Universidad del Azuay

Facultad de Ciencias de la Administración

Carrera de Ingeniería de Sistemas y Telemática

MÉTODO DE EVALUACIÓN DE CALIDAD DE HERRAMIENTAS DE DESARROLLO DE SOFTWARE LIBRE PARA CONSTRUIR ERP

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL GRADO EN INGENIERO EN SISTEMAS Y TELEMÁTICA

Autor:

Luis Fernando Guerrero Ortega

Director:

Ing. Catalina Verónica Astudillo Rodríguez

Codirector:

Ing. Paúl Esteban Crespo Martínez

Cuenca – Ecuador

2022

DEDICATORIA

Quiero dedicar este trabajo de titulación a Dios,
por brindarme la vida, salud y trabajo,
pero sobre todo por darme fuerzas
para sobrellevar cada situación vivida.

A mis padres, Walter y Lilia por su esfuerzo
y apoyo incondicional en todo momento,
por formarme con principios y valores
que me han sido de mucha ayuda
para cumplir con esta meta.

AGRADECIMIENTO

Agradezco a mi directora del presente trabajo de titulación, Catalina Astudillo, por su guía y confianza en todo momento.

También agradecer a mis amigos y compañeros, por cada uno de los momentos vividos a lo largo de este caminar.

ÍNDICE

Índice de Figuras

Índice de tablas

Índice de Anexos

RESUMEN

La industria del software evidencia un gran impulso a causa de la pandemia por el COVID-19, esto se demuestra en la creciente demanda de herramientas de desarrollo de software libre. Asegurarse que la calidad de la herramienta seleccionada por el desarrollador cumpla con sus requerimientos es fundamental, sin embargo, pese a que existen métodos de calidad, estos no se orientan en evaluar herramientas de desarrollo. El presente trabajo, apoyado en el modelo de investigación cuantitativo de Hernández, analiza los estándares de calidad McCall, Boehm, FURPS, Dromey, ISO 25010, EFFORT, Bertoa, Rawashdech, CBQM, Modelo Holístico y Modelo COCA. Como resultado, propone un nuevo método para evaluar la calidad de herramientas de desarrollo de software libre. Finalmente, evalúa la efectividad del método propuesto en un escenario de pruebas, que compara los frameworks Django, .NET Core y Laravel identificando la herramienta que cumple con los requerimientos de calidad para la construcción del sistema UDA-ERP.

Palabras clave: ERP, software libre, métodos de calidad, calidad de software, frameworks.

ABSTRACT

Software industry shows a great impulse due to the COVID-19 pandemic. This is demonstrated in the growing demand for free software development tools. Ensuring that the quality of the tool selected by the developer meets the requirements is fundamental. However, despite the existence of quality methods, they are not oriented to evaluate development tools. This paper, supported by Hernandez's quantitative research model, analyzes the quality standards McCall, Boehm, FURPS, Dromey, ISO 25010, EFFORT, Bertoa, Rawashdech, CBQM, Holistic Model and COCA Model. As a result, ithe study offers a new method to evaluate the quality of open source software development tools. Finally, it evaluates the effectiveness of the proposed method in a testing scenario, which compares Django, .NET Core and Laravel frameworks. The tool that meets the quality requirements for building the UDA-ERP system was identified.

Keywords: ERP, open source, quality methods, software quality, frameworks.

Translated by

Luis Fernando Guerrero

Introducción

Los modelos de calidad de software son una forma estandarizada de medir la capacidad de un producto de software para realizar las funciones esperadas de manera segura y sin fallas. Con la creciente tendencia en la industria del software, las empresas buscan el mecanismo idóneo que garantice la calidad de un producto, lo que conlleva a que estas empresas indaguen en la forma correcta de planificar y desarrollar programas que cumplan con los estándares esperados, y apliquen modelos de calidad que se adapten a sus necesidades.

La importancia al momento de desarrollar un producto de software hace que los arquitectos de software se apoyen en herramientas con el fin de ahorrar tiempo y esfuerzo. Sin embargo, en ocasiones, los desarrolladores pasan por alto la evaluación de calidad, lo que provoca posibles dificultades durante el proceso de construcción del software, llegando al punto de cambiar o abandonar una herramienta. Esto se puede evitar, al contar con un modelo de calidad orientado en analizar la calidad de éstas herramientas.

Durante los últimos años, se han creado varios modelos de calidad de software que están basados en modelos estándar (McCall, Boehm, FURPS, Dromey, ISO 25010) y que se adecúan a una necesidad específica, sin embargo, no existe un modelo orientado a analizar la calidad de herramientas de desarrollo de software.

Así también, en la actualidad se fomenta una cultura de software propietario por ser catalogado por la industria como poderoso y confiable, sin embargo, no respeta la libertad de los usuarios. El software es útil para sus usuarios solo si respeta su libertad, pero se sabe que los propietarios limitan las posibilidades al usuario de modificarlo e incluso llegan a restringir su uso. Por otro lado, el software libre se convierte en una puerta de entrada para que los usuarios exploren y aprendan del mismo, proporciona total libertad, auditabilidad, seguridad, sin monopolios y con un gran movimiento social.

El propósito de este trabajo de titulación es la definición de un método de calidad de software para herramientas de desarrollo de software libre. El método propuesto, se evalúa en un escenario de pruebas, el cual servirá como un análisis de soporte de las posibles herramientas de software libre a ser empleadas en la construcción del software UDA-ERP.

