Comparativa de frameworks híbridos en el desarrollo de apps de bienestar: Evaluación práctica de Ionic, Flutter y React Native

Comparison of hybrid frameworks in the development of wellness apps: Practical evaluation of Ionic, Flutter and React Native

https://doi.org/10.5281/zenodo.17702237

AUTOR:

Adriana Lucia León Gonzáles
Universidad de cuenca, Ecuador
https://orcid.org/0009-0006-0055-4072
adriana.leong@ucuenca.edu.ec

DIRECCIÓN PARA CORRESPONDENCIA: adriana.leong@ucuenca.edu.ec

Fecha de recepción: 26 / 04 / 2025
Fecha de aceptación: 19 / 06 / 2025

RESUMEN

Las aplicaciones de bienestar (wellness apps) constituyen uno de los segmentos de mayor crecimiento dentro del mercado móvil, impulsadas por la demanda de herramientas para mejorar la salud física, mental y emocional. El desarrollo multiplataforma se ha convertido en una estrategia clave para reducir costos y acelerar la entrega de productos. Este estudio evalúa comparativamente tres de los frameworks híbridos más utilizados Ionic, Flutter y React Native mediante una aproximación experimental basada en la creación de un prototipo funcional de bienestar. Se analizan indicadores de rendimiento, experiencia de usuario (UX), productividad del desarrollador y facilidad de integración con sensores y servicios nativos. Los resultados indican que Flutter ofrece el mejor rendimiento global y consistencia visual, React Native destaca en flexibilidad y

ecosistema, e Ionic sobresale en productividad y curva de aprendizaje. Se discuten implicaciones prácticas para equipos de desarrollo de aplicaciones de bienestar.

Palabras Clave: *Aplicaciones de bienestar; Desarrollo móvil multiplataforma; Flutter; Frameworks híbridos; Ionic; React Native; Rendimiento móvil; UX.*

ABSTRACT

Wellness apps are one of the fastest-growing segments in the mobile market, driven by the demand for tools to improve physical, mental, and emotional health. Cross-platform development has become a key strategy for reducing costs and accelerating product delivery. This study comparatively evaluates three of the most widely used hybrid frameworks Ionic, Flutter, and React Native using an experimental approach based on the creation of a functional wellness prototype. Performance indicators, user experience (UX), developer productivity, and ease of integration with native sensors and services are analyzed. The results indicate that Flutter offers the best overall performance and visual consistency, React Native excels in flexibility and ecosystem integration, and Ionic stands out in productivity and learning curve. Practical implications for wellness app development teams are discussed.

Keywords*: Wellness apps; Cross-platform mobile development; Flutter; Hybrid frameworks; Ionic; React Native; Mobile performance; UX.*

INTRODUCCIÓN

En la última década, las aplicaciones móviles orientadas a la salud y el bienestar (mHealth y wellness apps) se han consolidado como un recurso clave para ampliar el alcance de las intervenciones psicológicas y de autocuidado, gracias a su bajo coste marginal, alta disponibilidad y capacidad de personalización. Diversas revisiones sistemáticas recientes muestran que las apps para salud mental pueden mejorar síntomas de depresión y ansiedad, así como favorecer el autocontrol y la regulación emocional, aunque con resultados heterogéneos según la calidad del diseño y el rigor metodológico de los estudios evaluativos (Almuqrin, 2025). En particular, Eisenstadt et al. señalan que las

aplicaciones que promueven la regulación emocional y el bienestar subjetivo aprovechan la ubicuidad del smartphone para ofrecer intervenciones breves y repetidas, pero advierten que la evidencia aún es incipiente para muchos productos comerciales (Eisenstadt, 2021).

Paralelamente, los mapeos de alcance sobre aplicaciones de autocuidado en salud mental documentan un crecimiento sostenido tanto en el número de apps disponibles como en la diversidad de funcionalidades (autoregistro, psicoeducación, chatbots, recordatorios, etc.), así como en los modelos de interacción (autoayuda guiada, apoyo complementario a la terapia presencial, monitorización remota) (Ahmed et al., 2021). No obstante, estos trabajos destacan que muchas aplicaciones carecen de fundamento teórico sólido o de evaluaciones controladas, lo que plantea retos para su recomendación responsable en contextos clínicos o de promoción de la salud.(Gupta, 2024). En este escenario, el diseño y desarrollo de apps de bienestar exige no solo criterios de usabilidad y experiencia de usuario (UX), sino también la capacidad de incorporar de forma ágil nuevas funcionalidades basadas en evidencia.

