TOOLS FOR SOFTWARE QUALITY AUTOMATION IN MOBILE APPLICATIONS. SOFTWARE: PET FEEDING

José Alexander Vásquez Caruajulca, Estudiante[1], Kevin Antony Vera Saenz, Estudiante[1], Patricia Janet Uceda Martos, Doctora[1], Laura Sofía Bazán Díaz, Magíster[1]
[1]Universidad Privada del Norte, Perú, *N00036584@upn.pe, N00037217@upn.pe, patricia.uceda@upn.edu.pe*, laura.bazan@upn.pe

Abstract– ***The objective of the present research was to determine the use of automated tools to evaluate the quality of mobile applications based on the ISO/IEC 25000 standard, considering the dimensions of: maintainability, reliability, safety, performance or efficiency, functionality, compatibility, usability and portability in a case study: software product for feeding pets, developed in Java language, under the Android Studio IDE. The research was descriptive detailing the evaluation of software quality through the automated tools SonarQube and Firebase compatible with Android Studio. It was concluded with improvements in the maintainability characteristics, leaving pending the improvement of modification capacity; The quality of reliability was very good improving the start time of the application; The quality of performance/efficiency obtained an evaluation for improvement; in addition, the quality of the software was evaluated as functional, compatible, and portable.***

Keywords: software quality, ISO 25000, automated tool, mobile application, SonarQube, Firebase

Digital Object Identifier (DOI):
http://dx.doi.org/10.18687/LACCEI2022.1.1.534
ISBN: 978-628-95207-0-5 **ISSN:** 2414-6390

HERRAMIENTAS PARA AUTOMATIZACIÓN DE CALIDAD DE SOFTWARE EN APLICACIONES MÓVILES. SOFTWARE: ALIMENTACIÓN DE ANIMALES DOMÉSTICOS

José Alexander Vásquez Caruajulca, Estudiante[1], Kevin Antony Vera Saenz, Estudiante[1], Patricia Janet Uceda Martos, Doctora[1], Laura Sofía Bazán Díaz, Magíster[1]
[1]Universidad Privada del Norte, Perú, *N00036584@upn.pe*, *N00037217@upn.pe*, *patricia.uceda@upn.edu.pe*, laura.bazan@upn.pe

Resumen– El objetivo de la presente investigación fue determinar el uso de herramientas automatizadas para evaluar la calidad de aplicaciones móviles en base al estándar ISO/IEC 25000, teniendo en cuenta las dimensiones de: mantenibilidad, fiabilidad, seguridad, rendimiento o eficiencia, funcionalidad, compatibilidad, usabilidad y portabilidad en un estudio de caso: producto software para alimentación de animales domésticos, desarrollado en lenguaje Java, bajo el IDE Android Studio. La investigación fue descriptiva detallando la evaluación de la calidad de software a través de las herramientas automatizadas SonarQube y Firebase compatibles con Android Studio. Se concluyó con mejoras en las características de mantenibilidad, dejando pendiente la mejora de capacidad de modificación; la calidad de la fiabilidad fue muy buena mejorando el tiempo de inicio de la aplicación; la calidad del rendimiento/eficiencia obtuvo una evaluación por mejorar; además la calidad del software fue evaluada como funcional, compatible y portable.

Palabras clave: calidad de software, ISO 25000, herramienta automatizada, aplicativo móvil, SonarQube, Firebase

I. INTRODUCCIÓN

Desde hace muchos años, se ha venido analizando la importancia de la calidad de software como elemento que impacta en el ciclo de desarrollo y en el nivel de satisfacción de los usuarios. El portal ISO 25000 hace mención a la necesidad de contar con un modelo que permita determinar las características de la calidad que se evalúan al momento de estudiar las propiedades de un software. Asimismo, la calidad del producto de software se relaciona con el grado en el que dicho producto satisface los requisitos de los clientes o usuarios, aportándole valor [1].

La Organización Internacional de Normalización (ISO) tiene como parte de su rol, evaluar y actualizar constantemente los estándares de calidad y específicamente a nivel de calidad de producto de software se ha evidenciado evolución desde el estándar 9126, el cual fue publicado en el año 1991. La calidad se consideraba como el nivel de cumplimiento de características acordadas con el cliente, incluyendo calidad interna (construcción e implementación del software) y calidad externa (comportamiento y aceptación del producto) [2].

