.ModaAnguloIzquierdoFC.ToString() +
"°"); // Se añade ángulos izquierdos correctos PF
                    }
                    else if (item.EstadoEvaluacionF == "Incorrecto")
// Condicion si la posicion F es incorrecta
                    {
                        tableF.AddCell(item.ModaAnguloDerechoFI.ToString() +
"°");   //Se añade ángulos izquierdos incorrectos
                        tableF.AddCell(item.ModaAnguloIzquierdoFI.ToString() +
"°"); //Se añade ángulos izquierdos incorrectos
                    }
                    tableF.AddCell(item.EstadoEvaluacionF);
// Se añade el estado de posicion F
                }
                document.Add(tableF);
// Se agrega la tabla al documento pdf
            }
            catch (Exception ex)
            {
                MessageBox.Show("No se pudo generar el reporte PDF: " +
ex.Message); //Notificacion de error si el reporte no se pudo generar.
            }
            finally
            {
                document.Close();
// Cierra el documento PDF para liberar recursos

            }
            MessageBox.Show("Reporte generado con éxito en: " + filePath);
//Notificacion reporte generado con exito, seguido de la ubicacion
            contadorReporte++;
        }
        private void btnReporte_Click(object sender, RoutedEventArgs e)
//Metodo generar reporte
        {
            GenerarReportePdf();     //Se llama al método GenerarReportePdf
para generar el PDF del reporte
        }
    }
}
```

Anexo 10F. Entrenamiento Press Militar

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Timers;
using Microsoft.Kinect;
using Ubidots;
using System.Windows.Media.Media3D;
using Telegram.Bot;
```

```
using static Proyecto.ResultadosIncorrectos;             //Declaracion de
directiva para acceder a las propiedades de ResultadosIncorrectos
using System.Collections.ObjectModel;
using iTextSharp.text.pdf;
using iTextSharp.text;
using System.IO;
using Image = iTextSharp.text.Image;                    //Declaracion de alias
para el uso de Image
using System.Linq;                                      //Creacion de directiva
para manejo de las listas.
using System.Windows.Threading;
using System.Media;
namespace Proyecto
{
    public partial class PressMilitar : Window
    {
        private bool activacionCuerpo = false;         //Variable desactivacion
de cuerpo
        //Creacion de propiedad para almacenar colecciones tabla control
        public ObservableCollection<ControlRepeticionesI>
ControlRepeticionesIncorrectasI { get; set; } = new
ObservableCollection<ControlRepeticionesI>();
        public ObservableCollection<ControlRepeticionesF>
ControlRepeticionesIncorrectasF { get; set; } = new
ObservableCollection<ControlRepeticionesF>();
        //Creacion de propiedad para almacenar colecciones pdf
        public ObservableCollection<ControlReportePosicionInicial>
ControlRepeticionesIniciales { get; set; } = new
ObservableCollection<ControlReportePosicionInicial>();
        public ObservableCollection<ControlReportePosicionFinal>
ControlRepeticionesFinales { get; set; } = new
ObservableCollection<ControlReportePosicionFinal>();
        //Se llama a la variable del usuario de ingreso
        string nombreUsuario = AgregarUsuario.CurrentUsername;

        //Variables para el seguimiento de esqueleto
        public const double JointThickness = 3;
//Grosor de las líneas de unión dibujadas
        public const double ClipBoundsThickness = 10;
//Grosor de los rectángulos del borde del clip
        public const float InferredZPositionClamp = 0.1f;
//Constante para evitar que los valores Z de los puntos del espacio de la
cámara sean negativos
        public readonly Brush trackedJointBrush = new
SolidColorBrush(Color.FromArgb(255, 68, 192, 68)); //Pincel utilizado para
dibujar juntas que actualmente están rastreadas
        public readonly Brush inferredJointBrush = Brushes.Yellow;
//Pincel utilizado para dibujar juntas que se infieren actualmente.
        public readonly Pen inferredBonePen = new Pen(Brushes.Gray, 1);
//Bolígrafo utilizado para dibujar huesos que actualmente se infieren.
        public DrawingGroup drawingGroup;
//Grupo de dibujo para salida de renderizado corporal
        public DrawingImage imageSource;
//Imagen de dibujo que mostraremos.
        public CoordinateMapper coordinateMapper = null;
//Mapeador de coordenadas para asignar un tipo de punto a otro
        public BodyFrameReader bodyFrameReader = null;
//Lector de estructuras corporales
```

```
        public Body[] bodies = null;
//Creacion de array para guardar los cuerpos>
        public List<Tuple<JointType, JointType>> bones;
//Definicion de huesos
        public int displayWidth;
//Ancho de visualización (espacio de profundidad)
        public int displayHeight;
//Altura de la pantalla (espacio de profundidad)
        public List<Pen> bodyColors;

        //Creacion de campos
        public KinectSensor kinect;
// Campo que almacena una referencia a un objeto KinectSensor
        public ColorFrameReader LecturaFrameColor;
// Campo que almacena una referencia a un objeto LecturaFrameColor
        public WriteableBitmap MapaBits;
// Campo que almacena una referencia a un objeto MapaBits

        //Variables y campos para establecer conexion con Ubidots
        string apiKey = "BBUS-41a2f5af076c246639f66d4be1b76e40ea9";
        string variableMonitoreoId = "65dce502842f44000d0a13ae";
        ApiClient api;
        Variable variableMonitoreo;
        Timer timer;

        //Campo para el telegram
        private readonly TelegramBotClient botClient;

        //Variable y campos ubidots elevaciones laterales
        string variableIDCantidadRepeticiones = "6626f47ad37bf1000cdc9c66";
        Variable rPress;
        string variableIDPressC = "66160fbb48952e1897b83286";
        Variable rCorrectasPress;
        string variableIDPressI = "66160fbc1db34119ff6120ca";
        Variable rIncorrectasPress;

        // Variables para angulo deteccion de ejercicio

        // Rango de ángulos para la posición inicial y final
        private const int AnguloInicialMin = 25;                          //
Angulo minimo aceptable en posición inicial
        private const int AnguloInicialMax = 50;                          //
Angulo maximo aceptable en posición inicial
        private const int AnguloFinalMin = 2;                             //
Angulo minimo aceptable en posición final
        private const int AnguloFinalMax = 22;                            //
Angulo maximo aceptable en posición final

        // Angulos Posicion Correctas
        private const int AnguloInicialCorrectoMin = 33;
//Angulo minimo aceptable posicion inicial
        private const int AnguloInicialCorrectoMax = 43;
//Angulo maximo aceptable poscion inicial
        private const int AnguloFinalCorrectoMin =7;
//Angulo minimo aceptable posicion final
        private const int AnguloFinalCorrectoMax = 14;
//angulo mínimo aceptable posicion final
```

```
        //Angulos Posiciones Incorrectas
        private const int AnguloInicialBajoIncorrectoMin = 25;          //Manos
muy separadas del hombro
        private const int AnguloInicialBajoIncorrectoMax = 32;          //Manos
muy separadas del hombro
        private const int AnguloInicialAltoIncorrectoMin = 44;          //Mano
muy junta al hombro
        private const int AnguloInicialAltoIncorrectoMax = 50;          //Mano
muy junta al hombro

        private const int AnguloFinalBajoIncorrectoMin = 15;
//Brazos muy recogidos
        private const int AnguloFinalBajoIncorrectoMax = 22;
//Brazos muy recogidos
        private const int AnguloFinalAltoIncorrectoMin = 2;
//Brazos muy extendidos
        private const int AnguloFinalAltoIncorrectoMax = 6;
//Brazos muy e3xtendidos

        //Variables para angulos calculados
        int anguloDerecho;
        int anguloIzquierdo;
        int anguloDerechoC;
        int anguloIzquierdoC;
        int anguloDerechoI;
        int anguloIzquierdoI;

        //Variables para alamcenar los valores del calculo de la moda
        int modaAnguloDerechoIC;
        int modaAnguloIzquierdoIC;
        int modaAnguloDerechoII;
        int modaAnguloIzquierdoII;
        int modaAnguloDerechoFC;
        int modaAnguloIzquierdoFC;
        int modaAnguloDerechoFI;
        int modaAnguloIzquierdoFI;

        //Variables para validar los ejercicios de forma correcta e incorrecta
        private int totalRepeticionesRealizadas = 0;
        private int valorDefecto;
        private int repeticionesRestantes;
// Un valor predeterminado
        private int repeticionesCorrectas;
        private int repeticionesIncorrectas;
        private int correctas;
        private int incorrectas;
        //Contador de repeticiones a detalle
        private int numeroTotalRepeticiones = 1;
        //Contador para el nombre del pdf
        private static int contadorReporte = 1;
        //string
        string estadoEvaluacionI = ""; // Estado por defecto
        string estadoEvaluacionF = ""; //Estado por defecto


        enum EstadoEjercicio             //Declaración de enumeraciones
        {
            EnPosicionInicial,
```

```
            EnPosicionFinal,
        }
        private EstadoEjercicio estadoEjercicio =
EstadoEjercicio.EnPosicionInicial;               //Inicio de estado en posicion
inicial
        private int contadorFramesI;
        private int contadorFramesF;
        private int contadorAnguloInicialCorrectos;
        private int contadorAnguloFinalCorrectos;
        private int contadorAnguloInicialIncorrectos;
        private int contadorAnguloFinalIncorrectos;
        private const int FramesParaTresSegundos = 90;

        public PressMilitar()                            //Creacion de
constructor de esta clase
        {
            InitializeComponent();                       //Metodo para iniciar
componentes de UI
            InicializarKinect();                         //Llama al metodo para
iniciar el sensor kinect
            InicializarTimer();                          //Llama al metodo para
iniciar un time de ubidots
            Closed += PressMilitar_Closed;               //Se asocia el metodo
press militar al evento Closed
            ActualizarContadoresUI();                    //Llama a un metodo
para actualizar los contadores en la IU
            botClient = new TelegramBotClient("6722450347:AAFR9q-
qn_eJ38U9yk4pHx6UNnSC7_lTIGM");   //Inicializa un cliente de bot de Telegram
con el token
        }
        private void InicializarTimer()                  //Metodo para
inicializar temporizador
        {
            timer = new Timer(30000);                    //Creaci[on de
instancia para 30 segundos de temporizador
            timer.Elapsed += Timer_Elapsed;              //Se suscribe el método
Timer_Elapsed al evento Elapsed del temporizador
            timer.AutoReset = true;                      //Propiedad para
reiniciar automaticamente el temporizador
            timer.Start();                               //Inicia el
temporizador
        }
        private void Timer_Elapsed(object sender, ElapsedEventArgs e)
        {
            EnviarValorUbidots();
        }

        private void InicializarKinect()
        {
            this.kinect = KinectSensor.GetDefault();
            this.kinect.Open();
            this.LecturaFrameColor = this.kinect.ColorFrameSource.OpenReader();
//Acceso a datos de cámara RGB