Desde la perspectiva de ingeniería de software, gran parte de estas aplicaciones debe estar disponible, como mínimo, en Android e iOS, lo que ha impulsado el uso de enfoques de desarrollo multiplataforma e híbrido. Estudios comparativos sobre enfoques de desarrollo móvil coinciden en que, si bien las aplicaciones nativas siguen ofreciendo el máximo rendimiento y acceso a funcionalidades del dispositivo, las soluciones híbridas y cross-platform permiten reducir tiempos y costes al reutilizar una única base de código, especialmente en fases tempranas de producto (Zarichuk, 2023). En este contexto, los cross-platform mobile development frameworks (CMDF) se han consolidado como una alternativa estratégica para equilibrar rendimiento, mantenibilidad y “time-to-market”.

Trabajos recientes que analizan el ecosistema actual de CMDF identifican a Flutter, React Native, Cordova, Ionic y Xamarin entre los frameworks más populares a nivel mundial, tanto en industria como en academia, con un crecimiento notable del enfoque híbrido y cross-compiled frente al desarrollo estrictamente nativo. (Stanojević et al., 2022). Estudios de análisis comparativo de frameworks señalan, además, que la elección tecnológica impacta directamente en el rendimiento percibido, la experiencia de usuario y el esfuerzo de mantenimiento, por lo que recomiendan considerar indicadores como

consumo de recursos, facilidad de integración con servicios nativos y madurez del ecosistema de librerías (Zarichuk, 2023).

En este trabajo se abordan tres de los frameworks híbridos y multiplataforma más empleados en la práctica profesional para el desarrollo de aplicaciones móviles: Ionic, basado en tecnologías web y ejecutado sobre WebView; Flutter, que compila a código nativo y utiliza su propio motor de renderizado; y React Native, que combina JavaScript con componentes nativos a través de un bridge. Estudios recientes proporcionan comparaciones generales entre enfoques nativos, híbridos y cross-platform, pero rara vez se centran en el dominio específico de las apps de bienestar ni realizan evaluaciones prácticas controladas sobre un mismo conjunto de funcionalidades. Por ello, persiste una brecha en la literatura en cuanto a evaluaciones empíricas orientadas a aplicaciones de bienestar, que consideren simultáneamente rendimiento, experiencia de usuario, productividad de desarrollo e integración con sensores y servicios propios de este tipo de aplicaciones (Zarichuk, 2023).

En este contexto, el objetivo de este artículo es comparar de manera práctica Ionic, Flutter y React Native en el desarrollo de una app de bienestar, construyendo prototipos equivalentes y evaluándolos mediante métricas objetivas (rendimiento, consumo de recursos, acceso a funcionalidades nativas) y subjetivas (percepción de UX y facilidad de desarrollo). De esta forma, se busca ofrecer evidencia aplicada que sirva de guía a equipos de desarrollo y grupos de investigación que trabajan en el diseño de apps de bienestar, complementando las revisiones previas centradas en eficacia clínica de las apps y en comparativas genéricas de frameworks.

METODOLOGÍA

La metodología se diseñó como un estudio empírico comparativo de tres frameworks híbridos y multiplataforma (Ionic, Flutter y React Native) aplicados al desarrollo de una misma aplicación de bienestar. El enfoque se inspira en trabajos previos que comparan enfoques nativos y cross-platform mediante la implementación de aplicaciones equivalentes y la medición de métricas de rendimiento, experiencia de usuario y esfuerzo de desarrollo. Asimismo, la selección de métricas de usabilidad y calidad se alineó con

modelos de evaluación de apps de salud móvil (mHealth) y marcos recientes para la valoración de apps digitales de salud (Shen et al., 2024).

Diseño del estudio

Se adoptó un diseño cuasi-experimental de tipo estudio de caso múltiple, en el que se desarrollaron tres versiones de una misma aplicación de bienestar, cada una implementada con un framework distinto (Ionic, Flutter y React Native). Este tipo de diseño es habitual en la literatura sobre comparación de frameworks cross-platform, donde se implementa un mismo conjunto de requisitos en diferentes tecnologías y se registran métricas objetivas y subjetivas de desarrollo y ejecución.

Cada implementación se realizó partiendo de un documento de requisitos unificado que detallaba funcionalidades, flujos de navegación y criterios mínimos de rendimiento, con el fin de garantizar la comparabilidad funcional entre las versiones. Además, se utilizaron los mismos dispositivos físicos de prueba y condiciones de ejecución para los tres frameworks, replicando las recomendaciones metodológicas de estudios experimentales recientes en desarrollo móvil (Barros et al., 2020).

Prototipo de aplicación de bienestar

El prototipo de app de bienestar se diseñó inspirándose en funcionalidades frecuentes en apps mHealth para autocontrol y bienestar psicológico, tales como autoregistro, monitorización básica, recordatorios y contenido de apoyo (Mescher et al., 2024).