El presente trabajo está conformado por seis capítulos: el capítulo 1 explora los conceptos de software libre, software propietario y software ERP, así también, se analizan las herramientas de desarrollo de software libre, por último, se profundiza sobre los estándares de evaluación de calidad de software.

En el capítulo 2 se realiza un análisis de cada uno de los métodos de evaluación de calidad de software y se establece una comparación entre sus características y subcaracterísticas.

El capítulo 3 se propone el método de evaluación de calidad. Incluye la selección de las métricas de calidad que forman parte del método propuesto.

En el capítulo 4 se evalúa el método de calidad de software propuesto mediante un escenario de pruebas. El escenario se orienta a la selección del framework idóneo para el desarrollo del software UDA-ERP con herramientas de software libre.

1. Marco Teórico y Estado del Arte

En este apartado, se indaga entre las definiciones de software libre y privativo, determinando su enfoque, ventajas y desventajas, y su impacto en el desarrollo del software empresarial. Se analiza el software UDA-ERP de la Universidad del Azuay y su aporte a las MIPYMES. Así mismo, se analiza el enfoque de las herramientas de desarrollo de software libre y su aportación a la comunidad de desarrolladores.

Por último, se presentan los estándares de evaluación de calidad de software, modelos basados en estos estándares, un modelo holístico y un modelo de documentación.

1.1. Software libre

Según Anand (Anand, Krishna, Tiwari, & Sharma, 2018) el término software libre hace referencia al software que permite su uso, modificación y distribución de manera libre, con la única restricción de que cualquier versión reformada tenga la capacidad de difundirse con los términos originales de uso. Así mismo, (Stallman, 1986), define software libre como la libertad que tienen los usuarios para ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el software, siguiendo los cuatro tipos de libertades mencionados a continuación:

- Libertad 0: Libertad para ejecutar el programa para cualquier propósito.
- Libertad 1: Libertad para estudiar el programa y cambiarlo como se desee.
- Libertad 2: Libertad para redistribuir copias que ayuden a otros.
- Libertad 3: Libertad para distribuir copias de versiones modificadas y beneficiarse de los aportes de la comunidad.

Generalmente, el software de código abierto suele confundirse con software de prueba, gratuito, compartido o de archivos binarios de libre acceso. La diferencia radica en que, el software libre es completamente gratuito para que los desarrolladores lo empleen, en contraste, el software de prueba, el usuario puede experimentar con el software antes de decidir comprarlo, es decir, debe efectuar el pago por la licencia del software una vez finalizado el periodo de prueba (Dhir & Dhir, 2017).

1.2. Software de código abierto (OSS)

Levine & Prietula (2013) definen el software de código abierto como cualquier sistema de innovación o producción basado en colaboradores que interactúan para crear un servicio o producto de valor económico al alcance de contribuyentes y no contribuyentes por igual. El término código abierto se origina tiempo después de que la empresa Netscape compartiera el código fuente de su navegador web al mundo (Dreyfus, 1998). El surgimiento de esta propuesta, visualiza un nuevo mecanismo de interacción con los desarrolladores de software. Este enfoque se lo identifica como *código abierto* para que se diferencie del término *software libre* (Open Source Initiative, s.f.).

El desarrollo de OSS tiene como objetivos: apoyar a la difusión de tecnologías emergentes, otorgar a los usuarios software gratuito y a los desarrolladores la posibilidad de incorporar código fuente de terceros en sus implementaciones. La variedad de bibliotecas disponibles y evaluadas por la comunidad de desarrolladores de OSS, con frecuencia son incorporadas en otros sistemas, aportando en la consecución de un software funcional, reutilizable y un tiempo de lanzamiento más corto (Franco-Bedoya, Ameller, Costal, & Franch, 2017).

El enfoque de productos y/o proyectos basados en OSS ha revolucionado el campo de desarrollo de software. Las organizaciones que gestionan una cantidad considerada grande de datos, definen un paradigma para desarrollar el software apoyados por una comunidad, de modo que esta comunidad valide la calidad del software (Tyagi, Kumar, & Kumar, 2019).

1.3. Software propietario (PS)

El software privativo o propietario, también conocido como software comercial, cuenta con derechos de autor e incluye limitaciones para su uso, distribución o modificación según lo especificado por su desarrollador o editor (Dhir & Dhir, 2017).

El software propietario está controlado por licencias que se aplican a través de medidas legales, restringiendo a los usuarios la posibilidad de manipular el código, por este motivo, los distribuidores de PS no comparten código fuente, ya que al hacerlo están incumpliendo sus condiciones (Rappaport, 2018).

1.4. Software libre vs software propietario

Con la llegada de la ingeniería de software, investigadores y profesionales han batallado para hacer frente a los constantes cambios de requisitos en el software. Esta evolución plantea varios desafíos de investigación, especialmente en sistemas grandes y complejos (Talai & Bouras, 2017).

Camilo (2018) señala que la mayoría de estudios sobre la evolución del software, en cuestión de objetivos, procesos, economía e incluso las políticas de desarrollo, incluyen sistemas de código abierto. Esto se debe, a que los estudios de software patentado a menudo están sujetos a restricciones que limitan la cantidad de componentes que se pueden compartir.

Es necesario aclarar que software libre no precisamente significa que sea gratuito, el software libre puede ser comercial, así también, el software gratuito no siempre es código abierto. Lo mismo ocurre con el software propietario que puede incluir el código fuente, y esto no significa que es software libre (Gomez-Diaz, 2015).

