

Memoria del proyecto

Sistema para la mejora de la movilidad articular
basada en el uso de la estimación de posturas

Trabajo Fin de Máster

Ingeniería Informática

Febrero 2025



VNiVERSiDAD
D SALAMANCA

Autor

Sergio Salinero Santamaría

Tutores

André Fílipo Sales Mendes

Gabriel Villarrubia González

CERTIFICADO DE LOS TUTORES

D. André Fílipe Sales Mendes y D. Gabriel Villarrubia González, profesores del departamento de Informática y automática de la Universidad de Salamanca,

HACEN CONSTAR:

Que el trabajo titulado “Sistema para la mejora de la movilidad articular basada en el uso de la estimación de posturas”, que se presenta, ha sido realizado por Sergio Salinero Santamaría, con DNI 70838066K y constituye la memoria del trabajo realizado para la superación de la asignatura Trabajo de Fin de Máster en la titulación Máster en Ingeniería Informática de esta Universidad.

En Salamanca, a 10 de enero de 2025.

D. André Fílipe Sales Mendes

D. Gabriel Villarrubia González

RESUMEN

La creciente necesidad de realizar ejercicios físicos orientados a la rehabilitación y mejora de la movilidad de manera remota, motivada por factores como la distancia, las limitaciones de acceso a servicios especializados o la falta de recursos, ha generado una demanda social por herramientas tecnológicas que garanticen tanto la correcta ejecución de los ejercicios como un sentimiento personalizado del progreso de cada usuario.

En este contexto, el presente proyecto aprovecha los avances en visión computacional, específicamente mediante el uso de cámaras para capturar y analizar los movimientos del usuario. La plataforma emplea la tecnología de estimación de posturas conocida como BlazePose, desarrollada por Google, que permite detectar y mapear en tiempo real las posiciones articulares del cuerpo humano. A través de esta herramienta, el sistema genera un esqueleto virtual superpuesto a la imagen del usuario, simulando sus movimientos y ofreciendo retroalimentación inmediata sobre la precisión en la ejecución de los ejercicios.

Este enfoque resulta especialmente relevante en el ámbito de la rehabilitación, ya que permite detectar desalineaciones o errores en la postura que podrían comprometer los resultados terapéuticos. De este modo, los usuarios pueden corregir sus movimientos en tiempo real, garantizando que los ejercicios se realicen de manera segura y efectiva, reduciendo el riesgo de lesiones y optimizando el proceso de recuperación.

Una característica importante de la plataforma es su capacidad para ofrecer rutinas personalizadas de ejercicios, las cuales pueden ser diseñadas tanto por los usuarios como por profesionales de la salud. Estas rutinas pueden publicarse en la plataforma para que otros usuarios, con necesidades similares, las puedan ejecutar. Este componente colaborativo fomenta la creación y difusión de ejercicios útiles en diversos campos, como la rehabilitación física, la mejora de la movilidad articular, la prevención de lesiones y la mejora del rendimiento físico general.

Además de la creación de rutinas, el sistema realiza un seguimiento detallado de la actividad del usuario. Entre los parámetros monitorizados se encuentran el tipo de ejercicios realizados, las rutinas completadas, el tiempo de entrenamiento y las calorías estimadas consumidas durante las sesiones. Esta recopilación de datos permite generar estadísticas relevantes que ayudan a los usuarios a evaluar su progreso y mejorar su desempeño. Al mismo tiempo, este enfoque individualizado requiere de un sistema robusto de gestión y control de usuarios, lo cual está integrado en la plataforma para garantizar un uso eficiente y seguro de los datos.

El desarrollo del proyecto se ha llevado a cabo siguiendo las directrices del Proceso Unificado, un marco metodológico ampliamente utilizado en ingeniería de software que permite adaptar sus principios a los requisitos específicos del sistema propuesto. Este enfoque garantiza que cada etapa del desarrollo haya sido planificada y ejecutada con el objetivo de obtener una solución robusta, flexible y adaptada a las necesidades reales de los usuarios.

Palabras clave: BlazePose, MediaPipe, Visión computacional, salud, Bienestar personal.

SUMMARY

The growing need for performing physical exercises aimed at rehabilitation and improving mobility remotely, driven by factors such as distance, limited access to specialized services, or lack of resources, has created a social demand for technological tools that ensure both the correct execution of exercises and personalized progress monitoring for each user.

In this context, the present project leverages advancements in computer vision, specifically through the use of cameras to capture and analyze user movements. The platform utilizes BlazePose, a posture estimation technology developed by Google, which enables real-time detection and mapping of the human body's joint positions. Through this tool, the system generates a virtual skeleton overlaid on the user's image, simulating their movements and providing immediate feedback on the accuracy of exercise execution.

This approach is particularly relevant in the field of rehabilitation, as it allows the detection of misalignments or posture errors that could hinder therapeutic results. In this way, users can correct their movements in real time, ensuring that exercises are performed safely and effectively, reducing the risk of injuries, and optimizing the recovery process.

A key feature of the platform is its ability to provide **personalized exercise routines**, which can be designed either by users or by healthcare professionals. These routines can be published on the platform so that other users with similar needs can perform them. This collaborative component encourages the creation and dissemination of useful exercises across various fields, including physical rehabilitation, joint mobility improvement, injury prevention, and general physical performance enhancement.

In addition to routine creation, the system tracks users' activity in detail. The monitored parameters include the type of exercises performed, completed routines, training time, and estimated calories burned during the sessions. This data collection enables the generation of relevant statistics to help users assess their progress and improve their performance. At the same time, this individualized approach requires a robust user management and control system, which is integrated into the platform to ensure the efficient and secure handling of data.

The project development has been carried out in accordance with the guidelines of the **Unified Process**, a widely used framework in software engineering that allows the adaptation of its principles to the specific requirements of the proposed system. This approach ensures that each phase of the development process has been planned and executed with the aim of delivering a robust, flexible solution tailored to the real needs of users.

Palabras clave: BlazePose, MediaPipe, Computer vision, Health, Personal wellness

ÍNDICE

1. Introducción	11
2. Objetivos.....	13
2.1. Objetivos funcionales	13
2.2. Objetivos no funcionales	13
2.3. Objetivos personales	14
3. Estado del arte.....	15
3.1. Conceptos teóricos.....	15
3.1.1. Blazepose	15
3.1.2. Arquitectura de tres capas	17
3.1.3. Rest	17
3.2. Aplicaciones similares en el mercado.....	18
4. Técnicas y herramientas	20
4.1. Técnicas y herramientas empleadas en la api	20
4.1.1. Spring boot 4	20
4.1.2. Mysql	20
4.1.3. Postman	21
4.1.4. Eclipse ide for java developers.....	21
4.1.5. Visual studio code.....	21
4.2. Técnicas y herramientas empleadas en webapp	22
4.2.1. React	22
4.2.2. Material ui	22
4.3. Herramientas case	22
4.3.1. Visual paradigm	22
4.3.2. Ezestimate	23
4.3.3. Microsoft project.....	23
4.3.4. Jsdoc	23
4.3.5. Swagger	23
5. Aspectos relevantes del desarrollo	24
5.1. Marco de trabajo	24
5.2. Estimación del esfuerzo	25
5.3. Planificación temporal	26
5.4. Especificación de requisitos software	27
5.5. Análisis del sistema software.....	29
5.5.1. Modelo del dominio	30
5.5.2. Clases de análisis y descripción de la arquitectura	30
5.5.3. Realización de casos de uso en el modelo de análisis.....	31

5.6. Diseño del sistema software	32
5.6.1. Patrones arquitectónicos.....	32
5.6.2. Subsistemas de diseño	35
5.6.3. Clases de diseño	36
5.6.4. Realización de casos de uso en el modelo de diseño	40
5.6.5. Diseño de la base de datos.....	41
5.6.6. Modelo de despliegue.....	42
5.7. Implementación	43
5.7.1. Implementación del subsistema frontend	43
5.7.2. Implementación del subsistema backend.....	46
5.7.3. Implementación del subsistema datos	47
5.7.4. Implementación de BlazepPose	48
5.7.5. Implementación del sistema de reconocimiento de ejercicios	49
5.8. Pruebas.....	52
5.9. Funcionalidades del sistema	53
5.9.1. Usuario sin autenticar	53
5.9.2. Usuario autenticado	55
6. Conclusiones	70
7. Líneas futuras	71
8. Referencias.....	73

ÍNDICE DE FIGURAS

Figura 1: Tecnología de estimación de posturas.....	15
Figura 2: Aplicación similar, Kemtai	18
Figura 3: Resultados de la estimación del esfuerzo	26
Figura 4: Planificación temporal.....	27
Figura 5: Diagrama de actores.....	28
Figura 6: Diagrama de paquetes.....	28
Figura 7: Diagrama de casos de uso del paquete Gestión de rutinas	28
Figura 8: [UC-009] Elaborar una rutina	29
Figura 9: Diagrama de clases en el modelo de dominio.....	30
Figura 10: Arquitectura en el modelo de análisis.....	31
Figura 11: Diagrama de secuencia [UC-009] Elaborar una rutina	32
Figura 12: Patrón de capas.....	33
Figura 13: Patrón cliente-servidor	34
Figura 14: Patrón DAO	34
Figura 15: Patrón singleton.....	35
Figura 16: Subsistemas de diseño	35
Figura 17: Clase de diseño Servicios del subsistema Datos	36
Figura 18: Clase de diseño Controlador del subsistema Datos	37
Figura 19: Clase de diseño Modelo del subsistema Datos.....	37
Figura 20: Clase de diseño Servicios del subsistema Backend.....	38
Figura 21: Clase de diseño Controlador del subsistema Backend	38
Figura 22: Clase de diseño Modelo del subsistema Backend	39
Figura 23: Clase de diseño Utilidades del subsistema Backend	39
Figura 24: Clases de diseño Vistas del subsistema Frontend	40
Figura 25: Clase de diseño Controlador del subsistema Frontend	40
Figura 26: Realización de caso de uso [UC-009] Elaborar una rutina	41
Figura 27: Diseño de la base de datos.....	42
Figura 28: Diagrama de despliegue	42
Figura 29: Puntos clave devueltos por BlazePose	49
Figura 30: Ángulo entre dos rectas a raíz de tres puntos	50
Figura 31: Ejemplo 1: Curl de bíceps	51
Figura 32: Ejemplo 2: Flexiones.....	51
Figura 33: Ejemplo 3: Sentadillas.....	52
Figura 34: Iniciar sesión.....	53
Figura 35: Establecer contraseña	54
Figura 36: Envío del email en la recuperación de contraseña	54

Figura 37: Cambio de contraseña	55
Figura 38: Perfil de usuario	55
Figura 39: Cambio de contraseña	56
Figura 40: Eliminar cuenta	56
Figura 41: Ver categorías de ejercicios	57
Figura 42: Categoría de ejercicios	57
Figura 43: Eliminar todas las rutinas de una categoría.....	58
Figura 44: Publicar una rutina	59
Figura 45: Configuración de rutinas personales	60
Figura 46: Tecnología de estimación de posturas.....	61
Figura 47: Ejecución de rutinas.....	61
Figura 48: Gesto inicial	62
Figura 49: Estado de preparación	62
Figura 50: Comienzo de la rutina	63
Figura 51: Ejercicio completado.....	63
Figura 52: Tiempo de desahogo.....	64
Figura 53: Historial de rutinas	64
Figura 54: Eliminar todas las rutinas del historial	65
Figura 55: Estadísticas de usuario	66
Figura 56: Reiniciar las estadísticas.....	67
Figura 57: Visualización de los ejercicios.....	67
Figura 58: Ayuda sobre la gestión de ejercicios	68
Figura 59: Adición de ejercicios	69
Figura 60: Edición de un ejercicio	69
Figura 61: Eliminación de un ejercicio.....	69

1. INTRODUCCIÓN

El presente documento recoge la memoria del Trabajo de Fin de Máster titulado “Sistema para la mejora de la movilidad articular basada en el uso de la estimación de posturas”, realizado por el alumno Sergio Salinero Santamaría en la titulación Máster en Ingeniería Informática durante el curso académico 2024-2025. El proyecto ha sido dirigido por D. André Fílipe Sales Mendes y D. Gabriel Villarrubia González.

El envejecimiento de la población y el incremento de enfermedades relacionadas con la movilidad, como lesiones musculares, problemas articulares y enfermedades neurodegenerativas, han provocado una creciente demanda de programas de rehabilitación física. Sin embargo, factores como la falta de personal sanitario especializado, la escasez de infraestructura en zonas rurales y las dificultades para asistir a sesiones presenciales limitan el acceso a servicios de rehabilitación.

Al mismo tiempo, la realización de ejercicios de rehabilitación de manera autónoma en el domicilio sin una correcta supervisión puede generar resultados ineficaces o, en algunos casos, agravar las condiciones físicas del paciente debido a errores en la ejecución. Por tanto, existe una necesidad evidente de herramientas digitales que garanticen la correcta realización de los ejercicios, aporten retroalimentación en tiempo real y permitan un seguimiento preciso del progreso del usuario.

En los últimos años, los avances en visión computacional y estimación de posturas han permitido el desarrollo de aplicaciones innovadoras en el ámbito deportivo y sanitario. Estas tecnologías ofrecen la posibilidad de analizar los movimientos del cuerpo humano sin necesidad de sensores físicos, utilizando únicamente una cámara web [1-2]. Se emplea como tecnología de estimación de posturas BlazePose.

La plataforma propuesta pretende ser una herramienta innovadora en el ámbito de la salud y el bienestar físico. Al combinar tecnologías avanzadas de visión computacional con un enfoque personalizado y accesible, la solución busca optimizar los procesos de recuperación física, mejorar la calidad de vida de las personas con problemas de movilidad y facilitar el acceso a servicios de rehabilitación.

En el presente documento se expondrán los aspectos más relevantes del proyecto desarrollado, estructurado en las siguientes secciones:

- **Introducción:** Sección inicial cuya finalidad es ofrecer al lector una contextualización sobre la temática abordada, así como los objetivos y la función principal del proyecto.
- **Estado del arte:** Incluye una descripción de los conceptos teóricos necesarios para comprender adecuadamente el proyecto, así como un análisis como un análisis de las soluciones existentes en el mercado que guardan relación con el sistema desarrollado.
- **Técnicas y herramientas:** Presenta las tecnologías, técnicas y herramientas utilizadas durante el proceso de desarrollo del proyecto, destacando su utilidad y aportación.
- **Aspectos relevantes del desarrollo:** Se detallan los elementos más significativos del proceso de desarrollo del sistema, haciendo hincapié en las decisiones técnicas adoptadas y los desafíos encontrados.

- **Conclusiones:** Se presentan las conclusiones derivadas del desarrollo del proyecto, evaluando los resultados obtenidos en fusión de los objetivos planteados.
- **Líneas de trabajo futuras:** Se identifican posibles extensiones y mejoras del sistema, así como sus limitaciones actuales y el valor que podría aportar en contextos futuros.
- **Referencias:** Se listan las fuentes bibliográficas y documentales que han sido consultadas y utilizadas a lo largo del desarrollo del proyecto.

Adicionalmente, junto con este documento se entrega el código fuente del sistema desarrollado, así como su correspondiente documentación técnica, que se detalla en los siguientes anexos:

- **Anexo I. Plan de proyecto software:** Incluye la estimación del esfuerzo necesario y la planificación temporal de las actividades realizadas durante el desarrollo del proyecto.
- **Anexo II. Especificación de requisitos software:** Recoge la documentación relativa a los requisitos funcionales y no funcionales del sistema, detallando sus características y necesidades específicas.
- **Anexo III. Análisis del sistema software:** Contiene el modelo de análisis realizado sobre los requisitos, describiendo cómo se han estructurado y modelado para su implementación.
- **Anexo IV. Diseño del sistema software:** Proporciona la documentación del modelo de diseño del sistema, explicando la arquitectura adoptada y los componentes principales.
- **Anexo V. Documentación técnica:** Facilita la comprensión del código fuente, detallando su estructura, funciones implementadas y elementos técnicos relevantes.
- **Anexo VI. Manual de usuario:** Guía destinada al usuario final, que describe el procedimiento de instalación y las principales funcionalidades del sistema, facilitando su correcta interacción.

2. OBJETIVOS

El propósito central del proyecto es diseñar y desarrollar un sistema software que facilite la práctica de ejercicio físico mediante la aplicación de técnicas de estimación de posturas utilizando BlazePose. Además, el sistema permitirá analizar las actividades realizadas a través de la plataforma, así como fomentar la creación, personalización y compartición de rutinas de entrenamiento entre los usuarios.

2.1. OBJETIVOS FUNCIONALES

Los objetivos funcionales del sistema se muestran a continuación:

- **Gestión de autenticación:** El sistema deberá permitir a los usuarios establecer una contraseña e iniciar sesión. Además, deberá dar la posibilidad de cambiar su contraseña si ya habían establecido una.
- **Gestión de usuarios:** El sistema deberá gestionar la información de los usuarios, permitiendo visualizar y modificar parte de la información asociada.
- **Gestión de rutinas:** El sistema deberá gestionar la configuración de rutinas para su posterior ejecución. Además, se permitirán visualizar aquellas rutinas que sean guardadas
- **Gestión de ejercicios:** El sistema deberá gestionar la información correspondiente a los ejercicios y hacer uso de ella en el momento adecuado.
- **Gestión de estadísticas:** El sistema deberá obtener estadísticas en la ejecución de los ejercicios. Además, deberá interpretar y mostrar dichas estadísticas.

2.2. OBJETIVOS NO FUNCIONALES

Los objetivos no funcionales del sistema se muestran a continuación:

- **Seguridad:** El sistema debe asegurar la confidencialidad y protección de los datos de los usuarios que se encuentren registrados.
- **Mantenibilidad:** El sistema deberá estar diseñado de manera que facilite su mantenibilidad y permita realizar futuras actualizaciones con un mínimo de complejidad.
- **Escalabilidad:** El sistema deberá estar diseñado para gestionar de manera eficiente un incremento en la disponibilidad de recursos y en la cantidad de usuarios garantizando que no se produzcan disminuciones en el rendimiento.
- **Fiabilidad:** El sistema deberá gestionar adecuadamente cualquier interrupción o fallo, asegurando la consistencia y la integridad de los datos almacenados.
- **Rendimiento:** El sistema deberá operar optimizando al máximo el uso de los recursos computacionales disponibles.

2.3. OBJETIVOS PERSONALES

El desarrollo de este proyecto ha representado, desde el inicio, una oportunidad para poner a prueba mis capacidades en términos de aprendizaje, disciplina y constancia. He logrado llevar a cabo la creación completa de un sistema software con un nivel de complejidad significativo para un solo desarrollador, aplicando protocolos de desarrollo adquiridos durante los estudios universitarios y consolidando de manera firme los principios del Proceso Unificado.

Asimismo, este proyecto marca un hito importante, al coincidir con la culminación de mis estudios universitarios. A pesar de que las tecnologías empleadas no formaban parte del contenido previamente estudiado, los conocimientos adquiridos en el estudio me han permitido abordar y completar el desarrollo en los plazos establecidos, comprobando que los años de esfuerzo han rendido sus frutos.

Finalmente, a lo largo del proceso, he ampliado mis competencias técnicas al familiarizarme con las tecnologías actuales como React, Spring boot y BlazePose, además de reforzar mis conocimientos en lenguajes de programación como Java y JavaScript, lo que me ha permitido enriquecer aún más mi perfil profesional.

3. ESTADO DEL ARTE

3.1. CONCEPTOS TEÓRICOS

3.1.1. BlazePose

BlazePose es un modelo avanzado de estimación de posturas corporales desarrollado por Google Research y utilizado principalmente en la biblioteca MediaPipe [3-4]. Su principal objetivo es detectar y rastrear los puntos clave del cuerpo humano en tiempo real, lo que permite realizar un análisis preciso de movimientos y posturas. BlazePose ha sido diseñado específicamente para aplicaciones como fitness, deportes, rehabilitación, realidad aumentada y otras áreas donde el análisis corporal es fundamental.

Esta tecnología destaca por su capacidad para estimar con precisión la posición de 33 puntos clave del cuerpo humano, incluyendo articulaciones, extremidades y otros puntos de referencia importantes (Figura 1). Ofrece un nivel de detalle superior al resto de tecnologías similares, lo que lo hace ideal para capturar movimientos corporales completos y evaluar posturas dinámicas y complejas.

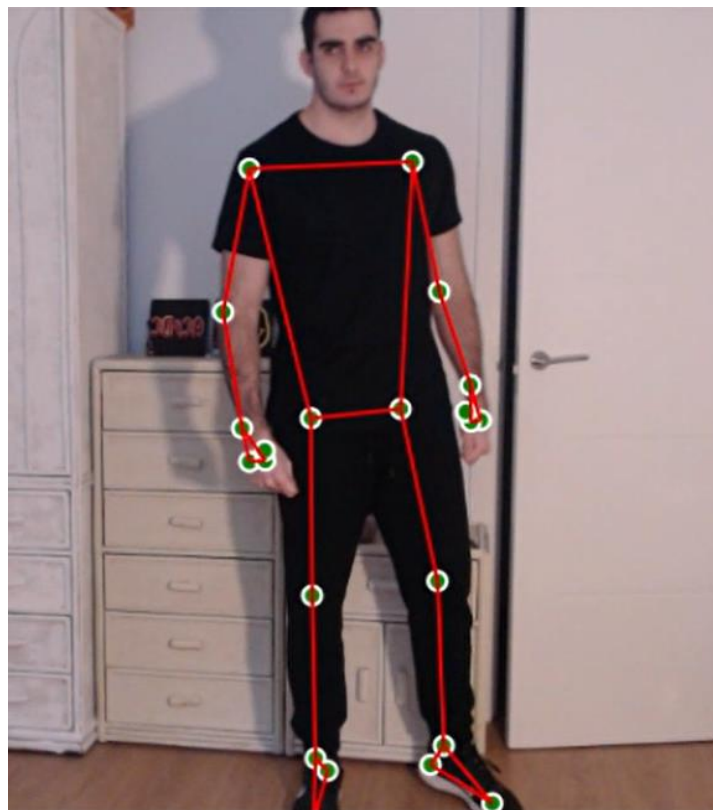


Figura 1: Tecnología de estimación de posturas

Utiliza una arquitectura basada en redes neuronales convolucionales (CNN) que ha sido optimizada para funcionar en una amplia gama de dispositivos y en tiempo real. El modelo se compone de tres etapas principales que trabajan en conjunto para ofrecer resultados precisos. La primera etapa es la detección de la persona, donde BlazePose identifica la presencia de una persona en la imagen o video. La segunda etapa es la estimación de posturas, en la que el modelo localiza los puntos clave del cuerpo detectado y los ajusta para maximizar la precisión. Finalmente, en la etapa de ajuste

refinado, BlazePose corrige y perfecciona las posiciones de los puntos clave para minimizar errores y asegurar un seguimiento preciso de los movimientos.

Este modelo presenta diversas ventajas que lo distinguen de otras soluciones disponibles. En primer lugar, BlazePose ofrece una alta precisión, ya que detecta 33 puntos clave en lugar de los 17 que suelen identificar otros modelos de estimación de posturas. Esto permite realizar un análisis más detallado en los movimientos y posturas corporales. Además, BlazePose está diseñado para realizar la detección en tiempo real, lo que lo hace ideal para aplicaciones interactivas y para su uso en dispositivos móviles. La versatilidad del modelo le permite adaptarse a diferentes escenarios, como rutinas de ejercicio, rehabilitación física, deportes y aplicaciones de realidad aumentada. Otra ventaja significativa es su eficiencia, ya que puede ejecutarse en dispositivos con recursos limitados, como smartphones, sin comprometer la precisión de los resultados.

No obstante, existen tecnologías con características similares:

- **PoseNet:** Es uno de los modelos más conocidos para la estimación de posturas corporales. Desarrollado por Google TensorFlow, está diseñado para detectar la posición de 17 puntos clave del cuerpo humano, incluidos los ojos, orejas, hombros, codos, muñecas, caderas, rodillas y tobillos [6]. Puede ejecutarse tanto en dispositivos móviles como en navegadores web utilizando TensorFlow.js, lo que lo hace muy accesible.
- **OpenPose:** Desarrollado por Carnegie Mellon University, es uno de los modelos más completos y precisos disponibles. Es capaz de detectar hasta 135 puntos clave del cuerpo, incluyendo manos, cara y pies [7]. Es ampliamente utilizado en proyectos de investigación y aplicaciones que requieren un análisis detallado de las posturas.
- **MoveNet:** Es un modelo de estimación de posturas desarrollado por Google, diseñado específicamente para aplicaciones en tiempo real. MoveNet puede detectar 17 puntos clave y está optimizado para dispositivos móviles y web, proporcionando un equilibrio entre velocidad y precisión [8].
- **MediaPipe Holistic:** Desarrollado también por Google, es una solución integral que combina la detección de cuerpo, manos y rostro. Utiliza diferentes modelos dentro de MediaPipe, como BlazePose para el cuerpo, MediaPipe Hands para las manos y MediaPipe Face Mesh para el rostro [9].
- **DeepPose:** Es uno de los primeros modelos de estimación de posturas basado en redes neuronales, desarrollado por Google. Este modelo utiliza técnicas de regresión directa para predecir las posiciones de los puntos clave [10].

A continuación, se presenta en la Tabla 1 una comparación entre las características más relevantes comunes para las tecnologías comentadas.