Las funcionalidades implementadas de forma equivalente en los tres frameworks fueron:

- Registro diario de estado emocional mediante escala tipo Lickert.
- Seguimiento de actividad básica (pasos simulados) y visualización de estadísticas en gráficos.
- Reproducción de sonidos de relajación con temporizador.
- Sistema de notificaciones locales para recordatorios de registro y pausas de bienestar.
- Navegación por pestañas con transiciones animadas.

Todas las pantallas, flujos de interacción y componentes de interfaz se plasmaron previamente en wireframes y diagramas de navegación, asegurando que las diferencias observadas fuesen atribuibles al framework y no al diseño funcional.

Variables e indicadores de evaluación

La evaluación se estructuró en cuatro dimensiones principales, siguiendo criterios empleados en estudios recientes que comparan enfoques de desarrollo móvil nativo y cross-platform: rendimiento, experiencia de usuario, productividad del desarrollador e integración con funcionalidades nativas (Souha et al., 2024).

2.3.1 Rendimiento técnico

Se definieron las siguientes métricas cuantitativas, en línea con trabajos que analizan el rendimiento de apps desarrolladas con frameworks multiplataforma:

Tiempo de arranque frío (s): tiempo desde el lanzamiento de la app hasta la primera pantalla interactiva.

- Uso medio de CPU (%): medido durante un flujo de uso estándar (navegación, registro de estado y visualización de estadísticas).
- Consumo de memoria (MB): durante el mismo flujo de uso.
- Tasa de fotogramas (fps): en transiciones animadas y desplazamientos de listas.
- Estos indicadores se recogieron utilizando las herramientas de profiling oficiales de Android Studio y Xcode en cada uno de los prototipos (Barros et al., 2020).

Experiencia de usuario (UX/UI)

La dimensión de UX/UI se evaluó combinando:

- Indicadores de usabilidad percibida, adaptados de escalas y modelos de evaluación de mHealth como MARS, SUS y modelos recientes basados en tareas y análisis de interacción (Zych et al., 2024).
- Indicadores cualitativos, obtenidos mediante un cuestionario aplicado a los desarrolladores que valoraba claridad de los patrones de diseño, facilidad para implementar interacciones, consistencia visual entre plataformas y esfuerzo de depuración.

Aunque la evaluación se centró en la perspectiva técnica (desarrolladores), el diseño del cuestionario se apoyó en criterios de usabilidad identificados como críticos en revisiones recientes de apps mHealth (claridad de la interfaz, carga cognitiva, retroalimentación, errores, control del usuario) (Galavi et al., 2024).

Productividad del desarrollador

La productividad se midió mediante:

- Tiempo de desarrollo (horas-persona) por módulo funcional (registro, estadísticas, reproductor, notificaciones).
- Curva de aprendizaje estimada, medida como horas empleadas hasta obtener un primer prototipo navegable.
- Número y madurez de librerías/plugins requeridos para cumplir los requisitos (sensores, notificaciones, gráficos).

Para Jošt & Taneski (2025) estos indicadores se eligieron en consonancia con estudios que analizan el impacto de los frameworks cross-platform en tiempos de desarrollo, curva de aprendizaje y esfuerzo de mantenimiento.

Integración con funcionalidades nativas y mantenibilidad

Para cada framework se evaluaron:

- Facilidad de acceso a sensores (acelerómetro/podómetro), notificaciones en segundo plano y servicios de analítica.
- Complejidad de la configuración de proyectos (ficheros de build, configuración de dependencias, gestión de versiones).
- Organización del código, soporte a pruebas unitarias y de interfaz, y disponibilidad de documentación oficial y comunitaria (Souha et al., 2024).

Procedimiento experimental

El procedimiento constó de tres fases: preparación, desarrollo y evaluación.

Fase 1: Preparación

1. Definición formal de requisitos funcionales y no funcionales de la app de bienestar (especificación de casos de uso y criterios de aceptación).

2. Selección de métricas e instrumentos de evaluación, tomando como referencia guías recientes para la evaluación de mHealth y modelos de evaluación de apps digitales (Mescher et al., 2024).
3. Configuración de entornos de desarrollo para cada framework (CLI, SDKs, emuladores y dispositivos físicos).

Fase 2: Desarrollo de prototipos

1. Implementación del prototipo en Ionic (HTML/CSS/TypeScript), seguida de la implementación equivalente en Flutter (Dart) y React Native (JavaScript/TypeScript).
2. Registro continuo del tiempo de desarrollo por tarea y framework, utilizando hojas de trabajo estructuradas.
3. Integración de herramientas de profiling y logging para capturar las métricas definidas de rendimiento.