El software propietario puede incluir software gratuito y software de dominio público (sin derechos de autor), esto se debe, a que el software gratuito está disponible sin costo para todos y puede ser utilizado por cualquier persona, para cualquier propósito sin restricciones (Anand et al., 2018).

La tabla 1 presenta el análisis de diversos factores y establece una comparativa entre el software de código abierto y el software propietario, de acuerdo a lo expuesto por los estudios de (Singh, Bansal, & Jha, 2015; Sirshar, Ali, & Ibrahim, 2019).

Tabla 1 Software de código abierto y software propietario en términos de complejidad y calidad

Factor	Software de código abierto	Software propietario
Costo	Costo menor o costo cero. No requiere de hardware o software costoso para el desarrollo.	El costo varía según la complejidad del software.
Seguridad	El desarrollo de código abierto no garantiza que el software esté libre de vulnerabilidades, sin embargo, permite a la comunidad de desarrolladores y usuarios, encontrarlas y solucionarlas.	Se considera seguro porque se desarrolla en un entorno controlado y el código es desarrollado por un solo individuo o equipo, lo que minimiza errores en el software y el riesgo de piratería.
Usabilidad	El software de código abierto está bien documentado.	Documentación y manuales de usuario detallados para una mejor comprensión de los desarrolladores de software.
Fiabilidad	Está disponible en varios sitios web lo que facilita los cambios en el código fuente, involucra a toda la comunidad para identificar y corregir posibles fallas.	Es desarrollado por especialistas, una vez terminado el software no se realizan modificaciones sin autorización. Las fallas se limitan a las detectadas por la organización.

Innovación	Proporciona a los desarrolladores el código fuente, lo que conlleva flexibilidad y facilita la innovación del software.	El acceso al código fuente es restringido, por tanto, no es posible modificar o realizar innovaciones en el software.
Transparencia	La estructura interna del software es visible para cualquier usuario.	El usuario no tiene acceso a los detalles internos, o de cualquier otro tipo del software usado en el desarrollo.
Estándares	Implementan estándares abiertos que puede ser revisados y/o modificados por los desarrolladores.	Los estándares son privados, solo pueden ser revisados y/o modificados por la organización.
Disponibilidad	Totalmente disponible de modo gratuito en la red.	Disponibles para la empresa que desarrolla el software. De modo gratuito, algunas empresas proporcionan demos por corto tiempo.
Soporte	Depende de la comunidad para enviar y recibir soporte a través de foros. Algunos proveedores dan soporte especializado con un costo adicional por el servicio.	Los proveedores del software ofrecen un soporte continuo a sus usuarios.
Licencias de Software	Mediante licencias Copy left obliga a que todas las adaptaciones y extensiones que se realicen al software también sean gratuitas.	Al ser el software propietario comercial implica un pago por la licencia.
Flexibilidad	A medida que las organizaciones buscan continuamente obtener más beneficios con menos esfuerzo, adoptan software de código abierto, el cual proporciona un valor comercial real, menores costos de TI y mayores oportunidades de innovación.	Al utilizar software patentado como Windows y Office, se requiere actualizar tanto el software como el hardware. Las actualizaciones deben instalarse para que funcionen correctamente.

Fuente: (Sirshar et al., 2019).

Las empresas de software propietario son reacias a compartir su código fuente al público, debido a que la venta de software es su principal fuente de ingresos. En contraparte, debido a que el costo de software libre es cero y la colaboración en la comunidad de desarrolladores es permanente, las empresas de software están aprovechando al máximo el paradigma del código abierto para los desarrollos comerciales. Es así, que los proyectos de software propietario evolucionan hacia el software de código abierto, siendo de alguna manera dominado por la filosofía de Open Source (Subhani, 2020).

1.5. Software ERP

La Planificación de Recursos Empresariales (ERP) se define como un tipo de software empresarial que permite la integración completa de la información de todas las áreas funcionales de una empresa, mediante una base de datos y, es accesible mediante una interfaz unificada y amigable (Addo-Tenkorang & Helo, 2011).

Hoy en día, el software ERP es considerado como una herramienta integral para toda empresa que requiera automatizar sus operaciones, y así, proporcionar información

exhaustiva a todos los niveles de la organización. Sin embargo, el alto costo de la integración de un ERP, el mantenimiento y la capacitación del personal, sumado al costo de las herramientas de desarrollo, son algunas de las razones por las que una empresa desisten de la idea de implementar este tipo de software, lo citado, ha ocasionado que empresas como TOYOTA se inclinen hacia OSS en busca de soluciones adecuadas, que reduzcan costos y que respalden sus operaciones (Kountouridou, Antoniou, & Stamelos, 2016).Es así, que en la oferta de sistemas ERP, se encuentran los llamados también ERP de software libre, sin embargo, cabe mencionar, que pese a ser catalogado para uso gratuito, tiene costes a considerar relacionados con: instalación, mantenimiento y aprendizaje (Economides & Katsamakas, 2006).

1.5.1. Software UDA-ERP

En el año 2015, como parte de un proyecto de investigación en la Universidad del Azuay – UDA, se propuso el desarrollo e implementación del Software UDA-ERP, el proyecto tiene como objetivo la vinculación entre la academia y la sociedad al proporcionar una solución informática que apoye en la gestión de micro, pequeñas y medianas empresas ecuatorianas, así mismo, su costo es accesible para este sector (Astudillo-Rodríguez, Crespo-Martínez, & Andrade-Dueñas, 2018).