Pero, pese a la existencia de elementos que permitían medir la calidad de software, existen investigaciones que planteaban metodologías que facilitaron el trayecto desde un modelo conformado por sendas medidas hacia un modelo con medidas base, que podían diversificarse en el tiempo, según el contexto y que a su vez podrían simplificar la evaluación de medidas que se desprenden de estas [3].

Callejas-Cuervo et al. desarrollaron una revisión sistemática donde se identificó la estructura de un modelo de calidad de software a nivel de: factor de calidad, criterio de calidad y métricas. Además, analizaron distinciones a nivel de calidad del proceso, calidad del producto y calidad en uso, presentando la evolución de modelos a nivel de proceso, como: ITIL, ISO/IEC 15504, Bootstrap, Dromey, Personal Software Process (PSP), Team Software Process (TSP), IEE/EIA 12207, Cobit 4.0, ISO 90003, Capability Maturity Model Integration (CMMI) e ISO/IEC 20000. Además, evaluaron la existencia de modelos a nivel de producto, como: McCall, Goal Question Metric (GQM), Boehm, FURPS, ISO 9126, Software Quality Assessment Exercise (SQAE), WebQEM e ISO25000 [4].

Martínez et al. analizaron el papel fundamental que jugaba la definición y análisis de requisitos en el éxito de los proyectos de software y de manera especial en la adecuada calidad de software; mencionaron que la flexibilidad y la adaptabilidad no estaban siendo aterrizadas de manera adecuada en las organizaciones. Es por ello que, la priorización de los requisitos y el uso de operadores de agregación de manera jerárquica permitió identificar a aquellos que fueron críticos para el éxito del proyecto y se ubicaron en las iteraciones iniciales y sucesivamente los demás requerimientos, demostrando buenos resultados a nivel de calidad de software [5].

Digital Object Identifier: (only for full papers, inserted by LACCEI).
ISSN, ISBN: (to be inserted by LACCEI).
DO NOT REMOVE

Si bien es cierto, los inicios de estudios de calidad de software se realizaron en aplicaciones transaccionales, poco a poco se trasladó el concepto a calidad de otro tipo de software; por ejemplo, Kuri et al. estudiaron aplicaciones de realidad virtual, pero aún no se tenía claro cómo los profesionales especializados en desarrollar aplicaciones de realidad aumentada podían enfrentar los desafíos de calidad del software y entregar cada vez mejores productos, por ello utilizaron criterios de calidad genéricos además de adoptar algunas de las métricas de software [6].

Pero, pese a los avances aún existe poca bibliografía que recomiende pruebas automatizadas de software y herramientas para mejorar la calidad de las aplicaciones móviles, por ello, de manera específica, con la presente investigación se busca un aporte para mejorar la calidad de software para aplicaciones móviles.

II. TRABAJOS RELACIONADOS

Perdomo y Zapata analizaron seis estados de Alfa: arquitectura seleccionada, demostrable, utilizable, listo, operativo y retirado, que representaban 30 indicadores de los 86 de la ISO/IEC 25000, donde demostraron una alta correlación con un 95% de aceptación, que permitieron garantizar un mayor nivel de calidad en los requisitos y por ende en la calidad del producto [7].

Marín, Trujillo y Buedo identificaron la importancia de gestionar actividades específicas de calidad en la disminución de costos para la corrección de defectos, cuyo objetivo se centraba en: detectar defectos, detectar defectos lo más temprano posible a la etapa donde se producen y prevenir defectos. Para ello desarrollaron una propuesta basada en el ciclo de Deming de mejora continua utilizando lista de chequeo, revisión documental, entrevistas y revisiones entre pares como técnicas o métodos para el desarrollo de actividades de calidad. Recomendaron asegurar el mayor esfuerzo a lo largo del ciclo del proyecto y no solo en las etapas de prueba [8].

Marín y Bautista evaluaron la calidad del producto bajo normas ISO/IEC 25000 en 5 tareas de un sistema transaccional de planillas, bajo arquitectura cliente - servidor, donde obtuvieron 1971.2 milisegundos como tiempo de respuesta, 2388 milisegundos como tiempo de espera, 10.93 milisegundos como tiempo de rendimiento luego de ejecutar 30 pruebas, 10.93 milisegundos promedio como rendimiento del sistema luego de ejecutar las 30 pruebas y 4316.6 Kb de uso del CPU luego de ejecutar 30 pruebas. Para ello utilizaron una entrevista semi estructurada [9].