            this.LecturaFrameColor.FrameArrived +=
LecturaFrameColor_FrameArrived; //Creacion de evento para cada cuadro de color
            FrameDescription colorFrameDescription =
this.kinect.ColorFrameSource.CreateFrameDescription(ColorImageFormat.Bgra);
//Se obtiene una descripcion del frame de color
```

```
            this.MapaBits = new WriteableBitmap(colorFrameDescription.Width,
colorFrameDescription.Height, 96.0, 96.0, PixelFormats.Bgr32, null);//Se
inicializa WriteableBitMap
            this.bodyFrameReader = this.kinect.BodyFrameSource.OpenReader();
//Lectura de los marcos del cuerpo
            this.DataContext = this;
            this.coordinateMapper = this.kinect.CoordinateMapper;
//Se obtiene el mapeador de coordenadas
            FrameDescription frameDescription =
this.kinect.DepthFrameSource.FrameDescription;    //Se obtiene las extensiones
de profundidad
            // obtiene el tamaño del espacio articular alto y ancho
            this.displayWidth = frameDescription.Width;
            this.displayHeight = frameDescription.Height;
            // Se define un hueso como una linea entre dos articulaciones
            this.bones = new List<Tuple<JointType, JointType>>();

            //Se crea una lista de huesos
            // Torso
            this.bones.Add(new Tuple<JointType, JointType>(JointType.Head,
JointType.Neck));
            this.bones.Add(new Tuple<JointType, JointType>(JointType.Neck,
JointType.SpineShoulder));
            this.bones.Add(new Tuple<JointType,
JointType>(JointType.SpineShoulder, JointType.SpineMid));
            this.bones.Add(new Tuple<JointType, JointType>(JointType.SpineMid,
JointType.SpineBase));
            this.bones.Add(new Tuple<JointType,
JointType>(JointType.SpineShoulder, JointType.ShoulderRight));
            this.bones.Add(new Tuple<JointType,
JointType>(JointType.SpineShoulder, JointType.ShoulderLeft));
            this.bones.Add(new Tuple<JointType, JointType>(JointType.SpineBase,
JointType.HipRight));
            this.bones.Add(new Tuple<JointType, JointType>(JointType.SpineBase,
JointType.HipLeft));

            // Brazo derecho
            this.bones.Add(new Tuple<JointType,
JointType>(JointType.ShoulderRight, JointType.ElbowRight));
            this.bones.Add(new Tuple<JointType,
JointType>(JointType.ElbowRight, JointType.WristRight));
            this.bones.Add(new Tuple<JointType,
JointType>(JointType.WristRight, JointType.HandRight));
            this.bones.Add(new Tuple<JointType, JointType>(JointType.HandRight,
JointType.HandTipRight));
            this.bones.Add(new Tuple<JointType,
JointType>(JointType.WristRight, JointType.ThumbRight));

            // Brazo izquierdo
            this.bones.Add(new Tuple<JointType,
JointType>(JointType.ShoulderLeft, JointType.ElbowLeft));
            this.bones.Add(new Tuple<JointType, JointType>(JointType.ElbowLeft,
JointType.WristLeft));
            this.bones.Add(new Tuple<JointType, JointType>(JointType.WristLeft,
JointType.HandLeft));
            this.bones.Add(new Tuple<JointType, JointType>(JointType.HandLeft,
JointType.HandTipLeft));
```

```
            this.bones.Add(new Tuple<JointType, JointType>(JointType.WristLeft,
JointType.ThumbLeft));

            // Pierna derecha
            this.bones.Add(new Tuple<JointType, JointType>(JointType.HipRight,
JointType.KneeRight));
            this.bones.Add(new Tuple<JointType, JointType>(JointType.KneeRight,
JointType.AnkleRight));
            this.bones.Add(new Tuple<JointType,
JointType>(JointType.AnkleRight, JointType.FootRight));

            // Pierna izquierda
            this.bones.Add(new Tuple<JointType, JointType>(JointType.HipLeft,
JointType.KneeLeft));
            this.bones.Add(new Tuple<JointType, JointType>(JointType.KneeLeft,
JointType.AnkleLeft));
            this.bones.Add(new Tuple<JointType, JointType>(JointType.AnkleLeft,
JointType.FootLeft));

            // Colores para los cuerpos
            this.bodyColors = new List<Pen>();
            this.bodyColors.Add(new Pen(Brushes.Red, 4));
            this.bodyColors.Add(new Pen(Brushes.Red, 4));
            this.bodyColors.Add(new Pen(Brushes.Red, 4));
            this.bodyColors.Add(new Pen(Brushes.Red, 4));
            this.bodyColors.Add(new Pen(Brushes.Red, 4));
            this.bodyColors.Add(new Pen(Brushes.Red, 4));

            // Crea el grupo de dibujo que se usa para dibujar.
            this.drawingGroup = new DrawingGroup();
            // Crea una fuente de imagen que se usa en el control de imagen
            this.imageSource = new DrawingImage(this.drawingGroup);
            this.Loaded += PressMilitar_Loaded;

            //Ubidots
            api = new ApiClient(apiKey);
// Inicializa el cliente de API de Ubidots.
            variableMonitoreo = api.GetVariable(variableMonitoreoId);
// Obtiene la inf de la variable de monitoreo en Ubidots
            rCorrectasPress = api.GetVariable(variableIDPressC);
// Obtiene la variable que almacena el número de repeticiones correctas
            rIncorrectasPress = api.GetVariable(variableIDPressI);
// Obtiene la variable que almacena el número de repeticiones incorrectas
            rPress = api.GetVariable(variableIDCantidadRepeticiones);
// Obtiene la referencia a la variable que almacena el número total de
repeticiones
        }

        private void EnviarValorUbidots()
//Metodo para enviar el valor a ubidots
        {
            // Obtener el valor actual de Kinect y enviarlo a Ubidots
            using (BodyFrame bodyframe = bodyFrameReader.AcquireLatestFrame())
//Se obtiene el último frame de cuerpocapturado por el sensor
            {
                if(bodyframe != null)
// Se verifica si el frame de cuerpo no es nulo
                {
```

```
                    Body[] bodies = new Body[bodyframe.BodyCount];
//Se crea un arreglo para almacenar los datos de los cuerpos detectados en el
frame
                    bodyframe.GetAndRefreshBodyData(bodies);
//Se actualiza el arreglo con los datos de los cuerpos detectados en el frame
                    foreach (var body in bodies)
//Recorre a través de cada cuerpo detectado en el frame
                    {
                        if (body.IsTracked)
//Se verifica si el cuerpo está siendo rastreado
                        {
                            // Si se detecta un cuerpo, enviar un 1 a Ubidots
                            variableMonitoreo.SaveValue(1);
                            return; // Sale del ciclo al detectar una persona
                        }
                    }
                    // Si no se detecta ningún cuerpo, enviar un 0 a Ubidots
                    variableMonitoreo.SaveValue(0);
                }
            }
        }

        public ImageSource ImageSource1                          //Declaracion
de propiedad
        {
            get
            {
                return this.MapaBits;                           //Permite
acceder a los frames a color y actualizar con los movimientos
            }
        }
        public ImageSource ImageSource                          //Declaracion de
propiedad
        {
            get
            {
                return this.imageSource;                       //Permite acceder
a los frames a color y actualizar con los movimientos
            }
        }
        private void PressMilitar_Loaded(object sender, RoutedEventArgs e)
        {
            if (this.bodyFrameReader != null)
//Verifica que la entrada de frames no sea nula
            {
                this.bodyFrameReader.FrameArrived +=
BodyFrameReader_FrameArrived; // Suscribe el método
BodyFrameReader_FrameArrived al evento FrameArrived del lector de frames de
cuerpo
            }
        }

        //Se crea un método para apagar el kinect y liberar el espacio de la
memoria RAM.

        private void PressMilitar_Closed(object sender, EventArgs e)
        {
            if (this.LecturaFrameColor != null & this.bodyFrameReader != null)
```

```
            {
                // Llama a Dispose para liberar los recursos del lector de
frames de cuerpo.
                this.LecturaFrameColor.Dispose();
                this.LecturaFrameColor = null;
                // Llama a Dispose para liberar los recursos del lector de
frames de color
                this.bodyFrameReader.Dispose();
                this.bodyFrameReader = null;
            }
            if (this.kinect != null)           // Verifica si el sensor Kinect
no es nulo
            {
                // this.kinect.Close();
                this.kinect = null;           // Establece el sensor Kinect a
null
            }
        }

        private void LecturaFrameColor_FrameArrived(object sender,
ColorFrameArrivedEventArgs e) //Metodo para frames de color
        {
            using (ColorFrame colorFrame = e.FrameReference.AcquireFrame())
//Se adquiere el frame mas reciente
            {
                if (colorFrame != null)
//Condicion cuando se obtiene un frame
                {
                    FrameDescription colorFrameDescription =
colorFrame.FrameDescription;        //Se obtiene la descripcion del frame
                    using (KinectBuffer colorBuffer =
colorFrame.LockRawImageBuffer()) ;         //bloqueo de bufer cuando se esta
procesando un frame
                    {
                        this.MapaBits.Lock();
//Bloqueo de bitmapa
                        if ((colorFrameDescription.Width ==
this.MapaBits.PixelWidth) && (colorFrameDescription.Height ==
this.MapaBits.PixelHeight))    //Verificacion de dimesiones del frame
                        {
                            //Copia de datos al bitmapa
                            colorFrame.CopyConvertedFrameDataToIntPtr(
//Llamado de metodo para copiar los datos de la imagen al bufer
                                        this.MapaBits.BackBuffer,
//Buffer para dibujar los datos de imagen
                                        (uint)(colorFrameDescription.Width *
colorFrameDescription.Height * 4),    //Cantidad de datos a copiar
                                        ColorImageFormat.Bgra);
//Se especifica el formato de color a usar
                            this.MapaBits.AddDirtyRect(new Int32Rect(0, 0,
this.MapaBits.PixelWidth, this.MapaBits.PixelHeight)); //Actualizacion de
interfaz al usuario
                        }
                        this.MapaBits.Unlock();
//Desbloqueo de bitmapa
                    }
                }
            }
```

```
        }
        public static int Moda(List<int> values)                                //
Método para calcular la moda de la lista de angulos
        {
            return values.GroupBy(v => v)                                       //
Agrupa los valores de la lista por su valor
                         .OrderByDescending(g => g.Count())                     //
Ordena los grupos en orden descendente por el número de elementos en cada grupo
                         .First()                                               //
Toma el primer grupo (el que tiene más elementos)
                         .Key;                                                  //
Devuelve el valor que más se repite en la lista
        }
        private async void BodyFrameReader_FrameArrived(object sender,
BodyFrameArrivedEventArgs e)                        //Metodo para la lectura de
frames del cuerpo
        {
            if (!activacionCuerpo)
//Verificacion para activacion de lectura de datos.