1.6. Herramientas de desarrollo de software libre

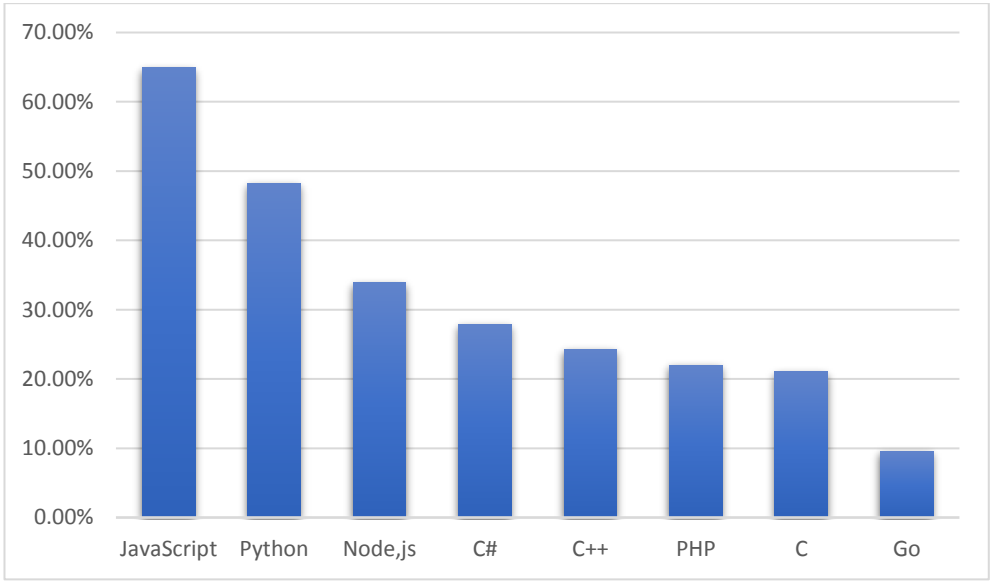
Como se muestra en la sección 1.4, la principal diferencia entre software libre y propietario es la libertad que tiene el usuario para disponer abiertamente del código fuente y realizar cambios, por lo tanto, las herramientas de desarrollo de software libre son un conjunto de librerías y componentes de libre acceso que facilitan el desarrollo de tareas de programación. Integran una serie de procedimientos implementados de forma nativa, obteniendo como resultado una arquitectura coherente, un código ligero y optimizado, lo que facilita el mantenimiento y su mejora continua (Yamami et al., 2019).

En el mercado del software se encuentran diversos lenguajes de programación que cuentan con frameworks para facilitar tanto el trabajo de desarrolladores independientes, como de empresas de desarrollo de software.

Algunas organizaciones han aplicado encuestas a desarrolladores de software, sobre las tecnologías de desarrollo que utilizan con mayor frecuencia, en el mismo sentido, otros estudios analizan frameworks de desarrollo web. Estos estudios aportan una visión clara sobre la diversidad de herramientas en este ámbito.

En la figura 1 se observa los resultados de la encuesta realizada por StackOverFlow (2021) a 80.000 desarrolladores, en la cual se detalla los lenguajes de programación más populares, sobresaliendo por noveno año consecutivo JavaScript con un 64,96%, seguido por tecnologías como Python y C#.

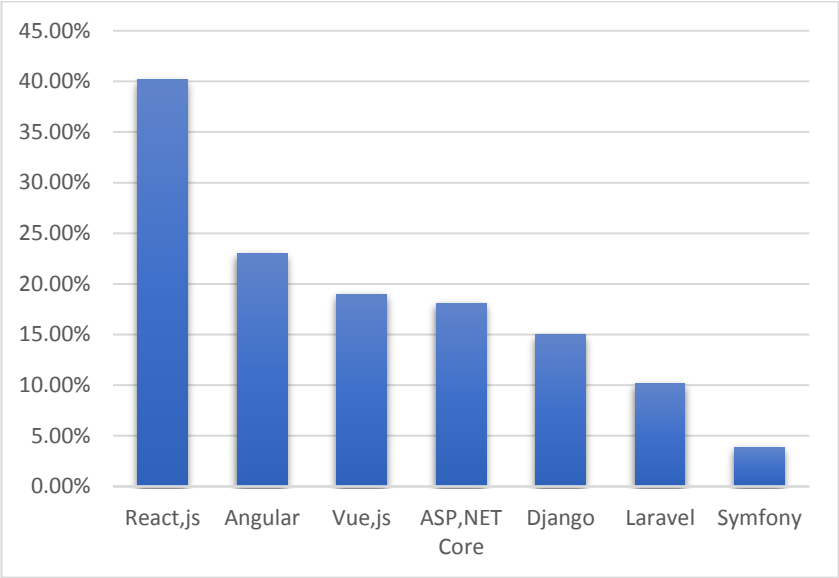
Figura 1 Lenguajes de Programación más populares



Fuente: Elaboración propia

En relación a frameworks de desarrollo web, React.js obtiene el primer lugar, seguido de cerca por frameworks como Angular, ASP.NET Core y Django. Esto se puede observar en la figura 2.