Calanzas et al. identificaron que, al especificar los requisitos bajo lenguaje natural se incrementaba la posibilidad de baja calidad del producto software, debido a la presencia de olores de código. Analizaron para ello 26 especificaciones de un sistema de administración financiero público con 870 requisitos de calidad. Luego de la verificación, el 44% de los requisitos presentaban olores, pero de ellos el 34.6% de los requisitos incluían lenguaje subjetivo, el 22% de ellos tenían referencia incompleta y el 16% de los requisitos fueron no comprobables [10].

Idri et al. evaluaron diversos requisitos que estaban presentes en aplicaciones móviles existentes para el seguimiento del embarazo, utilizaron el estándar ISO/IEC 25030 para la evaluación de la calidad, sugiriendo los requisitos y lista de verificación (características y sub características de calidad del producto de software). Además, utilizaron la norma ISO/IEC 25010 para evaluar la calidad del producto. Obtuvieron como resultados que los requisitos vinculados con las acciones de los usuarios y las características de la aplicación tenían mayor impacto en las sub características externas del modelo de calidad del producto de software [11].

Izurieta et al. identificaron estudios respecto a la deuda técnica del software, los cuales se centraban en la detección, cuantificación y toma de decisiones, pero, sobre todo evidenciaron la utilización de herramientas de análisis estático. Y en base a ello, dada la evolución de estudios de software, recomendaron implementar mejoras en los resultados de la deuda técnica durante el modelado y así poder alcanzar un impacto positivo en la arquitectura y demás consideraciones del producto [12].

Trisnadoli et al. realizaron una encuesta a diversos usuarios y crearon un foro grupal para definir los requisitos de calidad más confiables para un juego móvil relacionado con el turismo, desde el punto de vista del usuario y la calidad que deseaba obtener. Elaboraron además un modelo de calidad para juegos en celular. Este modelo incluía los criterios y sub criterios: 1. Usabilidad en uso: efectividad en el uso, eficiencia en el uso, satisfacción; 2. Flexibilidad en el uso: accesibilidad y 3. Seguridad: seguridad del usuario [13]. Este, fue uno de los pocos modelos identificados en la revisión de literatura específica para aplicaciones móviles.

Lenarduzzi et al. investigaron un caso de estudio con 185 desarrolladores junior en dos países, que desarrollaron 23 aplicaciones en diferentes lenguajes de programación y arquitectura. Determinaron el alto nivel de aprecio de SonarQube para la reducción de limpieza de código, nunca gastando más del 50% del tiempo estimado [14].

Marcilio et al. analizaron herramientas automáticas de análisis estático de software con SonarQube, extrajeron 421,976 problemas de 246 proyectos donde encontraron una baja resolución de proyectos, en promedio 13% de los problemas. Evidenciaron que existe una tendencia central de

solucionar problemas después de 18,99 días después de haber sido informados [15].

III. ESTADO DEL ARTE

A. Calidad del producto software

La serie de estándares ISO/IEC 9126 e ISO/IEC 25000 definen la calidad de un sistema software como el grado en el que el sistema satisface las necesidades implícitas y explícitas de los usuarios. Las necesidades se presentan en SQuaRe (Software Quality Requirements) a través de: modelo de calidad del producto software, modelo de calidad de datos y modelo de calidad en uso del sistema [16].

La finalidad del modelo de calidad del producto software es especificar y evaluar la calidad de los productos de software, a través de medidas internas y externas [16]. Existen diferentes modelos que permiten medir la calidad para productos software (Tabla I):

TABLA I
Modelos de calidad de producto software

Características	Modelos					
	ISO/IEC 25000	ISO/IEC 9126	FURPS	BOEHM	GILB	SQAE
Funcionalidad	x	x	x	x	x	x
Rendimiento/ Eficiencia	x	x	x	x		x
Compatibilidad	x					x
Fiabilidad	x	x	x	x	x	x
Usabilidad	x	x	x	x	x	
Seguridad	x					x
Mantenibilidad	x	x	x	x	x	x
Portabilidad	x	x		x		x
Calidad de uso		x				

A partir de la elaboración de la primera tabla comparativa (Tabla I), el modelo de calidad ISO/IEC 25000 ha actualizado e incorporado mayor número de características de calidad de software, es por ello que en la presente investigación se seleccionó este modelo.

