

PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR

PUCE TEC

TRABAJO DE INTEGRACIÓN CURRICULAR PREVIO A LA OBTENCIÓN DEL TÍTULO DE TECNÓLOGO SUPERIOR EN DESARROLLO DE SOFTWARE

"DESARROLLO DE APLICATIVO MÓVIL PARA LA OPTIMIZACIÓN DE LA GESTIÓN Y EL SEGUIMIENTO DE RUTINAS DE ENTRENAMIENTO FÍSICO EN EL GIMNASIO DE LA PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR SEDE IBARRA"

AUTOR: "*CONFORME BERMUDEZ JOSAFATH DANIEL*"

TUTOR: "*IBARRA ESTEVEZ JOSÉ LUIS*"

IBARRA – ECUADOR

AGOSTO, 2026

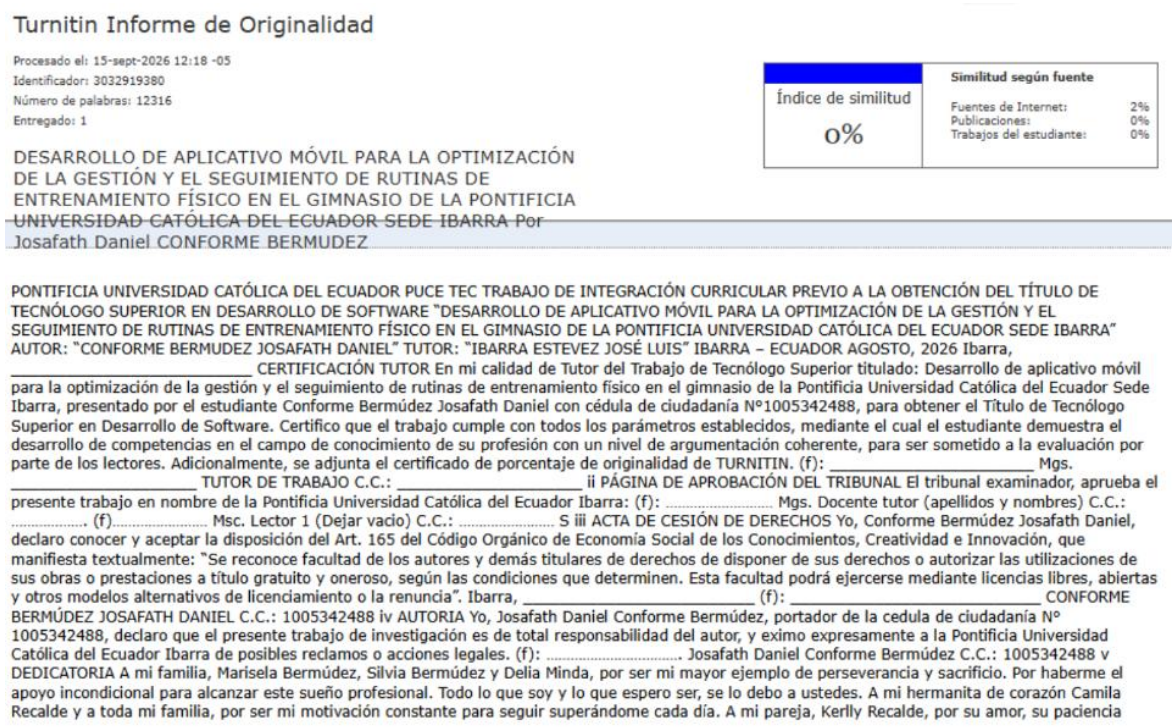
Ibarra, 16 de septiembre del 2026

CERTIFICACIÓN TUTOR

En mi calidad de Tutor del Trabajo de Tecnólogo Superior titulado: Desarrollo de aplicativo móvil para la optimización de la gestión y el seguimiento de rutinas de entrenamiento físico en el gimnasio de la Pontificia Universidad Católica del Ecuador Sede Ibarra, presentado por el estudiante Conforme Bermúdez Josafath Daniel con cédula de ciudadanía N°1005342488, para obtener el Título de Tecnólogo Superior en Desarrollo de Software.

Certifico que el trabajo cumple con todos los parámetros establecidos, mediante el cual el estudiante demuestra el desarrollo de competencias en el campo de conocimiento de su profesión con un nivel de argumentación coherente, para ser sometido a la evaluación por parte de los lectores.

Adicionalmente, se adjunta el certificado de porcentaje de originalidad de TURNITIN.



Turnitin Informe de Originalidad

Procesado el: 15-sept-2026 12:18 -05
Identificador: 3032919380
Número de palabras: 12316
Entregado: 1

DESARROLLO DE APLICATIVO MÓVIL PARA LA OPTIMIZACIÓN DE LA GESTIÓN Y EL SEGUIMIENTO DE RUTINAS DE ENTRENAMIENTO FÍSICO EN EL GIMNASIO DE LA PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR SEDE IBARRA Por Josafath Daniel CONFORME BERMUDEZ

Índice de similitud: 0%

Similitud según fuente
Fuentes de Internet: 2%
Publicaciones: 0%
Trabajos del estudiante: 0%

PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR PUCE TEC TRABAJO DE INTEGRACIÓN CURRICULAR PREVIO A LA OBTENCIÓN DEL TÍTULO DE TECNÓLOGO SUPERIOR EN DESARROLLO DE SOFTWARE "DESARROLLO DE APLICATIVO MÓVIL PARA LA OPTIMIZACIÓN DE LA GESTIÓN Y EL SEGUIMIENTO DE RUTINAS DE ENTRENAMIENTO FÍSICO EN EL GIMNASIO DE LA PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR SEDE IBARRA" AUTOR: "CONFORME BERMUDEZ JOSAFATH DANIEL" TUTOR: "IBARRA ESTEVEZ JOSÉ LUIS" IBARRA – ECUADOR AGOSTO, 2026 Ibarra, _____ CERTIFICACIÓN TUTOR En mi calidad de Tutor del Trabajo de Tecnólogo Superior titulado: Desarrollo de aplicativo móvil para la optimización de la gestión y el seguimiento de rutinas de entrenamiento físico en el gimnasio de la Pontificia Universidad Católica del Ecuador Sede Ibarra, presentado por el estudiante Conforme Bermúdez Josafath Daniel con cédula de ciudadanía N°1005342488, para obtener el Título de Tecnólogo Superior en Desarrollo de Software. Certifico que el trabajo cumple con todos los parámetros establecidos, mediante el cual el estudiante demuestra el desarrollo de competencias en el campo de conocimiento de su profesión con un nivel de argumentación coherente, para ser sometido a la evaluación por parte de los lectores. Adicionalmente, se adjunta el certificado de porcentaje de originalidad de TURNITIN. (f): _____ Mgs. _____ TUTOR DE TRABAJO C.C.: _____ ii PÁGINA DE APROBACIÓN DEL TRIBUNAL El tribunal examinador, aprueba el presente trabajo en nombre de la Pontificia Universidad Católica del Ecuador Ibarra: (f): ..... Mgs. Docente tutor (apellidos y nombres) C.C.: ....., (f)..... Msc. Lector 1 (Dejar vacio) C.C.: ..... S iii ACTA DE CESIÓN DE DERECHOS Yo, Conforme Bermúdez Josafath Daniel, declaro conocer y aceptar la disposición del Art. 165 del Código Orgánico de Economía Social de los Conocimientos, Creatividad e Innovación, que manifiesta textualmente: "Se reconoce facultad de los autores y demás titulares de derechos de disponer de sus derechos o autorizar las utilizaciones de sus obras o prestaciones a título gratuito y oneroso, según las condiciones que determinen. Esta facultad podrá ejercerse mediante licencias libres, abiertas y otros modelos alternativos de licenciamiento o la renuncia". Ibarra, _____ (f): _____ CONFORME BERMÚDEZ JOSAFATH DANIEL C.C.: 1005342488 iv AUTORIA Yo, Josafath Daniel Conforme Bermúdez, portador de la cedula de ciudadanía N° 1005342488, declaro que el presente trabajo de investigación es de total responsabilidad del autor, y eximo expresamente a la Pontificia Universidad Católica del Ecuador Ibarra de posibles reclamos o acciones legales. (f): ..... Josafath Daniel Conforme Bermúdez C.C.: 1005342488 v DEDICATORIA A mi familia, Marisela Bermúdez, Silvia Bermúdez y Delia Minda, por ser mi mayor ejemplo de perseverancia y sacrificio. Por haberme el apoyo incondicional para alcanzar este sueño profesional. Todo lo que soy y lo que espero ser, se lo debo a ustedes. A mi hermanita de corazón Camila Recalde y a toda mi familia, por ser mi motivación constante para seguir superándome cada día. A mi pareja, Kerlly Recalde, por su amor, su paciencia

Validar únicamente en FirmaEC.
Firmado electrónicamente por:
JOSE LUIS IBARRA ESTEVEZ

(f): _____

Mgs. Ibarra Estévez José Luis

TUTOR DE TRABAJO

C.C.: 1002640728

PÁGINA DE APROBACIÓN DEL TRIBUNAL

El tribunal examinador, aprueba el presente trabajo en nombre de la Pontificia Universidad Católica del Ecuador Ibarra:

Validar únicamente en FirmaEC.
Firmado electrónicamente por:
JOSE LUIS IBARRA ESTEVEZ

(f): ……

Mgs. Ibarra Estévez José Luis

C.C.: 1002640728

Galo Hernán Puetate Huera

Firmado digitalmente por Galo Hernán Puetate Huera
Fecha: 2026.09.21 08:29:42 -05'00'

(f)……

Mgs. Puetate Huera Galo Hernán

C.C.: 0401375787

ACTA DE CESIÓN DE DERECHOS

Yo, *Conforme Bermúdez Josafath Daniel*, declaro conocer y aceptar la disposición del Art. 165 del Código Orgánico de Economía Social de los Conocimientos, Creatividad e Innovación, que manifiesta textualmente: "Se reconoce facultad de los autores y demás titulares de derechos de disponer de sus derechos o autorizar las utilizaciones de sus obras o prestaciones a título gratuito y oneroso, según las condiciones que determinen. Esta facultad podrá ejercerse mediante licencias libres, abiertas y otros modelos alternativos de licenciamiento o la renuncia".

Ibarra, 15 de septiembre del 2026

(f):

CONFORME BERMÚDEZ JOSAFATH DANIEL

C.C.: 1005342488

AUTORIA

Yo, *Conforme Bermúdez Josafath Daniel*, portador de la cedula de ciudadanía N° 1005342488, declaro que el presente trabajo de investigación es de total responsabilidad del autor, y eximo expresamente a la Pontificia Universidad Católica del Ecuador Ibarra de posibles reclamos o acciones legales.

(f): …….

Josafath Daniel Conforme Bermúdez

C.C.: 1005342488

DEDICATORIA

A mi familia, Marisela Bermúdez, Silvia Bermúdez y Delia Minda, por ser mi mayor ejemplo de perseverancia y sacrificio. Por haberme el apoyo incondicional para alcanzar este sueño profesional. Todo lo que soy y lo que espero ser, se lo debo a ustedes.

A mi hermanita de corazón Camila Recalde y a toda mi familia, por ser mi motivación constante para seguir superándome cada día.

A mi pareja, Kerlly Recalde, por su amor, su paciencia infinita y por ser mi lugar de paz frente a cada desafío. Gracias por creer en mí incluso cuando yo mismo dudaba.

Y, finalmente, a mí mismo, por no rendirme y por haber logrado aquello que alguna vez pensé que podía ser imposible.

AGRADECIMIENTOS

En primer lugar, agradezco a la Pontificia Universidad Católica del Ecuador Sede Ibarra (PUCE-SI) y a todos los docentes de la carrera, por haberme brindado una formación académica y las herramientas técnicas necesarias para desarrollarme como un profesional íntegro.

Un agradecimiento especial a mi director de tesis, José Ibarra, por su guía, su tiempo y su retroalimentación constante a lo largo de este proyecto. Sus consejos fueron fundamentales para poder lograr el desarrollo de este trabajo.

A mi familia, por su esfuerzo incansable para darme la oportunidad de estudiar. Por sacrificar ciertas cosas para que yo pudiese culminar mis estudios. Gracias por comprender mis ausencias y por ser el motor principal de mi vida. Este logro es tan suyo como mío.

A mi madre, Marisela Bermúdez, por brindarme el apoyo suficiente, por estar conmigo en cada momento de tensión y esfuerzo, y por ser ese pilar fundamental y una fuente de motivación tan grande.

A mi tía, Silvia Bermúdez, por motivarme a continuar con mis estudios y brindarme el apoyo inicial que me permitió ingresar a la universidad que deseaba.

A mi pareja, Kerlly Recalde, por acompañarme durante todo este proceso y brindarme su apoyo incondicional. Gracias por escucharme, por tu empatía, por celebrar cada presentación, incluso en los momentos en que los nervios me invadían, y por sostenerme cuando las cosas se tornaban difíciles. Eres una parte fundamental de este logro.

A mis amigos, en especial a Johanna Pozo y a mis compañeros de carrera, por las horas de estudio compartido y por hacer de esta etapa universitaria una experiencia inolvidable. Las metas son más fáciles de alcanzar cuando se está rodeado de gente especial.

A mi gato, Panchito, por acompañarme en todos los desvelos y hacer más llevadera las noches frente a la computadora.

Finalmente agradezco a todos aquellos que, de una u otra forma, aportaron con un granito de arena en mi proceso de formación.

¡Muchas gracias!

ÍNDICE DE CONTENIDOS

CERTIFICACIÓN TUTOR ..... **ii**
PÁGINA DE APROBACIÓN DEL TRIBUNAL ..... **iii**
ACTA DE CESIÓN DE DERECHOS ..... **iv**
AUTORIA ..... **v**
DEDICATORIA ..... **vi**
AGRADECIMIENTOS ..... **vii**
ÍNDICE DE CONTENIDOS ..... **viii**
ÍNDICE DE TABLAS ..... **xi**
ÍNDICE DE FIGURAS ..... **xii**
RESUMEN ..... **xiii**
ABSTRACT ..... **xiv**
INTRODUCCIÓN **1**
CAPÍTULO I. ESTADO DEL ARTE **2**
1. Estado del Arte. **2**
1.1. Contextualización y Propósito 2
1.2. Problema de Investigación 2
1.3. Objetivos de la Investigación 3
1.3.1. Objetivo General 3
1.3.2. Objetivos Específicos 3
1.4. Fundamentos de Desarrollo de Software 4
1.4.1. Concepto y Evolución del Desarrollo de Software 4
1.4.2. Ciclo de Vida del Software 5
1.4.3. Metodologías de Desarrollo de Software 6
1.4.4. Análisis y Especificación de Requisitos Funcionales y No Funcionales 10
1.5. Arquitectura de Software 11
1.5.1. Concepto y Principios de la Arquitectura de Software 11
1.5.2. Arquitectura Cliente-Servidor y Arquitectura por Capas 11
1.5.3. Arquitectura Orientada a Servicios 13
1.5.4. Arquitectura de Microservicios 13
1.6. Desarrollo Móvil Multiplataforma 13
1.6.1. Concepto y Enfoques del Desarrollo Multiplataforma 13

1.6.2. Flutter como Framework de Desarrollo 13
1.6.3. Dart como Lenguaje de Programación 15
1.7. Desarrollo Backend 15
1.7.1. Node.js como Entorno de Ejecución 15
1.7.2. NestJS como Framework para el Desarrollo de APIs 15
1.8. Gestión y Persistencia de Datos 16
1.8.1. Sistemas Gestores de Bases de Datos (SGBD) 16
1.8.2. Bases de Datos SQL, NoSQL Análisis Comparativo 16
1.8.3. PostgreSQL como Motor de Base de Datos Relacional 17
1.8.4. pgAdmin como Herramienta de Administración 17
1.9. Integración de Sistemas y Documentación de Servicios 18
1.9.1. APIs REST y el Intercambio de Datos mediante JSON 18
1.9.2. Documentación de APIs con Swagger/Open API 18
1.10. Seguridad, Identidad y Control de Acceso 19
1.10.1. Fundamentos de Seguridad Informática y el Marco OWASP 19
1.10.2. Autenticación y Autorización 19
1.10.3. JSON Web Token (JWT) 19
CAPÍTULO II. MATERIALES Y MÉTODOS 21
2. Materiales y Métodos 21
2.1. Enfoque de la investigación 21
2.2. Tipo de Investigación 21
2.3. Diseño de la investigación 21
2.4. Población, Muestra, y Unidades de Estudio 21
2.4.1. Población 22
2.4.2. Muestra 22
2.4.3. Unidades de estudio 22
2.5. Desarrollo metodológico 22
2.5.1. Actores del sistema 22
2.5.2. Determinación de roles y responsabilidades 23
2.5.3. Especificación de requisitos de software 23
2.5.4. Fase de planificación 27
2.5.5. Fase de iteración 28
2.6. Diseño del sistema 29