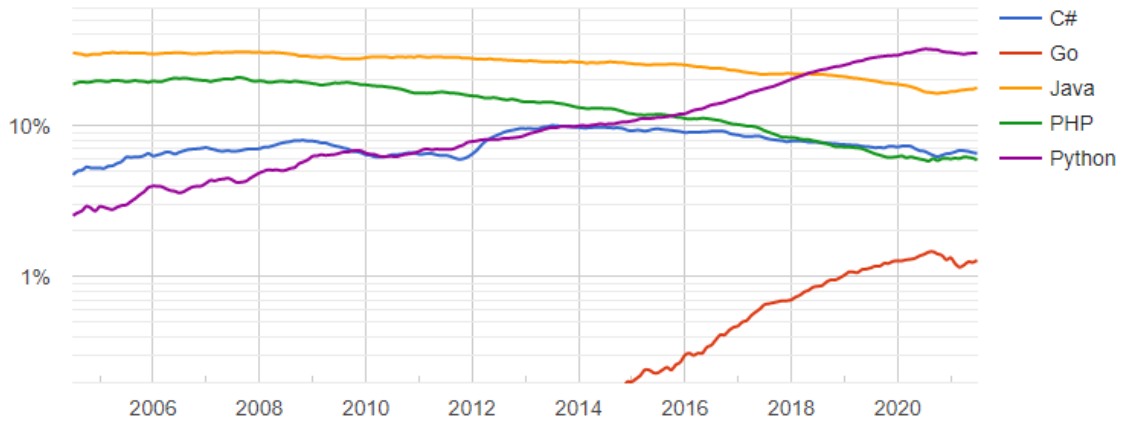
Figura 2 Web Frameworks más populares



Fuente: Elaboración propia

El índice de popularidad elaborado por PYPL (2021) toma datos brutos provenientes de Google Trends y representa el orden de clasificación de lenguajes de programación como se visualiza en la figura 3, en donde Python se posiciona como el lenguaje de programación de mayor popularidad, con un crecimiento en su frecuencia de búsqueda del 16,6% durante los últimos 5 años.

Figura 3 PYPL Popularidad de lenguajes de programación (Julio de 2021)



Fuente:("PYPL PopularitY of Programming Language index," 2021)

Un estudio realizado por Curie et al. (2019) muestra un análisis de frameworks para el desarrollo web backend, en donde contrasta cinco de los principales frameworks en aspectos de complejidad y seguridad. Los resultados se observan en la tabla 2.

Tabla 2 Comparación de frameworks de desarrollo web

	Django	**cakePHP**	**RUBY ON REAILS**	**LIFT**	**STRUTS**
Cloud Computing	X	X	X	X	X
HTML5 Support	X			X	
Debugging	X	X	X	X	X
ORM	X	X	X	X	X
Platform Support	Windows, Linux, OSX	Windows, Linux, OSX	Windows, Linux, OSX	Windows, Linux, OSX	Windows, Linux, OSX
Template Framework	X	X	X	X	X
Security	X	X	X	X	X

Fuente: (Curie et al., 2019)

Así mismo, en otro estudio comparativo de frameworks backend para desarrollo web, evalúa Django, Rails, Spring y Laravel. Los frameworks analizados cumplen con todos los criterios básicos a excepción de pequeñas diferencias. También se evidencia, que para el desarrollo de aplicaciones con enfoque empresarial, Spring y Django son los adecuados al ser catalogados como de mayor escalabilidad (Kaluža, Kalanj, & Vukelić, 2019). Esto se visualiza en la tabla 3.

Tabla 3 Comparación de frameworks de desarrollo web

	Laravel	RUBY ON REAILS	Django	Spring
Personalizado para grandes aplicaciones	0	0	1	1
Alta escalabilidad	0	0	1	1
Adaptado a principiantes	1	1	1	0
Bueno para el desarrollo de aplicaciones empresariales	0	0	1	1
Desarrollo rápido de prototipos	1	0	1	0
Una tendencia de Google en crecimiento	1	0	0	0
Puntos de los usuarios de Haker News	0.5	0.5	1	0.5
Puntos de los usuarios de Reddit	0.5	0.5	1	0.5
Clasificación por estrellas en GitHub	1	0.5	0.5	0.5
El número de etiquetas de Stack Overflow	0.5	1	0.5	0.5
TOTAL	**5.5**	**3.5**	**8**	**5**

Fuente: (Kaluža et al., 2019)

1.7. Estándares de evaluación de calidad de software

Según Yadav & Kishan (2020) en la actualidad existen varios modelos de calidad para evaluar productos de software. Estos modelos se basan en modelos básicos como: McCall, Boehm, FURPS, Dromey y la ISO 25010. A primera vista todos los modelos parecen ser iguales, sin embargo, la diferencia radica en el enfoque que presenta cada modelo, es decir, el propósito para el cual fueron creados. A continuación, se detallan los modelos básicos de calidad y sus características, así como también un modelo holístico y un modelo para documentación.

1.7.1. McCall

Propuesto por Jim McCall en 1977, está dirigido a desarrolladores para ser implementado durante el proceso de desarrollo de software. McCall posee 3 factores de calidad: Revisión, Operación y Transición del producto, como se observan en la tabla 4.

El modelo utiliza una jerarquía de criterios, factores y métricas para abordar la calidad interna y externa del software. Una de las principales aportaciones del modelo es la relación que se establece entre las métricas y los factores de calidad, sin embargo, en el modelo no se considera directamente la funcionalidad del software (Zuria, 2016).