Frente a la diversidad de características y sub características de la ISO 25000, se investigó la existencia de herramientas que permitían medirlas de manera automatizada (Tabla II), las cuales soportan lenguajes de programación específicos (Tabla III).

TABLA II
Herramientas que permiten medir calidad de producto software

Características	Herramientas de calidad			
	SonarQube	Designite	Ndepend	SoapUI
Fiabilidad	x			
Seguridad	x			
Mantenibilidad	x	x		
Calidad de uso		x	x	x

TABLA III
Herramientas de calidad y lenguajes de programación soportados

Lenguajes de programación	Herramientas de calidad			
	SonarQube	Designite	Ndepend	SoapUI
Java	x			
C++	x			
Kotlin	x			
C#		x	x	x
.Net			x	x

B. Aplicaciones móviles (App móvil)

Sanz y Galán las definen como un tipo de aplicación diseñada para ejecutarse en un dispositivo móvil, que en la mayoría de las veces cuenta con funcionalidades específicas y limitadas [17].

Frente a lo analizado, el objetivo de la investigación fue determinar el uso de herramientas automatizadas para evaluar la calidad de aplicaciones móviles en base al estándar ISO/IEC 25000, teniendo en cuenta las dimensiones de: mantenibilidad, fiabilidad, seguridad, rendimiento o eficiencia, funcionalidad, compatibilidad, usabilidad y portabilidad.

IV. MATERIALES Y MÉTODOS

La presente investigación fue descriptiva con estudio de caso: producto software para alimentación de animales domésticos, desarrollado en lenguaje Java, bajo el IDE Android Studio.

Según Guerrero y Guerrero, una investigación descriptiva permite describir contextos; esto es, detallar su funcionamiento y comportamiento. Con los estudios descriptivos se busca especificar los procesos y objetos sometidos a análisis [18].

Para la evaluación de la calidad del producto se evaluaron 7 requerimientos: (1) registro de mascota, (2) información de mascota, (3) listado de mascotas, (4) eliminar mascota, (5) registro de vacunas, (6) información médica y (7) vincular dispensador, presentes en los 4 módulos del sistema: mascota, dispensador, médico y vacunas; que fueron representados en

20 historias de usuario (Fig. 1) definidas en la etapa de análisis de la metodología Kanban utilizada. A nivel de código, se analizaron un total de 1535 líneas de código, en un período de 60 días, en función a la distribución de Sprints.

Número: 1	
Nombre historia: Detectar mascota	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Baja
Puntos asignados: 10	**Iteración asignada:** 1
Responsable:	
Descripción: Función que permite detectar a la mascota cuando se acerca al dispensador, haciendo que se el sensor infrarrojo se active.	
Validación: Se requiere se active el sensor para iniciar el proceso. Si no se activa: No hay mascota Si se activa: Si hay mascota	

Fig. 1 Ejemplo de historia de usuario: Detectar mascota

Para la evaluación de la calidad automatizada del producto se utilizaron las herramientas de software libre online SonarQube y Firebase, teniendo en cuenta 2 momentos de evaluación, la primera evaluación de acuerdo con el desarrollo del producto software original y la segunda evaluación luego de las mejoras teniendo en cuenta como referente la primera evaluación.

V. RESULTADOS

En los resultados, la evaluación de la dimensión de calidad en mantenibilidad con SonarQube (Tabla IV), muestran que tanto en la primera como segunda evaluación, se obtuvo una clasificación de muy buena; de igual manera, luego de la primera evaluación se mejoró la mantenibilidad disminuyendo significativamente la capacidad de condensación, los olores de código, la deuda técnica, la densidad del código repetido y la complejidad ciclomática, sin embargo que aún un muy alto riesgo en la capacidad de modificación.