Modelo	Puntos clave detectados	Precisión	Velocidad	Plataformas soportadas	Casos de uso
BlazePose	33	Alta	Muy rápido	Web, móvil y escritorio	Fitness, rehabilitación, realidad aumentada y deportes
PoseNet	17	Media	Rápido	Web, móvil, escritorio	Fitness, realidad aumentada y juegos
OpenPose	Hasta 135	Muy alta	Lento	Escritorio (requiere GPU)	Investigación, deportes y rehabilitación
MoveNet	17	Alta	Muy rápido	Web y móvil	Fitness y aplicaciones móviles
MediaPipe Holistic	Cuerpo, manos y rostro	Muy alta	Rápido	Web, móvil y escritorio	Soluciones integrales de análisis corporal
DeepPose	Variable	Media	Medio	Escritorio	Investigación y posturas complejas

Tabla 1: Comparación entre las tecnologías de estimación de posturas

3.1.2. ARQUITECTURA DE TRES CAPAS

La arquitectura de tres capas es un modelo de diseño de software que divide una aplicación en tres capas independientes y separadas: la capa de presentación, la capa de lógica de negocio y la capa de datos. Esta separación permite una mejor organización del código, facilita el mantenimiento, mejora la escalabilidad y promueve una mayor seguridad.

Cada capa tiene una responsabilidad clara y específica, lo que permite que los desarrolladores trabajen de forma modular y que las aplicaciones puedan escalarse de manera más eficiente. A continuación, se describe cada una de las capas en detalle:

- **Capa de presentación (Frontend):** Es la parte de la aplicación que interactúa directamente con el usuario final. Su principal responsabilidad es mostrar la información al usuario de forma clara y amigable, y recibir las entradas que ésta realiza.
- **Capa de lógica de negocio (Backend):** Es la parte central de la arquitectura. Se encarga de procesar las reglas de negocio de la aplicación, gestionar la lógica de la aplicación y controlar el flujo de datos entre el frontend y la base de datos.
- **Capa de datos (Servidor de base de datos):** Es la encargada de almacenar y gestionar toda la información necesaria para el funcionamiento de la aplicación. Esto incluye datos de usuarios, productos, transacciones, etc.

3.1.3. REST

La Transferencia de Estado Representacional, conocida como REST, es un enfoque arquitectónico empleado en el diseño de servicios web que sigue una serie de principios y restricciones bien definidas. Este concepto fue propuesto por Roy Fielding en su tesis doctoral en el año 2000 y, desde entonces, se ha consolidado como una de las metodologías más utilizadas en el desarrollo de aplicaciones web.

Los servicios web que implementan REST se basan en el protocolo HTTP para intercambiar información y aprovechan los métodos definidos en este protocolo, como GET, POST, PUT y DELETE, para realizar distintas operaciones sobre recursos disponibles.

En este proyecto, se ha optado por utilizar una API REST para implementar los servicios debido a las numerosas ventajas que ofrece. Entre ellas, destaca su simplicidad de comunicación, ya que emplea formatos de datos estándar como JSON y XML y utiliza URIs para identificar recursos. Además, REST facilita la escalabilidad y eficiencia del sistema, al mismo tiempo que asegura una independencia de la plataforma y una interoperabilidad entre diferentes tecnologías, lo que la convierte en una solución flexible y adaptable.

3.2. APLICACIONES SIMILARES EN EL MERCADO

Kemtai es una plataforma digital de ejercicios basada en tecnologías de visión por computadora, diseñada para ofrecer una experiencia de entrenamiento personalizada y adaptable en el entorno doméstico [11]. La plataforma está orientada a proporcionar a los usuarios sesiones de ejercicio asistido, mejorando su precisión en la ejecución de movimientos mediante el uso de estimación de posturas. Además, Kemtai resulta una herramienta útil para fisioterapeutas, ya que les permite extender el seguimiento de sus pacientes más allá de la clínica, asegurando que las rutinas de ejercicios prescritas se realicen correctamente incluso sin la supervisión directa de un especialista. La aplicación también ofrece un sistema de evaluación y registro del rendimiento físico, lo que facilita el monitoreo continuo del progreso de los usuarios.

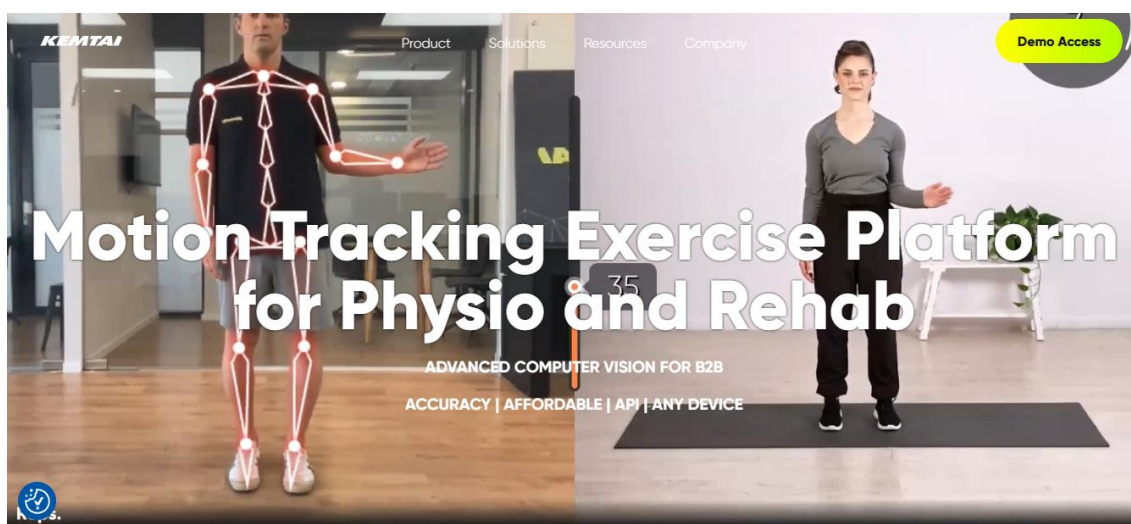


Figura 2: Aplicación similar, Kemtai

Esta aplicación incluye distintos modos de entrenamiento enfocados en diversas áreas, como movilidad, acondicionamiento físico, fuerza, flexibilidad y ejercicios básicos de fitness. En todos los casos, la tecnología de estimación de posturas constituye la base para el análisis y seguimiento de los movimientos de los usuarios.

La principal distinción entre Kemtai y la plataforma desarrollada en este proyecto radica en la incorporación de salas de entrenamiento virtuales, las cuales transforman

la propuesta en una red social de entrenamiento físico. Esta funcionalidad permite que los usuarios estén conectados entre sí, compartan sus rutinas de ejercicio y contribuyan a la creación de un entorno colaborativo y motivador en torno a la práctica de actividad física.

4. TÉCNICAS Y HERRAMIENTAS

A continuación, se detallan las técnicas, marcos de trabajo, herramientas y bibliotecas utilizadas durante el desarrollo del proyecto. Estas se agruparán en tres categorías:

- Técnicas y herramientas empleadas en la API
- Técnicas y herramientas empleadas en WebApp
- Herramientas CASE

4.1. TÉCNICAS Y HERRAMIENTAS EMPLEADAS EN LA API

4.1.1. Spring Boot 4

Spring Boot es un framework basado en Spring que simplifica el proceso de creación y despliegue de aplicaciones Java. Este marco está diseñado para facilitar el desarrollo de aplicaciones autónomas, listas para la producción y altamente configurables, sin la necesidad de una gran cantidad de configuración manual. Spring Boot sigue el principio de convección sobre configuración, lo que significa que proporciona configuraciones predeterminadas que funcionan bien en la mayoría de los casos, pero también permite personalizar el comportamiento de la aplicación según sea necesario.

Una de las características más destacadas es su capacidad de auto-configuración. Esta funcionalidad permite que el framework configure automáticamente componentes de la aplicación basándose en las dependencias presentes. Esto reduce la necesidad de configuraciones manuales y hace que el desarrollo sea mucho más ágil.

Además, es una excelente opción para la creación de microservicios, ya que permite desarrollar aplicaciones modulares, escalables y autónomas. Ofrece soporte completo para el desarrollo de aplicaciones y favorece la creación de servicios RESTful para aplicaciones basadas en APIs, pudiendo configurar su uso con JSON, XML u otros formatos de datos.

4.1.2. MySQL

MySQL es un sistema de gestión de bases de datos relacional (RDBMS) ampliamente utilizado, desarrollado inicialmente por la empresa sueca MySQL AB, que luego fue adquirida por Sun Microsystems y posteriormente por Oracle Corporation. MySQL es conocido por ser un software de código abierto, que utiliza el modelo de licencia GPL (General Public License), lo que permite su uso gratuito y su modificación por parte de los desarrolladores. Es uno de los sistemas de bases de datos más populares en el mercado y es ampliamente utilizado en aplicaciones web.

El sistema de bases de datos relacional se organiza en tablas con filas y columnas, y las relaciones entre las diferentes tablas se establecen mediante claves primarias y foráneas. Además, utiliza SQL (Structured Query Language) como el lenguaje estándar para interactuar con las bases de datos. SQL es utilizado para realizar operaciones de manipulación de datos (INSERT, UPDATE, DELETE), consultas (SELECT), creación y modificación de esquemas (CREATE, ALTER, DROP), y gestión de usuario y privilegios.

MySQL es conocido por su alto rendimiento, eficiencia y escalabilidad, ofreciendo herramientas como índices para optimizar la organización de los datos. Esto lo hace adecuado para aplicaciones que requieren el procesamiento rápido de grandes volúmenes de datos. Está optimizado para manejar consultas de bases de datos grandes y es capaz de gestionar una cantidad significativa de transacciones por segundo.

4.1.3. POSTMAN

Postman es una herramienta empleada para probar, documentar y gestionar solicitudes HTTP. Entre las funcionalidades que ofrece se destacan:

- La construcción y envío de solicitudes HTTP utilizando diversos métodos como GET, POST, PUT y DELETE.
- La organización de solicitudes en colecciones, así como la creación de entornos de trabajo con variables globales, lo que facilita su uso en distintos contextos.
- La ejecución de pruebas y la automatización de la validación de respuestas, permitiendo evaluar y asegurar el correcto comportamiento de las APIs.

4.1.4. ECLIPSE IDE FOR JAVA DEVELOPERS

Eclipse IDE es un entorno de desarrollo integrado (IDE) de código abierto y gratuito, diseñado específicamente para el desarrollo de aplicaciones en Java. Eclipse es uno de los IDE más populares y ampliamente utilizados en la comunidad de desarrolladores debido a su flexibilidad, extensibilidad y amplio soporte para diferentes tecnologías. Está basado en una arquitectura modular, lo que permite a los desarrolladores personalizar y agregar nuevas funcionalidades según sus necesidades.

4.1.5. VISUAL STUDIO CODE

Visual Studio Code es un editor de código fuente desarrollado por Microsoft, ampliamente reconocido por su versatilidad y eficiencia. Esta herramienta es utilizada por desarrolladores para escribir y depurar código en diversos lenguajes de programación.

Entre sus principales características se destacan las siguientes:

- **Compatibilidad con múltiples lenguajes de programación**, lo que permite su uso en una amplia variedad de proyectos de desarrollo.
- **Integración de extensiones y complementos**, lo que facilita la personalización del entorno de trabajo según las necesidades específicas de cada usuario.
- **Terminal integrada**, que ofrece un acceso rápido a la línea de comandos sin necesidad de salir del editor.
- **Capacidades avanzadas de depuración**, lo que permite a los desarrolladores detectar y corregir errores en su código de manera eficiente.

4.2. TÉCNICAS Y HERRAMIENTAS EMPLEADAS EN WEBAPP

4.2.1. React

React es una biblioteca de JavaScript de código abierto utilizada para construir interfaces de usuario (UI), principalmente en aplicaciones web de una sola página. Fue desarrollada originalmente por Facebook y lanzada en 2013, desde entonces se ha convertido en una de las herramientas más populares y ampliamente utilizadas en el desarrollo web moderno.

Una de las características más fundamentales de React es su arquitectura basada en componentes. Los componentes en React son piezas reutilizables de código que encapsulan tanto la lógica como la presentación de una parte específica de la interfaz de usuario.

4.2.2. Material UI

Material UI, conocido como MUI, es una popular biblioteca de componentes para la construcción de interfaces de usuario basadas en el diseño Material Design, un sistema de diseño desarrollado por Google. Esta biblioteca proporciona una colección de componentes reutilizables y personalizables que permiten a los desarrolladores crear aplicaciones web con un diseño coherente y atractivo, sin necesidad de desarrollar todos los elementos de la interfaz desde cero.

Está construido sobre React, lo que significa que está optimizado para ser utilizado en aplicaciones de React. Proporciona una amplia variedad de componentes listo para usar. Estos componentes siguen las mejores prácticas de desarrollo y están preconfigurados para ser fácilmente integrados en aplicaciones de React.

4.3. HERRAMIENTAS CASE

4.3.1. Visual Paradigm

Visual Paradigm es una herramienta de modelo visual ampliamente utilizada en el desarrollo de software y diseño de sistemas. Es una plataforma de software que ofrece soporte completo para el diseño, planificación y la documentación de sistemas, procesos y aplicaciones a través de diagramas y modelos visuales. Se enfoca en facilitar el análisis, diseño y comunicación en proyectos de desarrollo de software, proporcionando una amplia gama de herramientas para crear diagramas, generar documentación y gestionar el ciclo de vida del software.

4.3.2. EZEstimate

EZEstimate es una herramienta que facilita la evaluación de la cantidad de esfuerzo requerido para llevar a cabo un proyecto en la estimación del esfuerzo y la planificación temporal.

4.3.3. Microsoft Project

Microsoft Project es una herramienta de gestión de proyectos desarrollada por Microsoft, diseñada para facilitar la planificación, ejecución y supervisión de proyectos de manera eficiente. Su interfaz gráfica fácil de usar permite a los gerentes de proyectos y a los equipos de trabajo gestionar todas las fases del ciclo de vida del proyecto, optimizando los procesos de planificación y monitoreo.

Entre sus principales características se incluyen:

- **Planificación de proyectos:** Permite crear cronogramas detallados, establecer hitos y asignar tareas, facilitando la organización del proyecto.
- **Gestión de recursos:** Facilita la administración de recursos humanos, materiales y financieros, garantizando una distribución adecuada y eficiente de los mismos a lo largo del proyecto.
- **Monitoreo y control:** Ofrece herramientas para el seguimiento continuo del progreso del proyecto, permitiendo detectar desvíos en el tiempo, el costo o el alcance.
- **Generación de informes:** Proporciona opciones para la creación de informes detallados sobre el estado del proyecto, los recursos y los plazos, apoyando la toma de decisiones informadas y la comunicación con las partes interesadas.

4.3.4. JSDoc

JSDoc es un estándar utilizado para la documentación del código fuente en JavaScript. Permite estructurar la documentación mediante comentarios especiales insertados directamente en el código, con el objetivo de describir su funcionalidad, los parámetros que recibe, los tipos de datos involucrados, los valores de retorno y otros aspectos relevantes. Este sistema facilita la generación automática de documentación en formato HTML, lo que mejora la comprensión del código y su mantenimiento a lo largo del tiempo.

4.3.5. Swagger

Swagger es un conjunto de herramientas de código abierto ampliamente utilizado para el diseño, desarrollo, documentación y consumo de APIs RESTful. Facilita la creación de APIs de forma más eficiente y accesible, brindando una serie de características que permiten la interacción directa con las APIs, la generación automática de documentación y la validación de los endpoints.

5. ASPECTOS RELEVANTES DEL DESARROLLO

A continuación, se detallan los conceptos, las fases y las secciones más relevantes en el desarrollo del proyecto.

5.1. MARCO DE TRABAJO

En el desarrollo del proyecto se han seguido las directrices establecidas por el Proceso Unificado de Desarrollo Software. Este marco metodológico se caracteriza por ser un proceso dirigido por casos de uso, centrado en la arquitectura y basado en un enfoque iterativo e incremental.

- **Enfoque iterativo e incremental:** El Proceso Unificado se estructura en cuatro fases fundamentales que permiten desarrollar el sistema de manera gradual, a través de múltiples iteraciones. Cada una de estas fases se enfoca en objetivos específicos y contribuye a la evolución progresiva del proyecto:
 - **Inicio:** En esta fase se define el alcance del proyecto y se evalúa su viabilidad técnica y económica. Se realiza un análisis preliminar del mercado y se desarrollan los casos de negocio para justificar la necesidad del sistema.
 - **Elaboración:** Se identifican y detallan los requisitos del sistema mediante el análisis de los casos de uso. Durante esta etapa, se realiza un refinamiento del diseño inicial y se establecen las bases para la arquitectura del sistema.
 - **Construcción:** Con el sistema ya analizado y diseñado, se avanza en la implementación de los componentes principales. Esta fase se centra en el desarrollo del software siguiendo los requisitos previamente definidos.
 - **Transición:** Se prepara el sistema para su despliegue y puesta en producción. Esto incluye la validación del software, la corrección de errores, la documentación final y la capacitación de los usuarios.
- **Proceso dirigido por casos de uso:** Se utilizan los casos de uso como mecanismo central para capturar los requisitos funcionales del sistema. Los casos de uso definen las interacciones entre los usuarios y el sistema, lo que permite planificar las iteraciones de forma estructurada. Cada iteración se desarrolla mediante un conjunto de casos de uso que se abordan a lo largo de distintas disciplinas del desarrollo de software:
 - **Modelado del negocio:** Se analiza el contexto organizacional y los procesos existentes para identificar cómo el sistema puede satisfacer las necesidades del negocio.
 - **Requisitos:** Se documentan los requisitos funcionales y no funcionales del sistema, asegurando que se comprendan las expectativas del cliente.
 - **Análisis:** Se descompone el sistema en componentes para entender su comportamiento y definir sus responsabilidades.
 - **Diseño:** Se crean los modelos arquitectónicos que describen cómo se implementarán los requisitos en el sistema.
 - **Implementación:** Se lleva a cabo el desarrollo del sistema mediante la codificación de los componentes diseñados.

- **Pruebas:** Se realizan verificaciones y validaciones para asegurar que el sistema funcione de acuerdo con los requisitos especificados.
- **Centrado en la arquitectura:** El Proceso Unificado se basa en la premisa de que no existe un único modelo que pueda capturar todos los aspectos del sistema. Por ello, el desarrollo se enfoca en construir múltiples modelos y vistas arquitectónicas que permitan comprender el software desde distintas perspectivas. Esto incluye vistas estructurales, de comportamiento, de despliegue y de interacción, las cuales aseguran que el sistema sea flexible, mantenible y escalable. Este enfoque arquitectónico asegura que el sistema se construya sobre una base sólida que soporte tanto los requisitos actuales como los futuros cambios y evoluciones del proyecto.

5.2. ESTIMACIÓN DEL ESFUERZO

La estimación del esfuerzo en ingeniería de software consiste en determinar la cantidad de trabajo necesario para desarrollar un producto software, así como el tiempo y los recursos requeridos para completar el proyecto de manera exitosa.

En este caso, el proceso de estimación se ha llevado a cabo a través del cálculo del tamaño funcional del sistema, utilizando la métrica de Puntos de Casos de Uso (UCP). Esta metodología ha permitido obtener una estimación más precisa tras la especificación de los requisitos del sistema, proporcionando una visión objetiva sobre la complejidad y magnitud del proyecto [12].

Para realizar dicho cálculo, se ha empleado la herramienta EZEstimate, lo que ha facilitado la obtención de resultados estructurados y coherentes con la metodología adoptada.

Module
 Gestión de estadísticas
 Add Module Delete

Summary
 Total Modules: 5 Excel Report: Generate Report
 Use cases: Simple: 24 Average: 1 Complex: 0
 Actors: Simple: 0 Average: 0 Complex: 2

Add Actor / Use case
 Actor / Use case Name: Select Type: Complexity: Add

Tech / Env Factors
 Set Tech Factor Set Env Factors

Estimation Summary
 UAW: 6
 UUCW: 130
 UUCP = UAW + UUCW: 136
 TFactor: 32
 EFactor: 14
 TCF = 0.6 + (.01*TFactor): 0.92
 EF = 1.4 + (-0.03*EFactor): 0.98
 UCP = UUCP*TCF*EF: 122.6176
 Total Effort@ 8 Hrs/UCP: 735,7056

Use case / Actor List (Double click to delete)

Id	Module	Type	Name	complexity
1	Gestión de aut...	Actor	Usuario sin aut...	Complex
10	Gestión de usu...	Usecase	[UC-008] Elim...	Simple
11	Gestión de rutin...	Usecase	[UC-009] Elabor...	Simple
12	Gestión de rutin...	Usecase	[UC-010] Mostr...	Simple
13	Gestión de rutin...	Usecase	[UC-011] Public...	Average
14	Gestión de rutin...	Usecase	[UC-012] Acce...	Simple
15	Gestión de rutin...	Usecase	[UC-013] Elim...	Simple
16	Gestión de rutin...	Usecase	[UC-014] Limpia...	Simple
17	Gestión de rutin...	Usecase	[UC-015] Mostr...	Simple
18	Gestión de rutin...	Usecase	[UC-16] Limpiar...	Simple
19	Gestión de rutin...	Usecase	[UC-017] Ejecut...	Simple
2	Gestión de aut...	Actor	Usuario autenti...	Complex
20	Gestión de rutin...	Usecase	[UC-018] Reco...	Simple
21	Gestión de rutin...	Usecase	[UC-019] Termi...	Simple
22	Gestión de ejer...	Usecase	[UC-020] Mostr...	Simple
23	Gestión de ejer...	Usecase	[UC-021] Añadir...	Simple
24	Gestión de ejer...	Usecase	[UC-022] Modifi...	Simple
25	Gestión de ejer...	Usecase	[UC-023] Flimin...	Simple

Figura 3: Resultados de la estimación del esfuerzo

La Figura 3 presenta los resultados obtenidos en la estimación del esfuerzo. Para un análisis más detallado del proceso de cálculo correspondiente a dicha estimación, se puede consultar el Anexo I: Plan de proyecto.

5.3. PLANIFICACIÓN TEMPORAL

La planificación temporal en el desarrollo de software constituye un proceso fundamental para asegurar el éxito del proyecto [13]. Este proceso implica diversas etapas, descritas a continuación:

- **Definición de objetos y alcance:** Se establecen los propósitos del proyecto y los resultados que se espera alcanzar.
- **Identificación de hitos y tareas:** El proyecto se descompone en tareas más pequeñas y manejables, teniendo en cuenta su secuencia y las dependencias entre ellas.
- **Estimación de duración:** Se determina la duración de cada tarea, lo que permita calcular el tiempo total requerido para completar el proyecto.
- **Asignación de recursos:** Se asignan los recursos necesarios para cada tarea, como personal, herramientas, infraestructura y otros elementos indispensables para su ejecución.

- **Creación de un calendario de trabajo:** Se elabora un cronograma que detalla las fechas de inicio y finalización de cada tarea.
- **Seguimiento y ajustes:** Una vez definido el plan temporal, se realiza un seguimiento del progreso para identificar posibles retrasos o cambios en el alcance y realizar los ajustes necesarios.

La planificación temporal establecida para este proyecto ha permanecido sin modificaciones a lo largo de su desarrollo. Ha funcionado como una guía que orienta el orden de ejecución de las tareas de desarrollo y permite identificar retrasos potenciales o necesidades adicionales de recursos.

Para llevar a cabo la planificación temporal, se ha utilizado la herramienta Microsoft Project Professional 2016. La planificación realizada se detalla en la Figura 4, donde se especifican las distintas fases del proyecto junto con sus iteraciones, hitos y las fechas previstas de inicio y finalización.

FASE	ITERACIÓN	HITOS	FECHAS
INICIO	1ª ITERACIÓN	Se debe haber definido el alcance del sistema, los requisitos de información y los actores que lo componen, asegurando que la totalidad de los requisitos funcionales y no funcionales estén planteados y documentados.	02-09-2024 12-09-2024
ELABORACIÓN	2ª ITERACIÓN	Se debe haber definido el modelo de análisis completo en la que se definan los casos de uso correspondientes. Asimismo, se debe terminar una parte del diseño del sistema.	13-09-2024 09-10-2024
	3ª ITERACIÓN	La disciplina del análisis debe estar completa y se termina el diseño del sistema. Además, se debe haber empezado con la implementación del sistema.	09-10-2024 07-11-2024
CONSTRUCCIÓN	4ª ITERACIÓN	Se debe haber completado el diseño del sistema y continuado con la implementación del éste.	07-11-2024 05-12-2024
	5ª ITERACIÓN	Se completa la implementación del total del sistema junto con las pruebas correspondientes.	05-12-2024 12-12-2024
TRANSICIÓN	6ª ITERACIÓN	Se completará la documentación del proyecto, se optimizarán las medidas de seguridad, se procederá al despliegue de la aplicación y se llevarán a cabo pruebas integrales del sistema en su conjunto.	13-12-2024 29-12-2024

Figura 4: Planificación temporal

Para obtener una descripción más detallada del proceso de planificación temporal, se remite al Anexo I: Plan de proyecto.

5.4. ESPECIFICACIÓN DE REQUISITOS SOFTWARE

La especificación de los requisitos del software se ha llevado a cabo de acuerdo con la metodología propuesta por Durán y Bernárdez [14]. En este proceso, se han definido la estructura organizativa y los participantes involucrados en el proyecto, así como los objetivos y los requisitos del sistema. En relación con estos últimos, se incluyen tanto los requisitos de información como los requisitos funcionales y no funcionales.