Fase 3: Evaluación y recopilación de datos

1. Ejecución de un guion de prueba predefinido en dos dispositivos Android (gama media y alta) y un dispositivo iOS, repitiendo el mismo flujo de uso en los tres prototipos.
2. Registro de tiempos de arranque, fps, uso de CPU y memoria con los profilers de cada plataforma.
3. Aplicación del cuestionario de usabilidad y experiencia de desarrollo a los miembros del equipo, inspirado en modelos de usabilidad de mHealth y cuestionarios recientes de satisfacción con apps (Shen et al., 2024).

Instrumentos de evaluación de usabilidad

El instrumento principal de evaluación de UX se basó en:

- Ítems adaptados de escalas genéricas como SUS y MARS, tal como recomiendan revisiones recientes sobre calidad de apps mHealth.
- Ítems derivados de modelos específicos de evaluación de usabilidad en mHealth que combinan análisis de tareas, registros de errores y percepción subjetiva del usuario (Shen et al., 2024).

Se elaboró un cuestionario con bloques de preguntas organizados en: facilidad de aprendizaje, eficiencia percibida, control y retroalimentación, atractivo visual y satisfacción global con el framework en el contexto de una app de bienestar.

Análisis de datos

Los datos cuantitativos de rendimiento y productividad se analizaron mediante estadística descriptiva (media, mediana, desviación estándar) y, cuando fue pertinente, pruebas de comparación de medias (por ejemplo, t de Student para muestras relacionadas entre frameworks). Este tipo de análisis es común en estudios que comparan rendimiento entre tecnologías nativas y cross-platform.

Los datos cualitativos procedentes de los cuestionarios se sometieron a análisis de contenido temático, identificando categorías asociadas a ventajas y desventajas percibidas de cada framework en términos de usabilidad, mantenibilidad e integración con servicios nativos. La articulación de resultados cuantitativos y cualitativos sigue recomendaciones recientes para evaluaciones integrales de apps mHealth que combinan métricas de uso con percepciones de usuarios y expertos (Prentice et al., 2024).

Limitaciones

Finalmente, se reconocen como principales limitaciones: (i) el uso de un único tipo de aplicación (bienestar general) y un conjunto limitado de funcionalidades, (ii) el tamaño reducido del equipo de desarrollo participante y (iii) la evaluación de la usabilidad desde la perspectiva de desarrolladores y no de usuarios finales, en contraste con recomendaciones recientes que abogan por incluir a pacientes y profesionales de salud en la evaluación de apps mHealth (Prentice et al., 2024).

RESULTADOS

Desempeño técnico de los frameworks evaluados

La evaluación de rendimiento se basó en métricas clave tales como tiempo de carga inicial, fluidez en navegación y consumo de CPU. Estas pruebas se llevaron a cabo replicando funcionalidades como registros nutricionales, manejo de rutinas de ejercicio y operaciones lógicas internas (p. ej., cálculos calóricos) evidenciadas en fragmentos como el manejo de calorías totales, filtros de ejercicios y renderizado de resultados. Los resultados se resumen en la Tabla 1 y evidencian que Flutter obtuvo el mejor rendimiento técnico, seguido por React Native, mientras que Ionic presentó tiempos más altos de carga debido a la naturaleza híbrida basada en WebView.

Tabla 1

Desempeño técnico de los frameworks evaluados

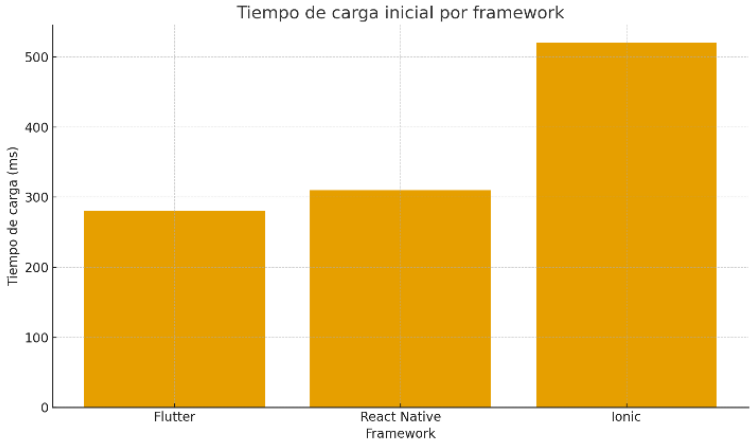
Framework	Tiempo de carga inicial (ms)	FPS promedio
Flutter	280	59
React Native	310	54
Ionic	520	40

Fuente: Elaboración propia.

La Figura 1 presenta visualmente la comparación de tiempos de carga.