TABLA IV
Evaluación de mantenibilidad con SonarQube

Sub características	**Evaluación 1**		**Evaluación 2**	
	Medición	**Resultado**	**Medición**	**Resultado**
Modularidad				
Capacidad de condensación	0.13	Muy buena	0.02	Muy buena
Olores de código	207		22	
Reusabilidad				
Deuda técnica	2.3% 2d 1h	A	1.8% 3h 4min	A
Capacidad de análisis				
Capacidad de pistas de auditoría	1	Muy buena	1	Muy buena
Densidad del código repetido	7%		0%	
Capacidad de modificación				
Complejidad ciclomática	184	Muy alto riesgo	176	Muy alto riesgo
Cobertura en líneas	0%		0%	

En la evaluación de la dimensión de calidad en fiabilidad con SonarQube (Tabla V), se observa que en la primera evaluación se obtuvo una mala evaluación de la confiabilidad en la tolerancia a fallos que fue mejorada de la clasificación C a la clasificación A de muy buena para la segunda medición.

TABLA V
Evaluación de fiabilidad con SonarQube

Sub características	**Evaluación 1**		**Evaluación 2**	
	Medición	**Resultado**	**Medición**	**Resultado**
Tolerancia a fallos				
Confiabilidad	C	Mala	A	Muy buena

Los resultados de la evaluación de la dimensión de calidad en rendimiento/eficiencia con Firebase (Tabla VI), para el tiempo de inicio de la aplicación, tienen una evaluación a mejorar, tanto en la primera como en la segunda evaluación, ya que se mantiene los 761 ms en la visualización de la primera pantalla y los 5min 6 s de visualización completa. En las estadísticas gráficas si hubo un cambio significativo mejorando las sincronizaciones verticales perdidas a 0%, la latencia de entrada alta a 75%, la mejora de los subprocesos de la IU lento, comandos de dibujo lentos y la carga de mapas de bit lentas. Finalmente, la utilización de recursos también se mejoró quedando luego de la segunda evaluación como favorable.

TABLA VI
Evaluación de rendimiento/eficiencia con Firebase

Sub características	**Evaluación 1**		**Evaluación 2**	
	Medición	**Resultado**	**Medición**	**Resultado**
Capacidad				
Tiempo de inicio de la aplicación	761 ms visualiz. 1° pantalla 5 min 6s visualiz. completa	A mejorar	761 ms visualiz. 1° pantalla 5min 6s visualiz. completa	A mejorar
Estadísticas gráficas				
Sincronizaciones verticales perdidas	14%	A mejorar	0%	Favorable
Latencia de entrada alta	0%		75%	
Subproceso de la IU lento	32%		2%	
Comandos de dibujo lentos	38%		0%	
Cargas de mapas de bits lentas	2%		0%	
Utilización de recursos				
CPU	24%	A mejorar	31%	Favorable
Memoria	288 KiB		61 KiB	
Internet	694 bytes/s		24 bytes/s	

En la evaluación de la calidad de la funcionalidad con Firebase (Tabla VII) se obtuvo que el producto software es

funcional teniendo en cuenta el acceso individual, el registro de mascota y el dispensador, tanto en la primera como en la segunda evaluación.

TABLA VII
Evaluación de funcionalidad con Firebase

Sub características	Evaluación 1		Evaluación 2	
	Medición	Resultado	Medición	Resultado
Cumplimiento de los requisitos				
Acceso individual (login)	Prueba de funcionalidad automática	Funcional	Prueba de funcionalidad automática	Funcional
Registro de mascota				
Dispensador				

La evaluación de la compatibilidad como dimensión de calidad utilizando Firebase y Android Studio (Tabla VIII) muestra una compatibilidad e interoperabilidad óptimas tanto en la primera como en la segunda evaluación, teniendo en cuenta las versiones aceptadas y la comunicación entre la base de datos y la aplicación.

TABLA VIII
Evaluación de compatibilidad con Firebase y Android Studio

Sub características	Evaluación 1		Evaluación 2	
	Medición	Resultado	Medición	Resultado
Coexistencia				
Mínima versión aceptada	5.0 (Lollipop)	Compatible	5.0 (Lollipop)	Compatible
Máxima versión aceptada	12.0 (S)		12.0 (S)	
Interoperabilidad				
BD Firebase y Android Studio	Interoperabilidad		Interoperabilidad	

En la dimensión de calidad de usabilidad con Firebase (Tabla IX), los resultados muestran que se han corregido errores significativamente los resultados de calidad, disminuyendo los problemas de accesibilidad, pero quedando aún pendiente por mejorar el tamaño de objetivos táctiles, el contraste bajo, las etiquetas de contenido y la implementación.