El sistema está compuesto por diversos actores que interactúan tanto con la plataforma como entre sí. La Figura 5 presenta una representación gráfica de dichos actores.

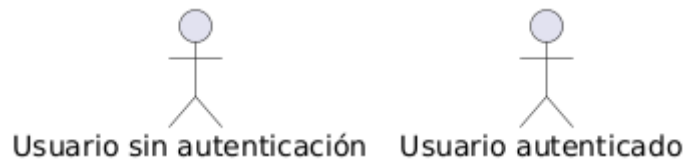


Figura 5: Diagrama de actores

El sistema se compone de múltiples subsistemas que segmentan la funcionalidad general, lo que permite una mejor gestión y manipulación de los requisitos funcionales. La Figura 6 presenta la estructura de los paquetes que conforman el sistema.



Figura 6: Diagrama de paquetes

Finalmente, cada paquete incluye una sección correspondiente a los casos de uso del sistema, los cuales se definen a través de diagramas de casos de uso y tablas de especificación según los lineamientos de Durán y Bernárdez. En la Figura 7 se presenta uno de los diagramas modelados, específicamente el correspondiente al paquete de Gestión de rutinas. Por su parte, la Figura 8 muestra la tabla de especificación del caso de uso [UC-009] Elaborar una rutina, que corresponde con la elaboración de una rutina de ejercicio físico.

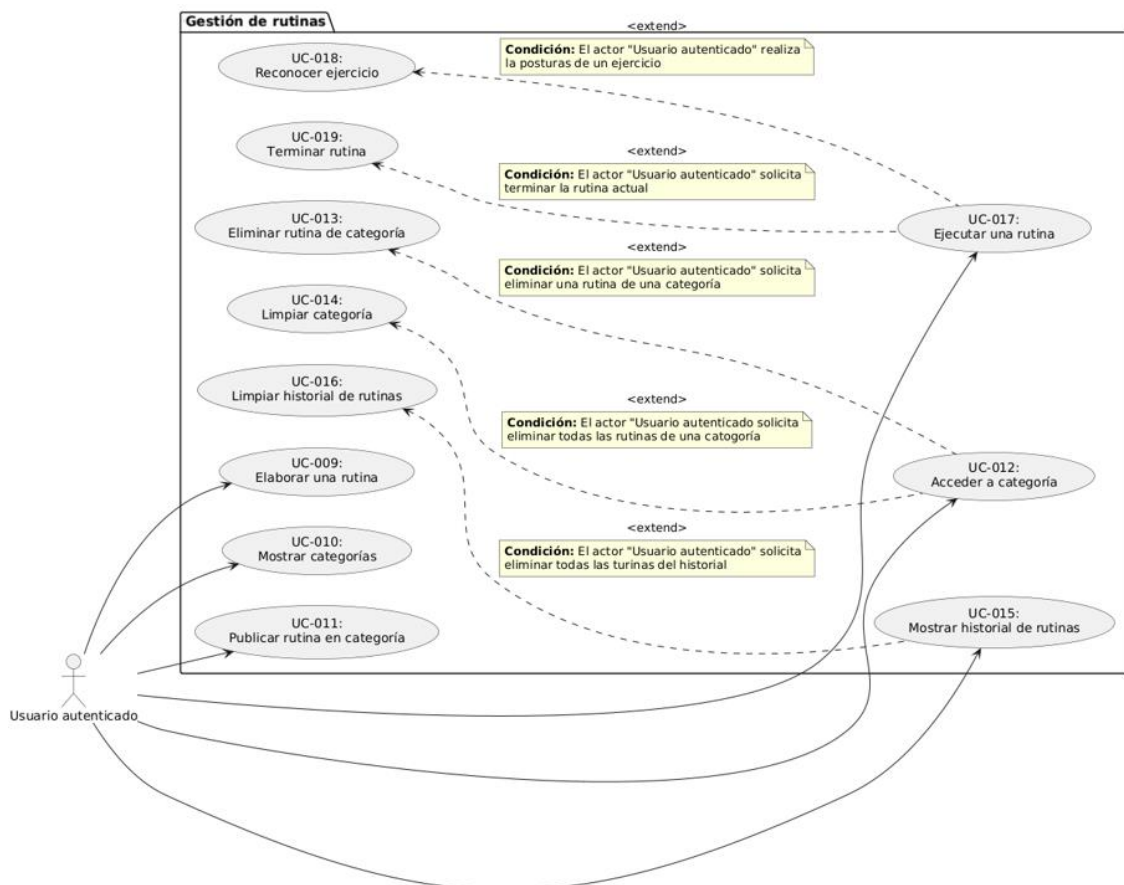


Figura 7: Diagrama de casos de uso del paquete Gestión de rutinas

UC-009	Elaborar una rutina	
Versión	1.0	
Autor	Sergio Salinero Santamaría	
Dependencias	[OBJ-003] Gestión de rutinas [IRQ-002] Información de rutinas [IRQ-003] Información de ejercicios	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor <i>[ACT-002] Usuario autenticado</i> solicita elaborar una rutina.	
Precondición	-	
Secuencia normal	Paso	Acción
	1	El actor <i>[ACT-002] Usuario autenticado</i> solicita elaborar una rutina.
	2	El sistema solicita al usuario la disposición de los ejercicios. Tipo, repeticiones y orden.
	3	El actor <i>[ACT-002] Usuario autenticado</i> introduce el tipo, repeticiones y orden de los ejercicios de la rutina.
	4	El sistema solicita al usuario el tiempo de descanso entre ejercicios.
	5	El actor <i>[ACT-002] Usuario autenticado</i> introduce el tiempo de descanso.
	6	El sistema procesa la información y pone a disposición del usuario el caso de uso <i>[UC-017] Ejecutar rutina</i> .
Postcondición	Una rutina ha sido configurada.	
Excepciones	Paso	Acción
	-	-
Rendimiento	Paso	Acción
	6	0.5 segundos
Frecuencia esperada	10 veces por hora	
Importancia	Alta	
Urgencia	Urgente	
Estado	Validado	
Estabilidad	Alta	
Comentarios	Ninguno	

Figura 8: [UC-009] Elaborar una rutina

Para obtener una descripción más detallada del proceso de definición de los requisitos del software, consulte el Anexo II: Especificación de requisitos.

5.5. ANÁLISIS DEL SISTEMA SOFTWARE

El análisis del sistema de software se llevó a cabo de manera inmediata después de la especificación de requisitos. Este análisis tiene como objetivo comprender y organizar la información relacionada con la planificación del sistema, permitiendo la identificación, evaluación y comprensión de las diversas características y comportamientos del mismo [15]. Para una descripción más detallada del proceso de análisis del software, consulte el Anexo III. Análisis del sistema.

5.5.1. Modelo del dominio

El modelo de dominio facilita la comprensión y representación de las clases conceptuales, entidades, relaciones, funciones y procesos que constituyen el contexto en el que opera el software. La Figura 9 presenta el modelo de dominio del sistema.

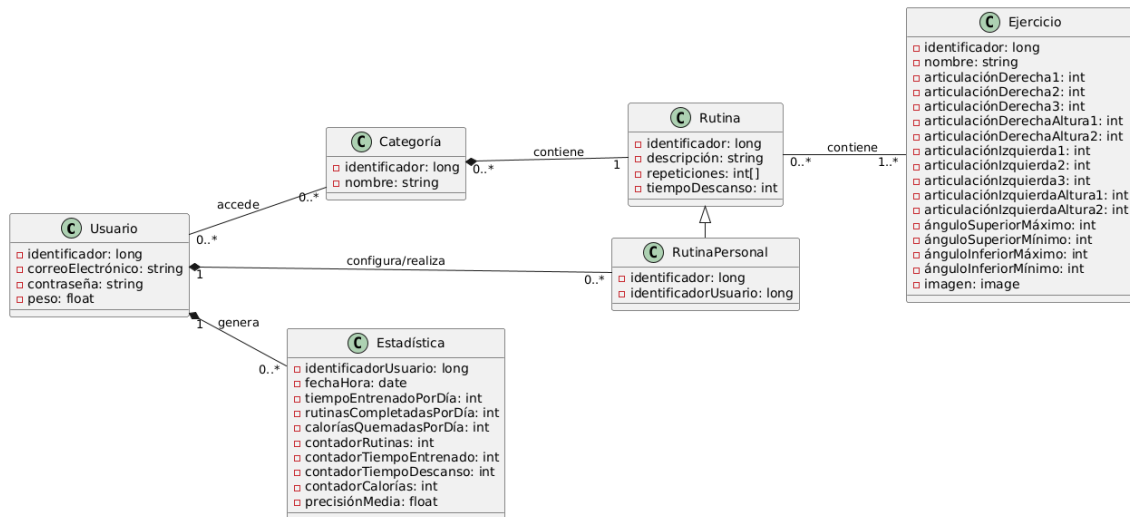


Figura 9: Diagrama de clases en el modelo de dominio

5.5.2. Clases de análisis y descripción de la arquitectura

Una clase de diseño es un tipo de clasificador que define un conjunto de objetos que comparten características, restricciones y semánticas comunes. En el contexto del modelo de análisis, se distinguen tres tipos de clases principales:

- **Clases de entidad:** Representan objetos de negocio que suelen ser persistentes y se almacenan dentro del sistema.
- **Clases de control:** Son responsables del procesamiento y gestión de la información.
- **Clases de interfaz:** Se encargan de presentar y comunicar la información al usuario y/o interactuar con otros componentes del sistema.

La Figura 10 muestra una descripción de la arquitectura del modelo de análisis, que incluye los paquetes y las clases de análisis.

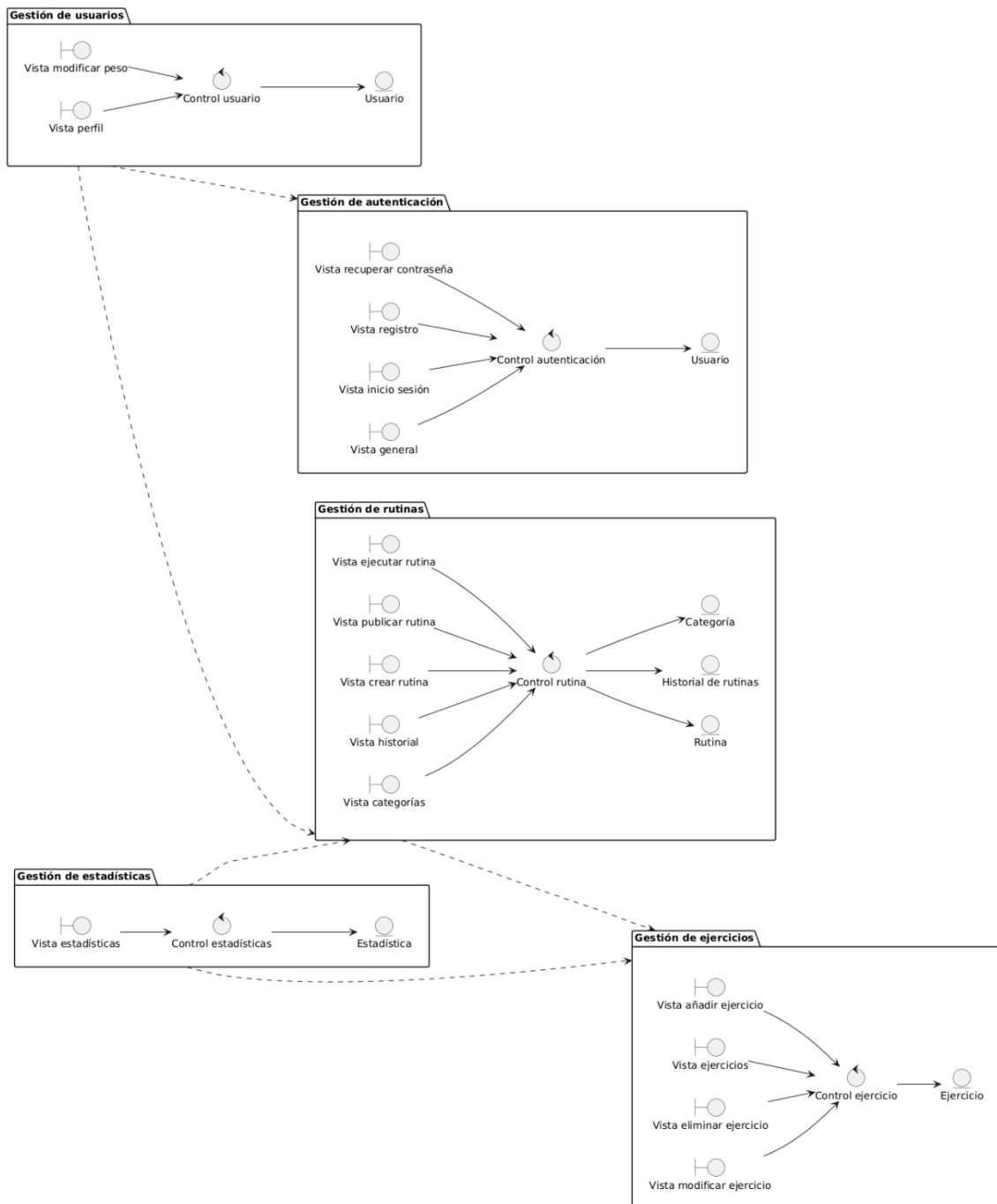


Figura 10: Arquitectura en el modelo de análisis

5.5.3. Realización de casos de uso en el modelo de análisis

A continuación, se describe el proceso de ejecución de los casos de uso, utilizando las clases de análisis correspondientes. En la Figura 11 se presenta el diagrama de secuencia asociado con el análisis del caso de uso [UC-009] Elaborar una rutina.

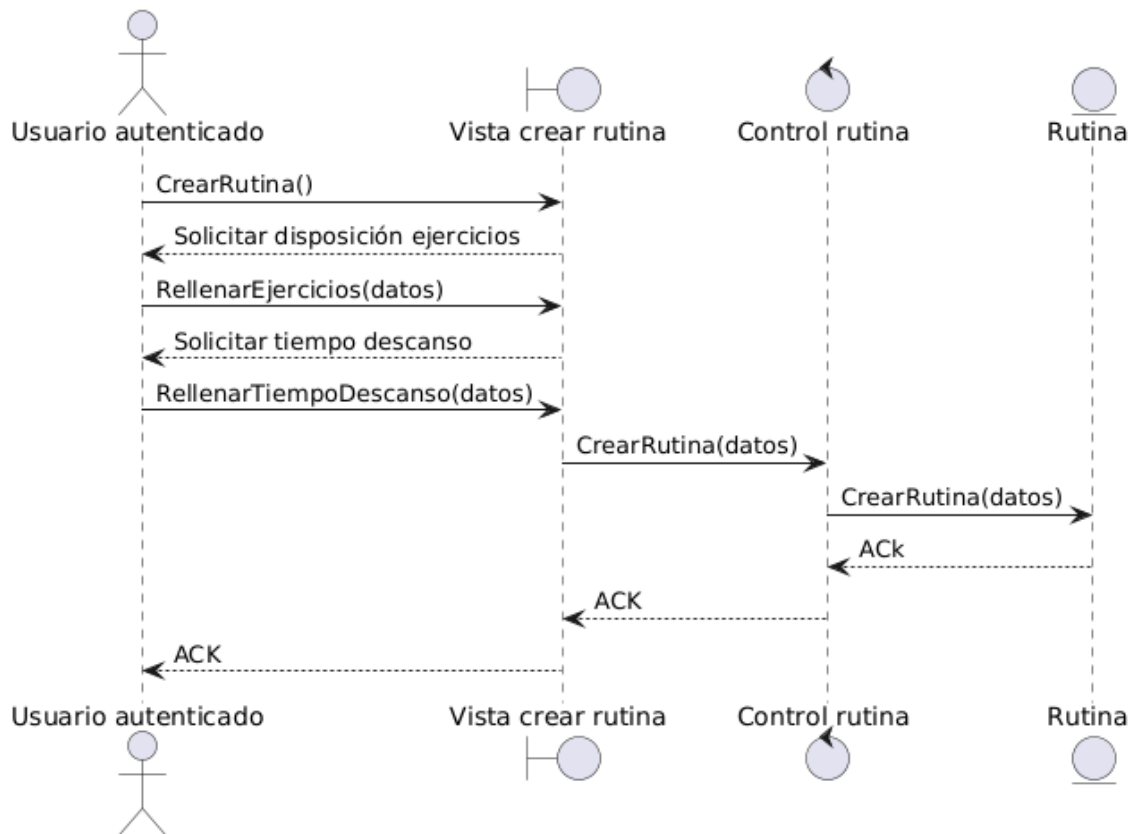


Figura 11: Diagrama de secuencia [UC-009] Elaborar una rutina

5.6. DISEÑO DEL SISTEMA SOFTWARE

El proceso de planificación y desarrollo de una solución software implica la definición de los requisitos del sistema, la identificación de los módulos y componentes necesarios, la especificación de la arquitectura del sistema y la determinación de las estructuras de datos y algoritmos empleados [16]. Asimismo, durante la etapa de diseño se analiza el impacto que dichos elementos tienen sobre el sistema.

Para obtener una descripción más detallada del proceso de diseño del sistema software, se puede consultar el Anexo IV: Diseño del sistema.

5.6.1. Patrones arquitectónicos

Los patrones arquitectónicos son soluciones probadas y documentadas para problemas recurrentes en el diseño y desarrollo de sistemas software [17]. A la hora de escoger estos patrones hay que tener en cuenta las tecnologías que se utilizarán en el desarrollo del sistema:

- **Estimación de posturas:** Técnicas de visión computacional que detectan figuras humanas en imágenes y vídeos, permitiendo determinar la posición de las articulaciones de un usuario.

- **Spring boot 4:** Tecnología diseñada para simplificar el desarrollo de la lógica de negocio de un sistema. Ofrece un enfoque modular que facilita la configuración, integración y despliegue de aplicaciones Java.
- **React:** Biblioteca JavaScript diseñada para la construcción de interfaces de usuario dinámicas y reutilizables en aplicaciones web.

5.6.1.1. Patrón de capas

El patrón de capas organiza la arquitectura lógica de un sistema en distintas capas, cada una con responsabilidades específicas y relacionadas, promoviendo una colaboración estructurada y un acoplamiento controlado, principalmente desde las capas superiores hacia las inferiores.

Las capas definidas por este patrón son las siguientes:

- **Capa de presentación:** Encargada de la interfaz de usuario, esta capa gestiona la visualización de información al usuario y facilita la interacción del mismo con el sistema.
- **Capa de lógica de negocio:** Corresponde al núcleo funcional de la aplicación. Aquí se establecen las reglas de negocio, se validan los datos y se gestionan procesos complejos que integran distintas operaciones.
- **Capa de datos:** Dedicada a la administración de la información, incluye la base de datos, los sistemas de almacenamiento y otros componentes relacionados con la gestión de datos.

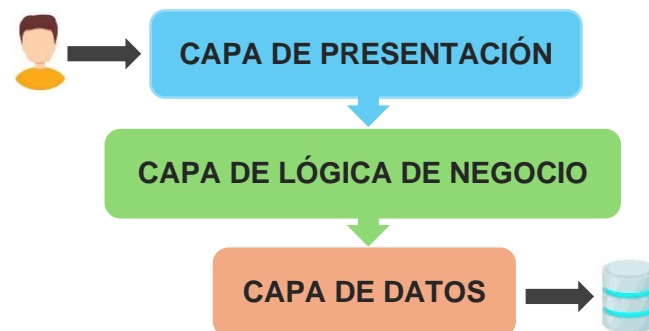


Figura 12: Patrón de capas

5.6.1.2. Patrón cliente-servidor

El sistema se organiza en dos componentes principales: el cliente y el servidor. Estos se comunican entre sí a través de una red utilizando protocolos de comunicación como HTTP o TCP/IP. El intercambio de información se basa en un modelo de solicitud-respuesta, en el cual el cliente emite una solicitud al servidor y permanece a la espera de una respuesta.

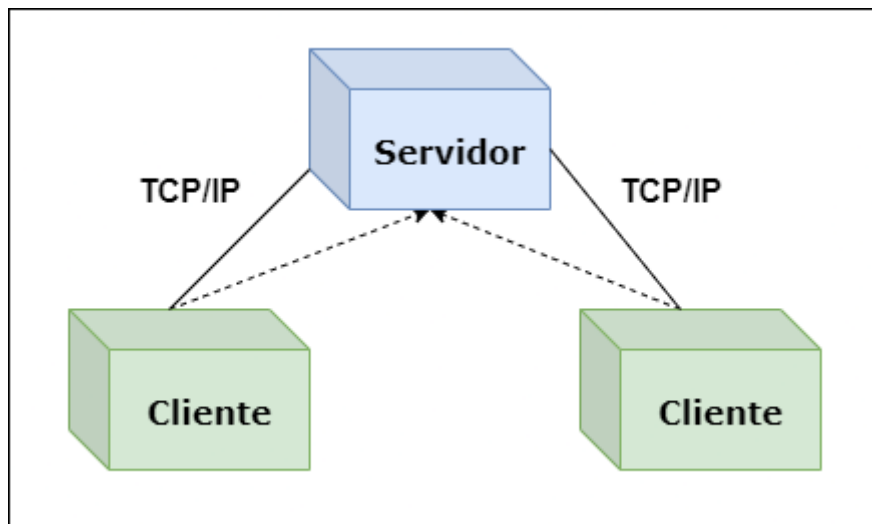


Figura 13: Patrón cliente-servidor

5.6.1.3. Patrón DAO

El patrón DAO (Data Access Object) facilita la separación entre la lógica de negocio y la lógica de acceso a los datos. Este enfoque introduce una capa de abstracción que actúa como intermediaria entre la capa de lógica de negocio y la capa de datos, garantizando que las modificaciones realizadas en una de estas capas no afecten a la otra.

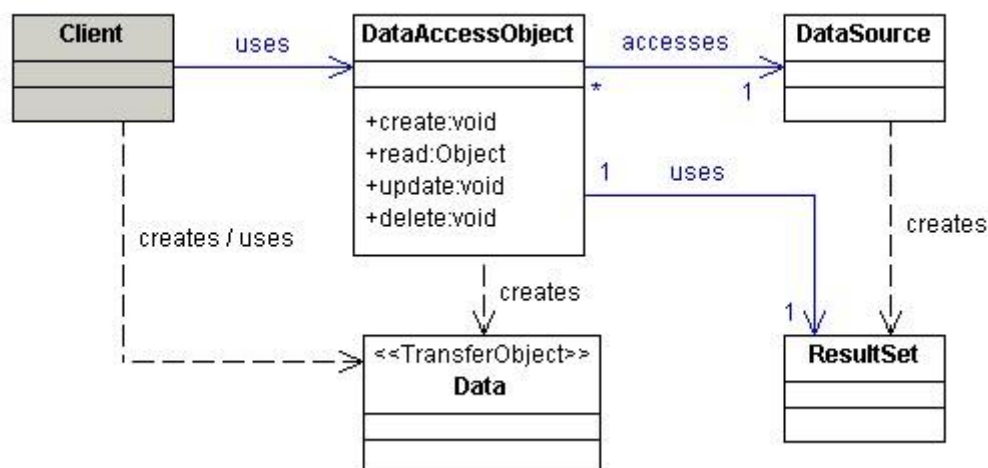


Figura 14: Patrón DAO

5.6.1.4. Patrón singleton

Se busca garantizar que una clase posea una única instancia, accesible de manera global y concreta. Este patrón será aplicado en el servidor, el cual debe ser único y actuar como un punto centralizado para todos los clientes.

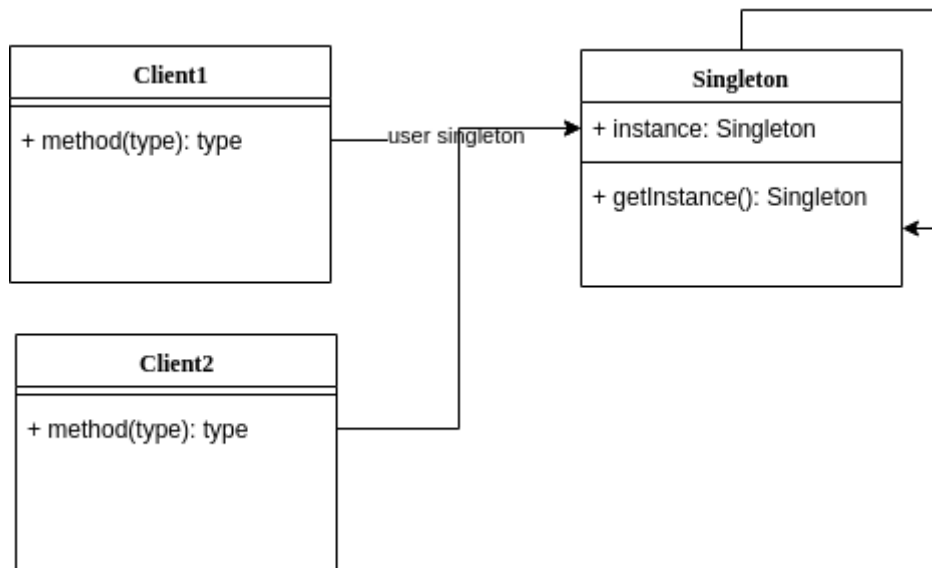


Figura 15: Patrón singleton

5.6.2. Subsistemas de diseño

Este apartado presenta una descripción de los subsistemas de diseño correspondientes a los distintos módulos que conforman el sistema. Se resalta la funcionalidad específica de cada componente, haciendo énfasis en su diseño para lograr un sistema más modular y escalable, lo que facilita tanto su mantenimiento como su futura evolución.