2.6.1. Arquitectura de Software 29
2.6.2. Organización de código Frontend 29
2.6.3. Diseño de la Base de Datos 30
2.6.4. Diseño de la Interfaz de Usuario 31
CAPITULO III. RESULTADOS Y DISCUSIONES 37
3.1. Análisis e interpretación de los resultados 37
3.1.1. Resultados del Desarrollo del Aplicativo 37
3.1.2. Cumplimiento de las Historias de Usuario 37
3.1.3. Verificación de Rendimiento de la Api 38
3.1.4. Métricas de Calidad del Software 40
CONCLUSIONES Y RECOMENDACIONES 41
Conclusiones 41
Recomendaciones 42
BIBLIOGRAFÍA 43
ANEXOS 46

ÍNDICE DE TABLAS

Tabla 1 Comparación de tecnologías para el desarrollo móvil multiplataforma 14
Tabla 2 Comparación de bases de datos relacionales y no relacionales 17
Tabla 3 Roles y Responsabilidades 23
Tabla 4 Autenticación de Usuarios 23
Tabla 5 Control de Acceso 24
Tabla 6 Configuración de Perfil Físico 24
Tabla 7 Visualización de Rutinas Diarias 24
Tabla 8 Cronometro de Entrenamiento 25
Tabla 9 Acceso a Educación Biomecánica 25
Tabla 10 Historial Rendimiento Físico 25
Tabla 11 Compatibilidad y Portabilidad 26
Tabla 12 Rendimiento y Eficiencia 26
Tabla 13 Usabilidad y Experiencia de Usuario (UX) 26
Tabla 14 Disponibilidad y Persistencia de Datos 26
Tabla 15 Escalabilidad y Mantenibilidad 27
Tabla 16 Disponibilidad de Recursos Multimedia 27
Tabla 17 Historias de Usuario 28
Tabla 18 Plan de iteraciones 28
Tabla 19 Tiempos de respuesta medidos por endpoint 38
Tabla 20 Matriz de Casos de Prueba ejecutados 47

ÍNDICE DE FIGURAS

Figura 1 "Modelo en cascada" o Ciclo de Vida del Software 6
Figura 2 Scrum 8
Figura 3 Metodología Kanban 9
Figura 4 Metodología Extreme Programming 10
Figura 5 Tipos de requerimientos no funcionales 11
Figura 6 Topología Cliente/Servidor 12
Figura 7 Arquitectura de software 29
Figura 8 ApiService 30
Figura 9 Diseño Base de Datos 31
Figura 10 Splash 32
Figura 11 Login 32
Figura 12 Home 33
Figura 13 Rutinas 33
Figura 14 Ejercicios 34
Figura 15 Guías 34
Figura 16 Registro de Progreso 35
Figura 17 Cronometro Tabata 35
Figura 18 Perfil Físico 36
Figura 19 Prueba de Rendimiento Login 39
Figura 20 Prueba de Rendimiento Rutinas 39
Figura 21 Prueba de Rendimiento Registro de Progreso 40
Figura 22 Endpoint Usuario 46
Figura 23 EndPoints Administrador 46
Figura 24 Login Con Credenciales Validas 48
Figura 25 Caso de Prueba Login Credenciales Invalidas 48
Figura 26 Endpoint de rutinas sin token (no autenticado) 49
Figura 27 Registro Nuevo Peso Corporal 49
Figura 28 Prueba de Rendimiento 50
Figura 29 Prueba LazyLoading 51

RESUMEN

El presente trabajo de titulación describe el desarrollo e implementación de una aplicación móvil orientada a la gestión de entrenamientos y seguimiento físico para los estudiantes de la Pontificia Universidad Católica del Ecuador Sede Ibarra (PUCE-SI). El principal objetivo es modernizar y optimizar la experiencia de los usuarios dentro del gimnasio universitario, proporcionando una herramienta de fácil acceso, este aplicativo cuenta con módulos que permiten el acceso a la aplicación mediante el uso de las credenciales institucionales, la visualización de rutinas preestablecidas, tener un seguimiento de su peso corporal, acceso a guías de ejecución de ejercicios para asegurar la correcta ejecución de los mismos, un cronometro integrado para la gestión de tiempos. A nivel técnico la solución está construida baso la arquitectura de cliente-servidor. Utilizando Flutter con el lenguaje Dart para la creación de la interfaz móvil multiplataforma, y la lógica del negocio y los servicios web fueron desarrollados usando NestJS. La persistencia de datos se gestionó mediante PostgreSQL y para el control de accesos se utilizó JWT. Los resultados del desarrollo resultaron en un aplicativo escalable y mantenible para futuro. En conclusión, el sistema automatiza procesos que anteriormente eran manuales, fomentando la autonomía deportiva y fortaleciendo el bienestar integral de la comunidad universitaria.

Palabras clave: *Aplicación móvil, gimnasio universitario, arquitectura cliente-servidor, desarrollo de software, Flutter.*

ABSTRACT

This project describes the development and implementation of a mobile application designed to manage training sessions and track physical activity for students at the Pontificia Universidad Católica del Ecuador Sede Ibarra (PUCE-SI). The main objective is to modernize and optimize the user experience at the university gym by providing an easily accessible tool. This app features modules that allow users to log in using their institutional credentials, view preset workout routines, track their body weight, access exercise guides to ensure proper form, and use a built-in stopwatch for time management. Technically, the solution is built on a client-server architecture. Flutter with the Dart programming language was used to create the cross-platform mobile interface, while the business logic and web services were developed using NestJS. Data persistence was managed using PostgreSQL, and JWT was used for access control. The result is a scalable and maintainable application for the future. In conclusion, the system automates processes that were previously manual, promoting athletic autonomy and strengthening the overall well-being of the university community.

Keywords; *Mobile app, university gym, client-server architecture, software development, Flutter.*

INTRODUCCIÓN

La transición digital ha generado nuevas oportunidades para mejorar la optimización de procesos en distintos campos, en estos se encuentra incluido el ámbito deportivo y universitario. El uso de aplicaciones móviles se ha vuelto más común al momento de realizar ejercicio, ya sea en los gimnasios o en el hogar, dichas aplicaciones permiten al usuario tener un apoyo virtual al momento de realizar sus entrenamientos. En este contexto el incorporar herramientas tecnológicas se vuelve una alternativa viable para poder modernizar la forma en que los estudiantes pueden interactuar con sus rutinas y tener un seguimiento de su evolución física.

El siguiente proyecto tiene como finalidad el desarrollo de un aplicativo móvil que ayude a los estudiantes de la Pontificia Universidad Católica del Ecuador Sede Ibarra (PUCE-SI) al momento de realizar sus entrenamientos, diseñada para que los estudiantes, mediante sus credenciales, tengan acceso a rutinas, guías técnicas, un cronometro para gestionar sus tiempos y también un registro peso corporal. De esta manera se busca facilitar el control de entrenamiento, promover la correcta ejecución de los ejercicios y brindar una herramienta que fortalezca la organización de la actividad física dentro del campus universitario.

Desde el punto de vista tecnológico, esta solución se apoya en una arquitectura cliente-servidor, utilizando Flutter como frontend y NestJS para la construcción del backend. Además, integra PostgreSQL como sistema de base de datos y JWT (JSON Web Token) como mecanismo de autenticación de los usuarios. Esta estructura permite mantener una comunicación entre ambas partes de manera segura y con una base sólida para la escalabilidad y mantenibilidad del sistema. Todo esto en conjunto responde a una necesidad técnica del entorno universitario, ofreciendo más que un aporte practico una oportunidad para poder fortalecer el uso de soluciones móviles orientadas al bienestar estudiantil.

CAPÍTULO I. ESTADO DEL ARTE

1. Estado del Arte

1.1. Contextualización y Propósito

En la actualidad el mundo del fitness se está insertando en el área de la tecnología, mejorando la eficiencia técnica al momento de monitorear progresos y optimizando el rendimiento deportivo, de hecho, como afirman Estrada-Marcén et al. (2020) en un estudio en la Universidad de Zaragoza, aproximadamente el 61.9% de los usuarios empleaban de manera activa tecnologías en sus entrenamientos en el centro de fitness o fuera de él. Además, en el mismo estudio, ”El 64.3% de los participantes tiene la creencia de que las tecnologías fitness mejoran la adherencia a la practica de actividad física.” (p.30). De esta manera, podemos ver como el implementar aplicativos se ha convertido en un asistente al momento de gestionar el rendimiento.

En el gimnasio de la Pontificia Universidad Católica del Ecuador existe una brecha sumamente amplia entre la digitalización, el acompañamiento físico y la infraestructura física. La gran mayoría de nuevos estudiantes no tiene una guía técnica de lo que deben realizar y cómo hacerlo, debido a la falta de un entrenador de tiempo completo para cada estudiante, el gimnasio solo cuenta con una persona encargada, quien debe realizar algunas otras actividades, esto genera un riesgo al momento de levantar las pesas y emplear una mala técnica sin supervisión alguna.

Ante este problema se ha presentado el desarrollo de un aplicativo móvil multiplataforma que gestiona rutinas, registra los progresos del peso corporal y brinda una guía biomecánica al estudiante para que tenga una mejor visión de como realizar correctamente un ejercicio sin sufrir algún riesgo. Mas allá de desarrollar una aplicación se ha logrado crear una herramienta de apoyo que ayude a la gestión deportiva, mediante el uso del framework Flutter con el lenguaje Dart; obteniendo así persistencia y acceso a las guías multimedia. Esta herramienta dota al estudiante de autonomía total, mientras que las guías interactivas reducen el riesgo físico, funcionando como un asistente virtual reduciendo la carga de trabajo para la instructora física.

1.2. Problema de Investigación

En el gimnasio universitario al carecer de planes estructurados, los usuarios frecuentemente realizas sus ejercicios sin una secuencia metodológica definida esto afecta la eficiencia de sus entrenamientos y limita la obtención de resultados, como resultado, los estudiantes no permanecen en sus actividades deportivas a lo largo del tiempo.

Dar solución a este problema puede verse desde 3 perspectivas claves, abordando lo que es el impacto directo en la comunidad estudiantil, el beneficio administrativo para la universidad y el aporte desde el desarrollo de software.

- **Desde la perspectiva práctica y social:** Este proyecto aporta a los estudiantes con una herramienta que puede actuar como un asistente, dado que aporta con una guía multimedia clara, que puede reducir la dependencia de un instructor físico y crear autonomía en los usuarios. Al mismo tiempo el módulo de seguimiento de avances individuales, crea una motivación con el estudiante al poder visualizar su evolución de una manera grafica.
- **Desde la perspectiva institucional y operativa:** para la PUCE Sede Ibarra el implementar esta herramienta, representaría la transición de un modelo tradicional hacia un ecosistema deportivo inteligente. Asimismo, se puede optimizar el tiempo de la entrenadora, quien podrá enforcase en usuarios que requieran atención especializada, lo que eleva la calidad de servicio del gimnasio universitario.
- **Desde la perspectiva tecnológica:** Finalmente, este proyecto justifica la práctica de conocimiento de desarrollo de software mediante la creación de un producto escalable. El uso de una metodología ágil, esto garantiza una adaptabilidad de los requerimientos de los usuarios mediante los ciclos cortos de desarrollo, a fin de que se pueda demostrar como la tecnología puede transformar positivamente la gestión deportiva.

1.3. Objetivos de la Investigación

1.3.1. Objetivo General

Desarrollar un aplicativo móvil enfocado en la gestión de rutinas, registro de progresos y guías multimedia, para la mejora en la orientación de los estudiantes, generando autonomía y optimizando la gestión operativa del gimnasio de la Pontificia Universidad Católica del Ecuador.

1.3.2. Objetivos Específicos

- **Analiza**r los fundamentos teóricos y los requerimientos funcionales y no funcionales relacionados con la gestión digital de rutinas de entrenamiento físico, para el establecimiento de las bases técnicas del aplicativo móvil.
- **Codificar** el aplicativo móvil mediante la integración de módulos de visualización practica de rutinas, registro de cargas y guías multimedia para la automatización de la gestión deportiva en el gimnasio universitario.

- **Validar** el funcionamiento y la usabilidad del aplicativo móvil mediante pruebas técnicas y evaluación de usuario, para la verificación de su eficacia en la orientación y control del entrenamiento físico.

1.4. Fundamentos de Desarrollo de Software

"El objetivo del desarrollo de software es crear un producto que satisfaga las necesidades de los usuarios y los objetivos empresariales de forma eficaz, repetible y segura. Los desarrolladores de software, programadores e ingenieros de software desarrollan software a través de una serie de pasos denominados ciclo de vida de desarrollo de software (SDLC)." (Quiroz-Vázquez, s.f). Gracias a esto, obtenemos una base conceptual que explica por qué el desarrollo de software no es crear una solución de forma improvisada, si no siguiendo una línea y etapas para poder alcanzar calidad y mantenibilidad.

Para un proyecto como el del gimnasio para la Universidad Católica del Ecuador sede Ibarra, este fundamento es importante porque se explica cómo se estructura el proceso, asimismo, ayuda a conectar la necesidad del usuario con una solución organizada.

1.4.1. Concepto y Evolución del Desarrollo de Software

El desarrollo de software a nivel de ingeniería implica la gestión de requerimientos y es un principio que crea soluciones para los usuarios, a través de una serie de pasos denominados ciclo de vida de desarrollo de software (SDLC), de una manera eficaz, repetible y segura, ahora las empresas utilizan un modelo DevOps, el cual se encarga de acelerar la entrega de aplicaciones y servicios de mayor calidad, dando así aplicaciones que tengan una mejora continua en el rendimiento del software, con mayor facilidad de automatización, enfocándose en modelos más adaptativos y no tan rígidos. En la actualidad el desarrollo de software se ha vuelto un proceso crucial, ya que el software es omnipresente y da vida a las aplicaciones y sistemas en gran parte de la vida moderna, que puede ser de todo tipo dependiendo la solución que se necesite. (Quiroz-Vázquez, s.f).

Esta evolución en el contexto del aplicativo para el gimnasio resulta relevante debido a que el sistema puede requerir ajustes mediante el tiempo de desarrollo y posterior a eso, permitiendo, implementar o actualizar módulos del sistema, de una manera escalable y segura.

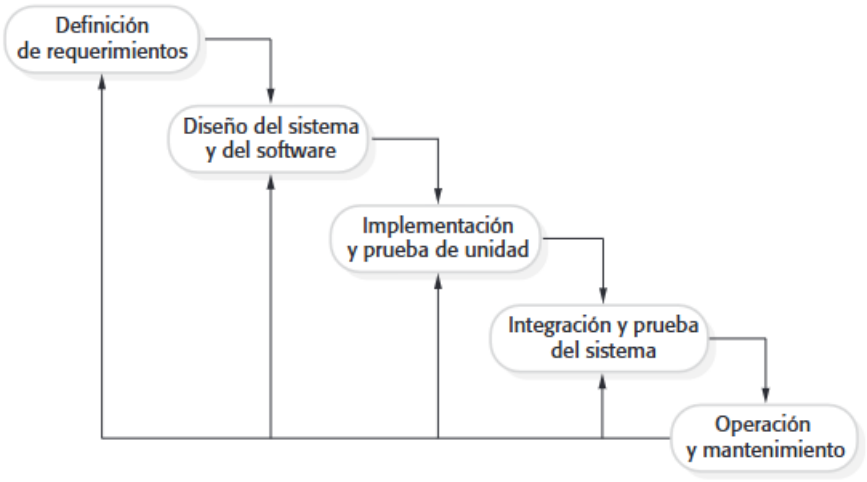
1.4.2. Ciclo de Vida del Software

El Ciclo de Vida del Desarrollo de Software (SDLC, por sus siglas en inglés) “se refiere a la metodología utilizada para planificar, crear, probar, implementar y mantener un software durante todo su ciclo de vida hasta su desmantelamiento.” (Swanlund et al., 2024). Así, este enfoque permite comprender que la construcción de las aplicaciones no se limita simplemente a la programación, sino, que viene de un método organizado y compuesto por diferentes fases para garantizar la calidad del sistema.