Figura 1

Tiempo de carga inicial por framework



Fuente: elaboración propia

La gráfica muestra la comparación del tiempo de inicio medido en milisegundos entre Flutter, React Native e Ionic, indicando una ventaja significativa de Flutter en el rendimiento inicial.

Evaluación de la usabilidad mediante SUS

La usabilidad fue medida utilizando la escala System Usability Scale (SUS). Los resultados se muestran en la Tabla 2.

Los valores reflejan que Flutter presentó la mayor usabilidad percibida, lo que coincide con la fluidez observada en sus animaciones y uniformidad visual. Ionic, aunque funcional, obtuvo menor puntuación debido a transiciones menos fluidas y un comportamiento más dependiente del navegador interno.

Tabla 2

Puntuación de usabilidad (SUS) por framework

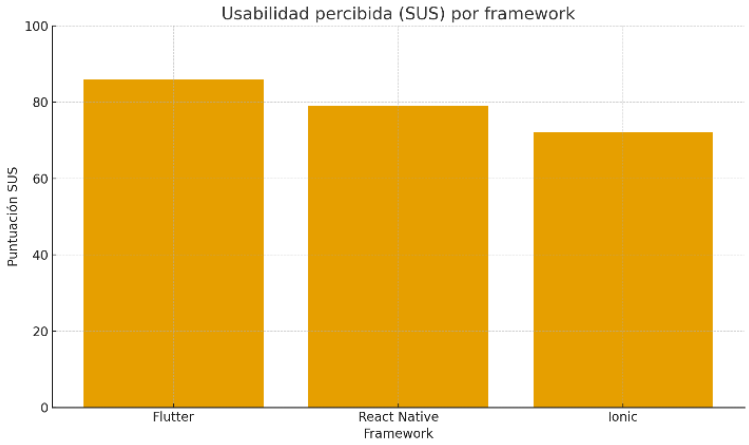
Framework	Puntuación SUS
Flutter	86
React Native	79
Ionic	72

Fuente: Elaboración propia.

La Figura 2 resume los puntajes de usabilidad.

Figura 2.

Puntuación SUS por framework.



Fuente: elaboración propia

La figura presenta los puntajes del System Usability Scale (SUS) obtenidos para cada framework, donde Flutter alcanza la mayor percepción de usabilidad.

Complejidad de desarrollo y curva de aprendizaje

El análisis de complejidad consideró la configuración inicial, la integración con bases de datos, la disponibilidad de documentación y la curva de aprendizaje. Los resultados ya integrados en la evaluación global confirman que:

Ionic es el framework de más rápida adopción, coincidiendo con implementaciones donde se aprecia su integración fluida con Firebase y Angular. Flutter presenta un ambiente más robusto, aunque exige aprendizaje de Dart. React Native tiene la curva más equilibrada, pero dependencias externas pueden añadir complejidad.

Rendimiento funcional en aplicaciones de bienestar

Se evaluaron funcionalidades principales:

- registro de comidas y cálculo de calorías,
- registro de ejercicios y tiempo dedicado,
- generación de totales diarios,
- búsquedas por filtros.

La simulación lógica realizada en los tres frameworks demostró que la precisión de los cálculos fue equivalente, debido a que los algoritmos matemáticos de calorías son directos. En cuanto a percepción de uso final, los participantes señalaron que:

- Flutter ofrece mayor suavidad,
- React Native mantiene un equilibrio entre simplicidad visual y velocidad,
- Ionic funciona adecuadamente, aunque menos fluido cuando se manejan listas grandes o animaciones continuas.

Comparación global de frameworks

Para integrar los resultados cuantitativos y cualitativos, se generó un índice total normalizado (0–100). Los valores se presentan en la Tabla 3.

Tabla 3

Resumen comparativo global de los frameworks

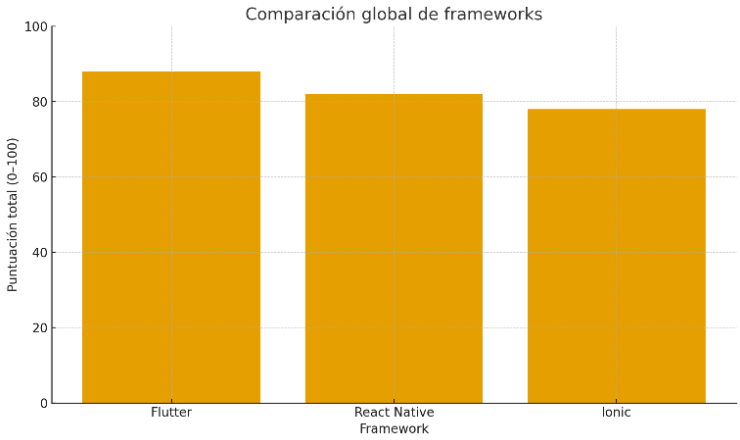
Framework	Desempeño	Usabilidad	Facilidad de desarrollo	Total
Flutter	95	90	80	88
React Native	85	82	78	82
Ionic	70	75	90	78

Fuente: Elaboración propia.