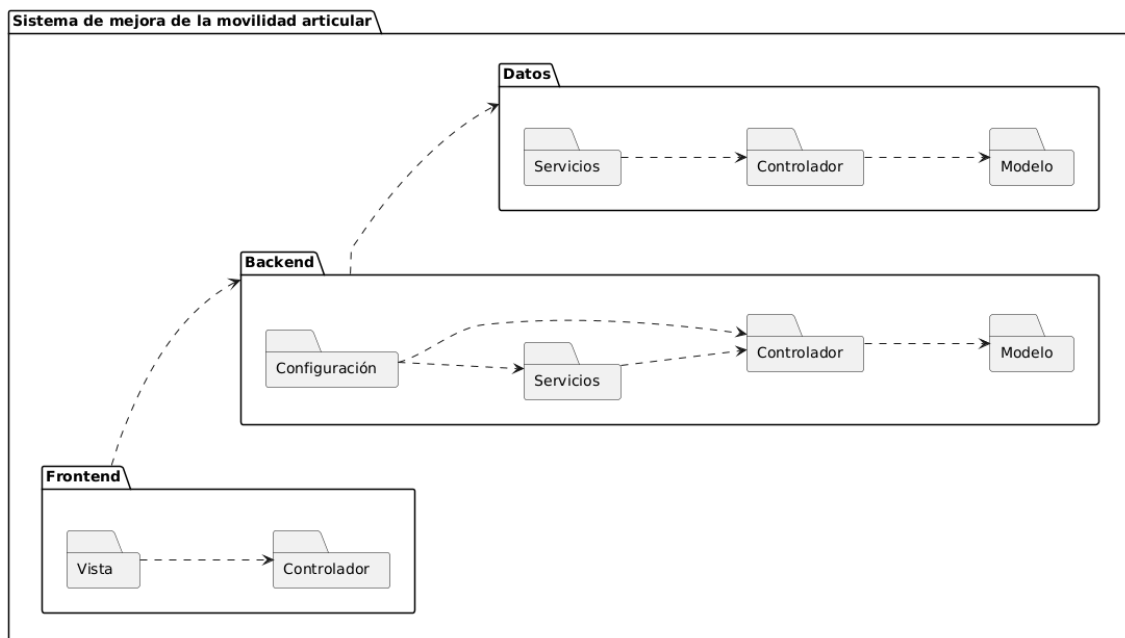


Figura 16: Subsistemas de diseño

Se dividen en tres módulos:

- **Frontend:** Alberga la parte visible del sistema, representando el conjunto de acciones con las que el usuario final podrá interactuar de forma directa.
- **Backend:** Encargado de gestionar la lógica de negocio, interconecta los componentes *Frontend* y *Datos*.
- **Datos:** Comprende un servidor cuyo fin es gestionar el acceso a los datos, así como operar sobre ellos.

5.6.3. Clases de diseño

Las clases de diseño constituyen una representación abstracta de una o más clases utilizadas en la implementación del sistema. A continuación, se detallan las clases de diseño que integran cada uno de los subsistemas previamente definidos.

5.6.3.1. Subsistema datos

SERVICIOS: Conjunto de servicios disponibles para componentes backend dentro del sistema.

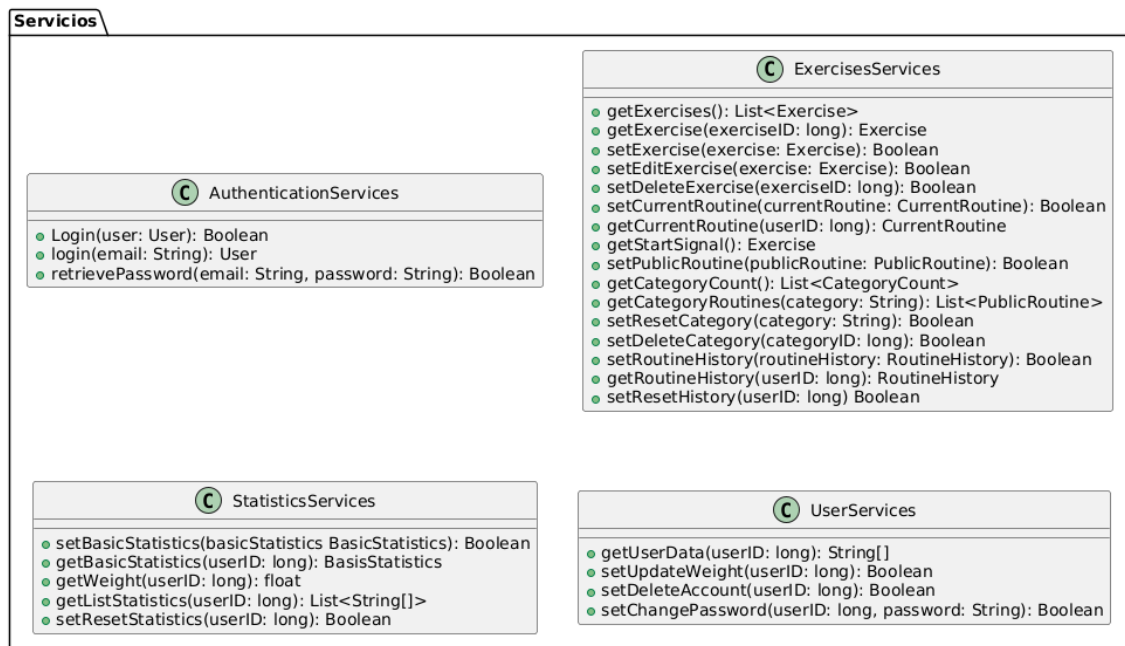


Figura 17: Clase de diseño Servicios del subsistema Datos

CONTROLADOR: Lógica y operaciones internas para gestionar el acceso a la base de datos una vez se ha solicitado un servicio.

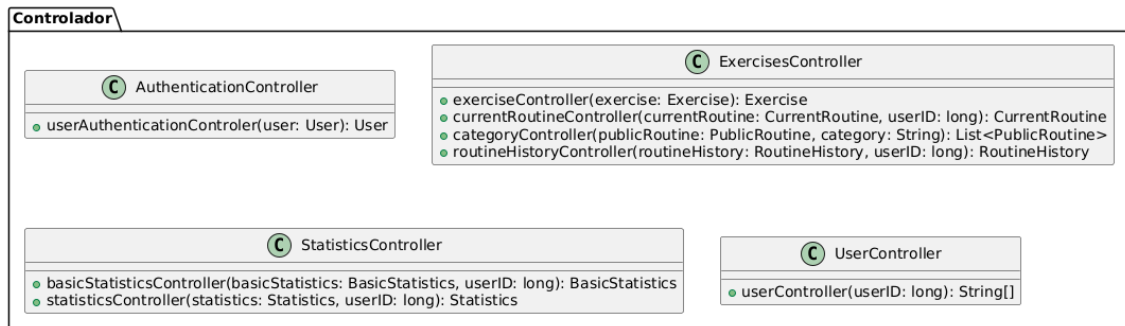


Figura 18: Clase de diseño Controlador del subsistema Datos

MODELO: Almacenamiento temporal de conjuntos de datos que serán utilizados en las operaciones con la base de datos.

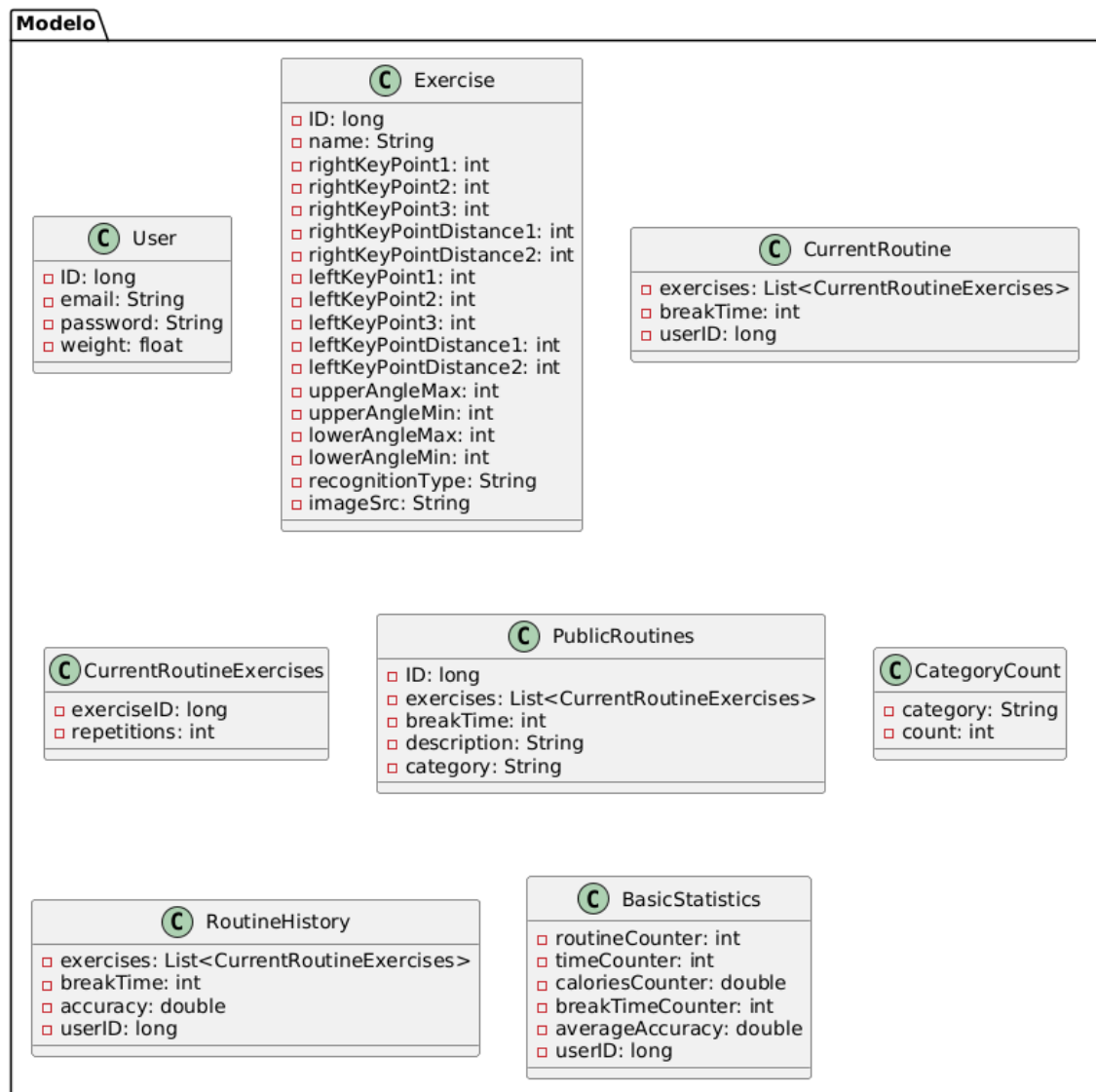


Figura 19: Clase de diseño Modelo del subsistema Datos

5.6.3.2. Subsistema Backend

SERVICIOS: Conjunto de servicios disponibles para componentes frontend dentro del sistema.

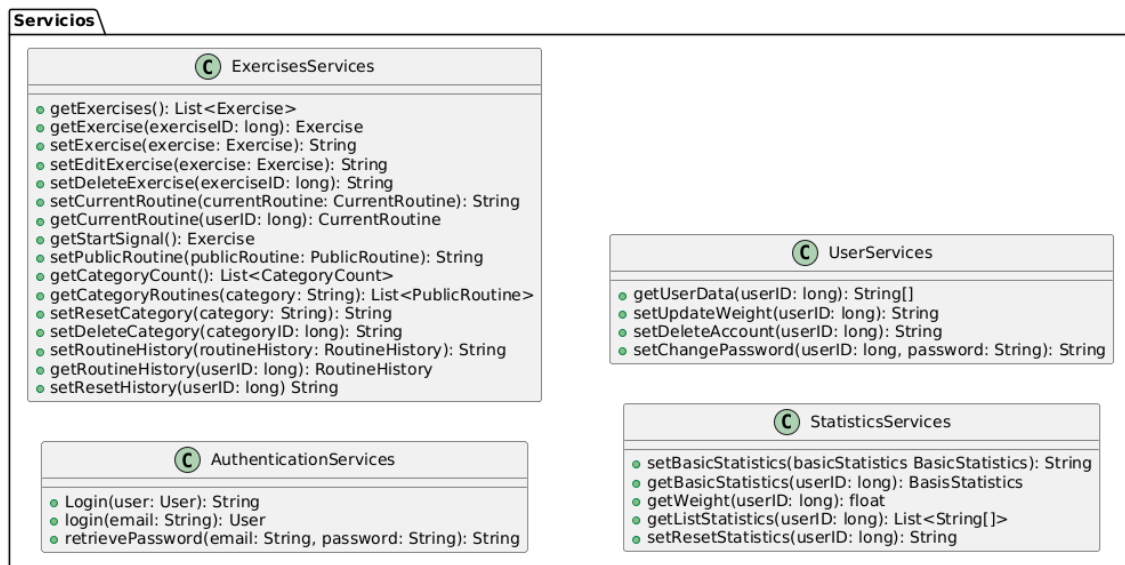


Figura 20: Clase de diseño Servicios del subsistema Backend

CONTROLADOR: Controlador de la lógica para gestionar la información recibida/solicitada por componentes software, estará interconectado con el servidor de base de datos.

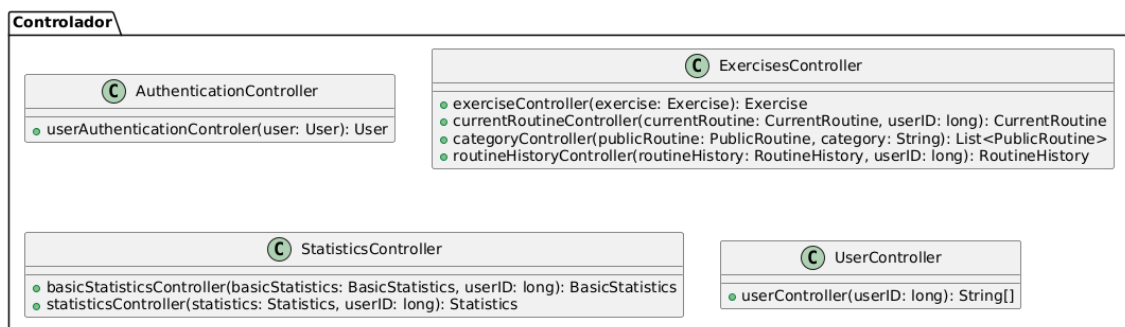


Figura 21: Clase de diseño Controlador del subsistema Backend

MODELO: Almacenamiento temporal de conjuntos de datos utilizados en la gestión de las peticiones a los servidores.

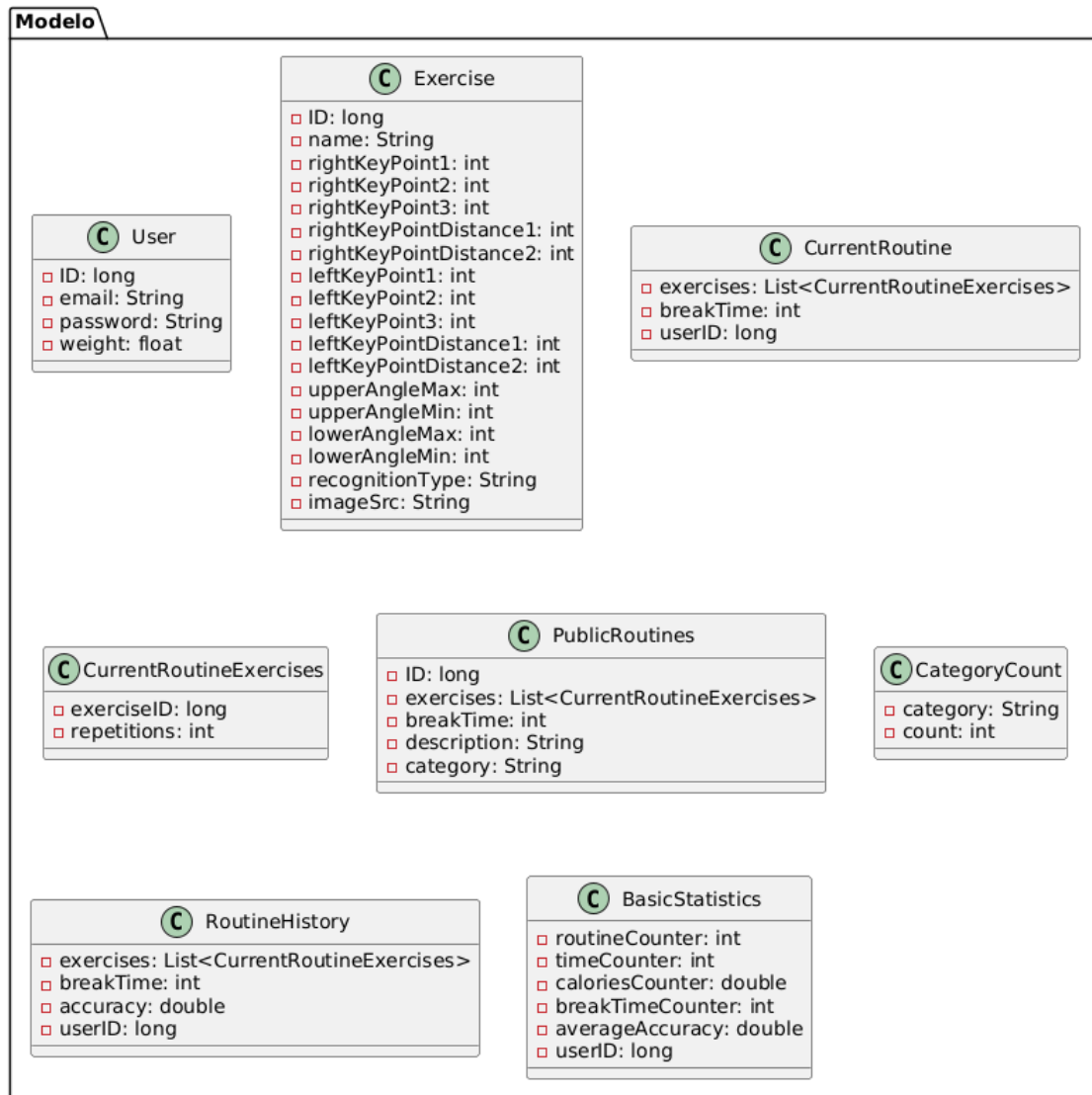


Figura 22: Clase de diseño Modelo del subsistema Backend

UTILIDADES: Conjunto de clases de configuración dedicadas a dar soporte al backend en operaciones internas.

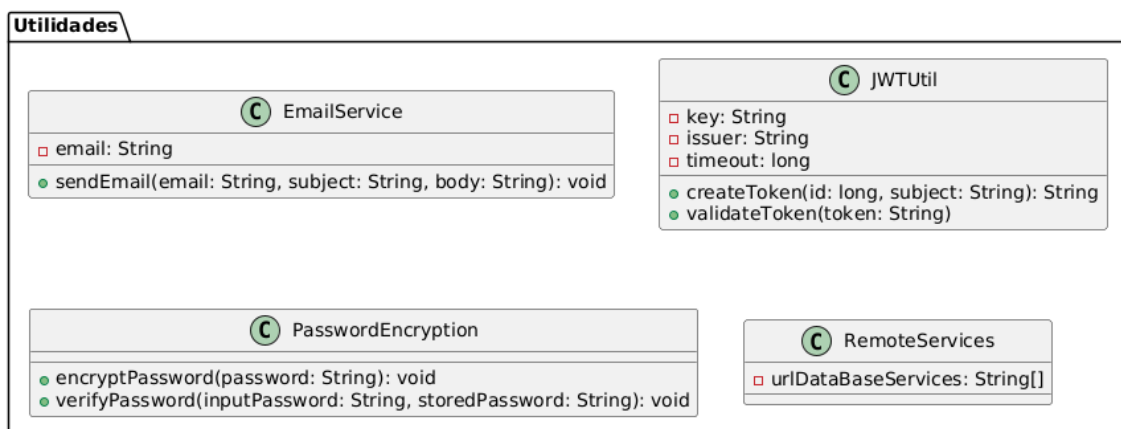


Figura 23: Clase de diseño Utilidades del subsistema Backend

5.6.3.3. Subsistema Frontend

VISTAS: Conjunto de clases que conforman la interfaz de usuario.

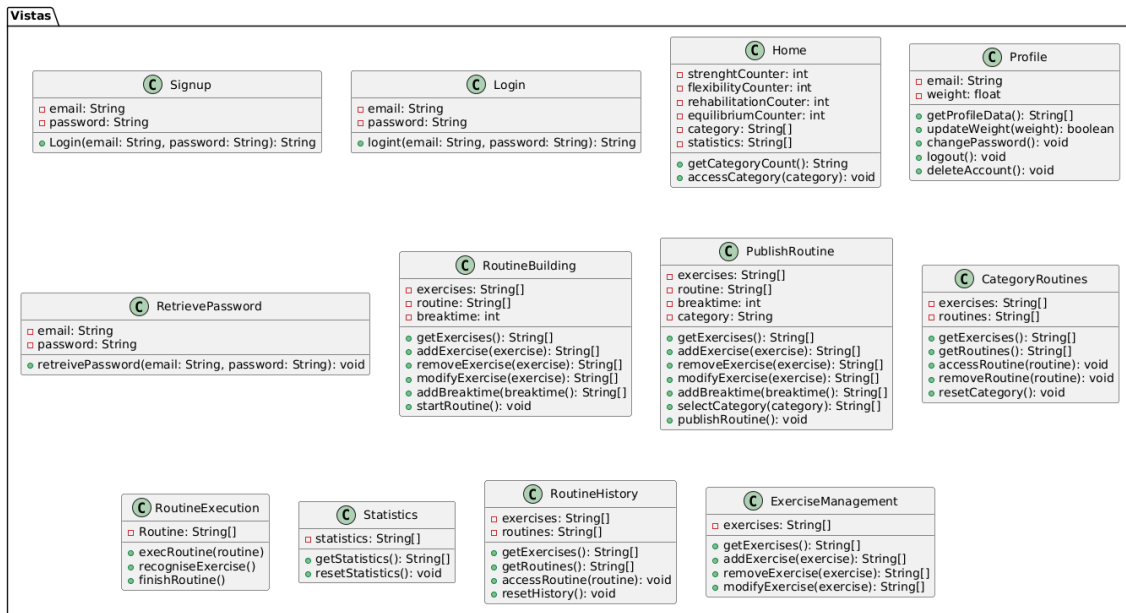


Figura 24: Clases de diseño Vistas del subsistema Frontend

CONTROLADOR: Lógica y operaciones de comunicación con el servidor backend dependiente de las acciones del usuario.

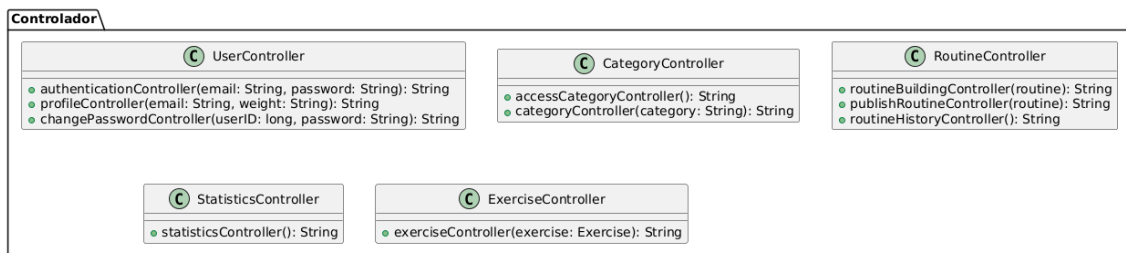


Figura 25: Clase de diseño Controlador del subsistema Frontend

5.6.4. Realización de casos de uso en el modelo de diseño

En esta sección se describirá de manera detallada la forma en que se llevará a cabo la implementación de cada uno de los casos de uso identificados durante la etapa de recopilación de requisitos y análisis del sistema.

En la Figura 26 se muestra la realización del caso de uso [UC-009] Elaborar una rutina.