                return;
            bool datosrecibidos = false;
// Indica si se han recibido datos de cuerpo
            using (BodyFrame bodyFrame = e.FrameReference.AcquireFrame())
// Se adquiere el frame de cuerpo más reciente
            {
                if (bodyFrame != null)
// Se comprueba si el frame de cuerpo adquirido no es nulo
                {
                    if (this.bodies == null)
// Inicializa el array de cuerpos si es nulo
                    {
                        this.bodies = new Body[bodyFrame.BodyCount];
                    }
                    //Se llena el array de cuerpos con los datos del frame de
cuerpo
                    bodyFrame.GetAndRefreshBodyData(this.bodies);
                    datosrecibidos = true;
                }
            }
            if (datosrecibidos)
//Si lectura de frames de cuerpo está activa
            {
                using (DrawingContext dc = this.drawingGroup.Open())
//Creacion de bloque para dibujar cada cuerpo
                {
                    dc.DrawRectangle(Brushes.Transparent, null, new Rect(0.0,
0.0, this.displayWidth, this.displayHeight)); //Dibujar el rectangulo en la
interfaz

                    int colorLapiz = 0;
                    foreach (Body body in this.bodies)
//Iteracion de bucle para cada cuerpo
                    {
                        Pen dibujarEsfero = this.bodyColors[colorLapiz++];
//Recorrido de colores para dibjar cada cuerpo

                        if (body.IsTracked)
//Condicion si el cuerpo es detectado
```

```
                    {
                        this.DrawClippedEdges(body, dc);
//Ajusta bordes del cuerpo para establecer los limites de deteccion
                        IReadOnlyDictionary<JointType, Joint> joints =
body.Joints;                                    //Declaracion de variable
"joints" para acceder a los valores de las articulaciones
                        Dictionary<JointType, Point> jointPoints = new
Dictionary<JointType, Point>();                  //Declaracion de diccionario
para acceder a los valores de las articulaciones

                        //Coversion de las articulaciones de un espacio de
3D a 2D para visualizacion del usuario
                        foreach (JointType jointType in joints.Keys)
//Iteracion de cada articulacion en el diccionario
                        {
                            CameraSpacePoint posicion =
joints[jointType].Position;                                     //Variable para
representar una posicion tridimencional
                            if (posicion.Z < 0)
//Verificacion que el valor de la componente z menor a 0
                            {
                                posicion.Z = InferredZPositionClamp;
//Ajuste de valor predeterminado, para que el valor z no sea confiable
                            }
                            DepthSpacePoint depthSpacePoint =
this.coordinateMapper.MapCameraPointToDepthSpace(posicion); //Estructura para
representar la posicion en tridimensional en bidimensional
                            jointPoints[jointType] = new
Point(depthSpacePoint.X, depthSpacePoint.Y);
//Almacenamiento de las posiciones convertidas y poder accerde mediante Point
                        }
                        this.DibujarCuerpo(joints, jointPoints, dc,
dibujarEsfero);


//************************************************************Procesamiento
validacion de ejercicio y tipo de
repeticiones*************************************************************
**//

                        Joint hombroDerecho =
body.Joints[JointType.ShoulderRight];           //Obetener articulaciones del
hombro derecho
                        Joint hombroIzquierdo =
body.Joints[JointType.ShoulderLeft];          //Obetener articulaciones del
hombro izquierdo
                        Joint codoDerecho =
body.Joints[JointType.ElbowRight];                // Obtener las articulaciones
del codo derecho
                        Joint munecaDerecha =
body.Joints[JointType.WristRight];              // Obtener las articulaciones
de la mano derecha
                        Joint codoIzquierdo =
body.Joints[JointType.ElbowLeft];               // Obtener las articulaciones
del codo izquierdo
                        Joint munecaIzquierda =
body.Joints[JointType.WristLeft];             // Obtener las articulaciones de
la mano izquierda
```

```
                            //Lado Derecho
                            Console.WriteLine($"Hombro derecho:
X={hombroDerecho.Position.X}, Y={hombroDerecho.Position.Y},
Z={hombroDerecho.Position.Z}");
                            Console.WriteLine($"Codo derecho:
X={codoDerecho.Position.X}, Y={codoDerecho.Position.Y},
Z={codoDerecho.Position.Z}");
                            Console.WriteLine($"Muneca derecha:
X={munecaDerecha.Position.X}, Y={munecaDerecha.Position.Y},
Z={munecaDerecha.Position.Z}");
                            //Lado izquierdo'
                            Console.WriteLine($"Hombro izquierdo:
X={hombroIzquierdo.Position.X}, Y={hombroDerecho.Position.Y},
Z={hombroDerecho.Position.Z}");
                            Console.WriteLine($"Codo izquierdo:
X={codoIzquierdo.Position.X}, Y={codoIzquierdo.Position.Y},
Z={codoIzquierdo.Position.Z}");
                            Console.WriteLine($"Muneca izquierda:
X={munecaIzquierda.Position.X}, Y={munecaIzquierda.Position.Y},
Z={munecaIzquierda.Position.Z}");

                            // Calcula el ángulo
                            anguloDerecho =
CalcularAngulo(hombroDerecho.Position,codoDerecho.Position,
munecaDerecha.Position);
                            anguloIzquierdo =
CalcularAngulo(hombroIzquierdo.Position,codoIzquierdo.Position,
munecaIzquierda.Position);

                            //Impresion de angulos
                            // Imprime el ángulo calculado en la consola
                            // Imprime el ángulo calculado en la consola
                            Console.WriteLine($"Angulo de Brazo Derecho
calculado: {anguloDerecho} °");
                            Console.WriteLine($"Angulo de Brazo Izquierdo
calculado: {anguloIzquierdo} °");

                            //Controlador de angulos
                            bool posicionInicialCorrecta =
EstaEnRango(anguloDerecho, AnguloInicialCorrectoMin, AnguloInicialCorrectoMax)
&&                 //Variable para verificar la posicion inicial correcta

EstaEnRango(anguloIzquierdo, AnguloInicialCorrectoMin,
AnguloInicialCorrectoMax);

                            bool posicionInicialIncorrecta =
EstaEnRango(anguloDerecho, AnguloInicialBajoIncorrectoMin,
AnguloInicialBajoIncorrectoMax) ||  //Variable para verificar la posicion
inicial incorrecta

EstaEnRango(anguloDerecho, AnguloInicialAltoIncorrectoMin,
AnguloInicialAltoIncorrectoMax) ||

EstaEnRango(anguloIzquierdo, AnguloInicialBajoIncorrectoMin,
AnguloInicialBajoIncorrectoMax) ||
```

```
EstaEnRango(anguloIzquierdo, AnguloInicialAltoIncorrectoMin,
AnguloInicialAltoIncorrectoMax);
                            //Posicion Final

                            bool posicionFinalCorrecta =
EstaEnRango(anguloDerecho, AnguloFinalCorrectoMin, AnguloFinalCorrectoMax) &&
//Variable para verificar la posicion final correcta

EstaEnRango(anguloIzquierdo, AnguloFinalCorrectoMin, AnguloFinalCorrectoMax);

                            bool posicionFinalIncorrecta =
EstaEnRango(anguloDerecho, AnguloFinalBajoIncorrectoMin,
AnguloFinalBajoIncorrectoMax) ||        //Variable para verificar la posicion
final incorrecta

EstaEnRango(anguloDerecho, AnguloFinalAltoIncorrectoMin,
AnguloFinalAltoIncorrectoMax) ||

EstaEnRango(anguloIzquierdo, AnguloFinalBajoIncorrectoMin,
AnguloFinalBajoIncorrectoMax) ||

EstaEnRango(anguloIzquierdo, AnguloFinalAltoIncorrectoMin,
AnguloFinalAltoIncorrectoMax);

                            // Proceso para contar una repetición
                            //Posicion Inicial
                            if (estadoEjercicio ==
EstadoEjercicio.EnPosicionInicial && EstaEnRango(anguloDerecho,
AnguloInicialMin, AnguloInicialMax) &&   //Condición para validar posicion
inicial
                                EstaEnRango(anguloIzquierdo, AnguloInicialMin,
AnguloInicialMax))
                            {
                                contadorFramesF = 0;
//Se resetea el contador de frames al cambiar de estado
                                contadorAnguloFinalIncorrectos = 0;
//Se reseta los contadores de posicion final
                                contadorAnguloFinalCorrectos = 0;
//Se resetea los contadores de posicion final
                                if (contadorFramesI == 0)
//Condición para reiniciar los valores de ángulos y moda calculados.