Independientemente de la metodología específica que se adopte, el SDLC se divide en etapas universales que rigen la evolución del producto. Según, Sommerville (2011). estas actividades fundamentales del desarrollo de software son:

1. **Análisis y definición de requerimientos.** Constituye una fase inicial, donde se define de manera detallada los recursos, restricciones y metas como una especificación del sistema. Dando como resultado una especificación formal de lo que se espera del sistema.
2. **Diseño del sistema y del software**. Se asignan los requerimientos, y se establece una arquitectura del sistema global para el correcto funcionamiento del sistema. Identificando y describiendo las relaciones de las funciones, las bases de datos y los módulos del mismo.
3. **Implementación y prueba de unidad.** Se codifica el diseño del sistema. En esta etapa se pone a prueba que cada módulo y función de manera individual del sistema validando que cumpla con su correcta función.
4. **Integración y prueba de sistema.** En esta etapa se fusionan los módulos individuales y se pone a prueba el correcto funcionamiento del sistema completo para asegurar que cumpla con los requisitos de software. Una vez realizado este proceso, se libera el sistema para el cliente.
5. **Operación y mantenimiento** Esta etapa es donde el sistema se instala y se pone en práctica, detectando errores para corregirlos y así mejorar la implementación de las unidades, esto se va realizando conforme se descubren nuevos requerimientos.

Figura 1
"Modelo en cascada" o Ciclo de Vida del Software



Nota. De Ingeniería de software (9a ed., p. 30), por I. Sommerville, 2011, Pearson Educación. Derechos de autor 2011 por Pearson Educación.

El ciclo de vida, representa un fundamento indispensable al momento del desarrollo del aplicativo, dado que, aplicando el mismo podemos asegurar una aplicativo controlado y seguro, que cumple los requisitos que necesita el gimnasio de la universidad Católica del Ecuador Sede Ibarra.

1.4.3. Metodologías de Desarrollo de Software

Las metodologías de desarrollo de software son métodos de trabajo para poder estructurar un sistema integrando varias técnicas que garantizan la seguridad y calidad del producto final. Estas metodologías han adquirido un papel importante, dado que las empresas se enfrentan a entornos volátiles, con incertidumbre y con una necesidad de responder con agilidad las demandas que existe en el mercado. Existen enfoques estructurados y rígidos como también modelos secuenciales iterativos y adaptativos, con esta evolución se puede ver la búsqueda de sistemas con alta eficiencia, calidad y con una solución a las necesidades del usuario final. (Ortega Ovalle, 2026)

Durante las últimas dos décadas, las metodologías ágiles han emergido como el enfoque dominante en la industria del software, desplazando progresivamente a los modelos tradicionales basados en planificación rígida. (Ortega Ovalle, 2026). El modelo tradicional presenta algunos problemas que ralentizan el proceso de desarrollo, en ocasiones, las especificaciones de requisitos cambian debido a que el cliente no tiene claro las necesidades y les resulta difícil determinar las mismas. Dada esta

razón y a medida que surgen nuevas tecnologías el mundo empresarial cambia constantemente su estructura. (Mahmoud Alrabaiah, 2022)

1.4.3.1. Metodologías Tradicionales

"Las metodologías tradicionales, como las primeras en su tipo, surgieron como guías para asegurar la creación de productos con un alto nivel de calidad. Estas se llevan a cabo de forma lineal, donde cada etapa depende de la anterior." (Guzmán Hidalgo et al., 2024). El articulo detalla que, si bien estas metodologías ofrecen ciertas ventajas relacionadas con el control del proyecto, también presentan fuertes limitaciones al momento en el que se requieren cambios frecuentes en los requerimientos.

Vamos a tomar como ejemplo el modelo en cascada, el cual se puede visualizar en la figura 1 "este modelo se conoce como "modelo en cascada" o ciclo de vida del software. El modelo en cascada es un ejemplo de un proceso dirigido por un plan; en principio, usted debe planear y programar todas las actividades del proceso, antes de comenzar a trabajar con ellas." (Sommerville, 2011).

Este modelo tiene como resultado diferentes documentos que deben ser firmados y autorizados para poder seguir en la siguiente fase, poniendo en práctica esto se vuelve un contratiempo, debido a que, en la práctica al momento de desarrollar, surgen retroalimentaciones de una fase a otra, no es lineal, de esa manera puede ser posible que los documentos deban modificarse para reflejar los cambios que se realizan.

Durante la fase final del método (operación y mantenimiento) el software se implementa al usuario, aquí es donde se descubren errores y omisiones de los requerimientos previos, y también la necesidad de una nueva funcionalidad, requiriendo así, al sistema evolucionar, lo que implica la repetición de ciertas etapas del proceso. (Sommerville, 2011).

1.4.3.2. Metodologías Agiles

Estas metodologías han ganado gran popularidad debido a sus beneficios en lo que es flexibilidad, adaptabilidad y eficiencias, y surgieron como respuesta a las limitaciones anteriores, este tipo de enfoques, se han ido añadiendo de manera amplia alrededor del mundo en el contexto de desarrollo de software, evidenciando así la capacidad de respuesta ante el cambio de los requisitos. (Osorio Marin, 2024).

Este tipo de metodologías encajan con el desarrollo del aplicativo móvil del gimnasio universitario, puesto que, el sistema requiere validar módulos de una manera progresiva, recibir retroalimentación de los estudiantes y ajustar la aplicación a sus requerimientos conforme sus

necesidades se expandan, por esto es que los modelos agiles se adaptan de mejor manera ante los modelos tradicionales.

Existen varias metodologías ágiles que sirven de manera eficaz al momento de desarrollar software; entre las más utilizadas se encuentran Scrum, Kanban y Extreme Programming (XP). Merchán-Narváez et al. (2024) las detallan de la siguiente manera:

- **Scrum.** Es un marco de trabajo ágil el cual se orienta a lograr que los equipos cooperen de manera efectiva en distintos tipos de proyecto. Se pueden apoyar en un conjunto de reglas, artefactos y roles definidos, estos elementos al brindan una estructura necesaria para que el producto final cumpla su objetivo. Tiene un enfoque incremental y se sustenta en la teoría de gestión de procesos, teoría la cual se basa en tres pilares: trasparencia, la inspección y la adaptación.

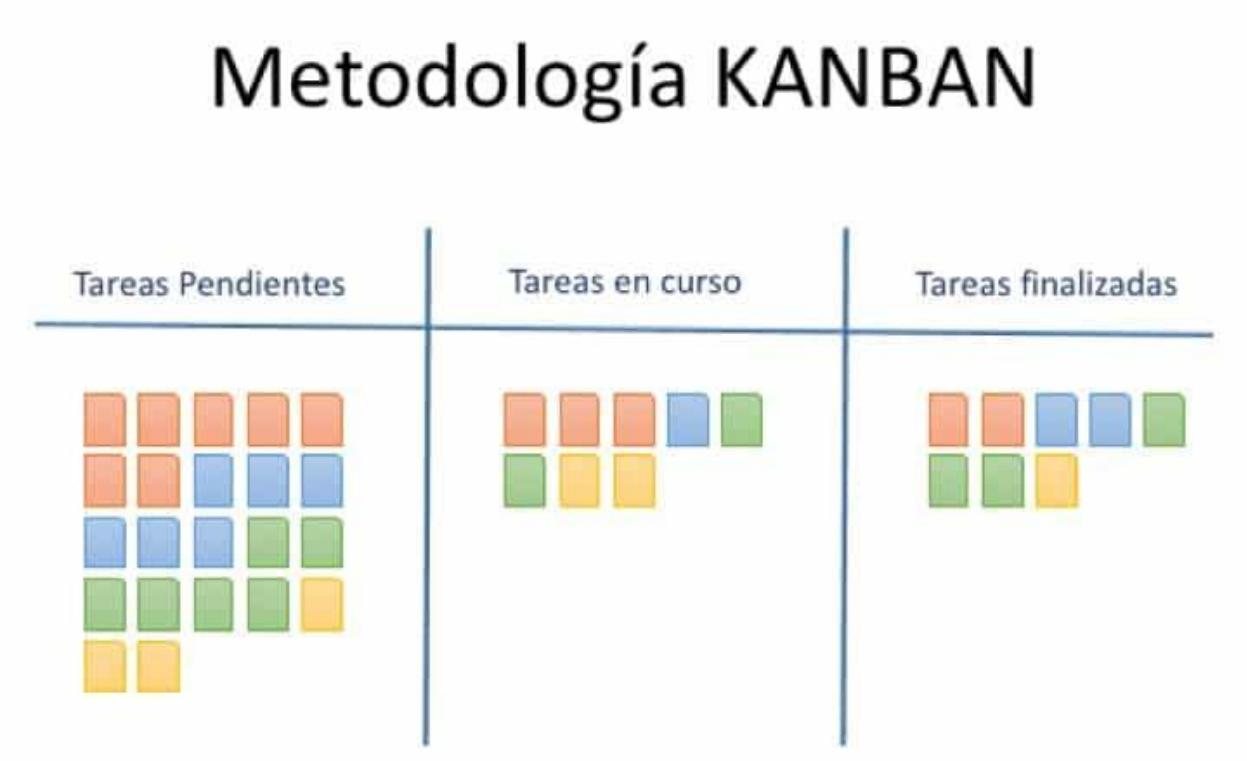
Figura 2
Scrum



Nota. De *Engineering Software Products: An Introduction to Modern Software Engineering* (p. 73), por I. Sommerville, 2011, Pearson Educación. Derechos de autor 2021 por Pearson Educación.

- **Kanban.** Es un método visual que se apoya en el uso de tableros. Estos tableros ayudan a que el equipo visualice gráficamente el flujo de trabajo y las tareas asignadas, y se estructuran en columnas que reflejan las diferentes fases del proceso. Cada tarea se representa a través de una tarjeta, la cual avanza de columna en columna conforme se acerca a su culminación.

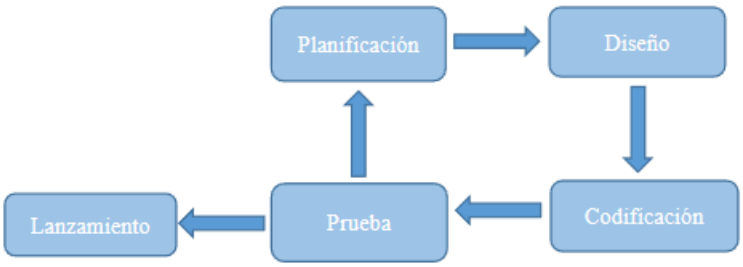
Figura 3
Metodología Kanban



Nota. De "Metodología Kanban (Metodología ágil)," por D. Calvo, 2018 (https://www.diegocalvo.es/metodologia-kanban-metodologia-agil/).

- **Extreme Programming (XP).** Esta metodología resalta la importancia de la interacción humana como una pieza clave para alcanzar el éxito en el desarrollo de software. Su enfoque se mantiene por cuatro pilares: la retroalimentación constante y el cliente, la búsqueda de soluciones simples y la apertura al cambio. Por esta razón resulta adecuada para proyectos en los que los requerimientos no están plenamente definidos.

Figura 4
Metodología Extreme Programming



Nota. De "Comparación de metodologías ágiles para el desarrollo de software," por N. J. Merchán-Narváez, E. E. Palma-Peralta y D. X. Poma-Japón, 2024, *MQRInvestigar, 8*(1), p. 5059 (https://doi.org/10.56048/MQR20225.8.1.2024.5052-5074). CC BY 4.0.

La metodología de desarrollo ágil XP, se adapta al aplicativo del gimnasio, debido a que la interacción humana es la base del aplicativo, el usuario al interactuar con el aplicativo, generará nuevas necesidades, y así añadiendo nuevos requerimientos, que, si usamos esta metodología resultará fácil de implementar o actualizar lo que ya estaba desarrollado.

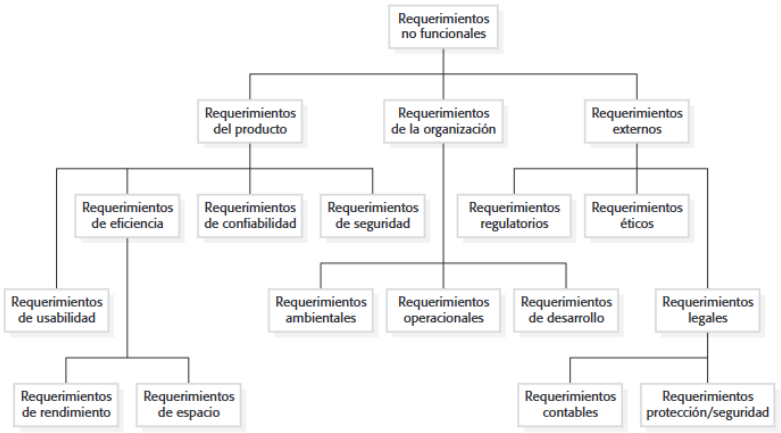
1.4.4. Análisis y Especificación de Requisitos Funcionales y No Funcionales

El análisis y la especificación de requisitos es la actividad en la cual se identifican las necesidades que el sistema debe satisfacer, desarrollando una base sobre la cual se diseña y se desarrolla la solución. (ISO/IEC/IEEE, 2018).

La mala definición de requerimientos puede llegar a tener un precio alto, debido a que estos mismos pueden causar errores, como requisitos incompletos, duplicados, incorrectos o incluso requisitos contradictorios, lo cual puede generar un proceso de alto costo y de mala calidad y al final un retraso en la entrega de los sistemas con clientes no satisfechos. (Universidad Nacional Abierta y Distancia, 2020).

La misma nos explica que estos requisitos se pueden clasificar en dos tipos los que son los requisitos funcionales y no funcionales; los requisitos funcionales son aquellos requisitos o funciones que el sistema debe ejecutar, y deben expresar sus capacidades y cualidades con el fin de satisfacer las necesidades del usuario; y los requisitos no funcionales los cuales describen restricciones que son externas al sistema, también se los puede conocer como restricciones. (Universidad Nacional Abierta y Distancia, 2020)

Figura 5
Tipos de requerimientos no funcionales



Nota. De Ingeniería de software (9a ed., p. 30), por I. Sommerville, 2011, Pearson Educación. Derechos de autor 2011 por Pearson Educación.

1.5. Arquitectura de Software

1.5.1. Concepto y Principios de la Arquitectura de Software

Podemos definir una arquitectura de software como lo hace en su informe Cristiá (2007), una arquitectura de software es una estructura o varias donde se pueden ver las propiedades y los módulos de un sistema y como es la relación que tiene cada uno de ellos.

El diseño arquitectónico actúa como una conexión esencial entre los requerimientos y la concepción del software, determinando así los módulos y sus relaciones, como resultado se obtiene un modelo que representa la comunicación entre los elementos que forman el sistema. (Sommerville, 2011)

Elegir un tipo de arquitectura puede resultar es un momento crítico del diseño de software, debido a que la arquitectura sirve como un pilar fundamental para que el proyecto tenga bases firmes y se pueda construir de una manera óptima, de esta manera se puede conseguir un desarrollo con calidad, buen rendimiento, escalable y con mantenibilidad a futuro. (Torassa Colombero et al., 2024).

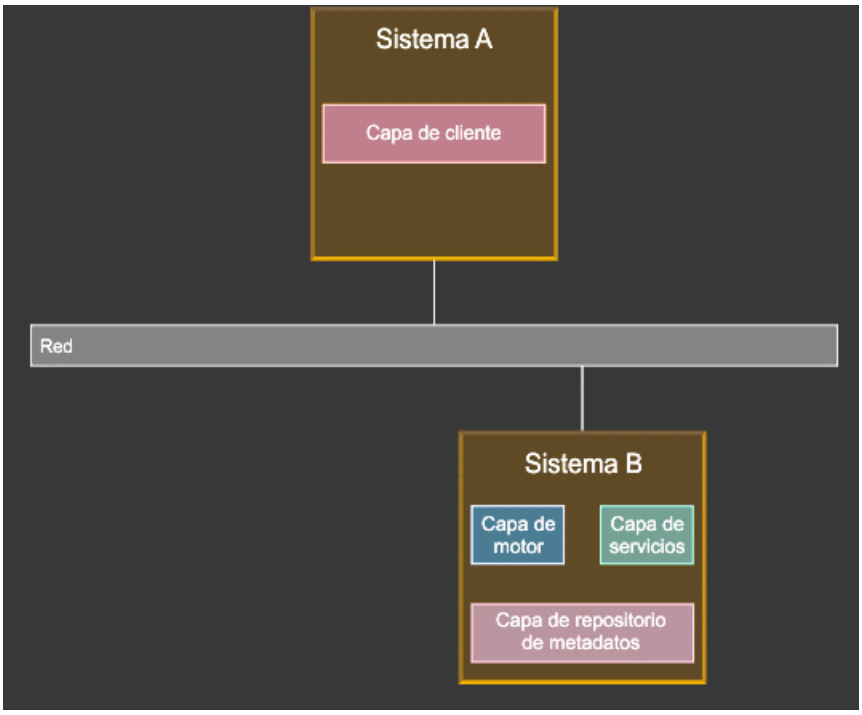
1.5.2. Arquitectura Cliente-Servidor y Arquitectura por Capas

La arquitectura cliente servidor separa la parte del usuario con los servidores, los usuarios por medio de la red hacen una petición al servidor, y este la devuelve, realizando este proceso con una

o más tareas al mismo tiempo, en esta arquitectura el cliente actúa como un consumidor de servicios, y el servidor es el que provee los servicios, centralizando así datos y lógica, favoreciendo el mantenimiento e integridad de la información. (Universidad de Valladolid, 2006). Lo cual es conveniente para el desarrollo de la aplicación de gimnasio debido a que el usuario es el que registra y obtiene información a través de la red, al momento de pedir una rutina y que la base de datos devuelva una rutina con los ejercicios que necesita.