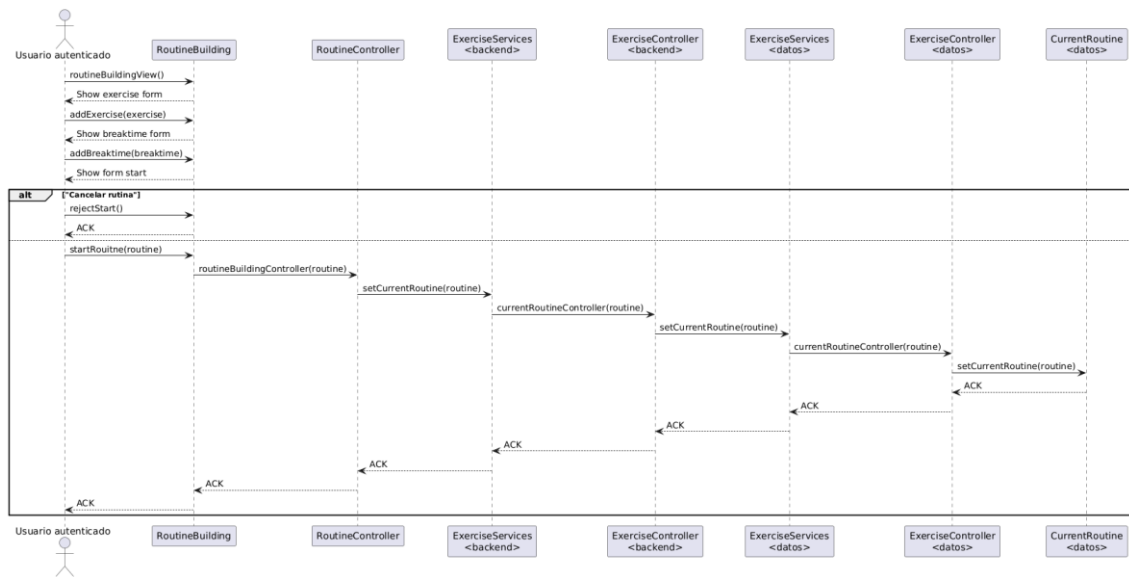


Figura 26: Realización de caso de uso [UC-009] Elaborar una rutina

5.6.5. Diseño de la base de datos

A continuación, se presenta la estructura organizativa de la base de datos del sistema. Se detallarán todas las tablas y atributos junto con sus respectivos tipos de datos. La base de datos utilizada será de tipo SQL, concretamente MariaDB.

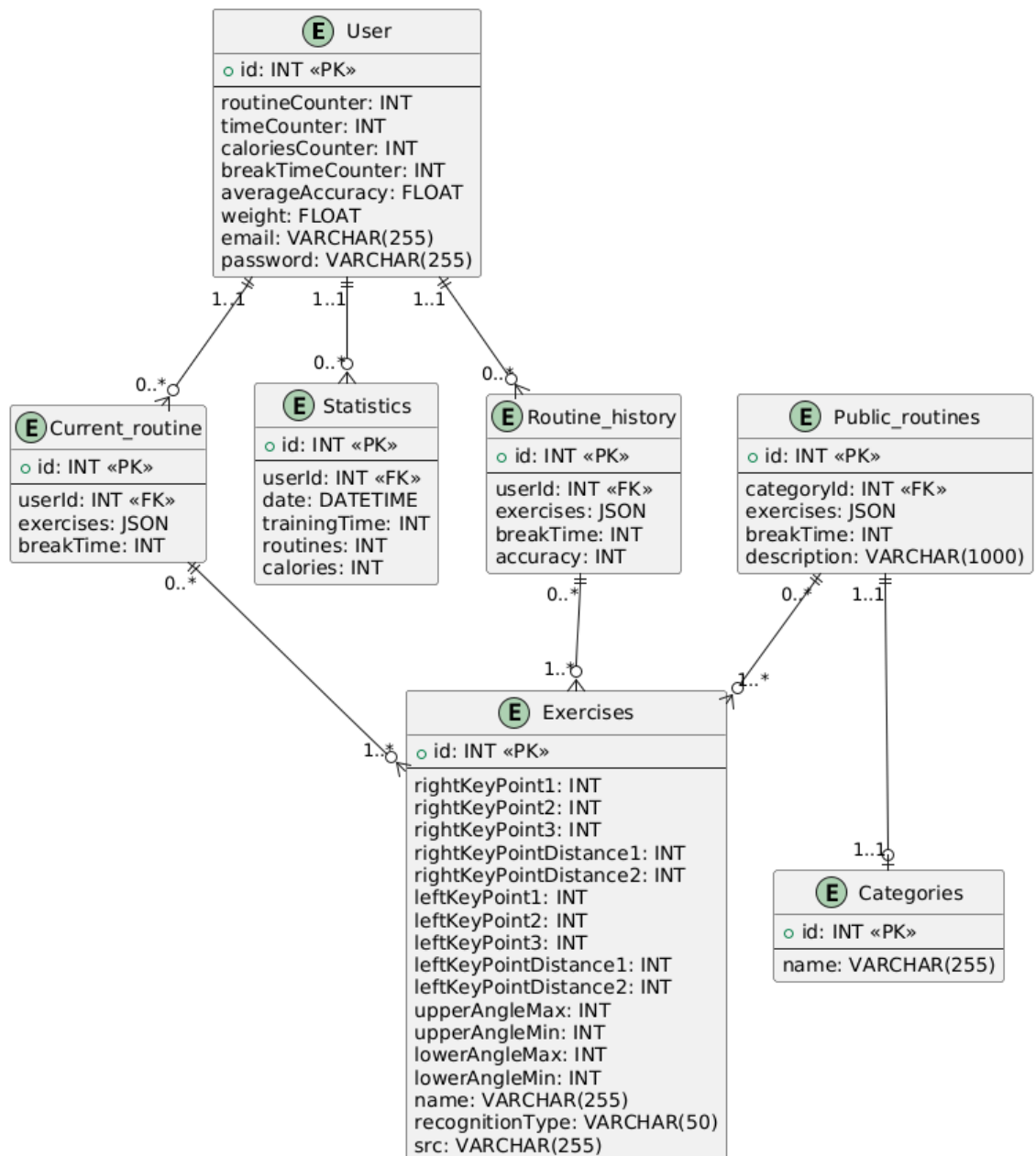


Figura 27: Diseño de la base de datos

5.6.6. Modelo de despliegue

El modelo de despliegue representa la distribución física del sistema, especificando cómo se asigna la funcionalidad a los distintos nodos de procesamiento. La Figura 28 ilustra la implementación de este diagrama.

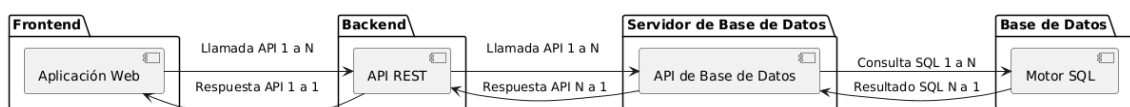


Figura 28: Diagrama de despliegue

El modelo de despliegue describe la estructura de un sistema durante su ejecución y está compuesto por cuatro nodos:

- **Frontend:** Parte del sistema que interactúa directamente con el usuario, formando la interfaz gráfica y los elementos que el usuario puede ver y manipular.
- **Backend:** Parte del sistema que se encarga de la lógica, procesamiento de datos y gestión del servidor. Es la capa que funciona detrás de escena, permitiendo que el frontend opere correctamente.
- **Servidor de base de datos:** Sistema que proporciona servicios de almacenamiento, gestión y acceso a bases de datos. Su función es responder las solicitudes del backend para operar en la base de datos garantizando integridad, seguridad y disponibilidad.
- **Base de datos:** Conjunto organizado de datos estructurados que se almacenan y gestionan de manera eficiente para facilitar su acceso, manipulación y actualización.

5.7. IMPLEMENTACIÓN

La ejecución de los requisitos funcionales ha representado la fase más demandante en términos de tiempo dentro del desarrollo del proyecto. Se han aplicado la planificación establecida, así como las herramientas, técnicas y especificaciones detalladas en el apartado 5. Aspectos relevantes del desarrollo.

Para obtener más información detallada sobre la documentación del código fuente, se recomienda consultar el Anexo V. Documentación técnica.

5.7.1. Implementación del subsistema Frontend

El subsistema Frontend comprende todo aquello que interactúa directamente con el usuario, conformando la parte visible del sistema y expresando las funcionalidades del mismo.

Ha sido desarrollado con la tecnología React. React es una biblioteca de JavaScript de código abierto usada especialmente para aplicaciones web. Permite crear componentes reutilizables que gestionan su propio estado y luego los combina para formar una interfaz más compleja.

Dentro del proyecto, en el directorio `/liveposes/src` se encuentra el código fuente distribuido de la siguiente manera:

- **/app:** Directorio dedicado a las configuraciones generales del proyecto. Se encuentra la página inicial (index) junto con la hoja de estilos global y el icono de la plataforma (favicon.ico).
- **/pages:** Aquí se hayan los ficheros JavaScript correspondientes a las páginas accesibles en la plataforma, siendo un total de 16, se explicarán más adelante.

- **/components:** La interfaz de usuario se divide en piezas pequeñas y reutilizables de código, facilitando el mantenimiento y la escalabilidad. Aquí se encuentran los componentes que son utilizados por las páginas.
- **/libraries:** Almacena los ficheros de funcionalidad que no se corresponden ni con las páginas ni con los componentes. Concretamente, se encuentra un único fichero encargado de ejecutar el algoritmo de detección de posturas.
- **/utils:** Directorio que contiene ficheros de configuración globales. Contiene un fichero *Colors* para unificar la gama de colores de la interfaz en la plataforma, y un fichero *RemoteServices* para gestionar el acceso unificado a los servicios del componente Backend.

Siendo más específico en la explicación, se procede a explicar detalladamente el contenido del directorio que contiene la práctica totalidad de la funcionalidad, */pages*.

Los ficheros contenidos en este directorio junto con su explicación se muestran a continuación:

- **Signup.js:** Permite el registro de usuarios en el sistema.
- **Login.js:** Permite el acceso de usuario en el sistema basado en la autenticación por usuario y contraseña.
- **ForgotPassword.js:** Servicio de recuperación de contraseña. Se solicita el correo electrónico al usuario para enviarle un correo con un enlace personalizado donde podrá modificar su contraseña.
- **PasswordRecovery.js:** Una vez recibido el correo electrónico, esta será la pantalla donde el usuario deberá proporcionar al sistema la nueva contraseña.
- **Home.js:** Es la página primera después de iniciar sesión. Su función es proporcionar acceso a las categorías de ejercicios, a la vez que permite accionar cualquiera de las funcionalidades principales, las cuales son:
 - Perfil de usuario
 - Mostrar categorías de ejercicios
 - Crear una rutina personal
 - Publicar una rutina en una categoría
 - Ver las estadísticas personales
 - Ver el historial de rutinas
 - Gestión de ejercicios físicos
- **CategoryRoutines.js:** Se encarga de permitir el acceso a las funcionalidades de las categorías de ejercicio, las cuales son:
 - Mostrar las rutinas contenidas en la categoría
 - Ejecutar cualquiera de las rutinas
 - Eliminar cualquiera de las rutinas
 - Eliminar todas las rutinas de la categoría
- **RoutineBuilding.js:** Encargado de permitir la ejecución de las funcionalidades relacionadas con la creación de una rutina de carácter personal:
 - Añadir un ejercicio a la rutina
 - Cambiar el tipo de ejercicio
 - Elegir el número de repeticiones del ejercicio
 - Eliminar un ejercicio
 - Modificar el orden de ejercicios en la rutina
 - Configurar tiempo de descanso

- Ejecutar la rutina
- **PoseRecognition.js:** El fichero más relevante y complejo, se encarga de gestionar la ejecución de las rutinas. En primer lugar, carga el modelo *BlazePose* mediante la API de *TensorFlow*. Seguidamente, ejecuta las siguientes funcionalidades:
 - Inicia el modelo de detección de posturas
 - Detecta la postura del usuario
 - Carga los datos de la rutina configurada
 - Reconoce la señal de inicio e inicia el cronómetro
 - Reconoce la realización de un ejercicio
 - Gestiona tiempos de descanso
 - Determina la terminación de la rutina y gestiona los datos de seguimiento obtenidos
- **PublishRoutine.js:** Se encarga de permitir la publicación de una rutina de ejercicios físicos en una categoría específica:
 - Añadir un ejercicio a la rutina
 - Cambiar el tipo de ejercicio
 - Elegir el número de repeticiones del ejercicio
 - Eliminar un ejercicio
 - Modificar el orden de ejercicios en la rutina
 - Configurar tiempo de descanso
 - Dar una descripción a la rutina
 - Seleccionar la categoría destino
 - Publicar la rutina
- **Profile.js:** Permite gestionar la información del usuario:
 - Mostrar la información del usuario
 - Actualizar el peso del usuario
 - Cambiar la contraseña
 - Cerrar sesión
 - Eliminar la cuenta del usuario
- **Statistics.js:** Se encarga de gestionar el acceso a los datos de seguimiento del usuario. Las funcionalidades que se permiten son las siguientes:
 - Mostrar las estadísticas personales
 - Reiniciar las estadísticas
- **RoutineHistory.js:** Encargado de permitir el acceso a la funcionalidad relacionada con el historial de rutinas:
 - Mostrar las rutinas realizadas por el usuario
 - Ejecutar cualquiera de las rutinas
 - Eliminar todas las rutinas del historial de rutinas
- **ExerciseManagement.js:** Permite la gestión de los ejercicios presentes en el sistema. La funcionalidad relacionada es:
 - Mostrar los ejercicios presentes en el sistema
 - Acceso a la adición de ejercicios
 - Acceso a la edición de cualquiera de los ejercicios
 - Eliminar cualquiera de los ejercicios
 - Acceso a la información de ayuda sobre la gestión de ejercicios
- **EditExerciseManagement.js:** Permite la edición de la mayoría de los parámetros contenidos en un ejercicio.

- **AddExerciseManagement.js:** Permite la edición de ejercicios a partir de la introducción de los parámetros necesarios
- **ExerciseManagementHelp.js:** Proporciona una ayuda para la gestión y manejo de la información presentes en los ejercicios físicos.

5.7.2. Implementación del subsistema Backend

Para el funcionamiento de la plataforma, el subsistema Frontend necesita gestionar cierta información, ya sea para mostrarla, insertarla, modificarla o actualizarla. Entre el Frontend y la información se encuentra el subsistema Backend. Se encarga de gestionar las peticiones del Frontend de forma segura y solicitar al subsistema Datos realizar las operaciones con los datos.

La tecnología utilizada es Java junto con la herramienta Spring Boot 4 para facilitar la creación de microservicios web.

Dentro del directorio *src/main/java/com/liveposes/main* se encuentra el código fuente, el cual está distribuido de la siguiente forma:

- **/:** Contiene la clase que inicia la aplicación Spring Boot.
- **/config:** Contiene la clase que configura el acceso a terceros, las operaciones permitidas y los directorios donde se permiten.
- **/controller:** Conjunto de clases dedicadas a gestionar las peticiones entrantes del subsistema Frontend.
El controlador de la aplicación Spring boot se divide en 4 clases conformando los conjuntos de funcionalidades presentes en el sistema:
 - **AuthenticationController.java:** Encargado de gestionar las peticiones entrantes relacionadas con la autenticación de los usuarios en el sistema.
 - **ExercisesController.java:** Maneja las peticiones relacionadas con los ejercicios físicos. Es la clase más interesante y con más complejidad.
 - **StatisticsController.java:** Controla las peticiones entrantes que se corresponden con la gestión de las estadísticas de usuario.
 - **UserController.java:** Se encarga de gestionar las peticiones relacionadas con la información del usuario.
- **/model:** Conformar el conjunto de clases dedicadas al almacenamiento temporal de datos, así como su correcto formateo y organización. Las clases presentes en el directorio son las siguientes:
 - **BasicStatistics.java**
 - **CategoryCount.java**
 - **CurrentRoutine.java**
 - **CurrentRoutineExercises.java**
 - **Exercise.java**
 - **PublicRoutine.java**
 - **RoutineHistory.java**
 - **User.java**
- **/services:** Conjunto de clases complementarias a las clases del controlador. Su finalidad es gestionar las solicitudes hacia el subsistema Datos en base a las peticiones del controlador. Las clases contenidas en el directorio son las siguientes:

- **AuthenticationServices.java**
- **ExercisesServices.java**
- **StatisticsServices.java**
- **UserServices.java**
- **/utils:** Conjunto de funcionalidad que es utilizada en repetidas ocasiones por el resto del código. Las clases presentes son:
 - **EmailService.java:** Clase encargada del envío del correo electrónico a un usuario concreto.
 - **JWTUtil.java:** Clase dedicada a crear y validar los tokens de acceso al sistema. Comprende la seguridad de acceso basado en JWT (Json Web Token) garantizando el legítimo acceso a las funcionalidades del sistema.
 - **PasswordEncryption.java:** Se encarga de encriptar y comprobar la veracidad de una contraseña.
 - **RemoteServices.java:** Comprende un acceso unificado a los servicios remotos.

5.7.3. Implementación del subsistema Datos

Para gestionar las operaciones en la información presente en el sistema, el subsistema Backend se comunica con el subsistema Datos. Esta estructura comprende una separación de responsabilidades, lo que favorece la modularidad en el sistema, aumentando la escalabilidad, reutilización, mantenibilidad y seguridad.

La tecnología utilizada es la misma que la del subsistema Backend. El lenguaje de programación es Java junto con la herramienta Spring Boot 4 para facilitar la creación de microservicios web.

Dentro del directorio *src/main/java/com/liveposes/main* se encuentra el código fuente, el cual está distribuido de la siguiente forma:

- **/:** Contiene la clase principal que inicia la aplicación Spring Boot. Esta clase es el punto de entrada del sistema y configura el contexto inicial de la aplicación.
- **/config:** Incluye la clase responsable de la configuración general del sistema. Esta clase define el acceso a servicios de terceros, las operaciones permitidas y las rutas o directorios autorizados.
- **/controller:** Este paquete agrupa las clases encargadas de gestionar las solicitudes entrantes desde el subsistema Frontend. Las clases de control están organizadas en función de las principales funcionalidades del sistema, de la siguiente manera:
 - **AuthenticationController.java:** Gestiona las solicitudes relacionadas con la autenticación de los usuarios, incluyendo el inicio de sesión y la validación de credenciales.
 - **ExercisesController.java:** Administra las peticiones relacionadas con los ejercicios físicos. Es la clase más compleja y central para la funcionalidad del sistema.
 - **StatisticsController.java:** Maneja las solicitudes correspondientes a la gestión y consulta de estadísticas de usuario.

- **UserController.java:** Se encarga de gestionar las peticiones relacionadas con la información del usuario, como su perfil y datos personales.
- **/model:** Este paquete incluye las clases destinadas al almacenamiento temporal de datos y su correcta estructuración. Estas clases actúan como representaciones de datos en el sistema, permitiendo su manipulación y transferencia. Las principales clases del modelo son:
 - **BasicStatistics.java**
 - **CategoryCount.java**
 - **CurrentRoutine.java**
 - **CurrentRoutineExercises.java**
 - **Exercise.java**
 - **PublicRoutine.java**
 - **RoutineHistory.java**
 - **User.java**
- **/services:** El paquete de servicios contiene clases que complementan a los controladores, proporcionando la lógica necesaria para gestionar las operaciones solicitadas en la información contenida en la base de datos. Las clases presentes son:
 - **AuthenticationServices.java**
 - **ExercisesServices.java**
 - **StatisticsServices.java**
 - **UserServices.java**
- **/utils:** Incluye clases utilitarias que proporcionan funcionalidades reutilizables en diferentes partes del sistema. Estas clases simplifican tareas recurrentes, mejorando la eficiencia del desarrollo. Los componentes destacados son:
 - **DBConnection.java:** Se encarga de gestionar la comunicación con la base de datos, así como el tipo de operación a realizar sobre la información.

5.7.4. Implementación de BlazePose

La implementación de BlazePose en JavaScript permite aprovechar esta avanzada tecnología de visión por computadora directamente en aplicaciones web, facilitando el análisis y seguimiento de los movimientos del cuerpo humano en tiempo real. Este proceso se realiza mediante una combinación de librerías especializadas y tecnologías optimizadas para el procesamiento de imágenes en el navegador.

Las librerías principales utilizadas son las siguientes:

- **TensorFlow.js:** Es una de las librerías más importantes en esta implementación. Permite ejecutar modelos de aprendizaje automático directamente en el navegador mediante JavaScript. Facilita la carga y ejecución del modelo preentrenado de BlazePose, proporcionando las herramientas necesarias para el análisis y procesamiento de imágenes en tiempo real.
- **MediaPipe BlazePose:** Desarrollada por Google, MediaPipe es un marco diseñado para crear soluciones de procesamiento de imágenes y videos. BlazePose, como solución específica de MediaPie, permite el

reconocimiento de las articulaciones clave del cuerpo, generando un mapa preciso de las posturas del usuario.

- **WebGL:** Aunque no es una librería de visión artificial per se, juega un papel crucial al permitir la aceleración por hardware (GPU) para el procesamiento eficiente de los datos visuales, garantizando que los modelos puedan ejecutarse con fluidez incluso en dispositivos con recursos limitados.

La implementación de BlazePose comienza con la integración del modelo en una aplicación web, en la cual se utiliza la API de acceso a dispositivos multimedia (MediaDevices API) para acceder a la cámara del dispositivo del usuario. Cada cuadro del video se captura y se envía al modelo de BlazePose cargado en TensorFlow.js. El modelo analiza la imagen y detecta los puntos clave del cuerpo, en la Figura 29 se muestran los 33 puntos que BlazePose devuelve tras el procesamiento de la imagen. Tras ello, se genera un esqueleto virtual con estos puntos clave, que representa la postura. El esqueleto se dibuja en superposición al video original, una otorgando retroalimentación clara y sin riesgo de malinterpretación.

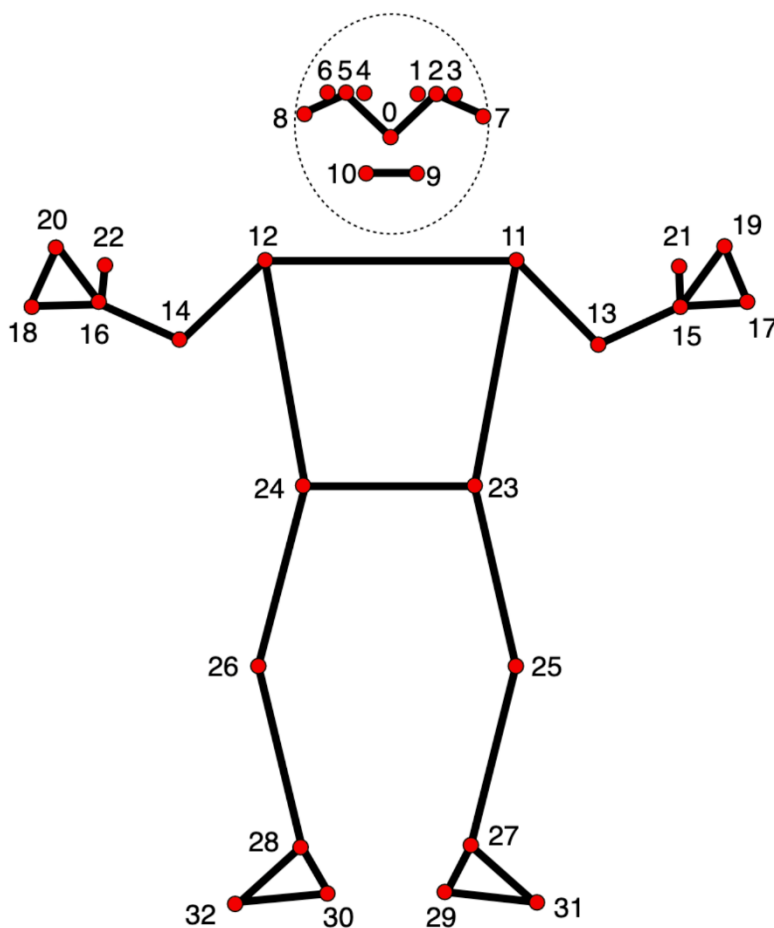


Figura 29: Puntos clave devueltos por BlazePose

5.7.5. Implementación del sistema de reconocimiento de ejercicios

El reconocimiento preciso de ejercicios mediante tecnologías de estimación de posturas requiere una adecuada representación de la información asociada a cada ejercicio. En este contexto, un ejercicio puede descomponerse en dos fases

fundamentales: una fase inicial y una fase final. Cada una de estas fases está definida por una postura específica, y al detectar correctamente en qué fase se encuentra el usuario y cuál fue la fase anterior, es posible determinar si se ha completado una repetición de un ejercicio concreto.

Bajo esta premisa, surge la necesidad de desarrollar un algoritmo unificado capaz de reconocer una amplia variedad de ejercicios de manera precisa y eficiente. Este algoritmo plantea tres enfoques principales para su funcionamiento:

- **Cálculo de ángulos entre rectas:** Este enfoque se basa en determinar la fase del ejercicio mediante el cálculo de los ángulos formados entre dos rectas, lo que permite evaluar si una articulación está doblada o extendida.
 - Para calcular un ángulo entre dos rectas, es necesario escoger tres de los puntos clave que nos genera la tecnología de estimación de posturas. Cada punto representa una articulación relevante.
 - Con estos tres puntos se puede calcular las ecuaciones de dos rectas y calcular su intersección, para más tarde calcular el ángulo en grados que forman utilizando la función de arcocoseno (Figura 30).