                                {
                                    modaAnguloDerechoIC = 0;
                                    modaAnguloIzquierdoIC = 0;
                                    modaAnguloDerechoII = 0;
                                    modaAnguloIzquierdoII = 0;
                                    anguloDerechoC = 0;
                                    anguloIzquierdoC = 0;
                                    anguloDerechoI = 0;
                                    anguloIzquierdoI = 0;
                                    var ultimoControlInicial =
ControlRepeticionesIniciales.LastOrDefault();  // Se obtiene el último control
de repeticiones iniciales de la colección
                                    if (ultimoControlInicial != null)
// Se verifica si se encontró un último control de repeticiones iniciales
                                    {
```

```
ultimoControlInicial.AngulosDerechosIC.Clear();                         // Se
limpia la colección de ángulos derechos IC

ultimoControlInicial.AngulosIzquierdosIC.Clear();                       // Se
limpia la colección de ángulos izquierdos IC

ultimoControlInicial.AngulosDerechosII.Clear();                         // Se
limpia la colección de ángulos derechos II

ultimoControlInicial.AngulosIzquierdosII.Clear();                       // Se
limpia la colección de ángulos izquierdos II
                                    }
                                    var nuevaPosicionInicial = new
ControlReportePosicionInicial
                                    {
                                        NumeroRepeticionT =
numeroTotalRepeticiones,                            // Se establece el número
total de repeticiones
                                        EstadoPosicionInicial = "Posicion
Inicial",                            // Se establece el estado de la posicion
                                        EstadoEvaluacionI = "",
// Se establece el estado de evaluacion por defecto
                                        ModaAnguloDerechoIC = 0,
// Se reinicia el valor de moda
                                        ModaAnguloIzquierdoIC = 0,
// Se reinicia el valor de moda
                                        ModaAnguloDerechoII = 0,
// Se reinicia el valor de moda
                                        ModaAnguloIzquierdoII = 0,
// Se reinicia el valor de moda
                                    };
                                    Dispatcher.Invoke(() =>
                                    {

ControlRepeticionesIniciales.Add(nuevaPosicionInicial);
                                    });
                                }
                                contadorFramesI++;
                                if (contadorFramesI >=1 && contadorFramesI <=
90)                                                 //Variable a verificar
                                {
                                    if (posicionInicialCorrecta)
                                    {
                                        anguloDerechoC =
CalcularAngulo(hombroDerecho.Position, codoDerecho.Position,
munecaDerecha.Position);
                                        anguloIzquierdoC =
CalcularAngulo(hombroIzquierdo.Position, codoIzquierdo.Position,
munecaIzquierda.Position);
                                        contadorAnguloInicialCorrectos++;

ControlRepeticionesIniciales.LastOrDefault()?.AngulosDerechosIC.Add(anguloDerec
hoC);       //Accede y añade valores a las listas angulo derecho C

ControlRepeticionesIniciales.LastOrDefault()?.AngulosIzquierdosIC.Add(anguloIzq
uierdoC);   //Accede y añade valores a las listas angulo izquiero C
                                    }
```

```
                                    if (posicionInicialIncorrecta)
                                    {
                                        anguloDerechoI =
CalcularAngulo(hombroDerecho.Position, codoDerecho.Position,
munecaDerecha.Position);
                                        anguloIzquierdoI=
CalcularAngulo(hombroIzquierdo.Position, codoIzquierdo.Position,
munecaIzquierda.Position);
                                        contadorAnguloInicialIncorrectos++;

ControlRepeticionesIniciales.LastOrDefault()?.AngulosDerechosII.Add(anguloDerec
hoI);      //Accede y añade valores a las listas angulo derecho I

ControlRepeticionesIniciales.LastOrDefault()?.AngulosIzquierdosII.Add(anguloIzq
uierdoI);  //Accede y añade valores a las listas angulo izquierdo I
                                    }
                                    if (contadorFramesI ==
FramesParaTresSegundos)
//Condicion cuando los frames llegan a 90
                                    {
                                        anguloDerecho = 0;
                                        anguloIzquierdo = 0;
                                        if (contadorAnguloInicialCorrectos >=
46)                                          //Condición para activar el
contador de posicion inicial correcta
                                        {
                                            modaAnguloDerechoIC =
Moda(ControlRepeticionesIniciales.LastOrDefault().AngulosDerechosIC);
//Cálculo de moda angulosderechosIC
                                            modaAnguloIzquierdoIC =
Moda(ControlRepeticionesIniciales.LastOrDefault().AngulosIzquierdosIC);//Cálcul
o de moda angulosiozquierdoIC
                                            estadoEvaluacionI = "Correcto";
                                            correctas++;
//Aumento de contador para la posicion correcta
                                            Dispatcher.Invoke(() =>
                                            {
                                                //Actualización de valores en
PDF

ControlRepeticionesIniciales.LastOrDefault().EstadoEvaluacionI =
estadoEvaluacionI;

ControlRepeticionesIniciales.LastOrDefault().ModaAnguloDerechoIC =
modaAnguloDerechoIC;

ControlRepeticionesIniciales.LastOrDefault().ModaAnguloIzquierdoIC =
modaAnguloIzquierdoIC;
                                            });
                                        }
                                        if (contadorAnguloInicialIncorrectos >
44)//Condición para activar posicion incorrecta
                                        {
                                            modaAnguloDerechoII =
Moda(ControlRepeticionesIniciales.LastOrDefault().AngulosDerechosII);
//Cálculo de moda angulosderechosII
```

```
                                            modaAnguloIzquierdoII =
Moda(ControlRepeticionesIniciales.LastOrDefault().AngulosIzquierdosII);
//Cálculo de moda angulosizquierdosII
                                            estadoEvaluacionI = "Incorrecto";
                                            incorrectas++;

//*************************************************************************
**************************************************//
                                            string detalle =
DetalleErrorPosicionInicial(modaAnguloDerechoII, modaAnguloIzquierdoII);
//Creación de variable, almacena el detalle de Pos.I
                                            var estadoIncorrectoI = new
ControlRepeticionesI                                        //Creación
de instancia para asignar informacion
                                            {
                                                NumeroRepeticionI =
numeroTotalRepeticiones,                                            //Se
agrega el valor de repeticion tabla control
                                                EstadoPosicionI = "Posicion
Inicial",                                       //Se agrega el
valor de posicion tabla control
                                                DetalleAnguloI = detalle
//Se agrga el detalle de posicion tabla control
                                            };

                                            Dispatcher.Invoke(() =>
//Actualización UI
                                            {

ControlRepeticionesIncorrectasI.Add(estadoIncorrectoI);
                                            });
                                            Dispatcher.Invoke(() =>
                                            {

ControlRepeticionesIniciales.LastOrDefault().EstadoEvaluacionI =
estadoEvaluacionI;

ControlRepeticionesIniciales.LastOrDefault().ModaAnguloDerechoII =
modaAnguloDerechoII;

ControlRepeticionesIniciales.LastOrDefault().ModaAnguloIzquierdoII =
modaAnguloIzquierdoII;
                                            });
                                        }//Fin condicional incorrecta
                                        estadoEjercicio =
EstadoEjercicio.EnPosicionFinal; // Cambia el estado
                                        //Aviso cambio de posicion UI
                                        Dispatcher.Invoke(() =>
                                        {
                                            txtIndicacion.Text = "EXTIENDA LOS
BRAZOS";
                                            // Considera la posibilidad de usar
un timer o un método asincrónico para borrar el mensaje después de un breve
periodo
                                            Task.Delay(2000).ContinueWith(t =>
Dispatcher.Invoke(() => txtIndicacion.Text = ""));
                                        });
                                    }
```

```
                                    Console.WriteLine($"*****POSICION
INICIAL*****");
                                    Console.WriteLine($"Contador de Frames
Posicion Inicial: {contadorFramesI}");
                                    Console.WriteLine($"Contador de angulos
iniciales Correctos: {contadorAnguloInicialCorrectos}");
                                    Console.WriteLine($"Contador de angulos
iniciales Incorrectos: {contadorAnguloInicialIncorrectos}");
                                    Console.WriteLine($"Angulo DERECHO
CORRECTO: {anguloDerechoC}");
                                    Console.WriteLine($"Angulo DERECHO
INCORRECTO: {anguloDerechoI}");
                                    Console.WriteLine($"Angulo IZQUIERDO
CORRECTO: {anguloIzquierdoC}");
                                    Console.WriteLine($"Angulo IZQUIERDO
INCORRECTO: {anguloIzquierdoI}");
                                    Console.WriteLine($"MODA ANGULO INCIAL
CORRECTO DERECHO: {modaAnguloDerechoIC}");
                                    Console.WriteLine($"MODA ANGULO INCIAL
INCORRECTO DERECHO: {modaAnguloDerechoII}");
                                    Console.WriteLine($"MODA ANGULO INCIAL
CORRECTO IZQUIERDO: {modaAnguloIzquierdoIC}");
                                    Console.WriteLine($"MODA ANGULO INCIAL
INCORRECTO IZQUIERDO: {modaAnguloIzquierdoII}");
                                    Console.WriteLine("Lista de ángulos
derechos II:");
                                    var listaAngulosDerechosII =
ControlRepeticionesIniciales.LastOrDefault()?.AngulosDerechosII;
                                    foreach (var anguloII in
listaAngulosDerechosII)
                                    {
                                        Console.WriteLine(anguloII);
                                    }
                                }
                            }
                            //Inicio estado Final
                            if (estadoEjercicio ==
EstadoEjercicio.EnPosicionFinal && EstaEnRango(anguloDerecho, AnguloFinalMin,
AnguloFinalMax) &&  //Activacion condicion final
                                EstaEnRango(anguloIzquierdo, AnguloFinalMin,
AnguloFinalMax))
                            {
                                contadorFramesI = 0;                        //
Resetea el contador de frames al cambiar de estado
                                contadorAnguloInicialIncorrectos = 0;
//Resetea loa contadores de angulos iniciales
                                contadorAnguloInicialCorrectos = 0;
//Resetea loa contadores de angulos iniciales
                                if (contadorFramesF == 0)
//Condicion cuando el contador de frames es 0
                                {
                                    anguloDerechoC = 0;
//Valor de angulo derecho correcto a 0
                                    anguloIzquierdoC = 0;
//Valor de angulo izquierdo correcto a 0
                                    anguloDerechoI = 0;
//Valor de angulo derecho incorrecto a 0
```

```
                                    anguloIzquierdoI = 0;
//Valor de angulo izquierdo incorrecto a 0
                                    var ultimoControlFinal =
ControlRepeticionesFinales.LastOrDefault();  //Variable para limpiar los
valores del pdf
                                    //Actualizacion de valores a 0
                                    if (ultimoControlFinal != null)
                                    {

ultimoControlFinal.AngulosDerechosFC.Clear();

ultimoControlFinal.AngulosIzquierdosFC.Clear();

ultimoControlFinal.AngulosDerechosFI.Clear();

ultimoControlFinal.AngulosIzquierdosFI.Clear();
                                    }
                                    //Actualizacion de listas a valores por
defecto
                                    var nuevaPosicionFinal = new
ControlReportePosicionFinal
                                    {
                                        NumeroRepeticionT =
numeroTotalRepeticiones,
                                        EstadoPosicionFinal = "Posicion Final",
                                        EstadoEvaluacionF = "",
                                        ModaAnguloDerechoFC = 0,
                                        ModaAnguloIzquierdoFC = 0,
                                        ModaAnguloDerechoFI = 0,
                                        ModaAnguloIzquierdoFI = 0
                                    };
                                    //Actualizacion el UI
                                    Dispatcher.Invoke(() =>
                                    {

ControlRepeticionesFinales.Add(nuevaPosicionFinal);
                                    });
                                }
                                contadorFramesF++;
//aumento de contador de frames
                                if (contadorFramesF >= 1 && contadorFramesF <=
90)          //Recorrido de los 90 frames
                                {
                                    if(posicionFinalCorrecta)
//Condicion que activa los contadores de angulos correctos
                                    {
                                        anguloDerechoC =
CalcularAngulo(hombroDerecho.Position, codoDerecho.Position,
munecaDerecha.Position);          //Valores para calcular angulo derecho C
                                        anguloIzquierdoC =
CalcularAngulo(hombroIzquierdo.Position, codoIzquierdo.Position,
munecaIzquierda.Position);  //Valores para calcular angulo izquierdo C
                                        contadorAnguloFinalCorrectos++;
//Aumento de contador de ángulos correctos

ControlRepeticionesFinales.LastOrDefault()?.AngulosDerechosFC.Add(anguloDerecho
C);                              //Agregacion de angulo derechosC a la lista
```

```
ControlRepeticionesFinales.LastOrDefault()?.AngulosIzquierdosFC.Add(anguloIzqui
erdoC);                          //Agregacion de angulo izquierdosC a la lista
                                    }
                                    if (posicionFinalIncorrecta)  //Condicion
que activa los contadores de angulos incorrectos.