Tabla 4 Modelo de calidad McCall

Factores	**Características**
Revisión del producto	Capacidad de mantenimiento, flexibilidad y comprobación
Transición del producto	Portabilidad, reutilización e interoperabilidad
Operación del producto	Corrección, fiabilidad, eficacia, integridad, usabilidad

Fuente: (Zuria, 2016)

1.7.2. Boehm

El modelo de Boehm presenta una jerarquía con tres características de alto nivel vinculadas a siete factores intermedios, que a su vez se vinculan a 15 características primitivas. El modelo de Boehm hace hincapié en la rentabilidad del mantenimiento e intenta definir la calidad del software mediante un conjunto predefinido de atributos y métricas (Zuria, 2016). Esto se puede observar en la tabla 5.

Tabla 5 Modelo de calidad Boehm

Factores	**Característica**	**Subcaracterística**
Utilidad general	Portabilidad	Independencia de los dispositivos, Integridad
Como es utilizado	Fiabilidad	Autocontención, Integridad, Precisión
	Eficiencia	Responsabilidad, accesibilidad
	Usabilidad	Accesibilidad, Comunicabilidad
Mantenibilidad	Testabilidad	Estructuración, responsabilidad, Accesibilidad
	Indestabilidad	Legibilidad, concisión, Estructuración, Autodescriptividad
	Flexibilidad	Estructuración, aumentabilidad

Fuente: (Zuria, 2016)

1.7.3. FURPS

El modelo Functionality, Usability, Reliability, Performance, Supportability conocido por sus siglas FURPS, fue definido por Rober Grady. La *funcionalidad* integra la capacidad y características del software, la *usabilidad* la facilidad de la interfaz, formación y documentación del usuario, la *fiabilidad* incluye la precisión, tiempo, frecuencia y gravedad de los fallos, el *rendimiento* mide el tiempo de respuesta, eficiencia, velocidad, eficiencia y uso de recurso, y el *soporte* incluye características como la capacidad de servicio, adaptabilidad y compatibilidad (Yadav & Kishan, 2020). Estas características se observan en la tabla 6.

Tabla 6 Modelo de calidad FURPS

Características	Sub características
Funcionalidad	Conjunto de características, capacidades, seguridad.
Usabilidad	Factores humanos, estética, documentación del usuario, material de formación.
Confiabilidad	Frecuencia y gravedad de los fallos, recuperación a fallos, tiempo entre fallos.
	Ingeniería humana.
Rendimiento	Velocidad, eficiencia, disponibilidad, tiempo de respuestas, tiempo de recuperación, utilización de recursos.
Compatibilidad	Probabilidad, extensibilidad, adaptabilidad, mantenibilidad, compatibilidad, configurabilidad, capacidad de servicio, capacidad de instalación, capacidad de localización.

Fuente: (Yadav & Kishan, 2020)

1.7.4. Dromey

Dromey propone la implementación de tres modelos de calidad: modelo de diseño, modelo de requerimientos y modelo de calidad de la implementación. Estos modelos se implementan en las etapas de desarrollo de un software. Este modelo considera la definición de la calidad y se centra en las propiedades del producto que se va a desarrollar, sin embargo, no especifica en detalle como evaluar cada uno de los factores de calidad propuestos (Nistala, Nori, & Reddy, 2019).

Las propiedades y características del modelo Dromey se muestran en la tabla 7 a continuación:

Tabla 7 Propiedades y características de calidad del modelo Dromey

Propiedades del Software	Características de calidad
Exactitud	Funcionalidad, confiabilidad.
Internas	Mantenibilidad, eficiencia, confiabilidad.
Contextuales	Mantenibilidad, reusabilidad, portabilidad, confiabilidad.
Descriptivas	Mantenibilidad, reusabilidad.

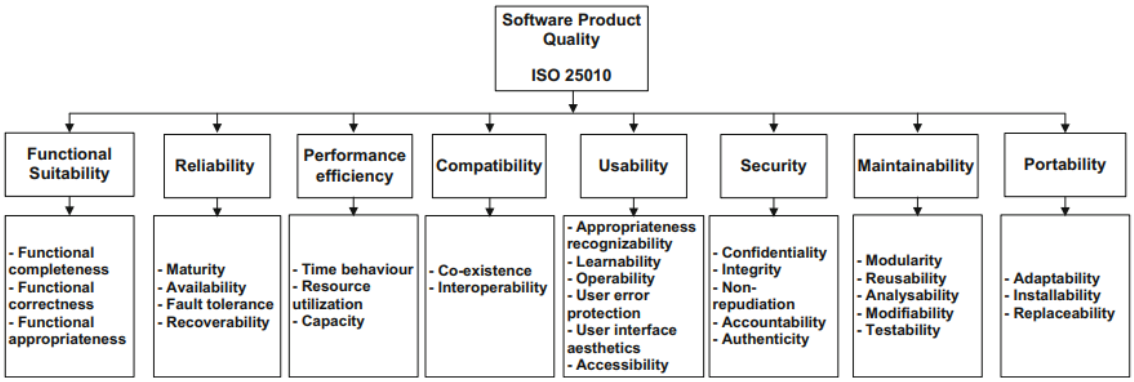
Fuente: Elaboración propia.

1.7.5. ISO 25010

La ISO 25010 se deriva de la ISO/IEC 9126, la cual define las características de calidad y describe un modelo de proceso para la evaluación de productos de software. La ISO 25010 integra modelos de calidad de uso y de calidad de producto. Según Abu Bakar & Razali (2013) es la norma internacional más utilizada para el control de calidad del software, hace referencia a *System and Software Quality Requirements and Evaluation* o la serie de normas SQuaRE.