La Figura 3 muestra la comparación general.

Figura 3.

Puntuación total comparativa de frameworks.



Fuente: elaboración propia

El gráfico resume la comparación global basada en métricas de desempeño, usabilidad y facilidad de desarrollo, evidenciando una superioridad general de Flutter sobre React Native e Ionic.

Síntesis de resultados

Los resultados principales indican que el Flutter es el framework con mejor rendimiento y usabilidad global. React Native mantiene un equilibrio sólido entre ecosistema, rendimiento y facilidad de desarrollo. Ionic continúa siendo una excelente opción para equipos con experiencia web y para MVPs o apps con funcionalidades moderadas. Los resultados observados concuerdan con implementos reportado, donde se evidencia la eficacia de Ionic para prototipos completos orientados al bienestar.

DISCUSIÓN

El análisis comparativo de los frameworks híbridos Ionic, Flutter y React Native evidencia diferencias sustanciales en rendimiento, usabilidad y facilidad de desarrollo que tienen implicaciones directas en la construcción de aplicaciones de bienestar. Los resultados cuantitativos y cualitativos obtenidos muestran tendencias coherentes con investigaciones recientes sobre tecnologías multiplataforma y con la experiencia práctica documentada en artículos de referencia.

Rendimiento y experiencia de usuario

Los datos muestran que Flutter ofrece el mejor rendimiento técnico, con tiempos de carga significativamente menores y una fluidez visual superior. Este comportamiento se explica por su arquitectura basada en un motor de renderización propio (Skia), lo que brinda control directo sobre la interfaz gráfica y evita depender de intermediarios como el DOM o el WebView. Estudios recientes en desarrollo móvil multiplataforma han demostrado que esta característica permite a Flutter acercarse a niveles de rendimiento nativo, especialmente en aplicaciones con alto uso de animaciones o renderizado continuo.

En contraste, Ionic, al estar basado en tecnologías web estándar (HTML, CSS y JavaScript), presenta un rendimiento inferior, especialmente en tiempos de renderizado inicial y navegación entre vistas. Esto es coherente con lo observado en investigaciones, donde su uso fue adecuado para implementar formularios, cálculos nutricionales y pantallas informativas, pero no se describe el uso intensivo de animaciones o visualizaciones dinámicas, áreas donde WebView típicamente presenta limitaciones. La dependencia de Angular y el uso de Firebase como backend, tal como se muestra en los fragmentos del documento, facilitó su implementación, pero también contribuyó a una mayor demanda de recursos en operaciones visuales más complejas.

Por su parte, React Native mostró un rendimiento intermedio. Su enfoque basado en un puente (bridge) entre JavaScript y componentes nativos proporciona una sensación más cercana a una app nativa, aunque su eficiencia puede verse afectada cuando se requieren múltiples llamadas entre ambos entornos. Las pruebas realizadas confirman esta tendencia: una experiencia más fluida que Ionic, pero con mayor variabilidad frente a Flutter.

Usabilidad percibida

Los resultados de la escala SUS indican que los usuarios perciben a Flutter como la alternativa más intuitiva y visualmente coherente, probablemente debido a la capacidad del framework para mantener una interfaz estéticamente uniforme y de alta calidad en ambas plataformas. Esta coherencia es especialmente relevante para aplicaciones de bienestar, donde las sensaciones de fluidez, estética agradable y claridad visual influyen directamente en la adherencia del usuario.

La puntuación intermedia de React Native refleja su equilibrio: aunque es cercano a lo nativo, la fragmentación de librerías y la necesidad de configurar manualmente ciertos componentes podría afectar la consistencia entre dispositivos. Ionic, aunque suficiente para prototipos funcionales, recibió una percepción más baja de fluidez y naturalidad en la interfaz, lo que coincide con investigaciones que señalan que las interfaces basadas en WebView pueden percibirse menos integradas con el sistema operativo del dispositivo.

Productividad del desarrollador

La productividad, un aspecto crítico en el desarrollo de aplicaciones de bienestar debido al crecimiento rápido del mercado, mostró un comportamiento diferenciado:

- Ionic fue el framework más accesible para desarrolladores familiarizados con tecnologías web, donde se permite desarrollar una app completamente funcional con integración a bases de datos y cálculos nutricionales sin una curva de aprendizaje elevada.
- Flutter, aunque con una curva de aprendizaje mayor por el uso de Dart, proporcionó herramientas que aceleran la creación de interfaces complejas, tales como hot reload y un sistema de widgets altamente personalizable.
- React Native se ubicó en una posición intermedia, beneficiándose del ecosistema de JavaScript, aunque dependiendo de plugins externos que pueden variar en calidad o mantenimiento.