                                    {
                                        anguloDerechoI =
CalcularAngulo(hombroDerecho.Position, codoDerecho.Position,
munecaDerecha.Position);          //Valores para calcular angulo derecho In
                                        anguloIzquierdoI =
CalcularAngulo(hombroIzquierdo.Position, codoIzquierdo.Position,
munecaIzquierda.Position);  //Valores para calcular angulo izquierdo In
                                        contadorAnguloFinalIncorrectos++;
//Aumento de contador de ángulos correctos

ControlRepeticionesFinales.LastOrDefault()?.AngulosDerechosFI.Add(anguloDerecho
I);                              //Agregacion de angulos derechosIn a la lista

ControlRepeticionesFinales.LastOrDefault()?.AngulosIzquierdosFI.Add(anguloIzqui
erdoI);                          //Agregacion de angulos izquierdosIn a la
lista
                                    }
                                    if (contadorFramesF ==
FramesParaTresSegundos)//Condicion cuando el contador llega a los 90 frames
                                    {
                                        anguloDerecho = 0;
//Se reinicia los valores de angulo derecho
                                        anguloIzquierdo = 0;
//Se reinicia los valores de angulo izquierdo
                                        if (contadorAnguloFinalCorrectos >= 46)
//Condición que ctiva la posicion final como correcta
                                        {
                                            modaAnguloDerechoFC =
Moda(ControlRepeticionesFinales.LastOrDefault().AngulosDerechosFC);
//Calculo de moda de la lista de valores de angulos derechos FC
                                            modaAnguloIzquierdoFC =
Moda(ControlRepeticionesFinales.LastOrDefault().AngulosIzquierdosFC); //Calculo
de moda de la lista de valores de angulos izquierdos FC
                                            estadoEvaluacionF = "Correcto";
//Actualizacion de estado correto
                                            correctas++;
//Aumento de contador para la posicion final correcta
                                            //Actualizacion de informacion en
el reporte pdf
                                            Dispatcher.Invoke(() =>
                                            {

ControlRepeticionesFinales.LastOrDefault().EstadoEvaluacionF =
estadoEvaluacionF;

ControlRepeticionesFinales.LastOrDefault().ModaAnguloDerechoFC =
modaAnguloDerechoFC;

ControlRepeticionesFinales.LastOrDefault().ModaAnguloIzquierdoFC =
modaAnguloIzquierdoFC;
                                            });
                                    }
```

```
                                        if (contadorAnguloFinalIncorrectos >
44) //Condición que ctiva la posicion final como incorrecta
                                        {
                                            modaAnguloDerechoFI =
Moda(ControlRepeticionesFinales.LastOrDefault().AngulosDerechosFI);
//Calculo de moda de la lista de valores de angulos derechos FIn
                                            modaAnguloIzquierdoFI =
Moda(ControlRepeticionesFinales.LastOrDefault().AngulosIzquierdosFI); //Calculo
de moda de la lista de valores de angulos izquierdos FIn
                                            estadoEvaluacionF = "Incorrecto";
//Actualizacion de estado final incorrecto
                                            incorrectas++;
//Aumento de contador para la posicion final incorrecta
                                            string detalle =
DetalleErrorPosicionFinal(modaAnguloDerechoFI, modaAnguloIzquierdoFI);
//Variable para verificar el detalle de la posicion FIn
                                            //Actualizacion de informacion en
la tabla de control
                                            var estadoIncorrectoF = new
ControlRepeticionesF
                                            {
                                                NumeroRepeticionF =
numeroTotalRepeticiones,
                                                EstadoPosicionF = "Posicion
Final",
                                                DetalleAnguloF = detalle
                                            };
                                            // Se agrega la instancia a la
colección que se muestra en la UI control
                                            Dispatcher.Invoke(() =>
                                            {

ControlRepeticionesIncorrectasF.Add(estadoIncorrectoF);

                                            });
                                            // Se agrega la instancia a la
colección que se muestra en la UI reporte
                                            Dispatcher.Invoke(() =>
                                            {

ControlRepeticionesFinales.LastOrDefault().EstadoEvaluacionF =
estadoEvaluacionF;

ControlRepeticionesFinales.LastOrDefault().ModaAnguloDerechoFI =
modaAnguloDerechoFI;

ControlRepeticionesFinales.LastOrDefault().ModaAnguloIzquierdoFI =
modaAnguloIzquierdoFI;
                                            });
                                        }  //Fin condicional incorrecta
                                        if (correctas == 2)
//Condicion para validar una repeticion completa como correcta
                                        {
                                            ReproducirSonidoP(true);
                                            repeticionesCorrectas++;  //Se
aumenta el contador de repeticiones correctas
```

```
rCorrectasPress.SaveValue(repeticionesCorrectas); //Se guarda el valor en la
plataforma en la nube Ubidots.
                                            correctas = 0;
//al finalizar los 90 frames el contador para validar una repeticion correcta
se reinicia.
                                        }
                                        //Para validar una repeticion como
incorrecta, se toma en cuenta si una de las posiciones tanto en inicial como
final es incorrecta, toda una repeticion se valida como incorrecta.
                                        if (incorrectas == 1 || incorrectas ==
2 || correctas == 1 && incorrectas == 1)  //Validacion para una repeticion
completa como incorrecta
                                        {
                                            ReproducirSonidoP(false);
                                            repeticionesIncorrectas++;  //Se
aumenta el contador de repeticiones incorrectas

rIncorrectasPress.SaveValue(repeticionesIncorrectas); //Se guarda el valor en
la platafome en la nube Ubidots
                                            incorrectas = 0;            //al
finalizar los 90 frames el contador para validar una repeticion incorrecta se
reinicia.
                                            correctas = 0;              //al
finalizar los 90 frames el contador para validar una repeticion correcta se
reinicia.
                                        }
                                        numeroTotalRepeticiones++;      //Se
aumenta el contador de repeticiones totales
                                        repeticionesRestantes--;        //Se
disminuye el valor de repeticiones totales a realizar
                                        totalRepeticionesRealizadas =
repeticionesCorrectas + repeticionesIncorrectas;  //Se suma los valores de
repeticones correctas e incorrectas
                                        ActualizarContadoresUI();      //
Actualizar valores en la paltalla de UI
                                        estadoEjercicio =
EstadoEjercicio.EnPosicionInicial; //Al finalizar las repeticiones se vuelve al
estado inicial
                                        Dispatcher.Invoke(() =>
                                        {
                                            txtIndicacion.Text = "RECOJA LOS
BRAZOS";
                                            Task.Delay(2000).ContinueWith(t =>
Dispatcher.Invoke(() => txtIndicacion.Text = ""));
                                        });
                                    }
                                    Console.WriteLine($"*****POSICION
FINAL*****");
                                    Console.WriteLine($"Contador de Frames
Posicion Inicial: {contadorFramesF}");
                                    Console.WriteLine($"Contador de angulos
iniciales Correctos: {contadorAnguloFinalCorrectos}");
                                    Console.WriteLine($"Contador de angulos
iniciales Incorrectos: {contadorAnguloFinalIncorrectos}");
                                    Console.WriteLine($"Angulo DERECHO
CORRECTO: {anguloDerechoC}");
```

```
                                    Console.WriteLine($"Angulo DERECHO
INCORRECTO: {anguloDerechoI}");
                                    Console.WriteLine($"Angulo IZQUIERDO
CORRECTO: {anguloIzquierdoC}");
                                    Console.WriteLine($"Angulo IZQUIERDO
INCORRECTO: {anguloIzquierdoI}");
                                    Console.WriteLine($"MODA ANGULO INCIAL
CORRECTO DERECHO: {modaAnguloDerechoFC}");
                                    Console.WriteLine($"MODA ANGULO INCIAL
INCORRECTO DERECHO: {modaAnguloDerechoFI}");
                                    Console.WriteLine($"MODA ANGULO INCIAL
CORRECTO IZQUIERDO: {modaAnguloIzquierdoFC}");
                                    Console.WriteLine($"MODA ANGULO INCIAL
INCORRECTO IZQUIERDO: {modaAnguloIzquierdoFI}");
                                    Console.WriteLine("Lista de ángulos
derechos FC:");
                                    var listaAngulosDerechosFC =
ControlRepeticionesFinales.LastOrDefault()?.AngulosDerechosFC;
                                    foreach (var anguloFC in
listaAngulosDerechosFC)
                                    {
                                        Console.WriteLine(anguloFC);
                                    }
                                }
                            }
                        }
                        this.drawingGroup.ClipGeometry = new RectangleGeometry(new
Rect(0.0, 0.0, this.displayWidth, this.displayHeight));   //Rectangulo del
limite de deteccion del cuerpo
                    }
                    if (repeticionesRestantes == 0)           //Condicion cuando
las repeticiones llegan a 0
                    {
                        repeticionesRestantes = valorDefecto; //El valor de
repetciones vuelve al valor seleccionado por el usuario
                        ActualizarContadoresUI();             //Se actualiza el
valor en la pantalla
                        double porcentajeIncorrectas =
(double)repeticionesIncorrectas / totalRepeticionesRealizadas;   //Se calcula
el porcenta para enviar el valor de alerta a telegram
                        if (porcentajeIncorrectas > 0.5)       //Se verifica que el
valor calculado supere el 50%
                        {
                            //Envio de alerta con el nombre y la cantidad de
repeticiones incorrectas ejecutadas.