El modelo de calidad propuesto en la norma ISO/IEC 25010 se observa en la figura 4, se compone de ocho características de calidad intrínsecas: la idoneidad funcional, fiabilidad, operabilidad, rendimiento, seguridad, compatibilidad, mantenibilidad y portabilidad. (Panduwiyasa, Saputra, Azzahra, & Aniko, 2021).

Figura 4 Modelo de Calidad ISO/IEC 25010



Fuente: (ISO, 2011)

Además de los modelos de calidad mencionados, existen otros métodos de calidad que toman como base para su propuesta a los modelos antes descritos y sirven de soporte al momento de evaluar la calidad de un software. Estos modelos se describen a continuación.

1.8. Modelos basados en estándares de calidad

1.8.1. EFFORT

El modelo de calidad propuesto por Aversano & Tortorella (2010) es una herramienta de medición para apoyar la evaluación de proyectos de software libre, define la calidad de un producto de software libre como la sinergia de tres componentes principales que se observan en la tabla 8 y son: calidad del producto desarrollado en el marco del proyecto, fiabilidad de la comunidad de desarrolladores y colaboradores, y el atractivo del producto por su área de captación específica.

Tabla 8 Modelo de calidad EFFORT

Componentes	**Características**
Calidad del producto de software	Portabilidad, mantenibilidad, rentabilidad, funcionalidad, usabilidad y eficiencia.
Fiabilidad de la comunidad	Desarrolladores, comunidad activa, herramientas de soporte, documentación.
Atractivo del producto	Adecuación funcional, difusión, costo de efectividad, reutilización.

Fuente: Elaboración propia.

1.8.2. Bertoa

El modelo planteado por Bertoa define un conjunto de atributos de calidad para estimar eficazmente componentes de software comerciales COTS (Commercial off the shelf). Dichos componentes pueden ser adoptados por empresas de desarrollo de software para construir software catalogado como de mayor complejidad (Sheoran & Sangwan, 2015). Los atributos de calidad y sus subcaracterísticas se pueden observar en la tabla 9.

Tabla 9 Modelo de calidad Bertoa

Características	**Subcaracterísticas (Tiempo de ejecución)**	**Subcaracterísticas (Ciclo vital)**
Funcionalidad	Precisión, seguridad	Idoneidad, Interoperabilidad, Cumplimiento
Fiabilidad	Madurez	Idoneidad
Usabilidad		Capacidad de aprendizaje, Comprensibilidad, Operabilidad
Eficiencia	Comportamiento temporal, Comportamiento de los recursos	
Mantenibilidad		Posibilidad de cambiar, Testabilidad
Portabilidad		Reemplazabilidad

Fuente: (Bertoa & Vallecillo, 2003)

1.8.3. Rawashdeh

Rawashdeh es otro modelo centrado en el uso de componentes COST y ha sido influenciado por los modelos ISO 1926 y Dromey. Las características del modelo, visualizadas en la tabla 10, se centran en el usuario del sistema, es decir, en la parte interesada, así también, en la calidad del producto final y en los procesos implicados (Suradi, Kahar, & Jamaluddin, 2018).

Tabla 10 Modelo de calidad Rawashdeh

Características	**Subcaracterísticas orientadas al producto**	**Subcaracterísticas orientadas al proceso**
Funcionalidad	Precisión, seguridad	Idoneidad, interoperabilidad, conformidad, compatibilidad
Fiabilidad	Recuperabilidad	Madurez
Usabilidad		Comprensibilidad, capacidad de aprendizaje, operabilidad, complejidad
Eficiencia	Comportamiento del tiempo, comportamiento de los recursos	
Mantenibilidad		Cambiabilidad, comprobabilidad
Gestionabilidad	Gestión de calidad	Gestión de calidad

Fuente: (Miguel, Mauricio, & Rodriguez, 2014)

1.8.4. CBQM

El modelo de calidad basado en componentes y conocido por sus siglas CBQM, soporta características de calidad, presentes en la tabla 11, que son apropiadas para las bibliotecas de componentes externos: componentes COTS, componentes preconstruidos y componentes de código abierto vinculados con el desarrollo orientado a componentes (Sheoran & Sangwan, 2015).

Tabla 11 Modelo de calidad CBQM

Usuario	**Desarrolladores**	**Integradores**
Certificación, simplicidad, rendimiento, económico, funcionalidad, eficiencia, usabilidad	Extensibilidad, exactitud, fiabilidad, portabilidad.	Modularidad, compatibilidad, rendimiento, complejidad, mantenibilidad, portabilidad, fiabilidad.

Fuente:(Dixit, 2013)

1.9. Modelo Holístico

Laaziri, Benmoussa, Khoulji, & Kerkeb (2019) en base a un estudio comparativo entre frameworks PHP, proporcionan un modelo de comparación de frameworks efectivo que fusiona siete dimensiones: características, multilenguaje, requisitos del sistema, arquitectura técnica, organización del código, integración continua, y finalmente, documentación y curva de aprendizaje. En la tabla 12 se muestran las especificaciones de las características.