TABLA IX
Evaluación de usabilidad con Firebase

Sub características	Evaluación 1		Evaluación 2	
	Medición	Resultado	Medición	Resultado
Accesibilidad				
Problemas	38	A mejorar	15	A mejorar
Tamaño de objetivos táctiles	1 advertencia 2 problemas menores 0 sugerencias		0 advertencias 1 problema menor 0 sugerencias	
Contraste bajo	0 advertencias 25 problemas menores 0 sugerencias		0 advertencias 9 problemas menores 0 sugerencias	
Etiquetas de contenido	6 advertencias 1 problema menor 2 sugerencias		1 advertencia 1 problema menor 2 sugerencias	
Implementación	1 advertencia 0 problemas menores 0 sugerencias		1 advertencia 0 problemas menores 0 sugerencias	

Finalmente, para la evaluación de la portabilidad como dimensión de calidad con Firebase y Android Studio (Tabla X), se obtuvo tanto en la primera como en la segunda evaluación, que el producto tiene una adaptabilidad buena, siendo portable. Asimismo, en instalación se mejoró la facilidad de instalación disminuyendo la dificultad y la baja portabilidad en la primera evaluación.

TABLA X
Evaluación de portabilidad con Firebase y Android Studio

Sub características	Evaluación 1		Evaluación 2	
	Medición	Resultado	Medición	Resultado
Adaptabilidad				
Firebase	Reutilización en Android, iOS, Web, C++ y Unity	Buena y portable	Reutilización en Android, iOS, Web, C++ y Unity	Buena y portable
Facilidad de instalación				
Instalación	Difícil, portabilidad baja		Mayor facilidad de instalación con portabilidad	

VI. DISCUSIÓN

Al igual que Marín, Trujillo y Buedo [19] que desarrollaron una propuesta basada en el ciclo de Deming para detectar defectos y mejorar la calidad de software, en la presente investigación se utilizó la metodología Kanban que incluye proceso de mejora continua en el ciclo de desarrollo de producto software y no solo en el proceso de pruebas.

Para la evaluación de olores de código, se utilizó SonarQube como herramienta automatizada, logrando disminuir el valor de 207 a 22. Respecto a la deuda técnica se redujo de 2.3% (2 días 1 hora) a 1.8% (3h 4min), valores significativos, en correspondencia a lo afirmado por Lenarduzzi et al. [14].

Se redujo la densidad del código repetido de 7% a 0%, resultado que debiera relacionarse con la densidad de complejidad ciclomática, que en el caso estudiado solo se redujo de 184 a 176, manteniendo un nivel muy alto de riesgo para la capacidad de modificación [20].

En la evaluación de fiabilidad con SonarQube se pasó de nivel de confiabilidad de C a A, valor significativo que guarda coherencia con lo mencionado por Marcilio et al. que mencionaron que las organizaciones con proyectos de software que utilizaron SonarQube resolvieron los problemas en los sistemas en un 13% [15].

Para la evaluación automatizada del rendimiento/ eficiencia se utilizó la herramienta Firebase, evidenciándose cambios en las sub características de sincronizaciones verticales perdidas de 14% a 0%, subproceso de la IU lento de 32% a 2%, comando de dibujos lentos de 38% a 0% y carga de mapas de bits lentas de 2% a 0%. Para la evaluación de utilización de recursos se redujeron resultados en memoria de 288 KiB a 61 KiB y uso de internet de 694 bytes a 24 bytes.

Para la evaluación automatizada de la funcionalidad se utilizó Firebase, pero previo a ello se corrigieron las historias de usuario, disminuyendo frases subjetivas que impactaban en la calidad, tal como lo mencionan Calanzas et al. [10].

Para la evaluación de compatibilidad se utilizó Firebase y Android Studio, encontrándose valores de compatibilidad e interoperabilidad.