                            await botClient.SendTextMessageAsync(chatId:
"5599983300", text: $"El usuario {nombreUsuario} ha completado
{repeticionesIncorrectas} de {totalRepeticionesRealizadas} repeticiones de
press militar de manera incorrecta");
                        }
                        numeroTotalRepeticiones = 1;           // Se reinicia el
valor a mostrar en reporte y control
                        totalRepeticionesRealizadas = 0;       //Se reinica el
valor de las repeticiones totales
                        porcentajeIncorrectas = 0;             //Se reinicia el
valor de porcentaje de repeticiones incorrectas
```

```
                    ActualizarContadoresUI();              //Se actualiza los
valorres en los campos de UI
                    BorrarCuerpo();                        //Se llama al metodo
para desaactivar la deteccion del esqueleto
                    activacionCuerpo = false;              //Se detiene el
seguimiento y procesamiento de frames
                }
            }
        }

        //Metodo para dibujar cuerpo, recibe 4 parámetros, articulaciones del
cuerpo, articulaciones en dos dimensiones, el cuadro de dibujo,boligrafo para
dibujar el cuerpo
        private void DibujarCuerpo(IReadOnlyDictionary<JointType, Joint>
joints, IDictionary<JointType, Point> jointPoints, DrawingContext
drawingContext, Pen drawingPen)
        {
            // Dibujar los huesos
            foreach (var bone in this.bones)
//Se itera sobre cada hueso en la colección de huesos
            {
                this.DibujarHuesos(joints, jointPoints, bone.Item1, bone.Item2,
drawingContext, drawingPen);  //Se llama al método DibujarHuesos para dibujar
cada hueso con las articulaciones y puntos capturados.
            }

            // Dibujar articulaciones
            foreach (JointType jointType in joints.Keys)
// Se recorre sobre cada tipo de articulación en el diccionario de
articulaciones
            {
                Brush drawBrush = null;
//Inicializa el esfero como nulo

                TrackingState trackingState = joints[jointType].TrackingState;
//Se obtiene el estado de seguimiento de la articulacion actual

                if (trackingState == TrackingState.Tracked)
//Condicion si la articulacion es rastreada
                {
                    drawBrush = this.trackedJointBrush;
//Se asigna un esfero para la articulacion rastreada
                }
                else if (trackingState == TrackingState.Inferred)
//Condicion si la articulacion rastreada es inferida
                {
                    drawBrush = this.inferredJointBrush;
//Se asigna un esfero para la articulacion inferida
                }

                if (drawBrush != null)
//Condicion si hay un esfero asignado
                {
                    drawingContext.DrawEllipse(drawBrush, null,
jointPoints[jointType], JointThickness, JointThickness);  // Se dibuja una
punto en la posición de la articulación con el pincel seleccionado
                }
            }
```

```
        }
        //Metodo borrar cuerpo
        private void BorrarCuerpo()
        {
            using (DrawingContext dc = this.drawingGroup.Open())
            {
                // Limpiar el área dibujando un rectángulo del tamaño completo
del área de dibujo con un esfero transparente
                dc.DrawRectangle(Brushes.Transparent, null, new Rect(0.0, 0.0,
this.displayWidth, this.displayHeight));
            }
        }

        // Método para dibujar los huesos entre dos articulaciones
        private void DibujarHuesos(IReadOnlyDictionary<JointType, Joint>
joints, IDictionary<JointType, Point> jointPoints, JointType jointType0,
JointType jointType1, DrawingContext drawingContext, Pen drawingPen)
        {
            Joint joint0 = joints[jointType0];     //Se obtiene la primera
articulación del diccionario usando jointType0
            Joint joint1 = joints[jointType1];     //Se obbtiene la segunda
articulación del diccionario usando jointType1
            if (joint0.TrackingState == TrackingState.NotTracked ||
joint1.TrackingState == TrackingState.NotTracked) //Condicion que verifica si
alguna de las articulaciones no es rastreada
            {
                return;                            //Si alguna de las
articulaciones no está siendo rastreada se sale del metodo
            }
            Pen drawPen = this.inferredBonePen;    //Se usa el bolígrafo para
huesos inferidos
            if ((joint0.TrackingState == TrackingState.Tracked) &&
(joint1.TrackingState == TrackingState.Tracked)) //Condicion cuando ambas
articulaciones están siendo rastreadas
            {
                drawPen = drawingPen;              //Se cambia el bolígrafo al
especificado en el parámetro si ambas articulaciones están rastreadas
            }

            drawingContext.DrawLine(drawPen, jointPoints[jointType0],
jointPoints[jointType1]);  //Se dibuja una línea entre las dos articulaciones.
        }

        // Método para dibujar bordes recortados alrededor del cuerpo rastreado
        private void DrawClippedEdges(Body body, DrawingContext drawingContext)
        {
            FrameEdges clippedEdges = body.ClippedEdges;
//Se obtiene los bordes recortados del cuerpo
            if (clippedEdges.HasFlag(FrameEdges.Bottom))
//Condicion que verifica si el borde inferior está recortado
            {
                drawingContext.DrawRectangle(
//Se dibuja un rectángulo rojo en el borde inferior
                    Brushes.Red,
//Se define el color de la linea
                    null,
//No hay borde
```

```
                    new Rect(0, this.displayHeight - ClipBoundsThickness,
this.displayWidth, ClipBoundsThickness)); //Se define la posición y tamaño del
rectángulo
            }
            if (clippedEdges.HasFlag(FrameEdges.Top))
//Condicion que verifica si el borde superior está recortado
            {
                drawingContext.DrawRectangle(
//Se dibuja un rectángulo rojo en el borde superior
                    Brushes.Red,
//Se define el color de la linea
                    null,
//No hay borde
                    new Rect(0, 0, this.displayWidth, ClipBoundsThickness));
//Se define la posición y tamaño del rectángulo
            }
            if (clippedEdges.HasFlag(FrameEdges.Left))
//Condicion que verifica si el borde izquierdo está recortado
            {
                drawingContext.DrawRectangle(
//Se dibuja un rectángulo rojo en el borde zquierdo
                    Brushes.Red,
//Se define el color de la linea
                    null,
//No hay borde
                    new Rect(0, 0, ClipBoundsThickness, this.displayHeight));
//Se define la posición y tamaño del rectángulo
            }
            if (clippedEdges.HasFlag(FrameEdges.Right))
//Condicion que verifica si el borde derecho está recortado
            {
                drawingContext.DrawRectangle(
//Se dibuja un rectángulo rojo en el borde derecho
                    Brushes.Red,
//Se define el color de la linea
                    null,
//No hay borde
                    new Rect(this.displayWidth - ClipBoundsThickness, 0,
ClipBoundsThickness, this.displayHeight)); //Se define la posición y tamaño del
rectángulo
            }
        }

        private void ReproducirSonidoP(bool esCorrecto)
        {
            try
            {
                string rutaSonido = esCorrecto ?
@"C:\Users\Lenovo\Documents\DOCUMENTOS ERIK\ESTUDIO\OCTAVO II\PRIMER
BIMESTRE\AVANCE11\Proyecto\imagenes\Acierto.wav"
                                               :
@"C:\Users\Lenovo\Documents\DOCUMENTOS ERIK\ESTUDIO\OCTAVO II\PRIMER
BIMESTRE\AVANCE11\Proyecto\imagenes\Error.wav";
                SoundPlayer player = new SoundPlayer(rutaSonido);
                player.Play();
            }
            catch (Exception ex)
            {
```

```
                Console.WriteLine("Error al reproducir el sonido: " +
ex.Message);
            }
        }

        //Metodo para calcular angulos
        private int CalcularAngulo(CameraSpacePoint puntoA, CameraSpacePoint
puntoB, CameraSpacePoint puntoC)    //Hombro "A",codo "B", muñeca "C"
        {
            //Calculo de vectores a partir de puntos
            Vector3D vectorAB = new Vector3D(puntoB.X - puntoA.X, puntoB.Y -
puntoA.Y, puntoB.Z - puntoA.Z);     //Vector hacia el hombro
            Vector3D vectorAC = new Vector3D(puntoC.X - puntoA.X, puntoC.Y -
puntoA.Y, puntoC.Z - puntoA.Z);     //Vector hacia el codo
            //Calculo de angulo por medio del producto escalar
            double productoPunto = Vector3D.DotProduct(vectorAB, vectorAC);
//Operación del producto punto entre los dos vectores
            double magnitudAB = vectorAB.Length;
//Calculo de la magnitud del vector hacia el hombro
            double magnitudAC = vectorAC.Length;
//Calculo de la magnitud del vector hacia el codo
            double cosenoAngulo = productoPunto / (magnitudAB * magnitudAC);
//Calculo del coseno del ángulo
            double anguloRadianes = Math.Acos(cosenoAngulo);
//Calculo del coseno en radianes
            //Conversion de radianes a grados
            double anguloGrados = anguloRadianes * (180.0 / Math.PI);
//Conversion de radianes a grados
            return (int)Math.Truncate(anguloGrados);
// Redondea y convierte a entero
        }

        // Método  que verifica si un valor está dentro del rango para validar
una repeticion correcta o incorrecta, posicion correcta o incorrecta en la
posicion inicial
        private bool EstaEnRango(int valor, int minimo, int maximo)  //Toma
tres valores para compararlos
        {
            return valor >= minimo && valor <= maximo;               //Devuelve
true si el valor del angulo está entre el mínimo y el máximo, de lo contrario
devuelve false.