Figura 6

Topología Cliente/Servidor



Nota. Tomado de "Topología cliente/servidor", IBM Docs., 2015, https://www.ibm.com/docs/es/iis/11.5.0?topic=topologies-clientserver-topology

Por otro lado, tenemos la arquitectura en capas la cual es una de las más utilizadas. "La arquitectura en capas consta en dividir la aplicación en capas, con la intención de que cada capa tenga un rol muy definido, como podría ser, una capa de presentación (UI), una capa de reglas de negocio (servicios) y una capa de acceso a datos (DAO), sin embargo, este estilo arquitectónico no define cuantas capas debe de tener la aplicación, sino más bien, se centra en la separación de la aplicación en capas (Aplica el principio Separación de preocupaciones (SoC))." (Reactive Programming, s.f.)

1.5.3. Arquitectura Orientada a Servicios

La Arquitectura orientada a Servicios (SOA) es una estructura de los sistemas en los que los módulos de dividen en diferentes servicios independientes. Para su implementación se han implementado lo que son contenedores Docker los cuales encapsulan las dependencias y evitan los conflictos de entono. Sin embargo, para superar los retos de escalabilidad es necesario la integración de plataformas de orquestación que gestiones eficientemente la carga del sistema. (Microsoft Learn, 2025).

1.5.4. Arquitectura de Microservicios

Debido a que para el desarrollo de la aplicación se utilizaron Apis, es necesario definir lo que son los microservicios, los microservicios, son servicios pequeños, independientes y escalables, cada microservicio responde a una funcionalidad concreta diferente. Una de las ventajas es que favorece la capacidad de mantenimiento en sistemas más complejos y escalables, también ofrecen una mayor capacidad de escalado, ya que cada servicio puede adaptarse según el requerimiento solicitado. (Microsoft Learn, 2025).

En el contexto del aplicativo para el gimnasio del campus, esta arquitectura resulta útil para la autenticación, el registro de rutinas, seguimiento del progreso incluso las notificaciones.

1.6. Desarrollo Móvil Multiplataforma

1.6.1. Concepto y Enfoques del Desarrollo Multiplataforma

El desarrollo de aplicaciones multiplataforma surge como una solución a la creación de software para múltiples plataformas. Este enfoque móvil permite usar solo una base de código que se puede utilizar tanto en Android como en IOS, en lugar de mantener dos desarrollos completamente separados, lo unifican a una sola fuente. Esto reduce el esfuerzo y acelera la entrega de los aplicativos móviles. De igual manera, esto reduce los costos de desarrollo y mejora la mantenibilidad del aplicativo. También, permite conservar el acceso a API nativas y sus capacidades en el momento que sea necesario. (JetBrains, 2026).

1.6.2. Flutter como Framework de Desarrollo

Flutter es un marco de trabajo (framework) para el desarrollo multiplataforma creado por Google, es de código abierto, se utiliza para el desarrollo frontend y full-stack ya que permite la creación para varias plataformas con un solo código base.

Aunque flutter fue lanzado en 2018 y su entorno principal eran las aplicaciones móviles, en la actualidad flutter es compatible con varias plataformas como: iOS, Android, web, Windows, MacOS y Linux. (Amazon Web Services, s.f.).

En comparación con el desarrollo de aplicaciones nativo —el cual resulta más costoso debido a que requiere más codificación y tiene el acceso pleno a toda la funcionalidad del dispositivo—, Flutter tiene características que logran hacer que el desarrollo multiplataforma sea fluido y con un gran rendimiento, ya que utiliza lenguaje Dart el cual se compila en código máquina, a su vez, tienen la herramienta de Hot Re load (recarga en caliente), que permite ver al desarrollador los cambios de código en tiempo real, ayudando a la reducción de los costos y tiempo de desarrollo. (Amazon Web Services, s.f.).

Esto sirve en el entorno del gimnasio debido a que los estudiantes no manejan un solo sistema operativo. Así, generando una solución que funcione tanto en Android como en iOS. Además, permite integrarse a servicios externos mediante API y seguir con la misma base de código.

A continuación, una tabla con la comparación de diferentes marcos de trabajo actuales para el desarrollo multiplataforma:

Tabla 1

Comparación de tecnologías para el desarrollo móvil multiplataforma

Tecnología	**Lenguaje**	**Uso compartido de código entre plataformas**	**Madurez de la comunidad**	**Ejemplos de aplicaciones**
Kotlin Multiplatform	Kotlin	Uso compartido flexible de lógica de negocio y UI entre plataformas, manteniendo código nativo de la plataforma donde sea necesario.	Creciendo rápidamente	Duolingo, McDonald's, Forbes, Philips, H&M, Bolt.
Flutter	Dart	La mayor parte de la lógica de la aplicación y la UI se comparten dentro de una única base de código Dart.	Grande y madura	eBay Motors, Alibaba, Google Pay, aplicaciones de ByteDance.
React Native	JavaScript, TypeScript	Lógica de negocio y componentes de UI compartidos entre plataformas, desde características individuales hasta aplicaciones completas.	Grande y madura	Microsoft Office, Teams, Xbox Game Pass, Facebook, Instagram.

Nota. Adaptado de *Cross-platform mobile development*, por JetBrains, s.f., Kotlin Documentation.

1.6.3. Dart como Lenguaje de Programación

Dart es un lenguaje de programación de código abierto creado por Google, el cual fue hecho para poder desarrollar aplicaciones rápidas en cualquier tipo de plataforma, con un lenguaje productivo. Dart, utiliza características modernas como la seguridad contra valores nulos (*null safety*) esto evita que una variable apunte a un espacio vacío, lo que causaría que el programa falle de forma inesperada. Asimismo, el lenguaje incluye una máquina virtual que compila en tiempo de ejecución (JIT) y, de igual manera permite compilar de manera directa con su sistema AOT de manera inmediata, esto elimina el retraso en la comunicación que hay en otros lenguajes de programación. Garantizando un rendimiento optimo. (Google, 2026).

1.7. Desarrollo Backend

1.7.1. Node.js como Entorno de Ejecución

Node.js es un entorno de ejecución de JavaScript, se construyó sobre el motor V8 de Google, está diseñado principalmente para construir aplicaciones de red escalables capaces de atender múltiples conexiones de manera concurrente sin bloquear el proceso. (Node.js, s.f.).

Node trabaja de manera en el que el modelo de entrada/salida actúa sin bloqueos con un bucle de eventos, permitiendo así utilizar de manera eficiente servicios backend que necesitan de varias respuestas al mismo tiempo de múltiples usuarios. (Node.js, s.f.).

Esto ayuda que la aplicación del gimnasio, tenga un buen rendimiento en su backend, permitiendo gestionar múltiples peticiones de los estudiantes en tiempo real.

1.7.2. NestJS como Framework para el Desarrollo de APIs

Para el desarrollo backend de la aplicación se utilizará NestJS, el cual es un framework que facilitara la creación del lado del servidor sobre Node.js. Según la documentación oficial, Nest proporciona una arquitectura de aplicación lista para usar que permite a los desarrolladores y equipos crear aplicaciones altamente comprobables, escalables, acopladas de forma flexible y de fácil mantenimiento. (NestJS, s.f.).

NestJS puede utilizar JavaScript moderno y TypeScript, lo que ofrece una estructura clara para organizar el proyecto. Para el proyecto de integración curricular, NestJS ayudará a que el backend del gimnasio quede ordenado y fácil de mantener, porque cada parte del sistema tendrá su propio módulo y su propia lógica.

1.8. Gestión y Persistencia de Datos

1.8.1. Sistemas Gestores de Bases de Datos (SGBD)

La gestión y persistencia de datos es esencial en un software que requiere almacenar la información de manera segura. Un sistema gestor de base de datos (SGBD) es el software que se especializa en almacenar datos de manera segura, organizar, recuperar y proteger los mismos de una aplicación, se podría decir que es el intermediario entre el backend y el almacenamiento físico de la información. Como dice, (Taipalus, 2023), en su revisión sobre comparaciones de rendimiento entre distintos SGBD, señala que para seleccionar un motor de base de datos se debe analizar qué tipo de dato y los requisitos del sistema más que en el desempeño del motor.

Esta observación es relevante debido a que el aplicativo del gimnasio universitario gestiona perfiles de estudiantes, rutinas, ejercicios y registros de progreso físico.

1.8.2. Bases de Datos SQL, NoSQL Análisis Comparativo

En la actualidad podemos dividir las bases de datos en tres grupos principales, bases de datos SQL, NoSQL y vectoriales, cada una de estas está orientada a diferentes tipos de datos.

Las bases de datos relacionales o SQL, son ideales para información ya estructurada y datos complejos, pero tienen un grande problema con la escalabilidad y sus esquemas rígidos. Mientras que las bases de datos no relacionales o NoSQL, son altamente escalables, y su manejan datos no estructurados o dinámicos de manera eficiente, su desventaja es que sacrifican la consistencia a cambio de la disponibilidad y son menos adecuadas para casos de uso tradicionales. (Moraru, 2026).

Por otro lado, tenemos las bases de datos vectoriales, las cuales han emergido como repuesta a la demanda de gestión de datos de alta dimensión, especialmente en aplicaciones que usan inteligencia artificial

Tabla 2

Comparación de bases de datos relacionales y no relacionales

Aspecto	SQL	NoSQL
Ventajas	El cumplimiento ACID garantiza una fuerte integridad de los datos	Altamente escalable para sistemas distribuidos de gran escala
	Robusto para datos estructurados y consultas complejas	Esquema flexible para datos dinámicos o no estructurados
	Ampliamente soportado y con estándares bien documentados	Alto rendimiento para necesidades en tiempo real y big data
Desventajas	Escalabilidad limitada a arquitecturas de servidor único	Consistencia más débil a favor de la disponibilidad
	El esquema rígido dificulta la adaptación a cambios	Menos adecuado para casos de uso transaccionales o relacionales
	Puede tener dificultades para manejar grandes volúmenes de datos no estructurados	Carece de la madurez y estandarización de los sistemas SQL

Nota. Tabla adaptada de *SQL vs NoSQL: A comparative study of performance in modern applications* de M. Moraru, 2026, Technical University of Moldova, https://repository.utm.md/handle/5014/34335.

1.8.3. PostgreSQL como Motor de Base de Datos Relacional

Postgres es un sistema gestor de base de datos de código abierto más avanzada del mundo. Su diseño ofrece rendimiento a nivel empresarial con funciones solidad y fiables.

Google Cloud la describe como la base de datos de código abierto más avanzada del mundo, entre sus características se puede ver que es un gestor que garantiza integridad y consistencia de datos, así como la compatibilidad con el estándar SQL. Entre sus ventajas están: la fiabilidad y seguras, ya que mantiene la durabilidad de los datos, además ofrece un modelo de autenticación seguro; alto rendimiento, al mantener sus datos de manera ordenada facilita las operaciones de CRUD de una manera más eficiente y su facilidad de monitoreo, ya que hay varias herramientas disponibles que permiten monitoreas la actividad de la base de datos. (Google Cloud, s.f.).

1.8.4. pgAdmin como Herramienta de Administración

pgAdmin es la principal herramienta grafica de administración para PostgresSql, está diseñado para usuarios principiantes como para avanzados. Ya que provee una interfaz gráfica que permite diseñar y modificar el esquema de la base de datos, ejecutar y depurar consultas SQL, a su vez

gestionar usuarios y permisos sin la necesidad de operar mediante una línea de comandos. (pgAdmin Development Team, s.f.)

Esto facilita la etapa de codificación contemplada para la base de datos del aplicativo para el gimnasio universitario, el ajuste de tablas, las rutinas y registro de procesos serán más fáciles de agregar.

1.9. Integración de Sistemas y Documentación de Servicios

1.9.1. APIs REST y el Intercambio de Datos mediante JSON

Las APIs Rest se han consolidado, a lo largo del tiempo, como uno de los enfoques más utilizados. Fueron diseñadas con el fin de abordar varias propiedades, entre ellas el rendimiento, la escalabilidad, portabilidad y la confiabilidad. Cuando un servicio cumple rigurosamente con las restricciones REST puede denominarse "RESTful". Para alcanzar esta denominación, la arquitectura de rest, se define en seis principios: cliente-servidor, interfaz uniforme, Interacciones sin estado, almacenable en cache, sistema en capas y código bajo demanda, el último es opcional (Czech Technical University in Prague, 2023).

JSON (JavaScript Object Notation) es un formato de intercambio de datos que para el humano resulta simple de leerlo y escribirlo y es aún más fácil para la maquina interpretarlo y generarlo. (JSON.org, s.f.). El uso combinado de REST y JSON permite extraer, enviar y consumir la información entre sistemas de manera eficiente, y reduciendo la complejidad de acoplar las bases de datos de cada aplicación (Mohd Kamir et al., 2022).

Para el presente proyecto, la comunicación entre la aplicación móvil desarrollada en Flutter y el backend desarrollado en NestJS (véase el apartado 1.8) se realiza mediante servicios REST que intercambian información en formato JSON: la visualización de rutinas, el registro de cargas de entrenamiento y la consulta del historial de progreso del estudiante enviaran por rutas HTTP, cuyo cuerpo de solicitud y respuesta se serializa y des serializa en JSON.

1.9.2. Documentación de APIs con Swagger/Open API

La documentación de las APIs es esencial para poder lograr que otros desarrolladores logren comprender y puedan consumir los servicios de manera correcta. Hay un formato estándar definido por OpenAPI Specification, donde dice que independientemente del lenguaje de programación, ya hay un formato estándar para describir interfaces HTTP. Esto guardado en un archivo con formato JSON o YAML permite describir la API de forma legible. (OpenAPI Initiative, 2024)

Aquí aparece Swagger, el cual es un conjunto de herramientas de código abierto que ayudan a diseñar APIs REST, entre estas herramientas se encuentran Swagger Editor, Swagger UI y Swagger Codegen, reduciendo así el trabajo manual. (SmartBear Software, s.f.).

1.10. Seguridad, Identidad y Control de Acceso

La seguridad es un componente sumamente especial en el desarrollo de aplicaciones móviles, nos permite proteger información y denegar permisos no autorizados. En un aplicativo como el del gimnasio universitario, resulta demasiado importante debido a que se manejan datos personales y operaciones que deben mantenerse integras y disponibles.

1.10.1. Fundamentos de Seguridad Informática y el Marco OWASP

La seguridad informática, se basa en un conjunto de principios que son orientados para proteger los sistemas. Y esto es importante debido a que los atacantes están de manera frecuente utilizando rutas para poder acceder a una aplicación y dañar el negocio u organización. Para resolver a estos casos, existe uno de los marcos más conocidos que es el OWASP Top 10, es una comunidad abierta, dedicada a identificar los riesgos más críticos en las aplicaciones. (OWASP Foundation, 2025).

En este contexto OWASP sirve como guía para fortalecer la seguridad de backend y de la aplicación del gimnasio universitario, permitiendo ejercer buenas prácticas para proteger las credenciales de los usuarios y así poder reducir el riesgo de filtración de información sensible.

1.10.2. Autenticación y Autorización

Según (Kosinski, s.f.), la autenticación y la autorización son procesos distintos pero que se encuentran relacionados en la gestión de identidades, la autenticación es la que verifica la identidad del usuario, la autenticación responde a un ¿Quién eres?, mientras que la autorización es la que proporciona los recursos que tiene permitido dicho usuario.

Pasando esto al tema del aplicativo móvil la autenticación verificará si el usuario que ingresa es un estudiante de la universidad, de caso contrario no tendrá acceso al aplicativo.