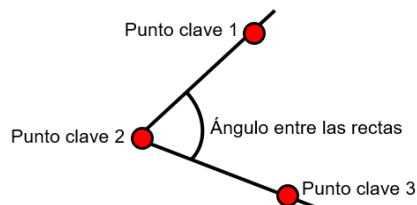


Figura 30: Ángulo entre dos rectas a raíz de tres puntos

Este método es útil para ejercicios en los que la posición angular de las articulaciones es el principal indicador de la correcta ejecución

- **Cálculo de ángulos y comparación entre distancias euclídeas:** En ciertos ejercicios, el cálculo de ángulos puede no ser suficientemente preciso debido a variaciones en la posición de la cámara o el ángulo de colocación del usuario. En estos casos, el algoritmo recurre a la comparación de alturas relativas entre dos puntos clave junto al cálculo de ángulos. Ejemplos:
 - Ejemplo 1: El ejercicio curl de bíceps consiste en levantar el peso sostenido en las manos (Figura 31), es decir, en la fase final del ejercicio, la articulación de la muñeca está por encima de la articulación del codo y el ángulo que forman las rectas correspondientes al brazo y antebrazo es un ángulo agudo, mientras que en la fase inicial del ejercicio, la muñeca está por debajo del hombro y el ángulo entre el brazo y antebrazo es obtuso.

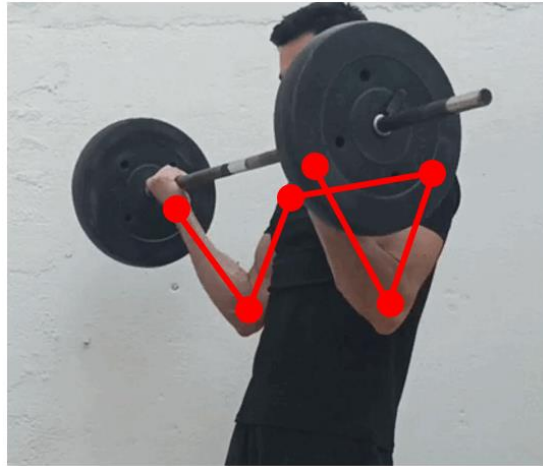


Figura 31: Ejemplo 1: Curl de biceps

- Ejemplo 3: Las flexiones son un ejercicio que implica el levantamiento del peso corporal mediante el uso de las extremidades superiores (Figura 32). Durante su ejecución, el hombro desciende hasta alcanzar, como mínimo, la altura del codo. En la fase inicial, la articulación del hombro se encuentra por encima del codo y el ángulo que forman las rectas correspondientes al brazo y al antebrazo forman un ángulo llano, mientras que en la fase final, el hombro se sitúa por debajo del codo y el ángulo es agudo.

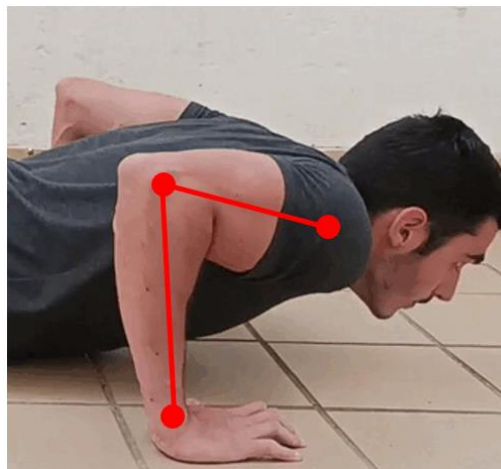


Figura 32: Ejemplo 2: Flexiones

- Las sentadillas son un ejercicio que implica levantar un peso, generalmente el propio peso corporal, mediante el uso de las extremidades inferiores (Figura 33). Durante su ejecución, la cadera desciende hasta alcanzar, como mínimo, la altura de las rodillas. En la fase inicial, la articulación de la cadera se encuentra por encima de la rodilla, formando un ángulo llano entre la pierna superior (fémur) y la inferior (tibia y peroné), mientras que en la fase final la cadera se sitúa por debajo de la rodilla y el ángulo es agudo.

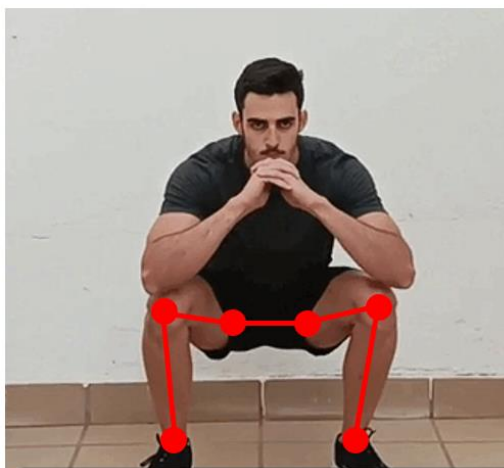


Figura 33: Ejemplo 3: Sentadillas

El objetivo principal de este algoritmo es ofrecer una solución flexible y precisa que permita reconocer una amplia variedad de ejercicios sin sacrificar la simplicidad en su implementación. De hecho, para implementarlo correctamente es necesario definir una serie de parámetros para cada ejercicio:

- Tres puntos clave para el cálculo de ángulos. Por ejemplo, en el curl de bíceps, estos puntos serían los correspondientes al hombro, codo y muñeca.
- Dos puntos clave para la comparación de alturas si procede, en el mismo ejemplo serían el codo y la muñeca
- Estos puntos deben ser identificados tanto para el lado derecho como izquierdo del cuerpo humano, resultando en 10 parámetros iniciales que otorgan bilateralidad.
- Cuatro parámetros correspondientes a los rangos de los ángulos:
 - Rango de ángulos para la fase inicial (ángulo máximo y mínimo).
 - Rango de ángulos para la fase final (ángulo máximo y mínimo).
- Un atributo de configuración del algoritmo, que indica cuál de las tres vertientes (ángulos, distancias o ambas) se utilizará para la evaluación.

5.8. PRUEBAS

La ejecución de prueba durante el desarrollo de un proyecto de software es un proceso esencial para verificar el correcto funcionamiento del sistema y detectar posibles errores inherentes al código en etapas tempranas.

En primer lugar, se llevaron a cabo pruebas unitarias para cada funcionalidad implementada, evaluando su comportamiento individual en el momento de su integración al sistema. Este enfoque permitió prevenir errores en cascada que pudieran surgir al incorporar nuevas funcionalidades dependientes.

Posteriormente, se realizaron pruebas a nivel de subsistema, donde cada componente fue evaluado de forma independiente. Tras su correcta validación, se llevaron a cabo pruebas globales con el objetivo de identificar y corregir posibles inconsistencias derivadas de la interacción entre los módulos ya existentes y los nuevos

elementos añadidos. Finalmente, una vez completado el desarrollo del sistema en su totalidad, se efectuaron pruebas integrales finales. Estas pruebas permitieron validar el funcionamiento completo del sistema, asegurando su estabilidad y correcto desempeño antes de proceder con su entrega final.

5.9. FUNCIONALIDADES DEL SISTEMA

A continuación, se presenta un resumen de las principales funcionalidades del sistema, organizadas según los roles de usuario definidos en la plataforma.

Para obtener información más detallada sobre la interacción del usuario con el sistema, se puede consultar el Anexo VI: Manual de usuario.

5.9.1. Usuario sin autenticar

Corresponde a la situación en la que un usuario aún no ha completado el proceso de autenticación en el sistema. En consecuencia, su interacción se encuentra restringida exclusivamente a acciones vinculadas con la autenticación y el acceso a la plataforma.

5.9.1.1. Iniciar sesión

En la Figura 34 se presenta la interfaz de usuario para acceder a la plataforma. El acceso se lleva a cabo utilizando el método tradicional de autenticación mediante usuario y contraseña.

El formulario incluye opciones para registrar un nuevo usuario y un mecanismo para recuperar la contraseña en caso de que sea olvidada.

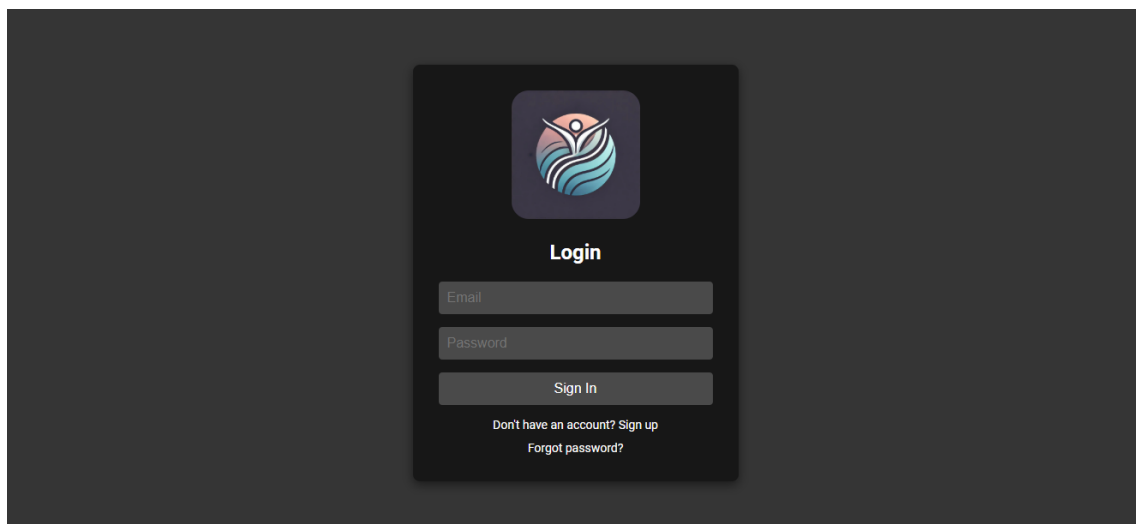


Figura 34: Iniciar sesión

5.9.1.2. Establecer contraseña

En la Figura 35 se presenta el formulario diseñado para llevar a cabo el registro de un usuario en el sistema. Esta vista incluye, además, la opción de iniciar sesión para aquellos usuarios que ya se encuentren registrados en el sistema.

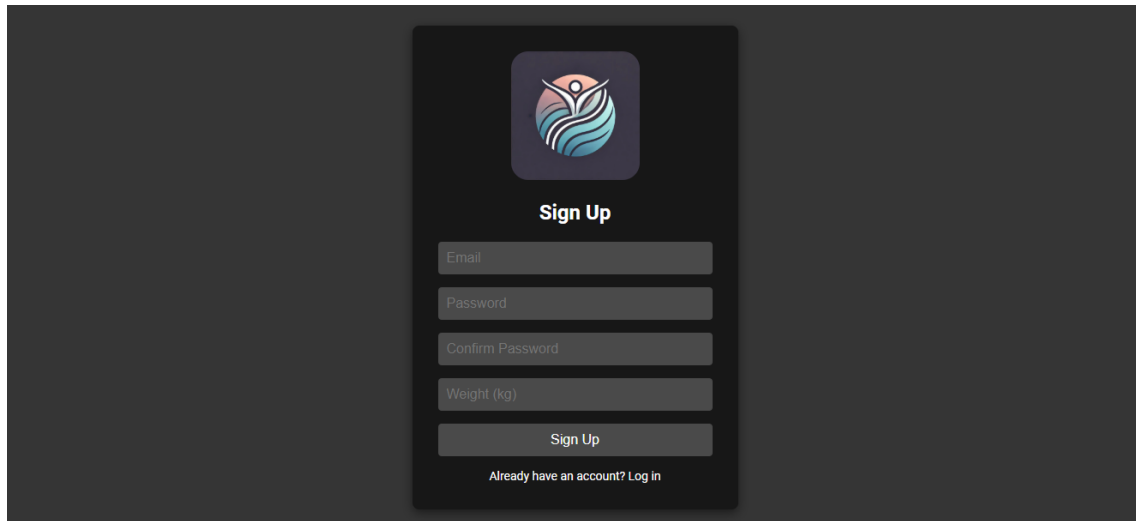
The image shows a 'Sign Up' form on a dark background. At the top is a logo featuring a stylized bird or person with arms raised, surrounded by blue and orange waves. Below the logo is the text 'Sign Up'. The form contains four input fields: 'Email', 'Password', 'Confirm Password', and 'Weight (kg)'. Below these fields is a 'Sign Up' button. At the bottom, there is a link that says 'Already have an account? Log in'.

Figura 35: Establecer contraseña

5.9.1.3. Establecer contraseña

El proceso de recuperación de contraseña implicar solicitar al usuario su dirección de correo electrónico, enviarle un mensaje con un enlace al servicio de restablecimiento de contraseña y permitirle reemplazar la contraseña anterior por una nueva. En la Figura 36 se ilustra el formulario destinado al envío del correo electrónico.

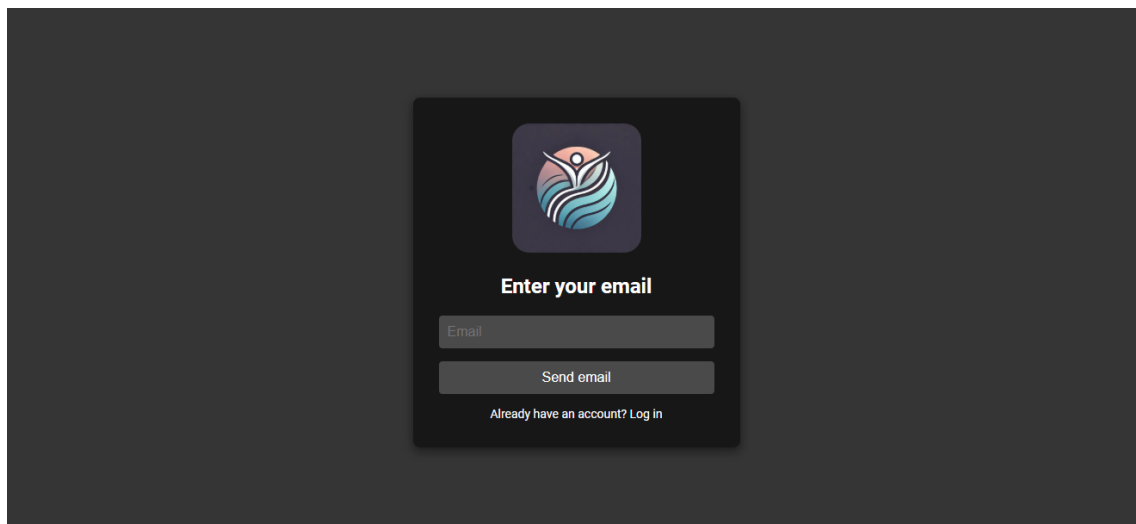
The image shows an 'Enter your email' form on a dark background. At the top is the same logo as in Figure 35. Below the logo is the text 'Enter your email'. The form contains one input field: 'Email'. Below this field is a 'Send email' button. At the bottom, there is a link that says 'Already have an account? Log in'.

Figura 36: Envío del email en la recuperación de contraseña

Tras enviarse el correo, el usuario debe acceder al enlace entregado, cuya interfaz es la mostrada en la Figura 37. Se le permitirá establecer una nueva contraseña.

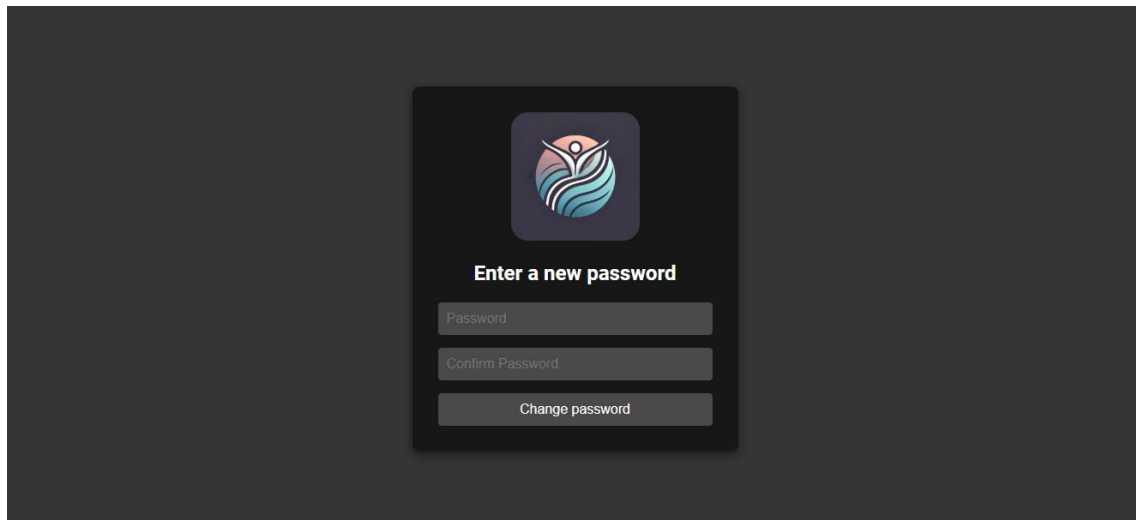


Figura 37: Cambio de contraseña

5.9.2. Usuario autenticado

Usuario que tiene acceso a la totalidad de la funcionalidad presentes en el sistema tras la autenticación. Se describen a continuación las acciones disponibles.

5.9.2.1. Gestionar datos de perfil

Se habilita la funcionalidad para visualizar y editar el peso del usuario, actualizar la contraseña y eliminar su cuenta del sistema. La Figura 38 presenta el diseño de la interfaz correspondiente, donde se facilita la visualización de esta información y el acceso a las funcionalidades mencionadas.

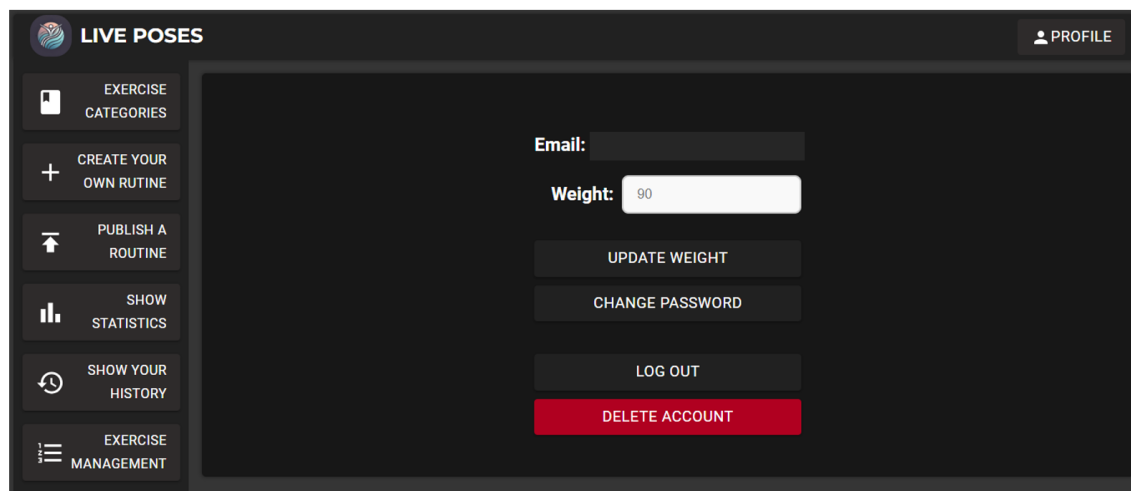


Figura 38: Perfil de usuario

El proceso de cambio de contraseña es idéntico al descrito en la recuperación de contraseña. El usuario debe introducir y confirmar la nueva contraseña (Figura 39).

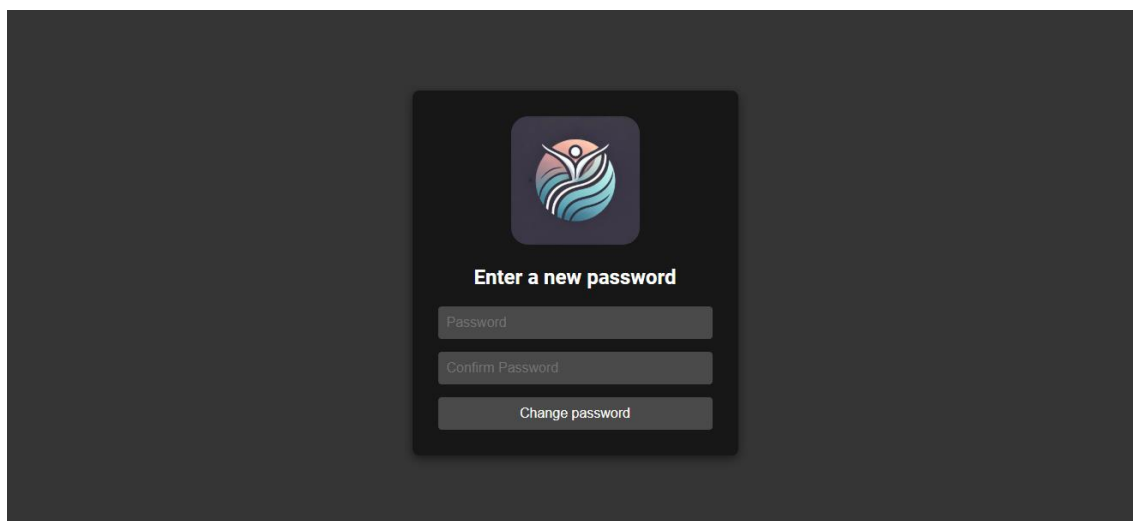


Figura 39: Cambio de contraseña

En el borrado de la cuenta aparecerá una ventana emergente solicitando la confirmación de eliminación. Se muestra en la Figura 40.

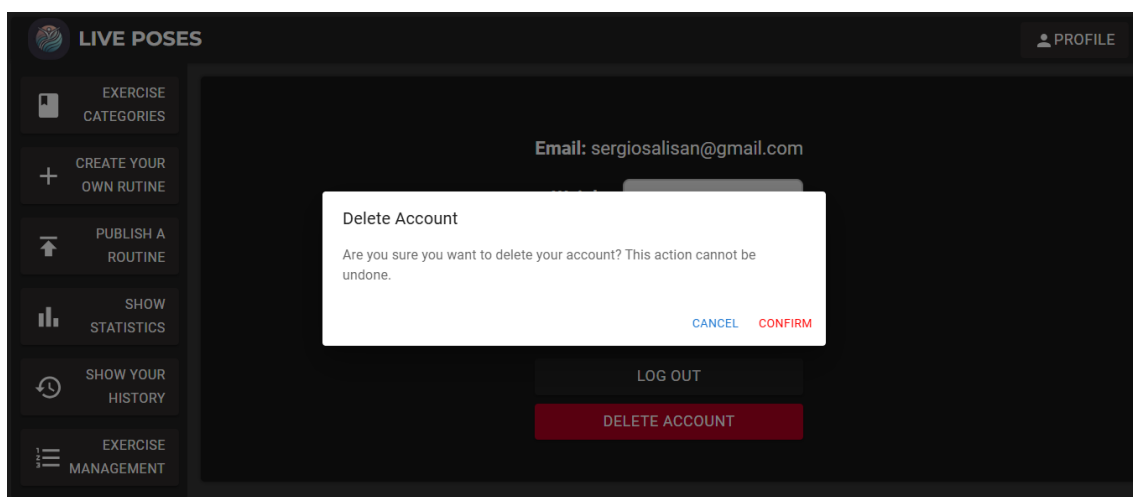


Figura 40: Eliminar cuenta

5.9.2.2. Visualizar categorías de ejercicios

Una categoría de ejercicio es un conjunto de rutinas que tienen en común su finalidad en base a los ejercicios que contienen. Por el momento se tienen cuatro categorías diferentes: rutinas de rehabilitación, flexibilidad y movilidad, fuerza y músculo y equilibrio. Se muestra a continuación (Figura 41).

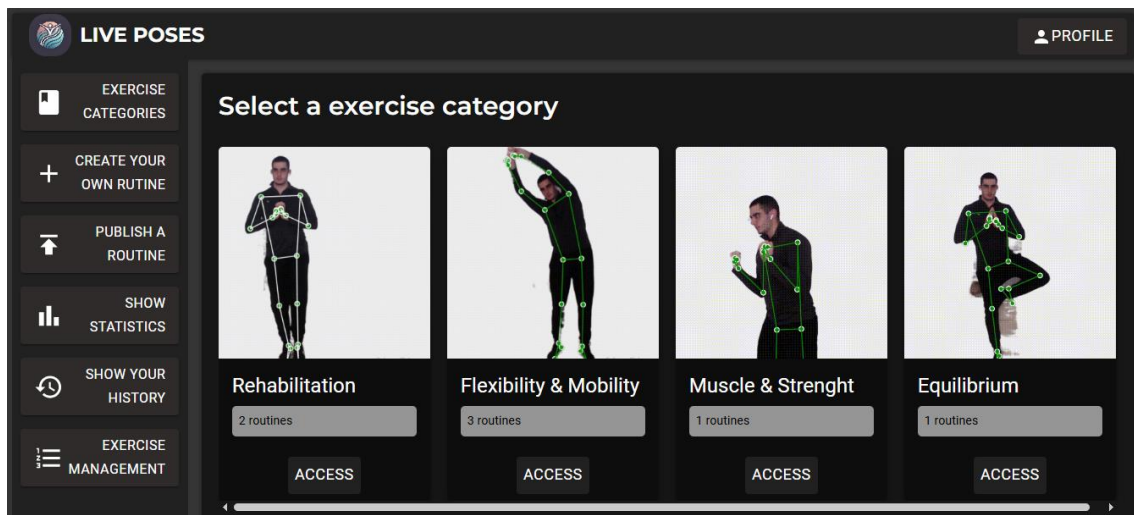


Figura 41: Ver categorías de ejercicios

Cuando se accede a una categoría, se disponen las rutinas contenidas en ella. En este momento se despliegan las opciones de poder realizar una rutina, poder eliminarla de la categoría o poder eliminar todas las rutinas de la categoría (Figura 42).