        }

        //Método para detalles de posicion inicial incorrectas
        private string DetalleErrorPosicionInicial(int promedioAnguloDerechoII,
int promedioAnguloIzquierdoII)
        {
            bool bajoDerechoI = EstaEnRango(promedioAnguloDerechoII,
AnguloInicialBajoIncorrectoMin, AnguloInicialBajoIncorrectoMax);
//Verificacion de condicion cuando la mano derecha esta muy separada del hombro
            bool altoDerechoI = EstaEnRango(promedioAnguloDerechoII,
AnguloInicialAltoIncorrectoMin, AnguloInicialAltoIncorrectoMax);
//Verificacion de condicion cuando la mano derecha esta muy cerca del hombro
            bool bajoIzquierdoI = EstaEnRango(promedioAnguloIzquierdoII,
AnguloInicialBajoIncorrectoMin, AnguloInicialBajoIncorrectoMax); //Verificacion
de condicion cuando la mano izquierda esta muy separada del hombro
```

```
            bool altoIzquierdoI = EstaEnRango(promedioAnguloIzquierdoII,
AnguloInicialAltoIncorrectoMin, AnguloInicialAltoIncorrectoMax); //Verificacion
de condicion cuando la mano izquierda esta muy cerca del hombro

            List<string> detalles = new List<string>();
//Lista para almacenar los detalles.

            if (bajoDerechoI || bajoIzquierdoI)
            {
                if (bajoDerechoI && bajoIzquierdoI)
//Condicion si ambas manos estan muy separadas de los hombros
                {
                    detalles.Add("Ambas manos muy separadas de los hombros");
//Se agrega el detalle a la tabla de control
                }
                else
                {
                    if (bajoDerechoI)
//Condicion cuando el brazo derecho en la posicion inicial está muy separada
del hombro
                    {
                        detalles.Add("Mano derecha muy separada del hombro");
//Se agrega el detalle a la tabla de control
                    }
                    if (bajoIzquierdoI)
//Condicion cuando el brazo izquiero en la posicion inicial está muy separada
del hombro
                    {
                        detalles.Add("Mano izquierda muy separada del hombro");
//Se agrega el detalle a la tabla de control
                    }
                }
            }
            if (altoDerechoI || altoIzquierdoI)
            {
                if (altoDerechoI && altoIzquierdoI)
//Condicion cuando ambas manos se encuentran muy cerca de los hombros
                {
                    detalles.Add("Ambas manos muy cerca de los hombros");
//Se agrega el detalle a la tabla de control
                }
                else
                {
                    if (altoDerechoI)
//Condicion cuando el brazo derecho en la posicion inicial está muy cerca del
hombro
                    {
                        detalles.Add("Mano derecha muy cerca del hombro");
//Se agrega el detalle a la tabla de control
                    }
                    if (altoIzquierdoI)
//Condicion cuando el brazo izquierdo en la posicion inicial está muy cerca del
hombro
                    {
                        detalles.Add("Mano izquierda muy cerca del hombro");
//Se agrega el detalle a la tabla de control
                    }
                }
```

```
            }
            // Verificar si no se añadió ningún detalle
            if (detalles.Count == 0)
            {
                return "Ángulo fuera de rango de detección";
//Se agrega el detalle a la tabla de control
            }

            return string.Join(" Y ", detalles);
//Se agrega el detalle a la tabla de control
        }

        //Método para detalles de posicion final incorrectas
        private string DetalleErrorPosicionFinal(int promedioAnguloDerechoFI,
int promedioAnguloIzquierdoFI)
        {
            bool bajoDerechoF = EstaEnRango(promedioAnguloDerechoFI,
AnguloFinalBajoIncorrectoMin, AnguloFinalBajoIncorrectoMax);
//Verificacion de condicion cuando el brazo derecho está muy recogido en PF
            bool altoDerechoF = EstaEnRango(promedioAnguloDerechoFI,
AnguloFinalAltoIncorrectoMin, AnguloFinalAltoIncorrectoMax);
//Verificacion de condicion cuando el brazo derecho está muy extendido en PF
            bool bajoIzquierdoF = EstaEnRango(promedioAnguloIzquierdoFI,
AnguloFinalBajoIncorrectoMin, AnguloFinalBajoIncorrectoMax);   //Verificacion
de condicion cuando el brazo izquierdo está muy recogido en PF
            bool altoIzquierdoF = EstaEnRango(promedioAnguloIzquierdoFI,
AnguloFinalAltoIncorrectoMin, AnguloFinalAltoIncorrectoMax);   //Verificacion
de condicion cuando el brazo izquierdo está muy extendido en PF

            List<string> detalles = new List<string>();
            if (bajoDerechoF || bajoIzquierdoF)
            {
                if (bajoDerechoF && bajoIzquierdoF)
//Condicion cuando ambos brazos estan muy recogidos en la PF
                {
                    detalles.Add("ambos brazos muy recogidos");
//Se agrega el detalle a la tabla de control
                }
                else
                {
                    if (bajoDerechoF)
//Condicion cuando brazo derecho está muy recogido en PF
                    {
                        detalles.Add("Brazo derecho muy recogido");
//Se agrega el detalle a la tabla de control
                    }
                    if (bajoIzquierdoF)
//Condicion cuando brazo izquierdo está muy recogido en PF
                    {
                        detalles.Add("Brazo izquierdo muy recogido");
//Se agrega el detalle a la tabla de control
                    }
                }
            }
            if (altoDerechoF || altoIzquierdoF)
            {
                if (altoDerechoF && altoIzquierdoF)
//Condicion cuando ambos brazos están muy extendidos
```

```
                {
                    detalles.Add("Ambos brazos muy extendidos");
//Se agrega el detalle a la tabla de control
                }
                else
                {
                    if (altoDerechoF)
//Condicion cuando el brazo derecho está muy extendido
                    {
                        detalles.Add("Brazo derecho muy extendido");
//Se agrega el detalle a la tabla de control
                    }
                    if (altoIzquierdoF)
//Condicion cuando el brazo izquierdo está muy extendido
                    {
                        detalles.Add("Brazo izquierdo muy extendido");
//Se agrega el detalle a la tabla de control
                    }
                }
            }
            // Devolver mensaje si no hay errores
            if (detalles.Count == 0)
            {
                return "Ángulo fuera de rango de detección";
//Se agrega el detalle a la tabla de control
            }

            return string.Join(" Y ", detalles);
        }
            private void ActualizarContadoresUI()
        {
            Dispatcher.Invoke(() =>
            {
                TextCorrectas.Text = repeticionesCorrectas.ToString();
                TextIncorrectas.Text = repeticionesIncorrectas.ToString();
                TextRaRealizar.Text = repeticionesRestantes.ToString();
            });
        }

        private void TextBox_TextChanged(object sender, TextChangedEventArgs e)
        {

        }

        // Evento que se ejecuta cuando el texto en el TextBox 'TextRaRealizar'
cambia
        private void txtRaRealizar_TextChanged(object sender,
TextChangedEventArgs e)
        {
            if (int.TryParse(TextRaRealizar.Text, out int numRepeticion))
// Se convierte el valor ingresado a un valor entero
            {
                repeticionesRestantes = numRepeticion;
//Si la conversión es exitosa, asigna el valor a la variable
repeticionesRestantes
                ActualizarContadoresUI();
//Actualizacion valores de UI
            }
```

```
            else
            {
                repeticionesRestantes = 0;
//Manejo en caso de que el texto no sea un número
            }
        }
        private void btnInicio_Press_Click(object sender, RoutedEventArgs e)
//Boton Menu
        {
            //Se inicializa las variables
            contadorFramesF = 0;
            contadorFramesI = 0;
            repeticionesCorrectas = 0;
            repeticionesIncorrectas = 0;
            repeticionesRestantes = valorDefecto;
            estadoEjercicio = EstadoEjercicio.EnPosicionInicial;
            Menu menu = new Menu();
            this.Close();
//Se cierra la ventana actual
            menu.Show();
//Se muestra la nueva ventana de menú
        }

        private void BtnIniciar_Click(object sender, RoutedEventArgs e)
//Boton INICIAR
        {
            //Se inicializa las variables
            contadorFramesF = 0;
            contadorFramesI = 0;
            contadorAnguloFinalIncorrectos = 0;
            contadorAnguloFinalCorrectos = 0;
            contadorAnguloInicialIncorrectos = 0;
            contadorAnguloInicialCorrectos = 0;
            activacionCuerpo = true;  // Activa el seguimiento y procesamiento
            repeticionesCorrectas = 0;
            repeticionesIncorrectas = 0;
            numeroTotalRepeticiones = 1;
            correctas = 0;
            incorrectas = 0;
            repeticionesRestantes = valorDefecto;
            estadoEjercicio = EstadoEjercicio.EnPosicionInicial;
            ActualizarContadoresUI();
            //Limpieza de campos de tablas de control y reporte pdf.