Para la evaluación de usabilidad, de manera automatizada se utilizó Firebase, disminuyendo de 38 a 15 los problemas, de 1 advertencia a 0 advertencias, de 2 problemas menores a 1 problema menor y 0 sugerencias; a nivel de etiquetas de contenido se disminuyeron de 6 advertencias a 1.

Para la evaluación de la portabilidad se utilizaron las herramientas Firebase y Android Studio; en ambos momentos la aplicación mantuvo una adaptabilidad buena demostrando ser portable y una mayor facilidad de instalación.

VII. CONCLUSIONES

El uso de la herramienta SonarQube permitió evaluar las dimensiones de calidad en mantenibilidad y fiabilidad, mientras que con la herramienta Firebase se logró evaluar las dimensiones de calidad en rendimiento/eficiencia, funcionalidad, compatibilidad, usabilidad y portabilidad, utilizando la interacción en algunas evaluaciones Android Studio.

La calidad de mantenibilidad luego de las dos evaluaciones alcanzó una clasificación de A, muy buena, dejando pendiente la mejora de la capacidad de modificación que permaneció en muy alto riesgo.

La calidad de la fiabilidad fue evaluada como muy buena, con una clasificación final de A. La calidad del rendimiento/eficiencia obtuvo una evaluación por mejorar en el tiempo de inicio de la aplicación, sin embargo, en las estadísticas gráficas se logró mejorar hacia un resultado favorable.

La calidad de la funcionalidad fue determinada como funcional; la calidad de la compatibilidad se evaluó como compatible en la coexistencia y de interoperabilidad, en ambas evaluaciones.

La calidad de la usabilidad se debe mejorar aún, por los aspectos de accesibilidad, y la calidad de la portabilidad se mejoró a una adaptabilidad buena, con facilidad de instalación y portabilidad.

VIII. RECOMENDACIONES

Es necesario ampliar y mejorar las herramientas de evaluación de calidad de software para que ayuden en el desarrollo de aplicaciones móviles, buscando cada vez mejores productos.

REFERENCIAS

[1] "ISO 25010." https://iso25000.com/index.php/normas-iso-25000/iso-25010.

[2] G. Pantaleo, "La máquina de hacer software," Jan. 2016, [Online]. Available: https://bibliotecaupn.elogim.com/auth-meta/login.php?url=https://search.ebscohost.com/login.aspx?direct=true&AuthType=ip,uid&db=edsbas&AN=edsbas.96918E8E&lang=es&site=eds-live

[3] D. St-Louis and W. Suryn, "Enhancing ISO/IEC 25021 quality measure elements for wider application within ISO 25000 series," in *IECON 2012 - 38th Annual Conference on IEEE Industrial Electronics Society*, Oct. 2012, pp. 3120–3125. doi: 10.1109/IECON.2012.6389400.

[4] M. Callejas-Cuervo, A. C. Alarcón-Aldana, and A. M. Álvarez-Carreño, "Modelos de calidad del software, un estado del arte," *Entramado*, vol. 13, no. 1, pp. 236–250, 2017, Available: https://www.redalyc.org/journal/2654/265452747018/html/

[5] M. Á. Q. Martínez, O. A. J. Antón, D. H. P. Moran, and M. Y. L. Vazquez, "Priorización de requisitos para una adecuada calidad de software," *Ser. Científica Univ. Las Cienc. Informáticas*, vol. 13, no. 6, Art. no. 6, May 2020, [Online]. Available: https://publicaciones.uci.cu/index.php/serie/article/view/597

[6] M. Kuri, S. A. Karre, and Y. R. Reddy, "Understanding Software Quality Metrics for Virtual Reality Products - A Mapping Study," in *14th Innovations in Software Engineering Conference (formerly known as India Software Engineering Conference)*, New York, NY, USA, Feb. 2021, pp. 1–11. doi: 10.1145/3452383.3452391.

[7] W. Perdomo and C. M. Zapata, "Software quality measures and their relationship with the states of the software system alpha: Medidas de la calidad del producto de software y su relación con los estados del alfa sistema de software.," *INGENIARE - Rev. Chil. Ing.*, vol. 29, no. 2, pp. 346–363, Apr. 2021, [Online]. Available: https://bibliotecaupn.elogim.com/auth-meta/login.php?url=https://ebsco.bibliotecaupn.elogim.com/login.aspx?direct=true&AuthType=ip,uid&db=a9h&AN=150919812&lang=es&site=eds-live

[8] A. Marin Diaz, Y. Trujillo Casañola, D. Buedo Hidalgo, "Apuntes para gestionar actividades de calidad en proyectos de desarrollo de software para disminuir los costos de corrección de defectos," *Ingeniare Rev. Chil. Ing.*, vol. 27, no. 2, pp. 319–327, Apr. 2019, doi: 10.4067/S0718-33052019000200319.