1.10.3. JSON Web Token (JWT)

JSON WEB TOKEN (JWT) es un estándar abierto que permite la verificación y autorización entre dos partes mediante JSON el cual es firmado de manera digital, permitiendo acceder a ambas partes de manera segura. Un JWT se compone por tres partes, se encuentra el header, payload y

signature, y esto es principalmente utilizado para el inicio de sesión de un usuario. Una de las ventajas del JWT es que son livianos y fáciles de aplicar en aplicaciones móviles. (IBM, 2026).

CAPÍTULO II. MATERIALES Y MÉTODOS

2. Materiales y Métodos

2.1. Enfoque de la investigación

El proyecto se basa en del enfoque mixto, tanto cualitativo como cuantitativo, se busca intervenir las necesidades del usuario mediante el desarrollo de un aplicativo funcional.

Desde la perspectiva cualitativa, se busca poder ver cuáles son las necesidades reales del estudiante mediante sus entrenamientos, medir las deficiencias del método actual y lograr capturar los requerimientos necesarios para una mejor experiencia de usuario.

Desde la perspectiva cuantitativa, se puede medir con datos exactos del rendimiento, los pesos, las series y la frecuencia de entrenamiento. De este modo, también podemos medir la eficiencia operativa que se gana frente al método tradicional

2.2. Tipo de Investigación

Se puede definir esta investigación como tecnológica o aplicada. Dado que, este estudio no persigue formulación de teorías, si no, la resolución de un problema real. De igual manera esta investigación poseerá un carácter descriptivo, porque se buscan las deficiencias de la manera en que se opera actualmente, describiendo e identificando, factores de riesgo y necesidades de guía biomecánica. Asimismo, dentro del ámbito de desarrollo de software se podrá identificar recursos funcionales y los recursos no funcionales que debe poseer el aplicativo para que le experiencia del usuario sea optima.

2.3. Diseño de la investigación

El proyecto tiene un enfoque claro, con un diseñó no experimental, ya que, se observa el fenómeno actual en su contexto natural, que es marcado por la gestión manual y la falta de guía y seguimiento. Este estudio detalla las condiciones físicas en la que se encuentran los estudiantes y la metodología que se utiliza al momento en el gimnasio de la PUCEI. Esto le da al usuario una respuesta que resuelve el problema de forma directa.

2.4. Población, Muestra, y Unidades de Estudio

Para el estudio se necesitaba definir quienes son los participantes que ayudaran con la información clave para la institución y el desarrollo del aplicativo.

2.4.1. Población

La población es el conjunto de usuarios activos que asisten al gimnasio de la universidad durante el periodo académico actual. Siento los principales los estudiantes y por otro lado el instructor encargado.

2.4.2. Muestra

La muestra ayuda a la captura de datos. Se necesita hablar directo con los estudiantes y que tengan un interés real en la tecnología para mejorar sus entrenamientos. Para esto, participara un grupo de 20 personas que asisten en distintos horarios para poder cubrir las diferentes dinámicas del gimnasio

2.4.3. Unidades de estudio

Este proyecto tiene dos unidades definidas, el primero es el estudiante, es el que nos da los detalles sobre la experiencia de usuario y nos cuenta su método de entrenamiento, y comparte si es necesario o no una guía multimedia como apoyo. Por otro lado, se encuentra la entrenadora, quien es la que define de mejor manera las rutinas de entrenamiento

2.5. Desarrollo metodológico

Para el desarrollo y construcción de este aplicativo destinado a la gestión deportiva en el gimnasio universitario se adoptó el enfoque en la metodología ágil de desarrollo Extreme Programming (XP), dado que, los entornos deportivos requieren alta flexibilidad, este enfoque permite desarrollar de una manera más iterativa, debido a que se desarrolla con una retroalimentación continua, permitiendo adaptarse rápidamente a los nuevos escenarios que pueden estar presentes y dar una solución exacta a las necesidades de los estudiantes.

2.5.1. Actores del sistema

Para este proyecto en base a su funcionalidad se definieron los siguientes actores, está el usuario, este es el encargado de utilizar la aplicación para poder acceder a los módulos de rutinas, registro y guías multimedia, el instructor, que es encargado de asignar o modificar las rutinas, finalmente el administrador, este es el encargado de toda la gestión de la plataforma, se encarga de administrar las cuentas de los usuarios e instructores, actualizar el catálogo general de ejercicios y visualizar las métricas generales de uso del gimnasio.

2.5.2. Determinación de roles y responsabilidades

Como se puede observar en la tabla 1 los siguientes se adaptan al desarrollo del aplicativo de la siguiente manera:

Tabla 3

Roles y Responsabilidades

Nombre	Rol	Categoría Profesional	Responsabilidad
Conforme Bermúdez Josafath Daniel	Desarrollador	Estudiante de Tecnología Superior en desarrollo de software	Diseñar, desarrollar y evaluar el sistema
Ibarra Estévez José Luis	Tutor	Ingeniero	Responsable de dar seguimiento al desarrollo del sistema

Nota. Conforme Josafath (2026).

2.5.3. Especificación de requisitos de software

2.5.3.1.Requisitos funcionales

Para la especificación detallada de las funcionalidades del aplicativo móvil, se diseñó una matriz de requisitos funcionales. Cada elemento se encuentra codificado bajo la nomenclatura **RF-[Número]**, definiendo su comportamiento, el actor responsable de su ejecución y el nivel de prioridad para su desarrollo dentro del ciclo de vida del proyecto.

Tabla 4

Autenticación de Usuarios

Código	RF-01
Nombre del requisito	Autenticación de usuarios
Descripción detallada	El sistema debe permitir a los estudiantes iniciar sesión en la aplicación móvil mediante sus credenciales institucionales (correo sin @pucesi.edu.ec y contraseña), verificando su identidad a través del Directorio Activo de la universidad.
Actor principal	Estudiante
Prioridad	Alta

Nota. Conforme Josafath (2026).

Tabla 5

Control de Acceso

Código	**RF-02**
Nombre del requisito	Control de acceso a la aplicación
Descripción detallada	El sistema debe autenticar al usuario mediante sus credenciales institucionales y validar que pertenezca al rol de estudiante para permitir el acceso a la aplicación móvil.
Actor principal	Estudiante
Prioridad	Alta

Nota. Conforme Josafath (2026).

Tabla 6

Configuración de Perfil Físico

Código	**RF-03**
Nombre del requisito	Configuración de perfil físico
Descripción detallada	El sistema debe solicitar al estudiante en su primer registro sus datos antropométricos iniciales (estatura, peso y fecha de nacimiento). En los registros posteriores, el sistema únicamente solicitara actualización del peso corporal, permitiendo tener el seguimiento de esta variable
Actor principal	Estudiante
Prioridad	Alta

Nota. Conforme Josafath (2026).

Tabla 7

Visualización de Rutinas Diarias

Código	**RF-04**
Nombre del requisito	Visualización de Rutinas Diarias
Descripción detallada	El aplicativo debe desplegar las rutinas de entrenamiento preestablecidas (Full Body, PPL y Upper/Lower), mostrando para cada una el listado de ejercicios asignados. Al seleccionar un ejercicio, se debe abrir una pantalla de detalle con la información completa del ejercicio y la opción de iniciar su ejecución.
Actor principal	Estudiante
Prioridad	Alta

Nota. Conforme Josafath (2026).

Tabla 8

Cronometro de Entrenamiento

Código	**RF-05**
Nombre del requisito	Cronometro de entrenamiento
Descripción detallada	El sistema debe proporcionar un cronómetro interactivo tipo Tabata que se active al iniciar un ejercicio, permitiendo al estudiante configurar el tiempo de ejercicio (segundos), tiempo de descanso (segundos) y número de series.
Actor principal	Estudiante
Prioridad	Alta

Nota. Conforme Josafath (2026).

Tabla 9

Acceso a Educación Biomecánica

Código	**RF-06**
Nombre del requisito	Acceso a Educación Biomecánica
Descripción detallada	El sistema debe desplegar un listado completo de todos los ejercicios disponibles en el catálogo. Al seleccionar un ejercicio, se debe abrir la pantalla de detalle con su descripción técnica y acceso al material audiovisual que demuestre la ejecución adecuada.
Actor principal	Estudiante
Prioridad	Alta

Nota. Conforme Josafath (2026).

Tabla 10

Historial Rendimiento Físico

Código	**RF-07**
Nombre del requisito	Historial de Rendimiento Grafico
Descripción detallada	El sistema debe generar gráficos estadísticos que muestren la evolución del peso corporal
Actor principal	Estudiante
Prioridad	Media

Nota. Conforme Josafath (2026).

2.5.3.2.Requisitos no funcionales

Tabla 11

Compatibilidad y Portabilidad

Código	**RNF-01**
Categoría	Compatibilidad y Portabilidad
Descripción detallada	El aplicativo debe ser multiplataforma, garantizando su correcto despliegue y funcionamiento nativo tanto en sistemas operativos Android como iOS, utilizando un único código base desarrollado con el framework Flutter.

Nota. Conforme Josafath (2026).

Tabla 12

Rendimiento y Eficiencia

Código	**RNF-02**
Categoría	Rendimiento y Eficiencia
Descripción detallada	El tiempo de respuesta en la carga de interfaces, visualización de rutinas y actualización de progreso no debe exceder los 2 segundos bajo condiciones normales de red. Este rendimiento se garantiza mediante la optimización de las consultas a la base de datos PostgreSQL y el uso de una API REST eficiente construida con NestJS.

Nota. Conforme Josafath (2026).

Tabla 13

Usabilidad y Experiencia de Usuario (UX)

Código	**RNF-03**
Categoría	Usabilidad y Experiencia de Usuario
Descripción detallada	La interfaz debe ser altamente intuitiva, aplicando patrones de Material Design y Cupertino. El usuario debe poder acceder a su rutina diaria y al material de prevención de lesiones (videos/guías) en un máximo de 3 interacciones (toques) desde la pantalla principal.

Nota. Conforme Josafath (2026).

Tabla 14

Disponibilidad y Persistencia de Datos

Código	**RNF-04**
Categoría	Disponibilidad y Persistencia de Datos
Descripción detallada	La aplicación debe almacenar en el dispositivo la información de inicio de sesión, garantizando su persistencia incluso después de cerrar o reiniciar la aplicación.

Nota. Conforme Josafath (2026).

Tabla 15
Escalabilidad y Mantenibilidad

Código	**RNF-05**
Categoría	Escalabilidad y Mantenibilidad
Descripción detallada	El código fuente del frontend debe centralizar la comunicación con el backend mediante una capa de servicios (*Service Layer*), separando la lógica de acceso a datos de la interfaz de usuario. Esto facilitará el mantenimiento y la incorporación de nuevas funcionalidades a medida que crezca el número de estudiantes inscritos por semestre.

Nota. Conforme Josafath (2026).

Tabla 16
Disponibilidad de Recursos Multimedia

Código	**RNF-06**
Categoría	Disponibilidad de recursos Multimedia
Descripción detallada	Para garantizar la optimización del consumo de red móvil, el sistema implementa carga diferida (lazy loading), asegurando que los recursos visuales solo consuman cuando el usuario interactúa explícitamente con ellos

Nota. Conforme Josafath (2026).

2.5.4. Fase de planificación

Durante esta fase se define la relación que tienen los actores y las acciones dentro de la aplicación, se describe quien realiza la acción, que beneficio se obtiene y permite priorizar el desarrollo de manera iterativa.

Tabla 17
Historias de Usuario

Código	Actor	Acción	Beneficio	Prioridad
HU-01	Estudiante	Iniciar sesión mediante las credenciales universitarias	Acceder a la plataforma de forma segura y proteger la información personal.	Alta
HU-02	Estudiante	Configurar mi perfil físico por primera vez, ingresando estatura, peso y fecha de nacimiento.	Establecer mis datos base para un seguimiento del peso corporal.	Alta
HU-03	Estudiante	Actualizar mi peso corporal en los registros posteriores al primero.	Mantener un seguimiento continuo y actualizado de mi evolución física.	Alta
HU-04	Estudiante	Visualizar las rutinas de entrenamiento (Full Body, PPL, Upper/Lower) con sus ejercicios asignados	Conocer el plan de entrenamiento disponible y seleccionar el que mejor se adapte a los objetivos.	Alta
HU-05	Estudiante	Seleccionar un ejercicio desde la rutina para ver su pantalla de detalle con información técnica y guía.	Comprender la ejecución correcta del ejercicio antes de realizarlo.	Alta
HU-06	Estudiante	Iniciar el cronómetro tipo Tabata configurando el tiempo de ejercicio, tiempo de descanso y número de series.	Gestión del tiempo de entrenamiento.	Alta
HU-07	Estudiante	Acceder al listado completo de ejercicios del catálogo para consultar sus guías biomecánicas.	Revisar la técnica correcta de cualquier ejercicio disponible en el gimnasio, previniendo lesiones.	Alta

Nota. Conforme Josafath (2026).

2.5.5. Fase de iteración

La fase de iteración se ejecuta en ciclos de desarrollo, cada iteración entrega funcionalidades completas y probadas, permitiendo la retroalimentación para ajustar el desarrollo según las necesidades identificadas.

Tabla 18
Plan de iteraciones

Iteración	Objetico	Entregable
Iteración 1	Implementar autenticación de usuarios y configuración de perfil físico.	Módulo de login funcional con almacenamiento local de token JWT y registro de datos antropométricos iniciales (estatura, peso y fecha de nacimiento).
Iteración 2	Desarrollar visualización de rutinas y cronómetro de entrenamiento.	Visualización de rutinas (Full Body, PPL, Upper/Lower), detalle de ejercicios con información técnica y cronómetro tipo Tabata configurable (tiempo de ejercicio, descanso y series).
Iteración 3	Implementar actualización de peso corporal, y guías biomecánicas.	Actualización de peso en registros posteriores; listado completo de ejercicios con acceso a guías biomecánicas mediante navegación.

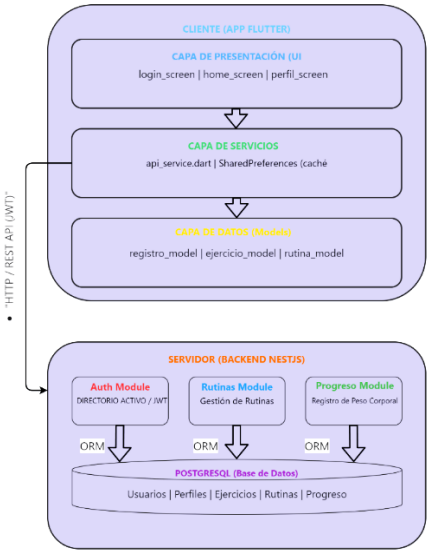
Nota. Conforme Josafath (2026).

2.6. Diseño del sistema

2.6.1. Arquitectura de Software

Esta arquitectura se diseñó bajo el patrón de cliente servidor, donde el frontend y el backend se comunican mediante el protocolo HTTP y a través de una API REST. Se divide en tres capas las cuales fueron diseñadas para maximizar la escalabilidad y facilitar las pruebas.

Figura 7
Arquitectura de software

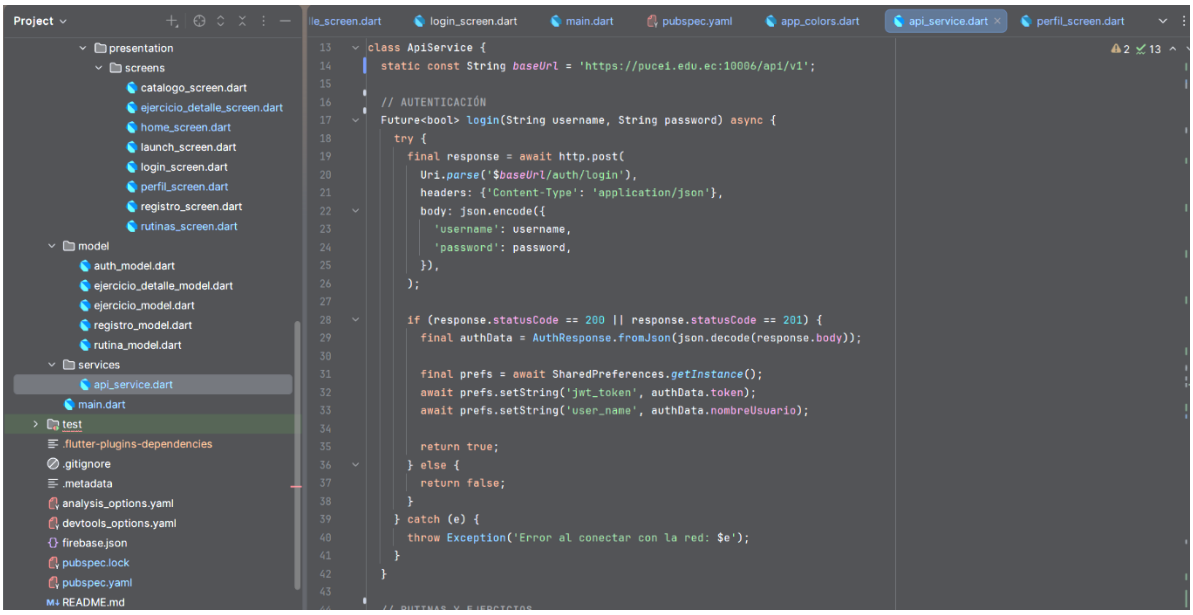


Nota. Josafath Conforme (2026)

2.6.2. Organización de código Frontend

Para la implementación del frontend se adoptó el patrón de capa de servicios (Service Layer), el cual fue descrito por Fowler (2002) como un mecanismo para poder centralizar la lógica del acceso a los datos y a la comunicación con los servidores remotos, desacoplando la lógica de la interfaz del usuario. Esto se ha aplicado a este proyecto en la clase ApiService, la cual es la encargada de gestionar las solicitudes HTTP hacia el backend, la serialización de los modelos de datos, y el almacenamiento local del token de sesión mediante SharedPreferences. Con esto las pantallas consumen directamente esta capa, sin necesidad de conocer los detalles de la comunicación HTTP subyacente.