            ControlRepeticionesIncorrectasI.Clear();
            ControlRepeticionesIncorrectasF.Clear();
            ControlRepeticionesIniciales.Clear();
            ControlRepeticionesFinales.Clear();
        }


        private void txtIndicacion_TextChanged(object sender,
TextChangedEventArgs e)
        {

        }

        private void btn8_Click(object sender, RoutedEventArgs e)
//Boton eleccion 8 repeticiones
```

```
        {
            //Se inicializa las variable de repeticiones restantes con el valor
de 8
            repeticionesRestantes = 8;
            valorDefecto = 8;
            ActualizarContadoresUI();
            rPress.SaveValue(valorDefecto);  //Envio de valor a ubidots
        }

        private void btn10_Click(object sender, RoutedEventArgs e)
//Boton eleccion 10 repeticiones
        {
            //Se inicializa las variable de repeticiones restantes con el valor
de 10
            repeticionesRestantes = 10;
            valorDefecto = 10;
            ActualizarContadoresUI();
            rPress.SaveValue(valorDefecto); //Envio de valor a ubidots
        }

        private void btn12_Click(object sender, RoutedEventArgs e)
//Boton eleccion 12 repeticiones
        {
            //Se inicializa las variable de repeticiones restantes con el valor
de 12
            repeticionesRestantes = 12;
            valorDefecto = 12;
            ActualizarContadoresUI();
            rPress.SaveValue(valorDefecto);  //Envio de valor a ubidots
        }
        private void Button_Click(object sender, RoutedEventArgs e)
//Boton eleccion 15 Repeticiones
        {
            //Se inicializa las variable de repeticiones restantes con el valor
de 15
            repeticionesRestantes = 15;
            valorDefecto = 15;
            ActualizarContadoresUI();
            rPress.SaveValue(valorDefecto);  //Envio de valor a ubidots
        }

        private void btnDetalleP_Click(object sender, RoutedEventArgs e)
        {
            var ventanaResultados = new ResultadosIncorrectos();
            ventanaResultados.DataContext = this; // Asegúrate de que
ResultadosIncorrectos tenga el DataContext adecuado para el binding
            ventanaResultados.Show();
        }

        //Metodo para generar reporte
        private void GenerarReportePdf()
        {
            string desktopPath =
Environment.GetFolderPath(Environment.SpecialFolder.Desktop);
            string filePath = Path.Combine(desktopPath,
$"PressMilitar_{nombreUsuario}{contadorReporte}.pdf");
            string imagePathLeft = Path.Combine(desktopPath,
"C:\\Users\\Lenovo\\Documents\\DOCUMENTOS ERIK\\ESTUDIO\\OCTAVO II\\PRIMER
```

```
BIMESTRE\\AVANCE 05\\Proyecto\\imagenes\\utn.jpg");  // Asumiendo la ubicación
de la imagen
            string imagePathRight = Path.Combine(desktopPath,
"C:\\Users\\Lenovo\\Documents\\DOCUMENTOS ERIK\\ESTUDIO\\OCTAVO II\\PRIMER
BIMESTRE\\AVANCE 05\\Proyecto\\imagenes\\zener.jpg");  // Asumiendo la
ubicación de la imagen

            Document document = new Document(PageSize.A4);
            try
            {
                PdfWriter.GetInstance(document, new FileStream(filePath,
FileMode.Create));
                document.Open();

                // Añadir imágenes
                Image imgLeft = Image.GetInstance(imagePathLeft);
                imgLeft.SetAbsolutePosition(30, document.Top - 10); // Ajusta
posición y tamaño según necesidad
                imgLeft.ScaleToFit(80, 80);
                document.Add(imgLeft);

                Image imgRight = Image.GetInstance(imagePathRight);
                imgRight.SetAbsolutePosition(document.PageSize.Width - 105,
document.Top - 5); // Ajusta posición y tamaño
                imgRight.ScaleToFit(80, 80);
                document.Add(imgRight);

                // Añadir título
                Paragraph titulo = new Paragraph("ZENER GYM\nReporte de
Evaluación de Ejercicio", FontFactory.GetFont("Times New Roman", 16,
Font.BOLD));
                titulo.Alignment = Element.ALIGN_CENTER;
                document.Add(titulo);
                Paragraph nombre = new Paragraph($"Usuario: {nombreUsuario}",
FontFactory.GetFont("Times New Roman", 16, Font.BOLD));
                nombre.Alignment = Element.ALIGN_CENTER;
                document.Add(nombre);

                // Espacio
                document.Add(new Paragraph("\n"));

                // Añadir subtitulo para la primera tabla
                Paragraph subTitleI = new Paragraph("Repeticiones en posición
inicial", FontFactory.GetFont("Times New Roman", 12, Font.BOLD));
                subTitleI.Alignment = Element.ALIGN_CENTER;
                document.Add(subTitleI);
                document.Add(new Paragraph("\n"));
                Paragraph rangocorrectoI = new Paragraph("Rango correcto 33° -
43°", FontFactory.GetFont("Times New Roman", 12, Font.BOLD));
                rangocorrectoI.Alignment = Element.ALIGN_CENTER;
                document.Add(rangocorrectoI);
                document.Add(new Paragraph("\n"));
                PdfPTable tableI = new PdfPTable(5);
// Ajusta número de columnas PI
                tableI.AddCell("Número");
// Columna número de repeticiones PI
                tableI.AddCell("Posición");
// Columna de posicion inicial o final
```

```
                tableI.AddCell("Ángulo brazo derecho");
// Columna para ángulos c/r brazo derecho PI
                tableI.AddCell("Ángulo brazo izquierdo");
// Columna para ángulos c/r brazo izquierdo PI
                tableI.AddCell("Estado de Posición");
// Columna para posicion correcta o incorrecta PI
                foreach (var item in ControlRepeticionesIniciales)
// Inicio de ciclo
                {
                    tableI.AddCell(item.NumeroRepeticionT.ToString());
// Se añade los valores de numero de repeticion PI
                    tableI.AddCell(item.EstadoPosicionInicial);
// Se añade el estado de posicion PI
                    if (item.EstadoEvaluacionI == "Correcto")
// Condicion si la posicion es correcta PI
                    {
                        // Mostrar sólo ángulos correctos
                        tableI.AddCell(item.ModaAnguloDerechoIC.ToString() +
"°");      // Se añade ángulos derechos correctos PI
                        tableI.AddCell(item.ModaAnguloIzquierdoIC.ToString() +
"°");    // Se añade ángulos izquierdos correctos PI
                    }
                    if (item.EstadoEvaluacionI == "Incorrecto")
// Condicion si la posicion es incorrecta PI
                    {
                        tableI.AddCell(item.ModaAnguloDerechoII.ToString() +
"°");      //Se añade ángulos izquierdos incorrectos PI
                        tableI.AddCell(item.ModaAnguloIzquierdoII.ToString() +
"°");    //Se añade ángulos izquierdos incorrectos PI
                    }
                    tableI.AddCell(item.EstadoEvaluacionI);
// Se añade el estado de posicion PI
                }
                document.Add(tableI);
// Se agrega la tabla PI al documento pdf PI

                // Espacio entre tablas
                document.Add(new Chunk("\n"));
                // Añadir subtitulo para la segunda tabla
                Paragraph subTitleF = new Paragraph("Repeticiones en posición
final", FontFactory.GetFont("Times New Roman", 12, Font.BOLD));
                subTitleF.Alignment = Element.ALIGN_CENTER;
                document.Add(subTitleF);
                document.Add(new Chunk("\n"));
                Paragraph rangocorrectoF = new Paragraph("Rango correcto 7° -
14°", FontFactory.GetFont("Times New Roman", 12, Font.BOLD));
                rangocorrectoF.Alignment = Element.ALIGN_CENTER;
                document.Add(rangocorrectoF);
                document.Add(new Paragraph("\n"));

                PdfPTable tableF = new PdfPTable(5);
// Ajusta número de columnas
                tableF.AddCell("Número");
// Columna número de repeticiones Pf
                tableF.AddCell("Posición");
// Columna de posicion inicial o final
                tableF.AddCell("Ángulo brazo derecho");
// Columna para ángulos c/r brazo derecho PF
```

```
                tableF.AddCell("Ángulo brazo izquierdo");
// Columna para ángulos c/r brazo izquierdo PF
                tableF.AddCell("Estado de Posición");
// Columna para posicion correcta o incorrecta PF
                foreach (var item in ControlRepeticionesFinales)
//Inicio de ciclo
                {
                    tableF.AddCell(item.NumeroRepeticionT.ToString());
// Se añade los valores de numero de repeticion PF
                    tableF.AddCell(item.EstadoPosicionFinal);
// Se añade el estado de posicion PF
                    if (item.EstadoEvaluacionF == "Correcto")
// Condicion si la posicion es correcta PF
                    {
                        tableF.AddCell(item.ModaAnguloDerechoFC.ToString() +
"°");       // Se añade ángulos derechos correctas PF
                        tableF.AddCell(item.ModaAnguloIzquierdoFC.ToString() +
"°");     // Se añade ángulos izquierdos correctos PF
                    }
                    if (item.EstadoEvaluacionF == "Incorrecto")
// Condicion si la posicion F es incorrecta
                    {
                        tableF.AddCell(item.ModaAnguloDerechoFI.ToString() +
"°");       //Se añade ángulos izquierdos incorrectos
                        tableF.AddCell(item.ModaAnguloIzquierdoFI.ToString() +
"°");     //Se añade ángulos izquierdos incorrectos
                    }
                    tableF.AddCell(item.EstadoEvaluacionF);
// Se añade el estado de posicion F
                }
                document.Add(tableF);
// Se agrega la tabla al documento pdf
            }
            catch (Exception ex)
            {
                MessageBox.Show("No se pudo generar el reporte PDF: " +
ex.Message);    //Notificacion de error si el reporte no se pudo generar.
            }
            finally
            {
                document.Close();
// Cierra el documento PDF para liberar recursos
            }
            MessageBox.Show("Reporte generado con éxito en: " + filePath);
//Notificacion reporte generado con exito, seguido de la ubicacion
            contadorReporte++;
//Se aumenta el contador de reporte
        }
        private void btnReportePdf_Click(object sender, RoutedEventArgs e)
//Metodo generar reporte
        {
            GenerarReportePdf();    //Se llama al método GenerarReportePdf para
generar el PDF del reporte
        }
    }
}
```

Anexo 10G. Conexión base de datos

```
using MySql.Data.MySqlClient;

namespace Proyecto
{
    internal class Conexion
    {
        public static MySqlConnection GetConnection()
//Agreagacion de un metodo publoc
        {
            string servidor = "localhost";
//Nombre del servidor
            string puerto = "3306";
//Puerto de comunicación
            string usuario = "root";
//Nombre de usuario
            string password = "eafloresm1";
//Contraseña de ingreso
            string bd = "sistema_usuarios";
//Nombre de la base de datos
            //Establecimiento de cadena de conexión
            string cadenaconexion = "server =" + servidor + "; port =" + puerto
+ "; user id =" + usuario + "; password =" + password + "; database =" + bd;

            MySqlConnection conexion = new MySqlConnection(cadenaconexion);
//Creación de objeto para la conexion
            return conexion;
//Devuelve el método de conexión.
        }
    }
}
```