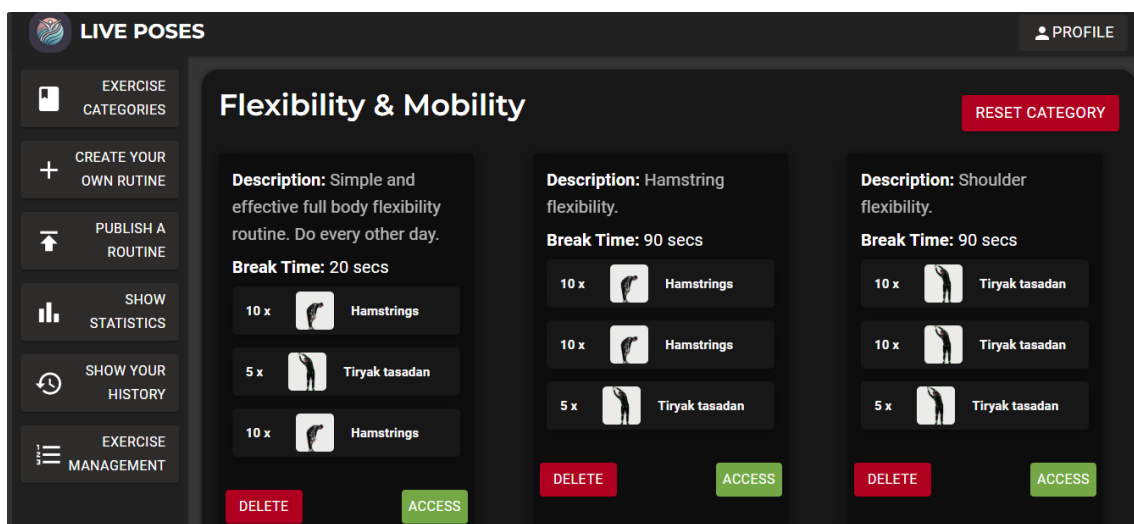


Figura 42: Categoría de ejercicios

En caso de que se quiera limpiar una categoría de ejercicios, aparecerá un mensaje de confirmación (Figura 43).

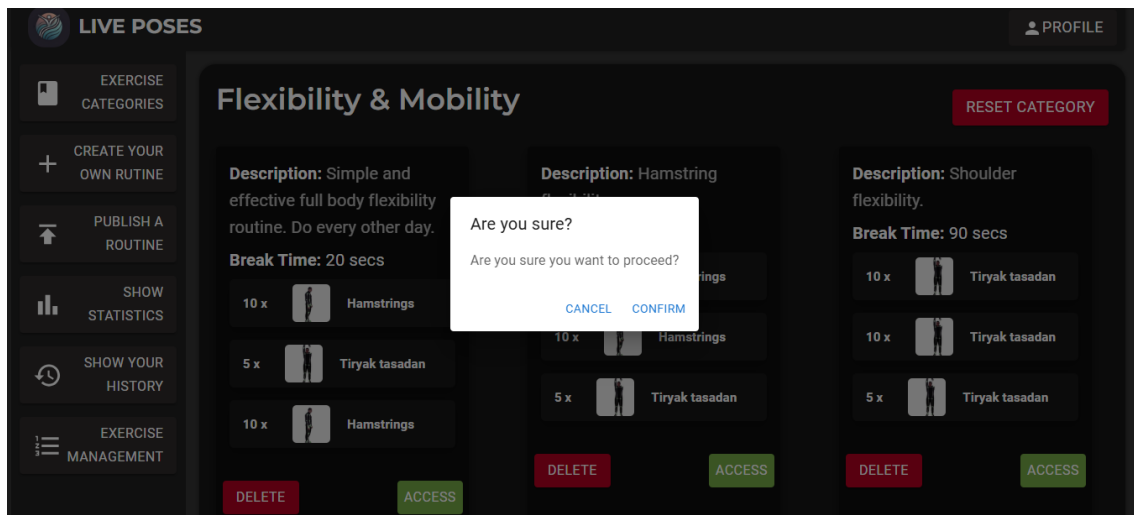


Figura 43: Eliminar todas las rutinas de una categoría

5.9.2.3. Publicar una rutina en categoría

Las rutinas presentes en las categorías pueden ser añadidas mediante un sistema de creación de rutinas (Figura 44). Consiste en añadir ejercicios, sus repeticiones y su disposición en orden, especificar el tiempo de descanso, opcionalmente una descripción y seleccionar la categoría a la que va destinado.

LIVE POSES PROFILE

EXERCISE CATEGORIES

+ CREATE YOUR OWN ROUTINE

PUBLISH A ROUTINE

SHOW STATISTICS

SHOW YOUR HISTORY

EXERCISE MANAGEMENT

Routine building

10 x Front leg raise + - [trash] [up] [down]

5 x Lunges + - [trash] [up] [down]

10 x Front leg raise + - [trash] [up] [down]

ADD EXERCISE

Set rest time (secs)

Give a description

Simple routine to improve hip mobility. Do it twice a week

89 words remaining

Select a category

Rehabilitation

PUBLISH ROUTINE

Figura 44: Publicar una rutina

5.9.2.4. Creación de una rutina personal

El sistema permite crear una rutina de carácter personal para ejecutarla inmediatamente después de su configuración. El proceso es idéntico al de publicar una rutina en una categoría, prescindiendo de su descripción y la selección de la categoría. En la Figura 45 se muestra un ejemplo de rutina configurada.

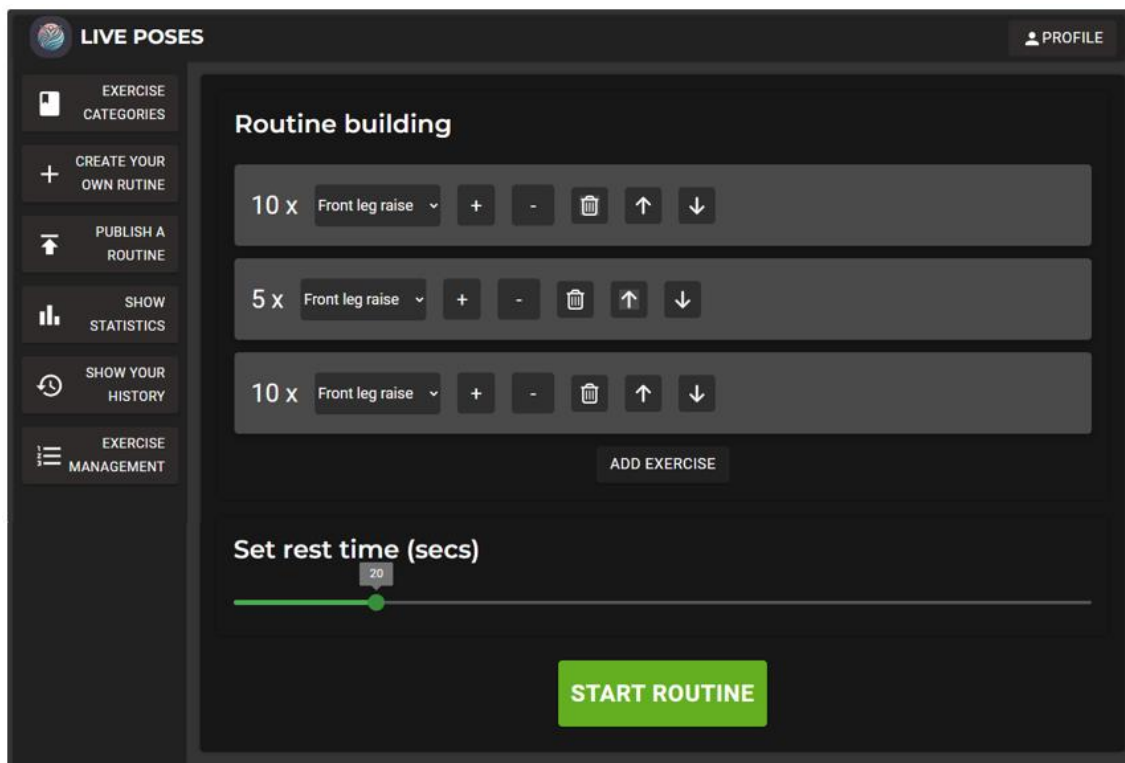


Figura 45: Configuración de rutinas personales

5.9.2.5. Ejecución de rutinas

Cuando una rutina es seleccionada/configurada, se podrá ejecutar haciendo uso de la tecnología de estimación de posturas. Se utiliza una implementación de la tecnología BlazePose, capaz de capturar y procesar la postura del cuerpo completo del usuario mediante una cámara web en formato de vídeo. Se ofrece una retroalimentación en tiempo de ejecución mostrando la postura actual del usuario. Además, se utiliza un algoritmo diseñado para el reconocimiento de posturas específicas, lo que permite ver qué ejercicio y cómo se está ejecutando. El sistema cuenta con la capacidad de contar repeticiones, recoger la precisión con la que se ha realizado el ejercicio y tomar el tiempo de ejecución.

En la Figura 46 se muestra la postura capturada por esta tecnología y su disposición en pantalla.

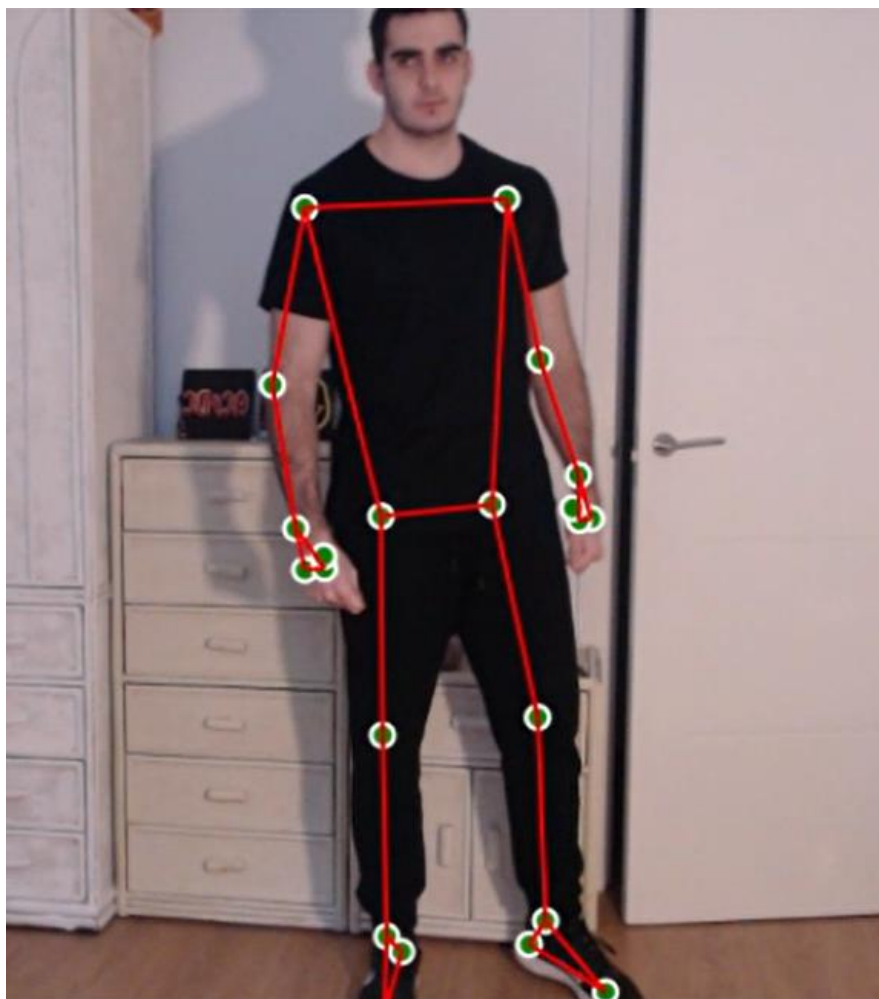


Figura 46: Tecnología de estimación de posturas

En la Figura 47 se muestra la disposición de los elementos en la pantalla, teniendo a la izquierda la postura del usuario, a la derecha la guía del ejercicio, abajo la toma de la precisión por cada repetición del ejercicio y el recuadro de información contiene el nombre del ejercicio, las repeticiones que se llevan y las restantes, el tiempo de ejecución y el tiempo de descanso restante.

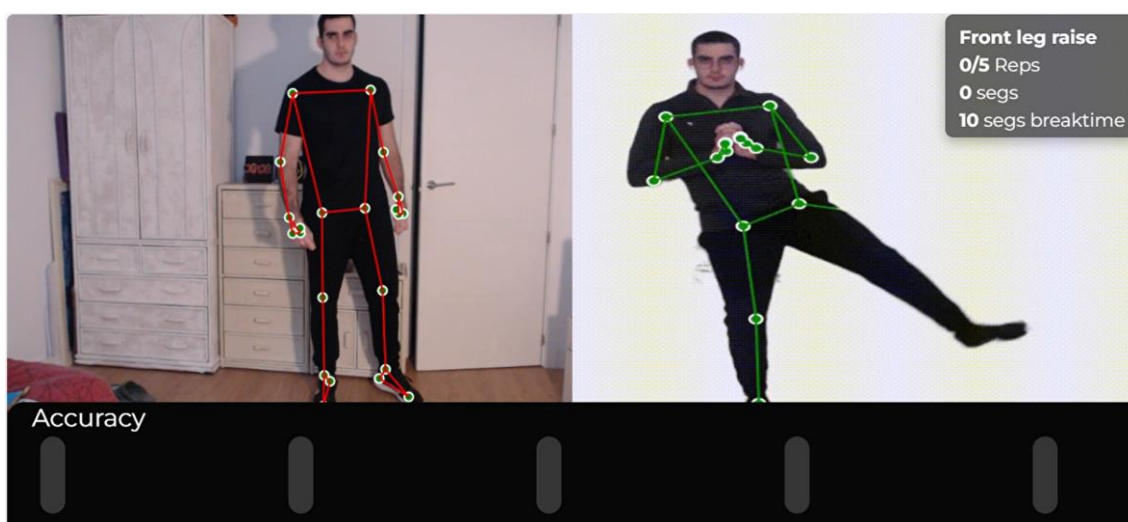


Figura 47: Ejecución de rutinas

Para comenzar una rutina se debe realizar un gesto con uno de los dos brazos indiferentemente, el cuál consiste en levantar la mano por encima de los hombros formando, más o menos, un ángulo recto entre su brazo y antebrazo. Este gesto se visualiza en la Figura 48.

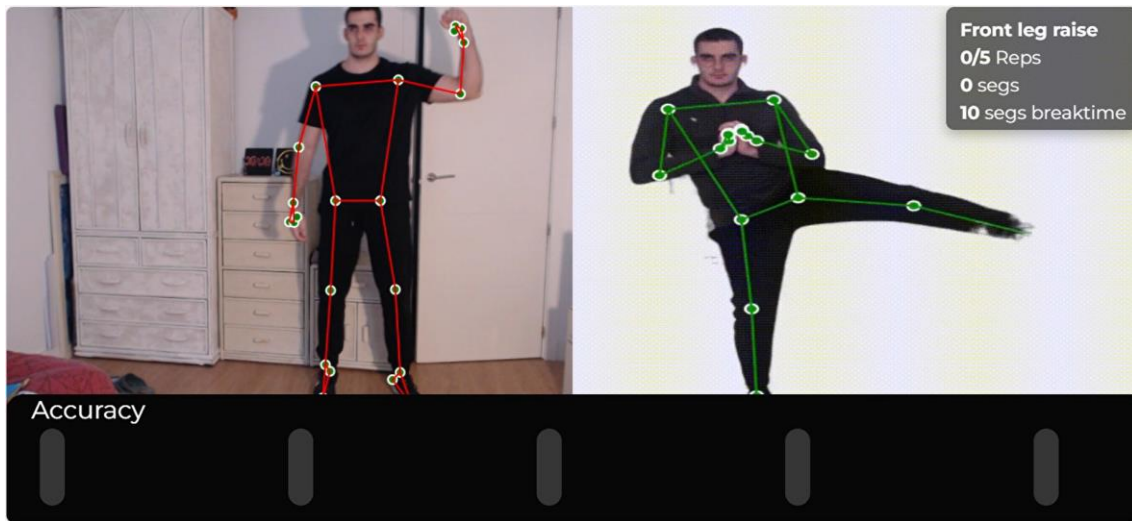


Figura 48: Gesto inicial

Tras realizar este gesto, el color de la postura dibujada en la pantalla cambiará de color desde rojo (bloqueado) al color naranja (bloqueado a punto de desbloquearse). Cuando se realiza el gesto se dejan 3 segundos para que el usuario se prepare (color naranja) (Figura 49).

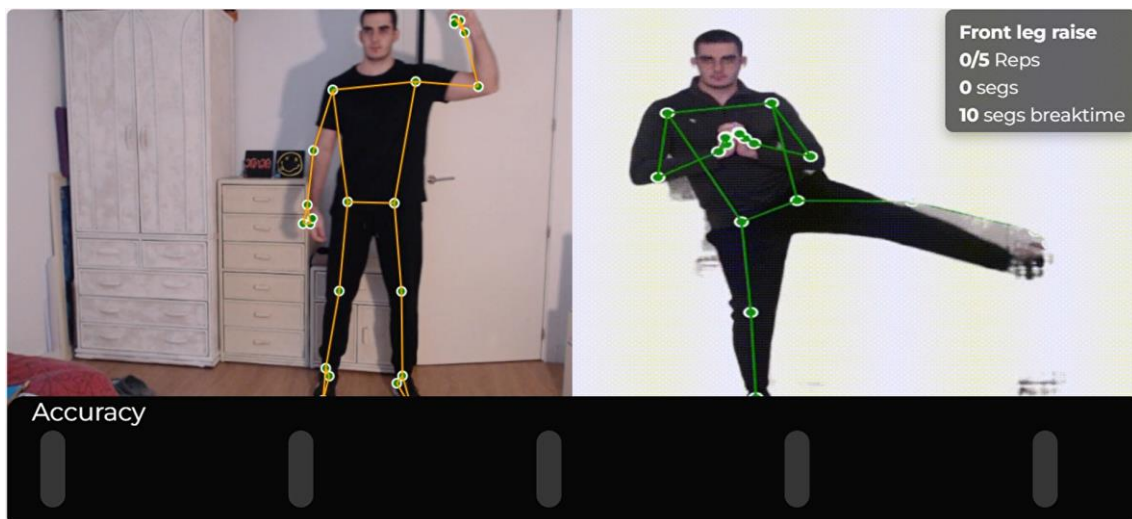


Figura 49: Estado de preparación

Cuando pasan los tres segundos, el esqueleto volverá a cambiar de color a blanco indicando el comienzo de la actividad, el contador de tiempo se pondrá en marcha y el ejercicio podrá ser ejecutado (Figura 50).

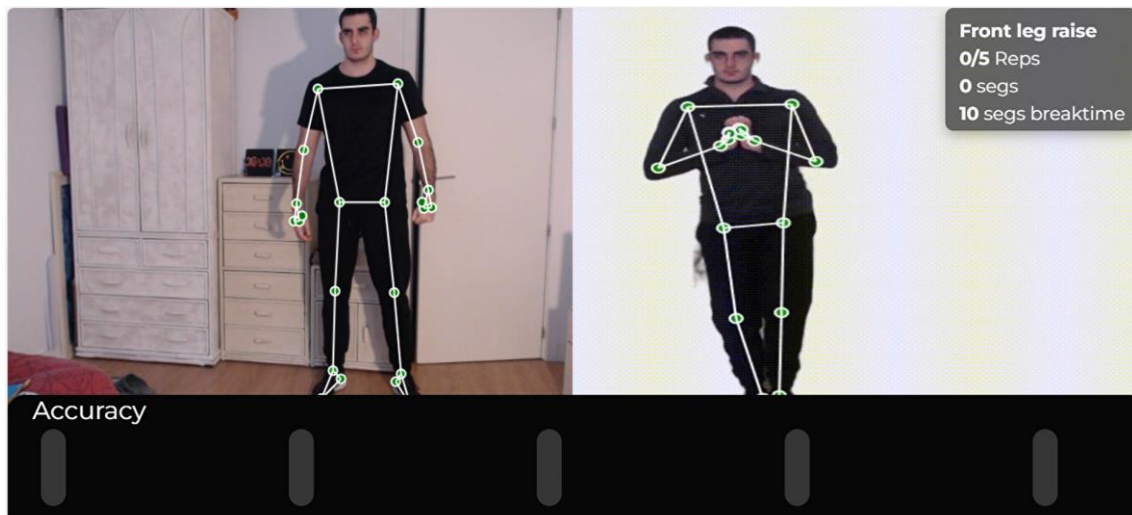


Figura 50: Comienzo de la rutina

Cuando se verifica que la ejecución de un ejercicio es correcta, el esqueleto se dibujará de color verde y el rectángulo de precisión correspondiente a esa repetición modificará su altura en base al rango de ejercicio realizado (Figura 51).

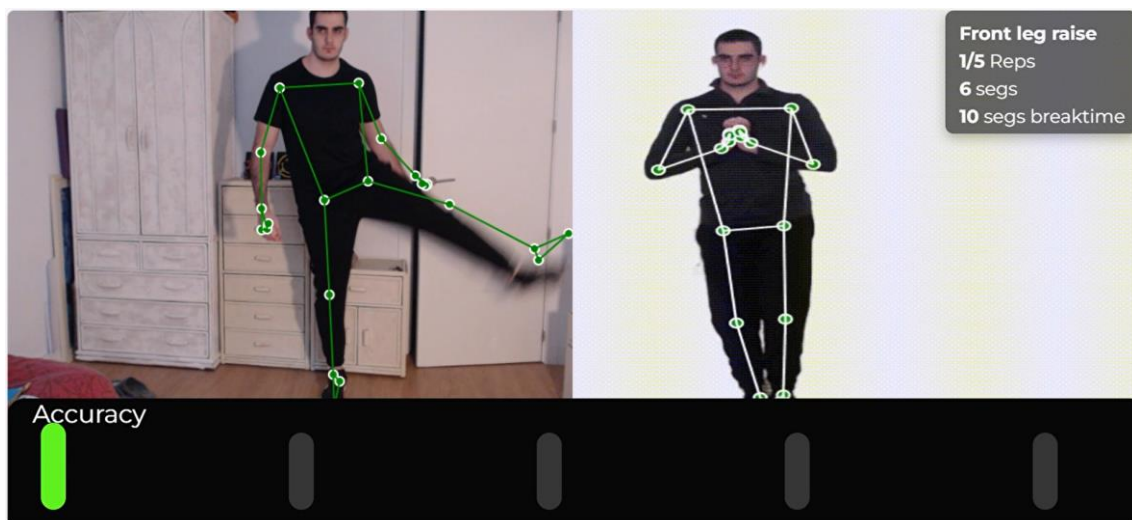


Figura 51: Ejercicio completado

Una vez completadas todas las repeticiones establecidas para un ejercicio específico, o si el usuario presiona el botón para avanzar al siguiente ejercicio, se procederá a realizar el tiempo de descanso correspondiente, en caso de que esté configurado. Durante el intervalo de tiempo asignado al descanso, el esqueleto cambia nuevamente a un color rojo para indicar el bloque de las funcionalidades. En este estado, el contador de repeticiones y de tiempo de ejecución permanecen desactivados (Figura 52). Al finalizar dicho período, se iniciará automáticamente la ejecución del próximo ejercicio.

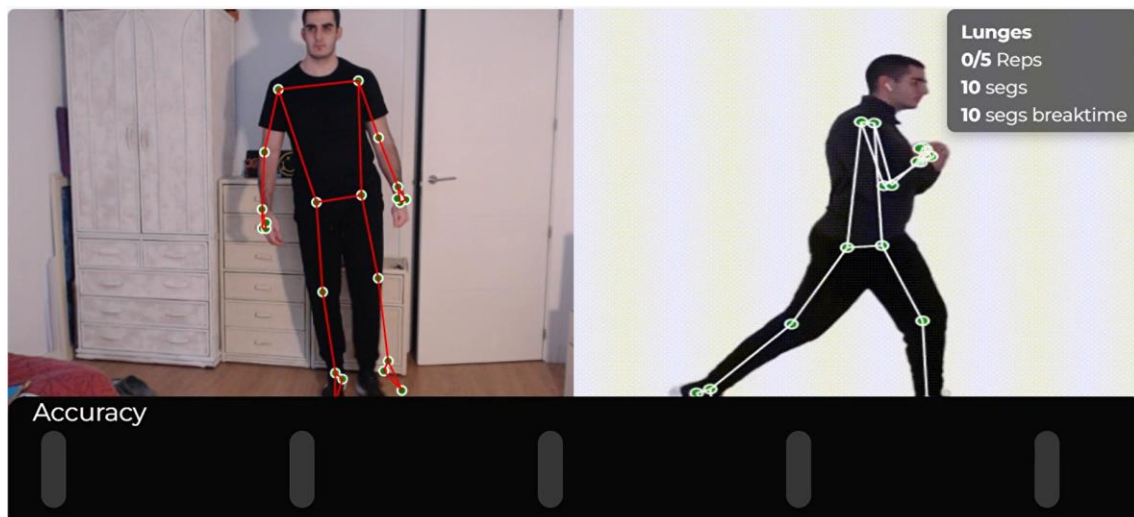


Figura 52: Tiempo de desahanso

5.9.2.6. Historial de rutinas

Cuando una rutina es ejecutada, se almacena en un conjunto llamado “Historial de rutinas”. Se almacenan y proyectan todas las rutinas que un usuario ha realizado en la plataforma pudiendo volver a ejecutarlas. En la Figura 53 se muestra la disposición de elementos en el historial de rutinas.