Figura 8
ApiService

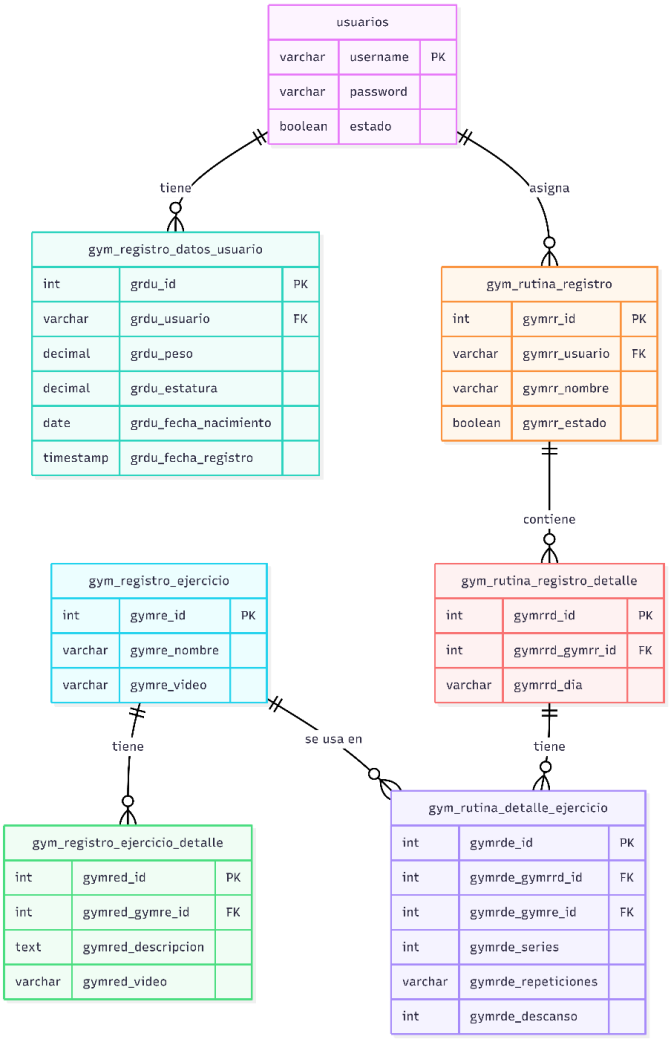


Nota. Conforme Josafath (2026).

2.6.3. Diseño de la Base de Datos

Esta sección describe la base de datos y la estructura que mantiene para la retención de información del sistema, garantizando integridad, escalabilidad y disponibilidad de los datos de la PUCE Sede Ibarra. El siguiente diagrama muestra las tablas organizadas por los diferentes módulos de la base de datos. Se indican claves primarias, claves foráneas y las cardinalidades entre entidades.

Figura 9
Diseño Base de Datos



Nota. Conforme Josafath (2026).

2.6.4. Diseño de la Interfaz de Usuario

Para el diseño del aplicativo el gimnasio se realizó siguiendo el manual de identidad de la Pontificia Universidad Católica del Ecuador, esta decisión garantiza una coherencia visual de la aplicación y fortalece el sentido de pertenencia de la comunidad universitaria.

El color primario (turquesa- #44C3DA) fue utilizado como un acento principal en la aplicación, según el Manual de Identidad de la PUCE (2017) este color detona armonía y se dirige a un público objetivo joven y actual, manteniendo así, la fidelidad a la institución.

El color secundario (Gris Oscuro-#393E49) se utilizó para generar contraste y juego complementario, se aplicó a los fondos de las tarjetas informativas, modales y contenedores

secundarios, y da como resultado una aplicación con una interfaz moderna, reduciendo la fatiga visual.

Esto fue realizado para que la aplicación se integre de manera natural con la universidad.

Figura 10
Splash



Nota. Conforme Josafath (2026).

Figura 11
Login



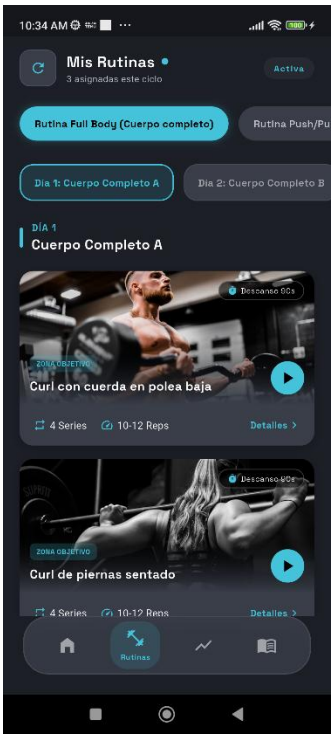
Nota. Conforme Josafath (2026).

Figura 12
Home



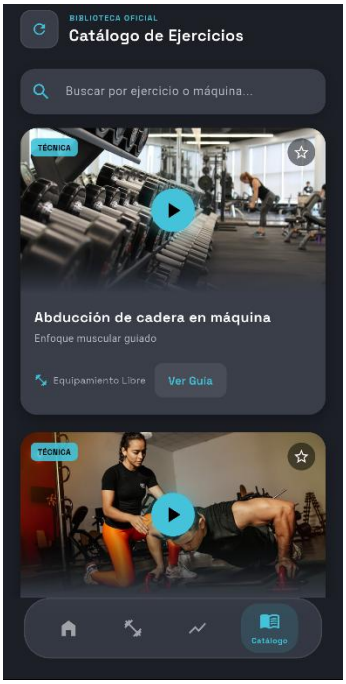
Nota. Conforme Josafath (2026).

Figura 13
Rutinas



Nota. Conforme Josafath (2026).

Figura 14
Ejercicios



Nota. Conforme Josafath (2026).

Figura 15
Guías



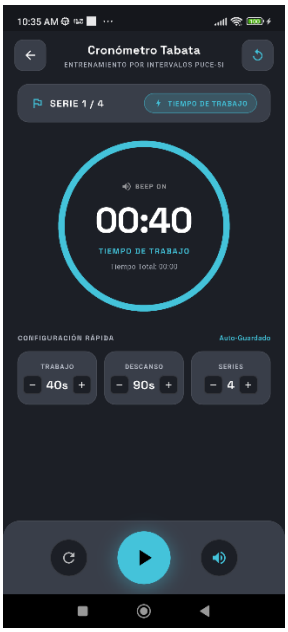
Nota. Conforme Josafath (2026).

Figura 16
Registro de Progreso



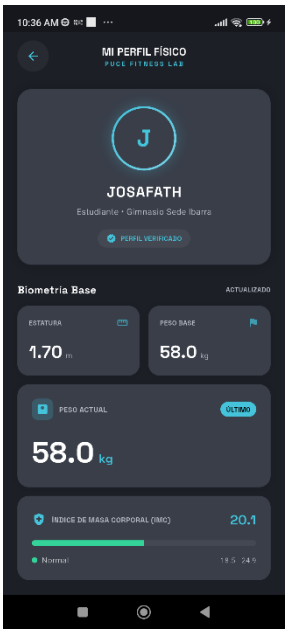
Nota. Conforme Josafath (2026).

Figura 17
Cronometro Tabata



Nota. Conforme Josafath (2026).

Figura 18
Perfil Físico



Nota. Conforme Josafath (2026).

CAPITULO III. RESULTADOS Y DISCUSIONES

3.1. Análisis e interpretación de los resultados

Este capítulo nos detalla de manera critica los resultados obtenidos con las historias de usuario. Para así poder comprender las dinámicas operativas del gimnasio universitario. Estos datos evidencian la eficiencia operativa del aplicativo como solución tecnológica construida con Flutter y su correcto funcionamiento al momento de iniciar sesión, ver sus rutinas y obtener acceso a las guías multimedia, asimismo. Del mismo modo, el funcionamiento de NestJS como backend, con la integración de PostgreSQL, como sistema de persistencia de datos y JWT para la autenticación y control de acceso

3.1.1. Resultados del Desarrollo del Aplicativo

Como resultado del aplicativo móvil, se obtuvo un sistema funcional capaz de atender los requerimientos identificados en el contexto del gimnasio universitario. Para esta aplicación se implementó una arquitectura cliente-servidor la cual se formó mediante Flutter como frontend y es la interacción del usuario y por otro lado NestJS estuvo del lado de Backend centralizando toda la lógica de negocio, la validación de reglas y el acceso a los datos almacenados. Dicha estructura, permitió separar las necesidades técnicas, mejorando el mantenimiento del sistema y facilitando la integración de nuevos módulos a futuro.

Durante las pruebas funcionales del sistema se comprobó que el usuario puede autenticarse de manera correcta, puede consultar rutinas y tener una guía de cómo se realiza el ejercicio. Desde la perspectiva técnica, el uso de JWT permitió mantener un modelo de autenticación stateless, en el que cada solicitud protegida es validada a partir del token enviado por el cliente en el encabezado de autorización.

3.1.2. Cumplimiento de las Historias de Usuario

Hu-01- Acceso a la aplicación mediante el uso de las credenciales universitarias. Esta historia detalla la manera en la que el estudiante ingresa al sistema utilizando las credenciales institucionales. Garantizando que solo personas autorizadas puedan utilizar el aplicativo dentro de la universidad. Validando si el usuario pertenece a un estudiante de la universidad. El proceso se respalda en un esquema de autenticación basado en JWT el cual permite validar la identidad del usuario, de esta forma se fortalece la seguridad del acceso.

HU-02- se cumple mediante la configuración inicial del perfil del estudiante, donde se registran los datos principales como la estatura, peso y la edad. Este registro sirve como base para el seguimiento corporal.

HU-03- Se cumple a partir de la actualización periódica del peso corporal en los registros posteriores al primero, apoyando con información útil para comparar sus cambios a lo largo del tiempo.

HU-04- Se cumple al mostrar las rutinas disponibles, como Full Body, Push Pull Legs y Upper Lower, junto con los ejercicios designados para cada una. Esto le permite al estudiante conocer el plan de entrenamiento con claridad.

HU-05- Al seleccionar el ejercicio se accede al detalle del mismo, donde el estudiante puede visualizar un video instructivo del ejercicio, detalles y una recomendación. Esto favorece a una correcta comprensión del ejercicio antes de realizarlo.

HU-06- se cumple al dar al usuario la opción de comenzar el ejercicio y poder medir sus tiempos con un cronometro de tipo Tabata, que puede configurarse con tiempo de ejercicio, tiempo de descanso y numero de series, apoyando la gestión de tiempo del usuario.

HU-07- se cumple al brindar al usuario un listado completo de los ejercicios que puede realizar, permitiendo de igual manera consultar las guías y recomendaciones técnicas. De esta manera el usuario puede revisar el ejercicio que desee de manera independiente.

3.1.3. Verificación de Rendimiento de la Api

Con el fin de verificar el cumplimiento del requisito no funcional RNF-02, se ha utilizado la herramienta Postman para ejecutar las solicitudes a los endpoints de manera directa, bajo condiciones normales de red. Cada endpoint fue ejecutado 10 veces consecutivas, registrando el tiempo de respuesta reportado por Postman en cada solicitud.

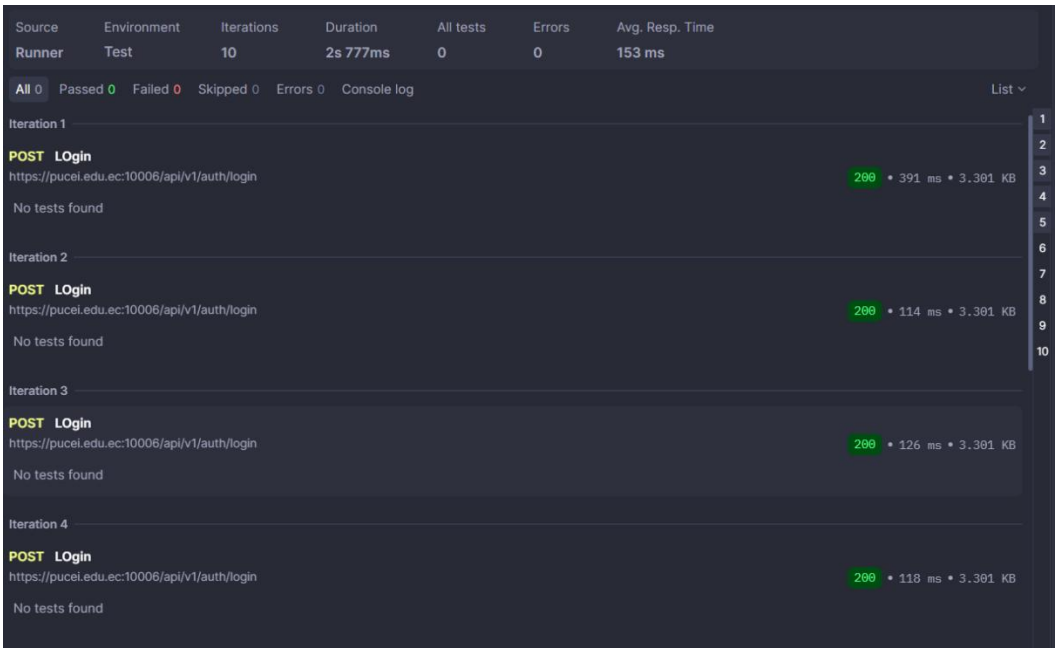
Tabla 19

Tiempos de respuesta medidos por endpoint

Endpoint	Método	Tiempo promedio (ms)	Tiempo máximo (ms)	Cumplimiento RNF-02
/auth/login	Post	153,4 ms	391 ms	Sí
/gym/usuario/rutinas	Get	39 ms	136 ms	Sí
/gym/usuario/registro	Get	27 ms	61 ms	Sí

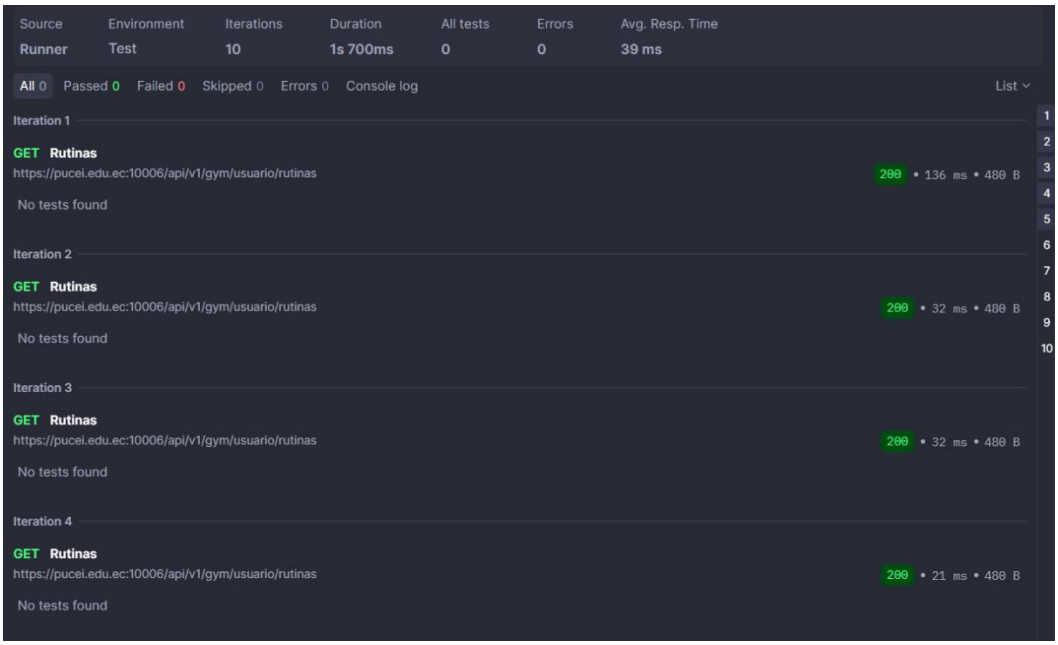
Nota. Conforme Josafath (2026).