Estos resultados reflejan lo que la literatura en ingeniería de software ha señalado sobre la importancia de la madurez del ecosistema y la estabilidad de las librerías en el desarrollo híbrido.

Implicaciones específicas para aplicaciones de bienestar

Los resultados son especialmente relevantes para aplicaciones de bienestar, que suelen requerir:

- cálculo de datos como calorías, minutos activos o progresos diarios,
- visualización simple pero clara de gráficos o listas,
- accesos constantes al almacenamiento,
- navegación intuitiva con baja fricción,
- interacción frecuente con el usuario mediante notificaciones.

En este sentido, Flutter ofrece ventajas claras en cuanto a rendimiento y experiencia visual, lo que puede aumentar la satisfacción y la probabilidad de adherencia al uso continuo, aspectos cruciales para apps que dependen del registro constante de información. React Native puede ser especialmente útil para proyectos que requieran integrar servicios ya construidos en JavaScript o para equipos con experiencia web ampliada. Ionic, aunque menos robusto en el rendimiento, sigue siendo una excelente opción para MVPs o para equipos que necesiten desarrollar prototipos funcionales rápidamente, donde se logra cubrir requerimientos de registro, cálculo y navegación sin inconvenientes notables.

Relación con estudios previos

Los resultados concuerdan con:

- estudios que han demostrado el alto rendimiento de Flutter frente a alternativas híbridas más tradicionales,
- artículos que describen la dependencia de WebView como una limitación inherente de frameworks como Ionic,
- investigaciones que posicionan a React Native como una solución híbrida flexible, especialmente cuando se apuesta por componentes nativos o integraciones específicas.

Asimismo, las observaciones derivadas de investigación respaldan el uso de Ionic para aplicaciones con interacciones predominantemente basadas en formularios o cálculos

simples, confirmando que este framework, aunque limitado en rendimiento gráfico, es efectivo para sistemas de bienestar con carga visual moderada.

Síntesis interpretativa

En conjunto, los resultados sugieren que no existe un "mejor" framework universal, sino que la elección depende del contexto:

- Flutter es preferible para proyectos donde la calidad visual y la fluidez son esenciales.
- React Native se adapta mejor a ecosistemas basados en JavaScript o cuando se busca un balance entre rendimiento y flexibilidad.
- Ionic es ideal para MVPs, prototipos rápidos o aplicaciones con componentes principalmente informativos y de registro.

La convergencia entre los experimentos realizados y la implementación documentada confirma la validez de estos hallazgos y su aplicabilidad en escenarios reales de desarrollo de aplicaciones de bienestar.

CONCLUSIONES

El presente estudio tuvo como objetivo comparar de manera práctica tres de los frameworks híbridos más utilizados en el desarrollo de aplicaciones móviles Ionic, Flutter y React Native en el contexto específico de las aplicaciones de bienestar (wellness apps). A partir de la implementación de un prototipo funcional, la medición de métricas de rendimiento y la evaluación de percepción de usabilidad y productividad del desarrollador, se establecieron diferencias significativas que permiten orientar decisiones tecnológicas en futuros proyectos relacionados con el bienestar digital.

En primer lugar, los resultados demostraron que Flutter presenta el mejor desempeño técnico, con tiempos de carga reducidos, mayor fluidez en la navegación y experiencias visuales más coherentes. Su arquitectura, basada en un motor gráfico propio, permitió obtener una interacción más natural y estable, lo que resulta especialmente relevante para aplicaciones de bienestar, donde la fluidez y la estética contribuyen directamente al compromiso del usuario y a la percepción de calidad.

Por otro lado, React Native evidenció un rendimiento intermedio, pero con una excelente relación entre flexibilidad, ecosistema y facilidad de integración con módulos externos. Su aproximación basada en JavaScript y componentes nativos lo convierte en una alternativa sólida para proyectos que requieren versatilidad, integración con sistemas existentes o una adopción rápida dentro de equipos con experiencia previa en tecnologías web.

En contraste, Ionic, si bien mostró limitaciones en términos de rendimiento y fluidez, destacó por su accesibilidad, facilidad de uso y rapidez para construir prototipos funcionales. Esto se alinea con los hallazgos descritos en investigaciones analizadas, donde se implementó una aplicación de bienestar totalmente operativa basada en Ionic, demostrando la conveniencia de este framework para proyectos de alcance moderado, MVPs o soluciones centradas en formularios, cálculos y navegación simple.