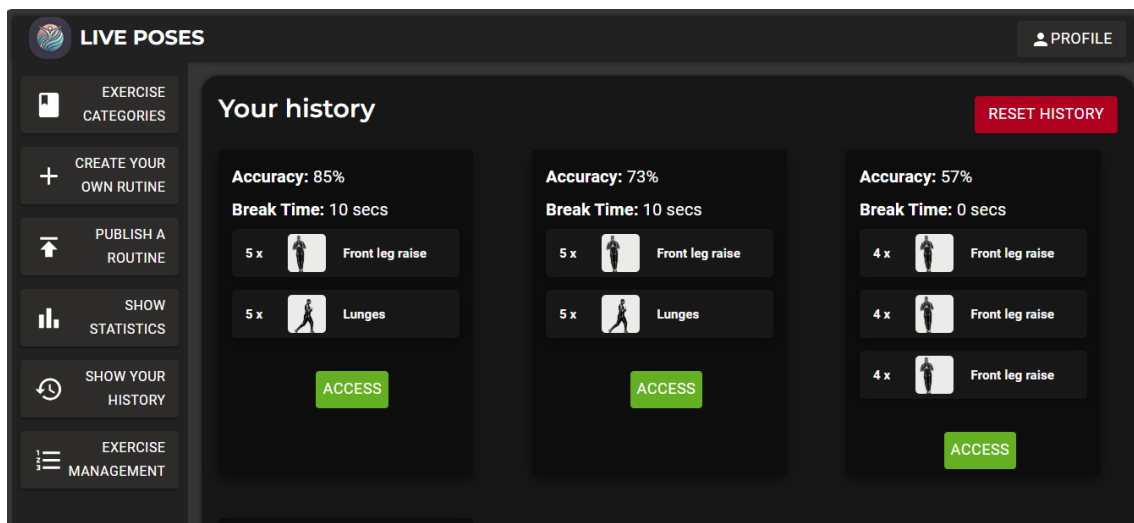


Figura 53: Historial de rutinas

Además, se permite limpiar el historial de rutinas. Se mostrará un mensaje de confirmación de la eliminación y, si se procede, se eliminarán todas las rutinas del historial (Figura 54).

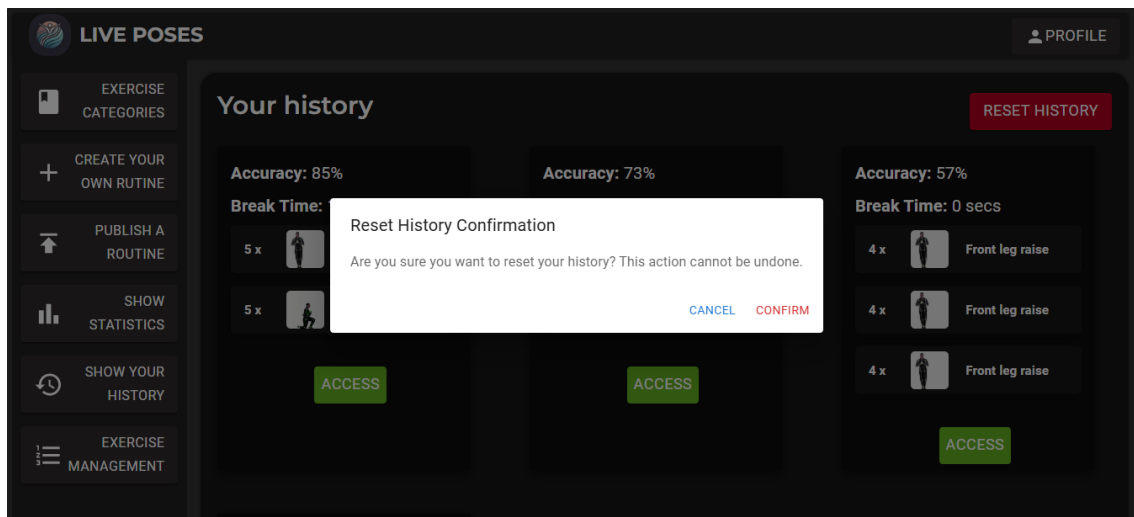


Figura 54: Eliminar todas las rutinas del historial

5.9.2.7. Estadísticas del usuario

Las estadísticas generadas por el sistema en base a la actividad del usuario en la plataforma son:

- Contador de rutinas realizadas en total
- Contador de tiempo entrenado total
- Calorías quemadas en total
- Contador de tiempo descansado total
- Precisión media
- Rutinas realizadas por días
- Tiempo entrenado por días
- Calorías quemadas por días

En la Figura 55 se muestran la disposición gráfica de estas estadísticas. Se observa que se han utilizado dos gráficos de barras y uno de líneas.

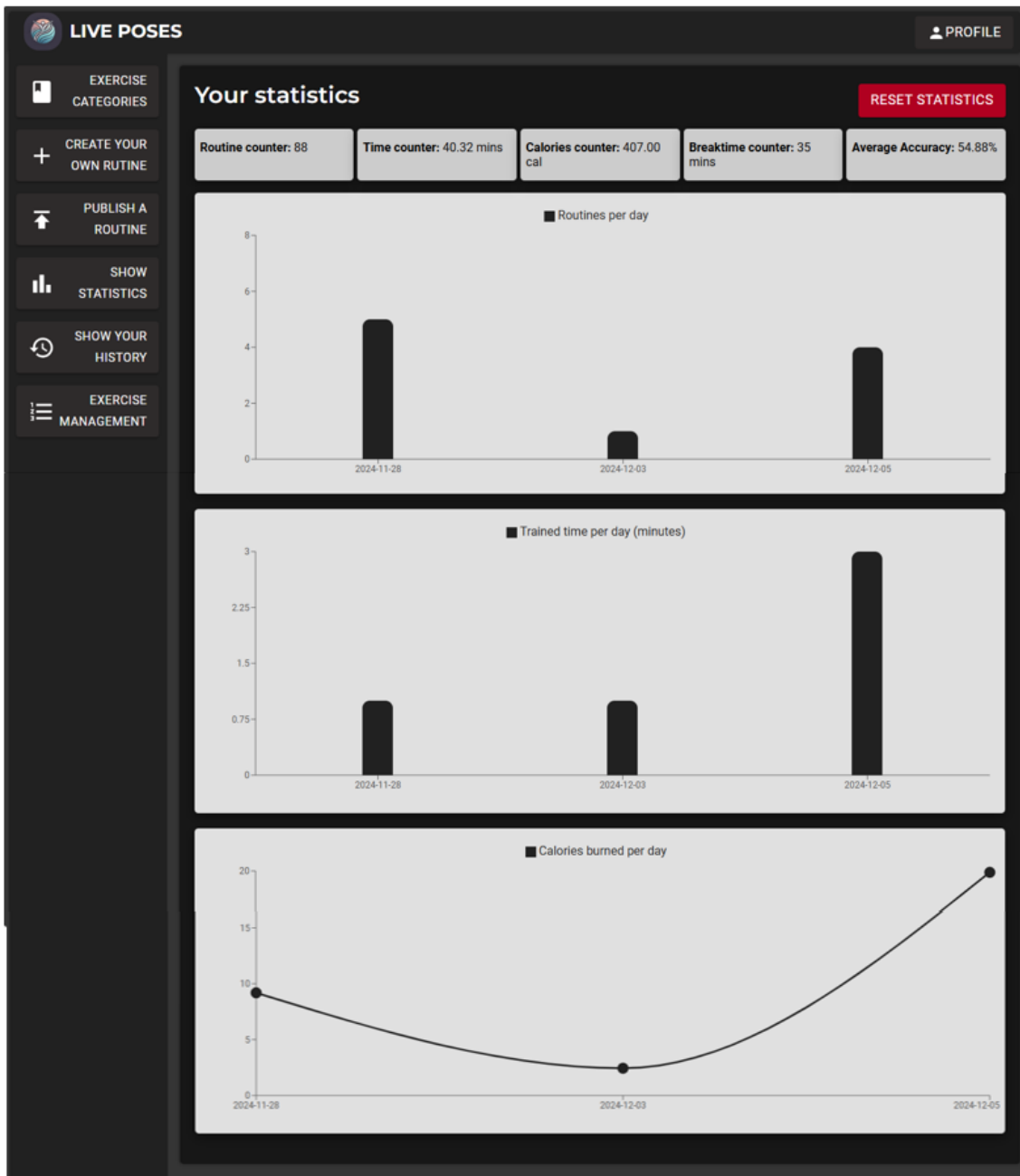


Figura 55: Estadísticas de usuario

Las calorías quemadas se determinan utilizando la Ecuación 1, que establece que el gasto calórico es el producto del peso corporal del usuario, el tiempo total de entrenamiento expresado en horas y el factor MET (Unidad Metabólica de Reposo). Para actividades anaeróbicas, que son el enfoque principal de esta aplicación, el valor del factor MET oscila entre 7.0 y 8.0 [18-19].

$$\text{Calorías quemadas} = \text{Peso} \times \text{Tiempo entrenado en horas} \times \text{MET}$$

Ecuación 1: Cálculo de calorías quemadas

Asimismo, cabe la posibilidad de reiniciar las estadísticas. Se utiliza, de nuevo, un mensaje de confirmación. Se visualiza en la Figura 56.

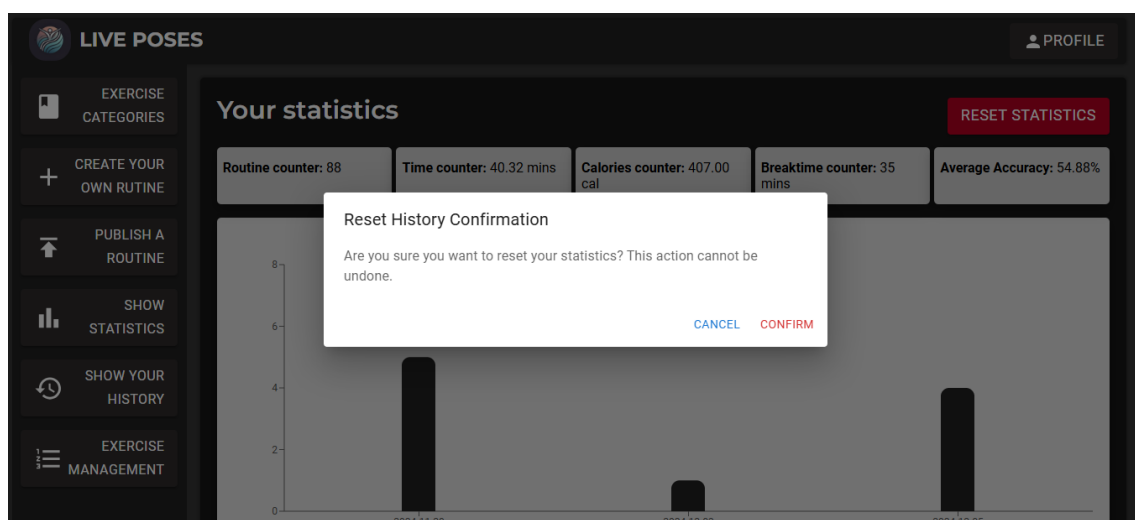


Figura 56: Reiniciar las estadísticas

5.9.2.8. Gestión de ejercicios

La plataforma tiene la capacidad de añadir, eliminar y modificar los ejercicios físicos que componen de una forma u otra las rutinas. En primer lugar, se muestran los ejercicios presentes en el sistema (Figura 57).

The screenshot shows the 'LIVE POSES' application interface. The main area is titled 'Current exercises' and features a table of exercises. Each row includes a pencil icon, a red 'X' button, the exercise name, an image, and several numerical columns representing different metrics. A 'HELP' button is in the top left of the table area, and an 'ADD EXERCISE' button is in the top right.

		Name	Image	Righth articulation 1	Righth articulation 2	Righth articulation 3	Righth articulation to distance 1	Righth articulation to distance 2	Left articulation 1
		Front leg raise		12	24	26	-1	-1	11
		Lunges		24	26	28	24	26	23
		Hamstrings		12	24	26	16	26	11
		Tiriyak tasadan		12	24	26	-1	-1	11
		Biceps curl		12	14	16	14	16	11
		Pull ups		12	14	16	16	10	11
		Standing crunch		24	26	28	26	29	23

Figura 57: Visualización de los ejercicios

La adición de ejercicios es un proceso que puede ser complicado. Para ello, se ha creado una guía sobre los elementos necesarios para que el algoritmo funcione junto con su significado (Figura 58).

LIVE POSES

PROFILE

EXERCISE CATEGORIES

CREATE YOUR OWN ROUTINE

PUBLISH A ROUTINE

SHOW STATISTICS

SHOW YOUR HISTORY

EXERCISE MANAGEMENT

Exercises management help

An exercise is based on moving one or more limbs from point A to point B. During this process, at least one joint is bent. The exercise is considered completed when the movement transitions from point A to point B and back to point A, meaning the initial position is reached, the maximum extension point (final position) is attained, and the initial position is returned to.

To determine this, the algorithm calculates angles and distances between the key joints involved in the exercise. Calculating the angle formed by two lines requires knowing the equations of the lines, which implies knowledge of three points corresponding to the three key joints of the exercise. Additionally, the angle ranges for the exercise must be provided: one range (maximum and minimum) for the initial position and another for the final position.

The algorithm is also capable of measuring Euclidean distances between joints, requiring specification of which joints to use for these measurements. These parameters are necessary to define one side of the body. Therefore, except for angles that are identical for both the right and left sides, the data must be provided for both sides individually. An image is displayed with the numbers corresponding to the joints.

In summary, the data required to be input into the algorithm are as follows:

- RightKeyPoint1: Integer First right human part articulation to calculate the angle
- RightKeyPoint2: Integer Second right human part articulation to calculate the angle
- RightKeyPoint3: Integer Third right human part articulation to calculate the angle
- RightKeyPointDistance1: Integer First right human part articulation to compare distances
- RightKeyPointDistance2: Integer Second right human part articulation to compare distances
- LeftKeyPoint1: Integer First left human part articulation to calculate the angle
- LeftKeyPoint2: Integer Second left human part articulation to calculate the angle
- LeftKeyPoint3: Integer Third left human part articulation to calculate the angle
- LeftKeyPointDistance1: Integer First left human part articulation to compare distances
- LeftKeyPointDistance2: Integer Second left human part articulation to compare distances
- UpperAngleMax: Integer Upper part of the exercise maximum allowed angle
- UpperAngleMin: Integer Upper part of the exercise minimum allowed angle
- LowerAngleMax: Integer Lower part of the exercise maximum allowed angle
- LowerAngleMin: Integer Lower part of the exercise minimum allowed angle

Figura 58: Ayuda sobre la gestión de ejercicios

La adición de ejercicios radica en dar valores a los parámetros descritos en la sección de ayuda, Figura 59.

LIVE POSES PROFILE

EXERCISE CATEGORIES

CREATE YOUR OWN ROUTINE

PUBLISH A ROUTINE

SHOW STATISTICS

SHOW YOUR HISTORY

EXERCISE MANAGEMENT

Add an exercise

Name	Image	Righth articulation 1	Righth articulation 2	Righth articulation 3	Righth articulation to distance 1	Righth articulation to distance 2	Left articulation 1	Left articulation 2	Left articulation 3
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>

ADD EXERCISE

Figura 59: Adición de ejercicios

Modificar un ejercicio consiste en dar nuevos valores a los parámetros de un ejercicio ya existente (Figura 60).

LIVE POSES PROFILE

EXERCISE CATEGORIES

CREATE YOUR OWN ROUTINE

PUBLISH A ROUTINE

SHOW STATISTICS

SHOW YOUR HISTORY

EXERCISE MANAGEMENT

Current exercises

Name	Image	Righth articulation 1	Righth articulation 2	Righth articulation 3	Righth articulation to distance 1	Righth articulation to distance 2	Left articulation 1	Left articulation 2
Front	/_next/static/media/FrontLeg	12	24	26	-1	-1	11	23

EDIT EXERCISE

Figura 60: Edición de un ejercicio

En la eliminación de un ejercicio se utiliza un mensaje de confirmación (Figura 61).

Reset History Confirmation

Are you sure you want to remove the exercise? This action cannot be undone.

[CANCEL](#) [CONFIRM](#)

Figura 61: Eliminación de un ejercicio

6. CONCLUSIONES

Al concluir el desarrollo del presente proyecto, se ha llevado a cabo un análisis exhaustivo para verificar el cumplimiento de los objetivos inicialmente planteados, permitiendo así extraer conclusiones fundamentadas. Como resultado, se confirma la implementación satisfactoria de la totalidad de los objetivos funcionales y no funcionales establecidos al inicio.

El sistema desarrollado integra de manera efectiva la tecnología BlazePose y una cámara web para la captura de la postura del usuario, permitiendo procesar y analizar la información obtenida. Asimismo, se generan y presentan estadísticas relevantes que aportan valor al usuario. De igual forma, se ha incorporado un sistema para compartir rutinas a través de las categorías de ejercicios, ampliando las posibilidades de uso de la plataforma hacia un enfoque formativo y facilitando una experiencia similar a la de una red social. Es importante destacar la implementación de un sólido control de usuarios, que permite gestionar roles diferenciados y la administración eficiente de la información asociada a cada perfil.

Además, se han resuelto el principal problema identificado en la fase inicial del proyecto. La aplicación de técnicas de estimación de posturas ha permitido diseñar un sistema capaz de reconocer la correcta ejecución de ejercicios. Esta funcionalidad proporciona al usuario una herramienta visual que facilita el aprendizaje adecuado de los movimientos, reduciendo el riesgo de lesiones y promoviendo una práctica segura. Esta característica resulta imprescindible en el ámbito de la rehabilitación, movilidad articular o cualquier otra actividad motriz o deportiva.

Se ha implementado un sistema de creación y publicación de rutinas personalizadas. Los usuarios pueden seguir rutinas diseñadas por otros miembros de la comunidad o implementar las suyas propias. Este enfoque fomenta la interacción y la compleción del sistema con diferentes tipos de rutinas y ejercicios.

Adicionalmente, la naturaleza flexible y modular del sistema facilita la incorporación de nuevos ejercicios y la configuración de rutinas orientadas a ámbitos específicos.

El desarrollo de este proyecto ha permitido consolidar los conocimientos adquiridos a lo largo de los últimos años, aplicándolos en un contexto práctico y resaltando la relevancia de seguir metodologías y buenas prácticas en el desarrollo de software. Culminar este proyecto representa una experiencia gratificante, aunque también ha evidenciado áreas susceptibles de mejora. A pesar de haber solventado la mayor parte de los inconvenientes detectados, el aprendizaje derivado de este proceso será de gran utilidad para optimizar futuros proyectos en términos de eficiencia y calidad.

7. LÍNEAS FUTURAS

Una vez concluido el desarrollo del sistema de software, se plantean diversas mejoras y la incorporación de nuevas funcionalidades:

- Implementar un sistema de adición de ejercicios mejorado basado en la utilización de la estimación de posturas. El método actual puede llegar a ser confuso si no se tienen claros qué parámetros conforman el algoritmo y cuál es el significado de cada uno de ellos. Es por esto que se propone un nuevo método con el siguiente flujo de ejecución cuando un usuario quiere añadir un ejercicio a la plataforma:
 - El usuario indica qué vertiente del algoritmo es conveniente utilizar (cálculo de ángulos, comparación de distancias o ambas).
 - El usuario indica mediante un esqueleto interactivo los puntos clave correspondientes a las articulaciones presentes en el ejercicio.
 - El usuario indica las articulaciones que se van a dedicar a la comparación de distancias, si procede.
 - Se captura la postura del usuario correspondiente a la fase inicial del ejercicio. Los ángulos y distancias calculados en esa captura serán los asociados a la fase inicial del ejercicio.
 - Se captura una segunda vez la postura del usuario, esta vez correspondiéndose con la fase final del ejercicio. Los ángulos y distancias calculados serán los asociados a la fase final del ejercicio.
- Se propone optimizar la escalabilidad del sistema mediante la implementación de índices adecuados en la base de datos, lo que permitirá mejorar significativamente los tiempos de respuesta en las consultas y operaciones realizadas. Asimismo, se sugiere habilitar la arquitectura del sistema para soportar múltiples servidores tanto de backend como en la base de datos. Esto mejorará la disponibilidad, estabilidad y distribución de la carga, garantizando continuidad operativa ante fallos.
- Utilizar trajes sensoriales especializados con el fin de capturar de manera precisa y detallada la postura del usuario durante la interacción con el sistema. Esta tecnología avanzada permite una mejora significativa en la exactitud de los datos recopilados, lo cual resulta especialmente relevante en estudios enfocados en el análisis postural. Asimismo, facilita una determinación más precisa de parámetros clave, tales como ángulos articulares, alineaciones corporales y distancias entre puntos de referencia específicos. Esto optimizará la experiencia y la efectividad de las rutinas.
- El algoritmo de reconocimiento de ejercicios actualmente desarrollado es capaz de determinar si un ejercicio ha sido realizado conociendo las tres articulaciones clave involucradas en el ejercicio, otras dos para calcular distancias y, con estos datos, calcular ángulos y distancias. Este enfoque permite utilizar el mismo algoritmo para reconocer una multitud de ejercicios. No obstante, puede que no sea suficiente en algunos casos y puede que se esté sacrificando precisión por reutilización. Es por ello que

se propone actualizar el algoritmo o desarrollar un algoritmo individualizado para cada ejercicio.

8. REFERENCIAS

- [1] Wang, J., Qiu, K., Peng, H., Fu, J., & Zhu, J. (2019, October). Ai coach: Deep human pose estimation and analysis for personalized athletic training assistance. In *Proceedings of the 27th ACM international conference on multimedia* (pp. 374-382).
- [2] Huang, Z., Liu, Y., Fang, Y., & Horn, B. K. (2018, October). Video-based fall detection for seniors with human pose estimation. In *2018 4th international conference on Universal Village (UV)* (pp. 1-4). IEEE.
- [3] Lugaresi, C., Tang, J., Nash, H., McClanahan, C., Uboweja, E., Hays, M., ... & Grundmann, M. (2019). Mediapipe: A framework for building perception pipelines. *arXiv preprint arXiv:1906.08172*.
- [4] Mroz, S., Baddour, N., McGuirk, C., Juneau, P., Tu, A., Cheung, K., & Lemaire, E. (2021, December). Comparing the quality of human pose estimation with blazepose or openpose. In *2021 4th International Conference on Bio-Engineering for Smart Technologies (BioSMART)* (pp. 1-4). IEEE.
- [5] Pauzi, A. S. B., Mohd Nazri, F. B., Sani, S., Bataineh, A. M., Hisyam, M. N., Jaafar, M. H., ... & Mohamed, A. S. A. (2021). Movement estimation using mediapipe blazepose. In *Advances in Visual Informatics: 7th International Visual Informatics Conference, IVIC 2021, Kajang, Malaysia, November 23–25, 2021, Proceedings 7* (pp. 562-571). Springer International Publishing.
- [6] Kendall, A., Grimes, M., & Cipolla, R. (2015). PoseNet: A convolutional network for real-time 6-dof camera relocalization. In *Proceedings of the IEEE international conference on computer vision* (pp. 2938-2946).
- [7] OpenPose. Accedido: 05-09-2024. Disponible en: [OpenPose: Main Page](#)
- [8] MoveNet. Accedido: 05-09-2024. Disponible en: [MoveNet: Ultra fast and accurate pose detection model. | TensorFlow Hub](#)
- [9] MediaPipe Holistic. Accedido: 05-09-2024. Disponible en: [Holistic - mediapipe](#)
- [10] Toshev, A., & Szegedy, C. (2014). Deeppose: Human pose estimation via deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1653-1660).
- [11] Kemtai. Accedido: 05-09-2024. Disponible en: [Motion Tracking Exercise Platform for Physio and Fitness | Kemtai](#)
- [12] Moreno García, M. N. *Transparencias Gestión de Proyectos. Estimación del esfuerzo.*
- [13] Moreno García, M. N., Navarro Cáceres, M. *Transparencias Gestión de Proyectos. Planificación temporal.*
- [14] García Peñalvo, F. J., Moreno García, M. N., Rodríguez-Aragón, J. F., Zato Domínguez, C. *Transparencias de Ingeniería del Software I, Tema 6.*
- [15] García Peñalvo, F. J., Moreno García, M. N., Rodríguez-Aragón, J. F., Zato Domínguez, C. *Transparencias de ingeniería del Software I, UML.*
- [16] Moreno García, M. N., García Peñalvo, F. J. *Transparencias de Ingeniería del Software II, Tema 3.*
- [17] Moreno García, M. N. *Transparencias de Ingeniería del Software II, Tema 1.*
- [18] Cálculo de calorías quemadas. Accedido: 24-11-2024. Disponible en: [Calcular Calorías por Entrenamiento | Virtuagym ES](#)
- [19] Relación entre la energía consumida y la actividad física (MET). Accedido: 24-11-2024. Disponible en: [Calcular Calorías por Entrenamiento | Virtuagym ES](#)

- [20] Núñez-Vinaveirán, T., Sánchez, M., Millán, P., Martínez-Méndez, J. R., Iglesias, C., Casado-Pérez, C., & García-de-Lorenzo, A. (2014). Estimación del gasto energético en el paciente quemado mediante la utilización de ecuaciones predictivas: revisión bibliográfica. *Nutrición Hospitalaria*, 29(6), 1262-1270.