Figura 19
Prueba de Rendimiento Login



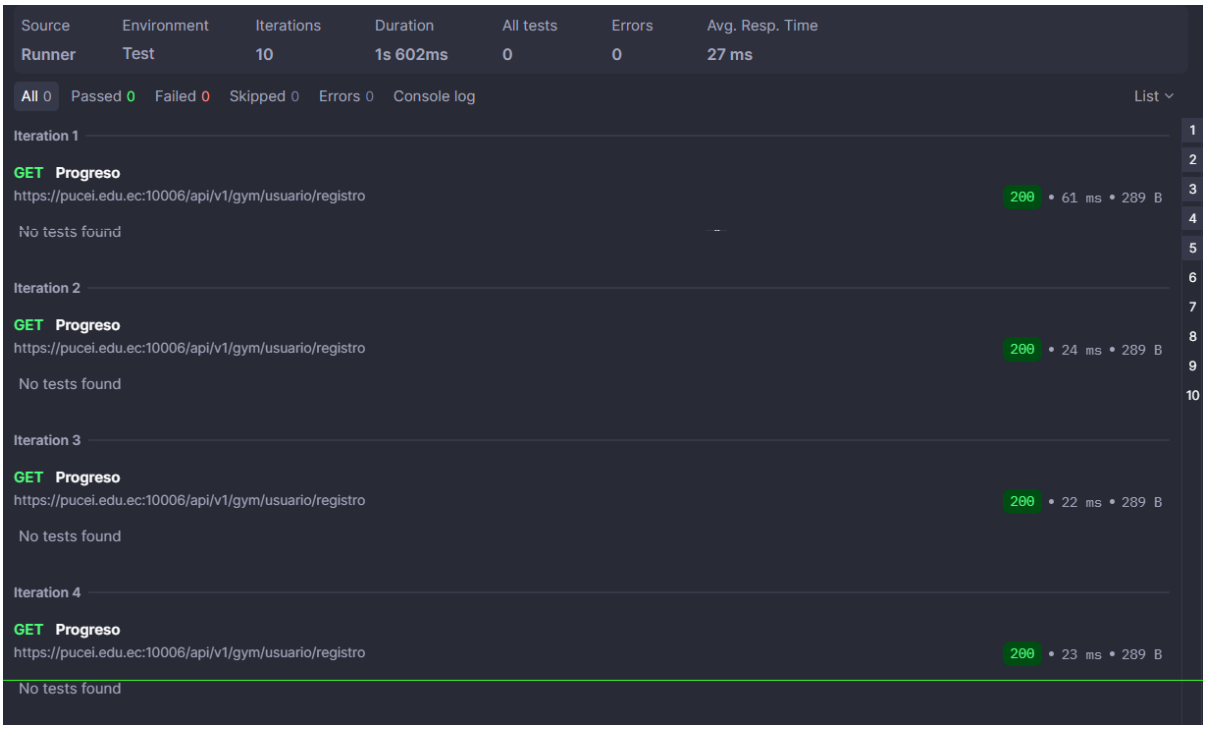
Nota. Conforme Josafath (2026).

Figura 20
Prueba de Rendimiento Rutinas



Nota. Conforme Josafath (2026).

Figura 21
Prueba de Rendimiento Registro de Progreso



Nota. Conforme Josafath (2026).

3.1.4. Métricas de Calidad del Software

Las métricas del software desarrollado se evaluaron en consideración de la funcionabilidad, confiabilidad, usabilidad y mantenibilidad, tomando como referencia el comportamiento de la aplicación mediante su uso. Para esto, se realizaron pruebas en cada módulo de la aplicación validando el comportamiento de las características según lo esperado, corroborando su correcto funcionamiento.

Los resultados que se obtuvieron muestran que la aplicación responde de manera correcta a los requerimientos obtenidos, aunque siempre pueden identificarse oportunidades de mejora, el aplicativo cumple con un nivel de calidad adecuado para su propósito académico, ofreciendo un apoyo funcional para los usuarios del gimnasio universitario.

CONCLUSIONES Y RECOMENDACIONES

Conclusiones

Se desarrollo y se implementó con éxito una aplicación móvil integral para la gestión de rutinas y seguimiento físico, además la integración del módulo de autenticación garantiza que el acceso a la plataforma este estrictamente reservado para la comunidad universitaria resguardando la integridad de la información institucional. De esta manera se logró responder a las necesidades identificadas en el contexto del gimnasio universitario.

Además, la arquitectura basada en Flutter, NestJS, PostgreSQL y JWT proporciono una interacción estable entre el aplicativo móvil y el backend, tanto así, como un acceso controlado de la información. Los resultados obtenidos fueron positivos y cumplen con las funcionalidades principales definidas en las historias de usuario.

Recomendaciones

Se recomienda continuar con el fortalecimiento de la aplicación, integrando nuevas funcionalidades orientadas al seguimiento del progreso del estudiante como la integración de reportes gráficos y métricas de rendimiento, personalización de las rutinas, de igual manera, se recomienda implementar un almacenamiento local (modo offline) para futuras actualizaciones, por último la optimización continua de los recursos multimedia, dado que, se debe mantener una auditoria constante de los videos explicativos.

Además, se recomienda completar la migración hacia una arquitectura por capas en la totalidad de los módulos del frontend.

BIBLIOGRAFÍA

Amazon Web Services. (s.f.). *¿Qué es Flutter?* Amazon Web Services, Inc.: [https://aws.amazon.com/es/what-is/flutter/

Calvo, D. (7 de Abril de 2018). *Metodología Kanban (Metodología ágil)*. Diego Calvo: https://www.diegocalvo.es/metodologia-kanban-metodologia-agil/

Cristiá, M. (2007). *Introducción a la Arquitectura de Software.* Universidad Nacional de Rosario. https://www.fceia.unr.edu.ar/~mcristia/Introduccion_a_la_Arquitectura_de_Software.pdf

Czech Technical University in Prague. (2023). *Lecture 06: REST.* Course B6B36EAR - Czech Technical University in Prague: https://cw.fel.cvut.cz/b231/_media/courses/b6b36ear/lectures/lecture-06-rest-s.pdf

Estrada-Marcén, N., Sánchez Bermúdez, J., Simón Grima, J., y Casterad Sera, J. (01 de 07 de 2020). *Uso de dispositivos fitness por parte de usuarios de gimnasios.* https://doi.org/10.47197/retos.v38i38.73108

Fowler, M. (2002). *Patterns of Enterprise Application Architecture.* Addison-Wesley Professional.

Google. (2026). *Dart overview.* Dart: https://dart.dev/overview](https://dart.dev/overview

Google Cloud. (s.f.). *What is PostgreSQL?* Google Cloud: https://cloud.google.com/discover/what-is-postgresql?hl=es

Guzmán Hidalgo, C., Jaramillo Alba, J., Pulla Vasquez, T., y Sarmiento Rios, S. (14 de 08 de 2024). Evolución de las metodologías de desarrollo de software en la gestión académica. *Ciencia Latina Revista Científica Multidisciplinar, 8*(4), pp. 2468-2475. https://doi.org/10.37811/cl_rcm.v8i4.12490

IBM. (2 de 12 de 2015). *Topología cliente/servidor*. IBM Docs: https://www.ibm.com/docs/es/iis/11.5.0?topic=topologies-clientserver-topology

IBM. (16 de 01 de 2026). *JSON Web Token (JWT).* IBM Documentation: https://www.ibm.com/docs/es/cics-ts/6.x?topic=cics-json-web-token-jwt

ISO/IEC/IEEE. (2018). *ISO/IEC/IEEE 29148:2018(E) — Systems and software engineering — Life cycle processes — Requirements engineering.* IEEE. https://doi.org/10.1109/IEEESTD.2018.8559686

JetBrains. (2026). *Cross-platform mobile development.* Kotlin Documentation: https://kotlinlang.org/docs/multiplatform/cross-platform-mobile-development.html

JSON.org. (s.f.). *Introducción a JSON.* JSON.org: https://www.json.org/json-es.html

Kosinski, M. (s.f.). *Autenticación o autorización: ¿cuál es la diferencia?* IBM: https://www.ibm.com/es-es/think/topics/authentication-vs-authorization

Mahmoud Alrabaiah, H. (2022). *Agile Beeswax: Agile Development Methodology for Mobile Applications.* Universidad de Granada. Universidad de Granada. http://hdl.handle.net/10481/75443

Merchán-Narváez, N., Palma-Peralta, E., y Poma-Japón, D. (1 de 8 de 2024). Comparación de metodologías ágiles para el desarrollo de software. *MQRInvestigar*, pp. 5052-5074. https://doi.org/10.56048/MQR20225.8.1.2024.5052-5074

Microsoft Learn. (17 de junio de 2025). *Arquitectura orientada a servicios - .NET*. Microsoft Learn: https://learn.microsoft.com/es-es/dotnet/architecture/microservices/architect-microservice-container-applications/service-oriented-architecture

Mohd Kamir, Y., Mustafa, M., y Azli, I. (2022). Design and Implement of REST API for Data Integration. *2022 International Conference on Engineering and Emerging Technologies (ICEET)*. https://doi.org/10.1109/ICEET56468.2022.10007414

Moraru, M. (2026). *SQL vs NOSQL: A comparative study of performance in modern applications.* Technical University of Moldova. https://repository.utm.md/handle/5014/34335

NestJS. (s.f.). *Introduction*. NestJS Documentation: https://docs.nestjs.com/

Node.js. (s.f.). *Sobre Node.js*. Node.js: https://nodejs.org/es/about

OpenAPI Initiative. (2024). *OpenAPI Specification v3.1.1*. OpenAPI Initiative: https://spec.openapis.org/oas/v3.1.1.html

Ortega Ovalle, M. (2026). Tendencias actuales en metodologías de desarrollo de software: una revisión sistemática. *Ciencia Latina Revista Científica Multidisciplinar*, *10*(2), 2233-2250. https://doi.org/10.37811/cl_rcm.v10i2.23290

Osorio Marin, D. (2024). *Uso de las metodologías ágiles en proyectos de desarrollo de software (2019-2023): una revisión sistemática de la literatura.* Corporación Universitaria Minuto de Dios (UNIMINUTO). https://hdl.handle.net/10656/21371

OWASP Foundation. (2025). *About OWASP Top 10 - 2025*. OWASP: https://owasp.org/Top10/2025/0x01_2025-About_OWASP/

pgAdmin Development Team. (s.f.). *pgAdmin 4 development documentation*. pgAdmin: https://www.pgadmin.org/docs/pgadmin4/development/

Pontificia Universidad Católica del Ecuador. (2017). *Manual de Identidad PUCE*. https://www.pucesi.edu.ec/webs2/wp-content/uploads/2024/05/normativas/planeacion-y-gestion-academica-administrativa/manuales/Manual-de-Identidad-PUCE-V007-072017.pdf

Quiroz-Vázquez, C. (s.f). *IBM*. IBM: https://www.ibm.com/es-es/think/topics/software-development

Reactive Programming. (s.f.). *Arquitectura en Capas*. Reactive Programming: https://reactiveprogramming.io/blog/es/estilos-arquitectonicos/capas

SmartBear Software. (s.f.). *About Swagger Specification.* Swagger Documentation: https://swagger.io/docs/specification/v3_0/about/

Sommerville, I. (2011). *Ingeniería de Software* (9a ed.). Pearson Educación. https://gc.scalahed.com/recursos/files/r161r/w25469w/ingdelsoftwarelibro9_compressed.pdf

Swanlund, S., Wilson, V., Kapoor, V., Gabbard, M., y Srinath, C. (2024). Software development life cycle. *Journal of the Society for Clinical Data Management, 4*(1). https://doi.org/10.47912/jscdm.343

Taipalus, T. (2023). Database management system performance comparisons: A systematicliterature review. *Journal of Systems and Software*. https://doi.org/10.1016/j.jss.2023.111872

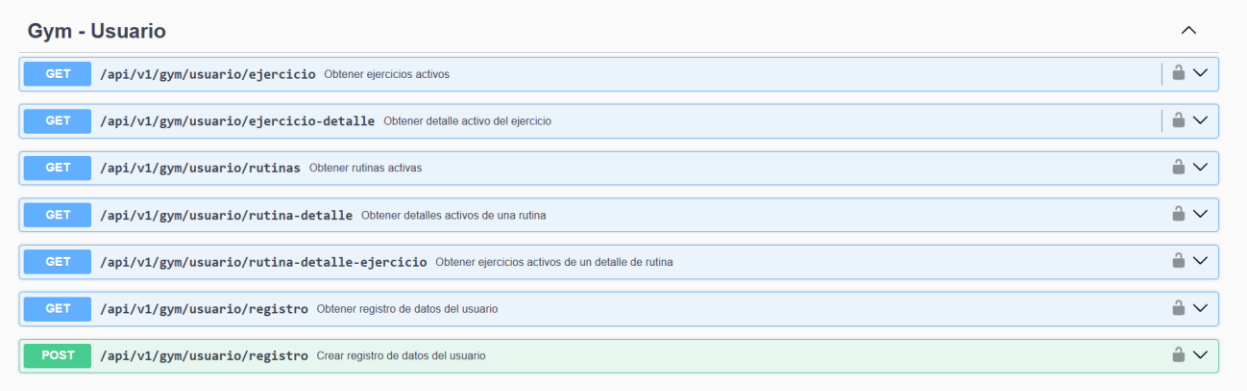
Torassa Colombero, V., Estelles, J., Gallegos, L., y Lopez, P. (2024). Monolitos vs. Microservicios en Arquitectura de Software. *JAIIO, Jornadas Argentinas de Informática, 10*(5), pp. 42-54. https://revistas.unlp.edu.ar/JAIIO/article/view/17984

Universidad de Valladolid. (2006). *Tema 2: El modelo cliente/servidor.* Departamento de Informática, E. U. Informática en Segovia, Universidad de Valladolid. https://www.infor.uva.es/~fdiaz/sd/2005_06/doc/SD_TE02_20060305.pdf

Universidad Nacional Abierta y Distancia. (2020). *Unidad 2: Levantamiento de información y determinación de Requerimientos.* Ingeniería de Requisitos.: https://repository.unad.edu.co/reproductor-ova/10596_35614/que_son_los_requerimientos.html

ANEXOS

Figura 22
Endpoint Usuario



Nota: Visualización de los endpoints por los cuales la aplicación permite al usuario ver datos.

Figura 23
EndPoints Administrador



Nota: Visualización de los endpoints por los cuales el administrador modifica los datos de la aplicación.