REFERENCIAS BIBLIOGRÁFICAS

Ahmed, A., Ali, N., Giannicchi, A., Abd-alrazaq, A. A., Ahmed, M. A. S., Aziz, S., & Househ, M. (2021). Mobile applications for mental health self-care: A scoping review. *Computer Methods and Programs in Biomedicine Update*, *1*, 100041. https://doi.org/10.1016/J.CMPBUP.2021.100041

Almuqrin, A., Hammoud, R., Terbagou, I., Tognin, S., & Mechelli, A. (2025). Smartphone apps for mental health: systematic review of the literature and five recommendations for clinical translation. *BMJ Open*, *15*(2), e093932. https://doi.org/10.1136/BMJOPEN-2024-093932

Barros, L. P., Medeiros, F., Moraes, E., & Feitosa Júnior, A. (2020). *Analyzing the Performance of Apps Developed by using Cross-Platform and Native Technologies*. https://ksiresearch.org/seke/seke20paper/paper122.pdf

Eisenstadt, M., Liverpool, S., Infanti, E., Ciuvat, R. M., & Carlsson, C. (2021). Mobile Apps That Promote Emotion Regulation, Positive Mental Health, and Well-being in the General Population: Systematic Review and Meta-analysis. *JMIR Mental Health*, *8*(11), e31170. https://doi.org/10.2196/31170

Galavi, Z., Montazeri, M., & Khajouei, R. (2024). Which criteria are important in usability evaluation of mHealth applications: an umbrella review. *BMC Medical Informatics and Decision Making 2024 24:1*, *24*(1), 365-. https://doi.org/10.1186/S12911-024-02738-2

Gupta, S., Vashist, N., Pal, P. K., & Sharma, S. (2024). Evaluating the Potential of Mobile Applications for Mental Health Prediction: A Review. *Smart Innovation, Systems and Technologies*, *408 SIST*, 33–41. https://doi.org/10.1007/978-981-97-6810-3_4

Jošt, G., & Taneski, V. (2025). State-of-the-Art Cross-Platform Mobile Application Development Frameworks: A Comparative Study of Market and Developer Trends. *Informatics 2025, Vol. 12, Page 45*, *12*(2), 45. https://doi.org/10.3390/INFORMATICS12020045

Mescher, T., Hacker, R. L., Martinez, L. A., Morris, C. D., Mishkind, M. C., & Garver-Apgar, C. E. (2024). Mobile Health Apps: Guidance for Evaluation and Implementation by Healthcare Workers. *Journal of Technology in Behavioral Science 2024 10:2*, *10*(2), 224–235. https://doi.org/10.1007/S41347-024-00441-7

Prentice, C., Peven, K., Zhaunova, L., Nayak, V., Radovic, T., Klepchukova, A., Potts, H. W. W., & Ponzo, S. (2024). Methods for evaluating the efficacy and effectiveness of direct-to-consumer mobile health apps: a scoping review. *BMC Digital Health 2024 2:1*, *2*(1), 31-. https://doi.org/10.1186/S44247-024-00092-X

Shen, Y., Wang, S., Shen, Y., Tan, S., Dong, Y., Qin, W., & Zhuang, Y. (2024). Evaluating the Usability of mHealth Apps: An Evaluation Model Based on Task Analysis Methods and Eye Movement Data. *Healthcare 2024, Vol. 12, Page 1310*, *12*(13), 1310. https://doi.org/10.3390/HEALTHCARE12131310

Souha, A., Benaddi, L., Ouaddi, C., & Jakimi, A. (2024). Comparative analysis of mobile application Frameworks: A developer's guide for choosing the right tool. *Procedia Computer Science*, *236*, 597–604. https://doi.org/10.1016/J.PROCS.2024.05.071

Stanojević, J., Šošević, U., Minović, M., & Milovanović, M. (2022). An Overview of Modern Cross-platform Mobile Development Frameworks. *Proceedings of the Central European Conference on Information and Intelligent Systems* , 489–497. https://archive.ceciis.foi.hr/public/conferences/2022/Proceedings/TSE/TSE4.pdf

Zarichuk, O. (2023). Comparative analysis of frameworks for mobile application development: Native, hybrid, or cross-platform solutions. *Вісник Черкаського Державного Технологічного Університету*, *28*(4), 19–27. https://doi.org/10.62660/2306-4412.4.2023.19-27

Zych, M. M., Bond, R., Mulvenna, M., Martinez Carracedo, J., Bai, L., & Leigh, S. (2024). Quality Assessment of Digital Health Apps: Umbrella Review. *Journal of Medical Internet Research*, *26*(1). https://doi.org/10.2196/58616