[9] E. R. Marín C. and J. J. Bautista G., "Evaluación de la calidad de producto de software bajo normas ISO/IEC 25000: Caso de estudio sistema de planillas de la Municipalidad Provincial de Chiclayo," 2021. [Online]. Available: https://repositorio.uss.edu.pe/bitstream/handle/20.500.12802/8086/Marin%20Chaman%20Edson%20%26%20Bautista%20Guti%c3%a9rrez%20Juan_.pdf?sequence=6&isAllowed=y

[10] A. T. S. Calazans *et al.*, "Quality Requirements and the Requirements Quality: The indications from Requirements Smells in a Financial Institution Systems," in *Proceedings of the XXXIII Brazilian Symposium on Software Engineering*, New York, NY, USA, Sep. 2019, pp. 472–480. doi: 10.1145/3350768.3350782.

[11] A. Idri, M. Bachiri, and J. L. Fernández-Alemán, "A Framework for Evaluating the Software Product Quality of Pregnancy Monitoring

Mobile Personal Health Records," *J. Med. Syst.*, vol. 40, no. 3, p. 50, Dec. 2015, doi: 10.1007/s10916-015-0415-z.
[12] C. Izurieta, G. Rojas, and I. Griffith, "Preemptive Management of Model Driven Technical Debt for Improving Software Quality," in *Proceedings of the 11th International ACM SIGSOFT Conference on Quality of Software Architectures*, New York, NY, USA, May 2015, pp. 31–36. doi: 10.1145/2737182.2737193.
[13] A. Trisnadoli, I. Muslim, and W. Novayani, "Software Quality Requirement Analysis on Educational Mobile Game with Tourism Theme," *J Softw*, 2016, doi: 10.17706/jsw.11.12.1250-1257.
[14] V. Lenarduzzi, V. Mandić, A. Katin, and D. Taibi, "How long do Junior Developers take to Remove Technical Debt Items?," in *Proceedings of the 14th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, New York, NY, USA, Oct. 2020, pp. 1–6. doi: 10.1145/3382494.3422169.
[15] D. Marcilio, R. Bonifácio, E. Monteiro, E. Canedo, W. Luz, and G. Pinto, "Are static analysis violations really fixed? a closer look at realistic usage of SonarQube," in *Proceedings of the 27th International Conference on Program Comprehension*, Montreal, Quebec, Canada, May 2019, pp. 209–219. doi: 10.1109/ICPC.2019.00040.
[16] C. Calero, C. C. Muñoz, and M. G. P. Velthuis, *Calidad del producto y proceso software*. Editorial Ra-Ma, 2010.
[17] R. L. Sanz and R. Galán López, *Introducción a la movilidad: 4G/LTE y el desarrollo de aplicaciones Android*. Dextra Editorial, 2014. [Online]. Available: https://elibro.bibliotecaupn.elogim.com/es/ereader/upnorte/43939
[18] G. Guerrero and C. Guerrero, *Metodología de la investigación*. Grupo Editorial Patria, 2015. [Online]. Available: https://elibro.bibliotecaupn.elogim.com/es/ereader/upnorte/40363
[19] "Apuntes para gestionar actividades de calidad en proyectos de desarrollo de...: EBSCOhost.", [Online]. Available: https://pwebebsco.bibliotecaupn.elogim.com/ehost/pdfviewer/pdfviewer?vid=1&sid=99239844-b21a-4916-8053-dd0fc6e17f72%40redis
[20] Colexio Profesional de Enxeñaría en Informática de Galicia, "I Jornada sobre calidad del producto software e ISO 25000, Santiago de Compostela, 10 de junio de 2014". [Online]. Available: https://www.cpeig.gal/sites/default/files/libros/Libro%2BJornadas%2BGalicia%2BCalidad%2BSoftware.pdf