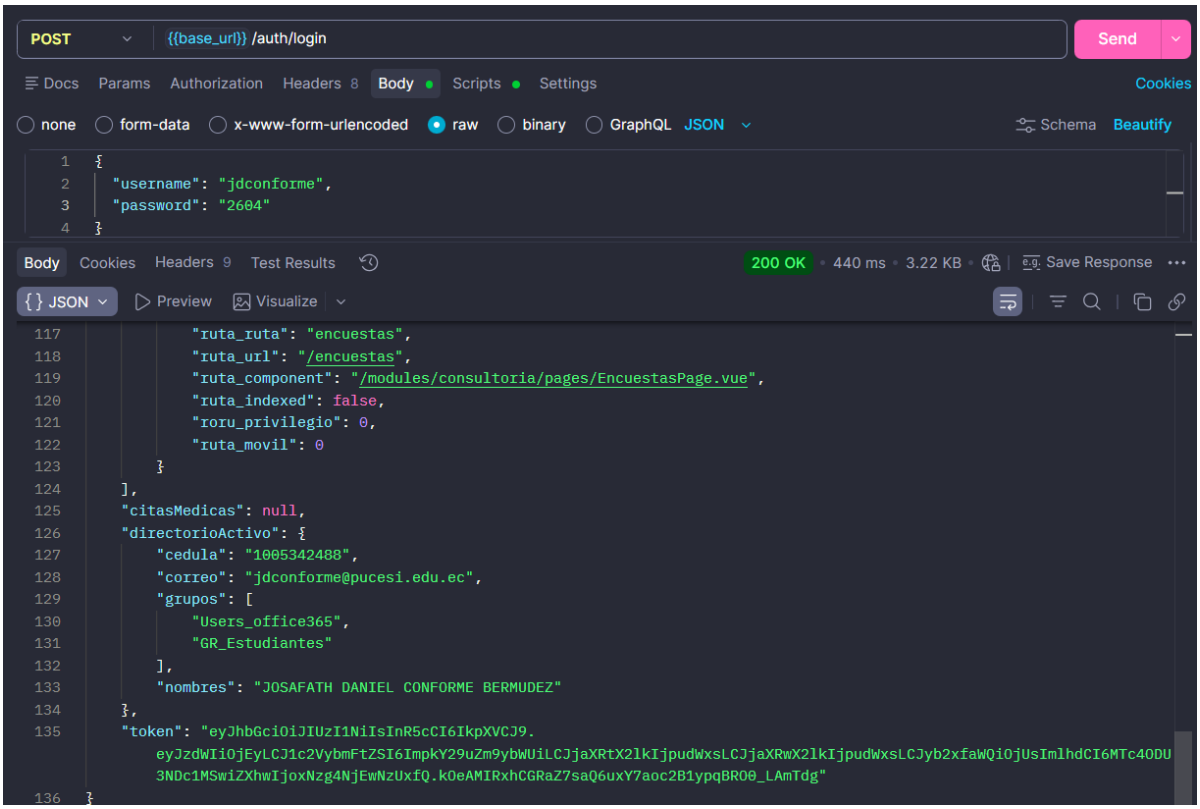
Tabla 20

Matriz de Casos de Prueba ejecutados

ID	Modulo Asociado	Descripción	Resultado Esperado	Resultado Obtenido	Estado
CP-01	Autenticación (HU-01)	1. Ingresar credenciales válidas en el BODY con Postman. 2. Presionar "Iniciar Sesión".	El servidor valida con Directorio Activo y devuelve código 200 OK junto con el token JWT.	El token JWT se recibe correctamente. Código 200 Ok devuelto.	Exitoso
CP-02	Seguridad (RF-02)	1. Interceptar petición HTTP en Postman hacia el endpoint /gym/usuario/rutinas. 2. Eliminar el token JWT de la cabecera Authorization.	El servidor debe denegar el acceso a los datos y devolver un error HTTP 401 Unauthorized.	El endpoint bloqueó la transacción protegiendo los datos, devolviendo el código 401.	Exitoso
CP-03	Rendimiento (RNF-02)	1. Acceder a la vista "Mis Rutinas" 2. Medir el tiempo de renderizado de la UI y descarga de datos.	La aplicación debe cargar la información en menos de 2 segundos.	Gracias a la implementación de caché en memoria y Future.wait en Flutter, la carga promedió ~800ms.	Exitoso
CP-04	Gestion (HU-03)	1. Ingresar un nuevo peso corporal (ej. 75.5 kg) en el formulario. 2. Enviar la petición POST a /gym/usuario/registro.	El servidor debe devolver 201 Created y la UI debe actualizar automáticamente la gráfica de historial.	El registro persistió en PostgreSQL; la interfaz móvil actualizó la vista reactivamente.	Exitoso
CP-05	Entrenamiento (HU-06)	1. Iniciar el cronómetro Tabata configurando 40s de trabajo y 90s de descanso. 2. Esperar que el contador llegue a cero.	El cronómetro debe hacer la transición automática a la fase de descanso e incrementar el número de serie.	Las transiciones de estado (setState) funcionaron sin bloquear el hilo principal de la aplicación.	Exitoso
CP-06	Multimedia (RNF-06)	1. Navegar por el catálogo de ejercicios. 2. Monitorear el consumo de red.	Los videos no deben descargarse a menos que el usuario presione el botón de reproducción (Lazy Loading).	El controlador de YouTube se inicializa en pausa, ahorrando ancho de banda móvil del estudiante.	Exitoso

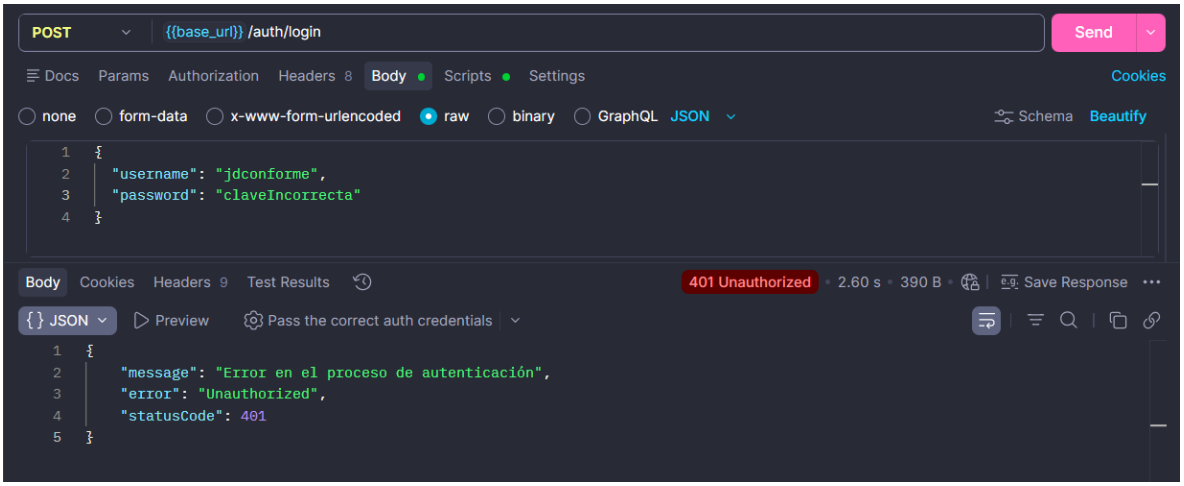
Nota: matriz de casos de prueba ejecutados desde la perspectiva del cliente.

Figura 24
Login Con Credenciales Validas



Nota: Captura de pantalla de Postman con la visualización del token devuelto y la respuesta 200 ok cumpliendo con el caso de prueba 01.

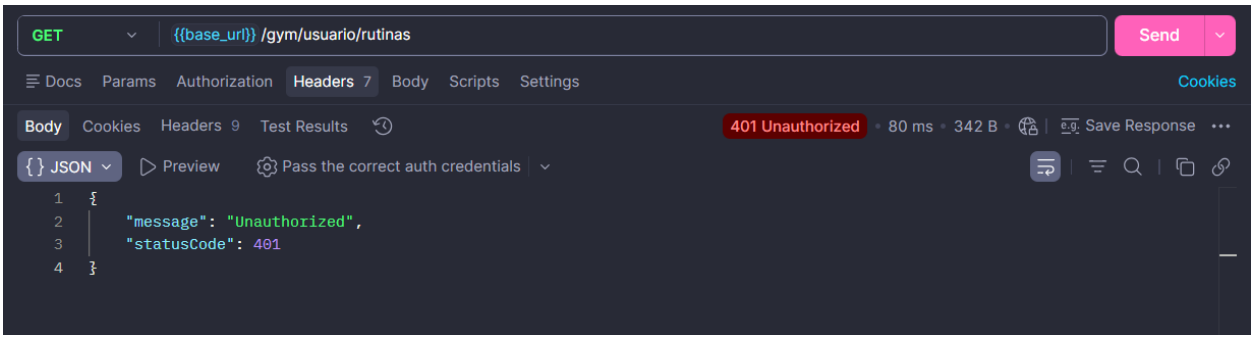
Figura 25
Caso de Prueba Login Credenciales Invalidas



Nota: Captura de pantalla de Postman con Credenciales Invalidad y la respuesta.

Figura 26

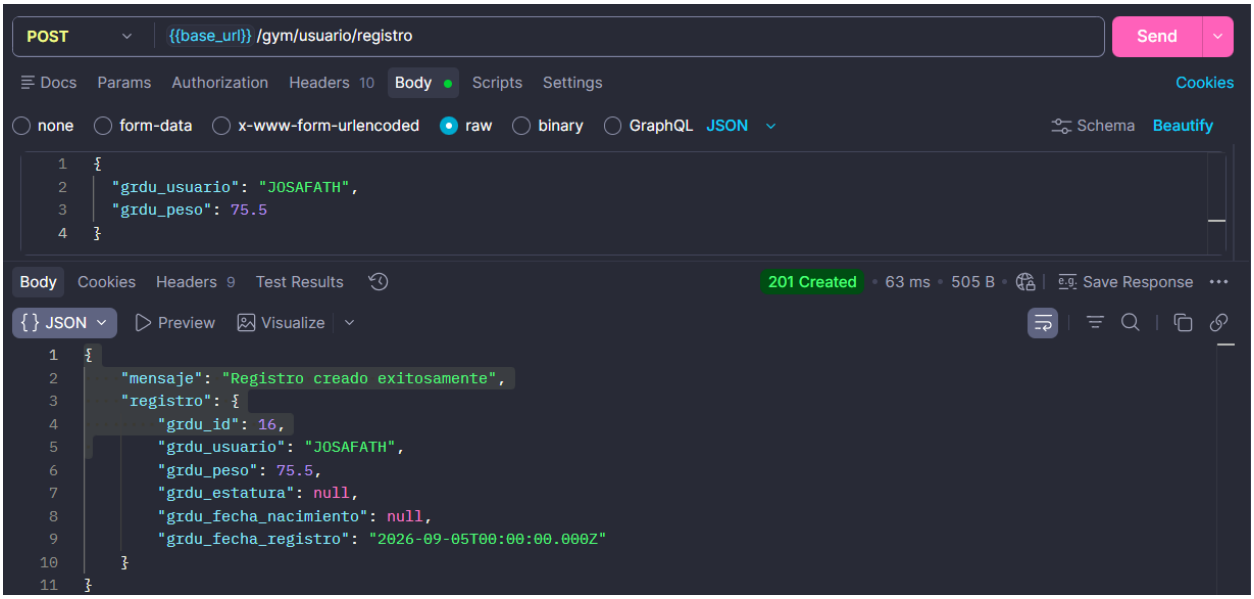
Endpoint de rutinas sin token (no autenticado)



Nota: Captura de pantalla de Postman de la obtención de rutinas sin un token autorizado con respuesta 401 cumpliendo con el caso de prueba 02.

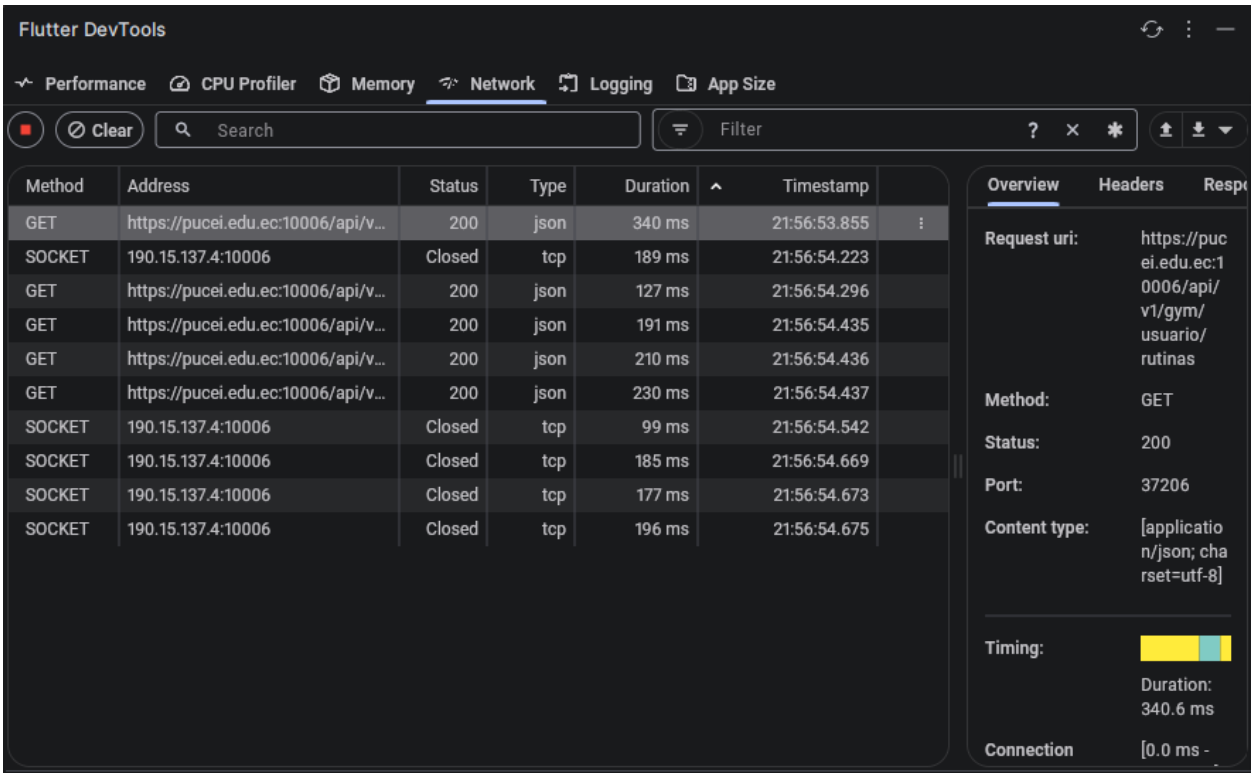
Figura 27

Registro Nuevo Peso Corporal



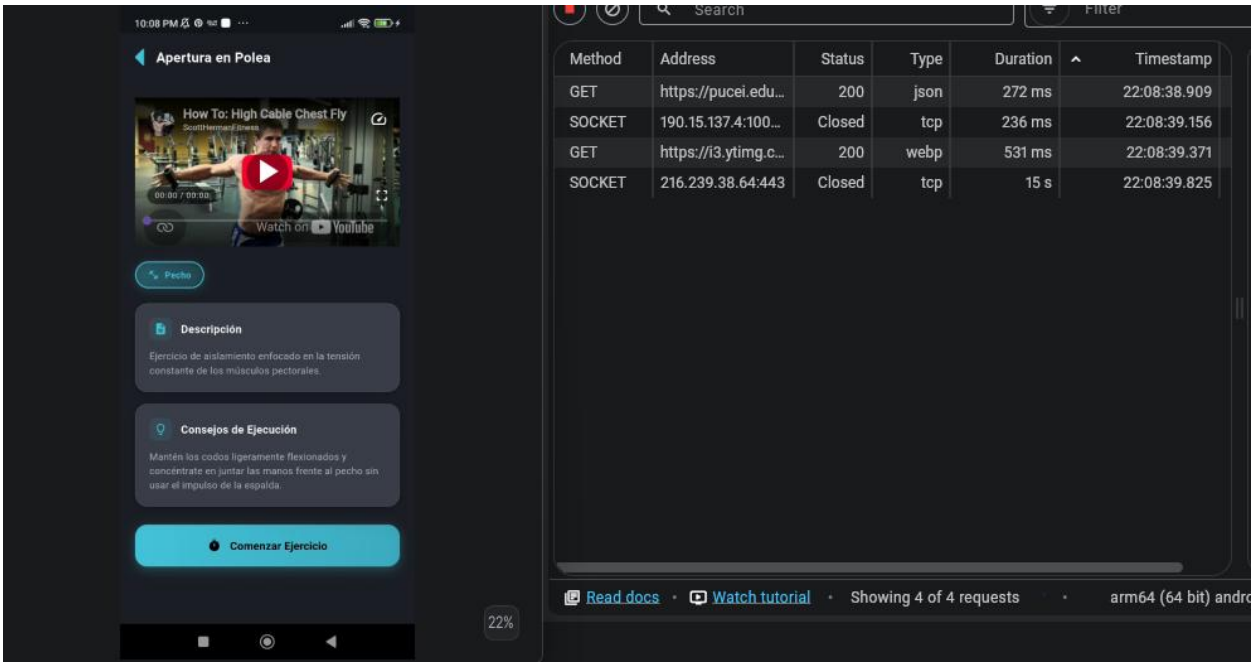
Nota: Captura de pantalla de Postman del registro de nuevo peso corporal para un usuario autorizado con respuesta 201 cumpliendo con el caso de prueba 04.

Figura 28
Prueba de Rendimiento



Nota: Captura de pantalla de Android Studio usando Flutter DevTools obteniendo parámetros de rendimiento de internet cumpliendo con el caso de prueba 02.

Figura 29
Prueba LazyLoading



Nota: Captura de pantalla de Android Studio usando Flutter DevTools obteniendo parámetros de rendimiento de internet donde se logra ver que no ha iniciado la descarga del video porque el usuario no ha dado click en play cumpliendo con el caso de prueba 06.