Tabla 12 Modelo Holístico

Características	Scafollding
Multilingüe	Contenido multilingüe
	Sistema operativo
	Lenguaje de programación
	Etiqueta
	Sistema multiusuario
	Autoenfoque
	Pingback
	Extensiones/Plugins
Requisitos del sistema	Motor de procesamiento de imágenes
	WYSIWYG-Editor
	Múltiples proyectos
	Páginas externas
	Estadísticas de usuario
	Control de revisión
	Compatible con PSR-0
	Generación de código de máquina
	Lenguaje de programación
	Instalación
	Núcleos
	Acceso a la base
	Contenedor de servicio
	Modelo de motor
Arquitectura técnica	Formularios y validadores
	Caché y rendimiento
	Herramientas de depuración y desarrollo
	Panel de administración
	Buscador
	Despachador de eventos y middleware
	API REST
	Centinela
	patrones

	Compatibilidad de la API REST
	Scaffolding
	MVC
Organización del código	Modelado HTML
	Instalación a través del composer
	ORM
	Múltiples drivers de soporte de BD, almacenamiento
	Búsqueda completa del texto
	Soporte CI, AQ
	Travis Ci
	Estilo CI
Integración continua (CI)	Drone
	Codificación
	Circulo CI
	Jenkins
	Curva de aprendizaje
	Complejidad de instalación
Curva de aprendizaje y documentación	Eventos / Comunidad
	Documentación
	Certificación

Fuente:(Laaziri, Benmoussa, Khoulji, Larbi, & Yamami, 2019)

1.10. Modelo COCA

COCA es un modelo de calidad simple para evaluar la calidad de la documentación del usuario (Alchimowicz & Nawrocki, 2016). Está compuesto por cuatro características de calidad ortogonales las cuales son:

- **Integridad:** Grado en que la documentación proporciona toda la información que necesitan los usuarios para utilizar el software.
- **Operabilidad:** Grado en que la documentación tiene atributos que la hacen útil y fácil de usar.
- **Exactitud:** Grado en que las descripciones proporcionadas en la documentación son correctas.
- **Apariencia:** Grado en que la información contenida en la documentación se presenta de forma estética.

2. Análisis y comparación de los métodos de evaluación de calidad de software

A partir de la revisión del estado del arte presentado en el capítulo 1, se analizan nueve métodos de calidad de software: McCall, Boehm, FURPS, Dromey, ISO 25010, EFFORT, Bertoa, Rawashdeh y CBQM. De los métodos revisados, se extraen las características y subcaracterísticas incluidas en cada modelo, finalmente se establece una tabla comparativa entre los modelos analizados.

2.1. McCall

La tabla 13 presenta en detalle los tres factores de calidad que presenta el modelo de McCall, con sus respectivas características y subcaracterísticas (McCall, Richards, & Walters, 1977).

Tabla 13 Características y subcaracterísticas del Modelo de calidad McCall

Factores de calidad	Característica	Definición	Subcaracterística
Operación del producto	**Exactitud**	La medida en que un software cumple con la especificación de sus requisitos.	Trazabilidad
			Completitud
			Consistencia
	Fiabilidad	La medida en que un software realiza sus funciones previstas sin fallas.	Consistencia
			Precisión
			Tolerancia al error
	Eficiencia	La cantidad de recursos de hardware y código que el software necesita para realizar una función.	Eficacia de ejecución
			Eficiencia de almacenamiento
	Integridad	Hasta qué punto el software puede controlar que una persona no autorizada acceda a los datos o al software.	Control de acceso
			Auditoría de acceso
	Usabilidad	El grado de esfuerzo requerido para aprender, operar y comprender las funciones del software.	Operabilidad
			Entrenamiento
			Comunicatividad
Revisión de producto	**Mantenibilidad**	El esfuerzo necesario para detectar y corregir un error durante la fase de mantenimiento.	Simplicidad
			Concisión
			Autodescriptivo
			Modularidad
	Testabilidad	El esfuerzo necesario para mejorar un programa de software operativo.	Sencillez
			Instrumentación
			Autodescriptividad
			Modularidad
	Flexibilidad		Sencillez

		El esfuerzo requerido para verificar que un software cumple con los requisitos especificados.	Expandibilidad
			Generalidad
			Modularidad
Transición de producto	**Portabilidad**	El esfuerzo necesario para trasladar un programa de una plataforma a otra.	Simplicidad
			Sistema de software independiente
			Independencia de la máquina
	Reutilización	La medida en que el código del programa se puede reutilizar en otras aplicaciones.	Sencillez
			Generalidad
			Modularidad
			Sistema de software independiente
			Independencia de la máquina
	Interoperabilidad	El esfuerzo necesario para integrar dos sistemas entre sí.	Modularidad
			Comunicación de comunicaciones
			Datos comunes

Fuente: Elaboración propia

2.2. Boehm

La tabla 14 visualiza los factores de calidad propuestos en el modelo Boehm, así como las características y subcaracterísticas (Zuria, 2016).

Tabla 14 Características y subcaracterísticas del Modelo de calidad Boehm

Factores	Características	Definición	Subcaracterísticas
Utilidad General	Portabilidad	Medida en que puede operar en equipos con una configuración diferente a la actual.	Independencia Del Dispositivo
			Autocontención
Utilidad Tal Como Está	Fiabilidad	Medida en que se espera que realice las funciones previstas de manera satisfactoria.	Autocontención
			Precisión
			Completitud
			Robustez / Integridad
			Consistencia
	Eficiencia	Medida en que cumple su propósito sin desperdicio de recursos.	Responsabilidad
			Eficiencia Del Dispositivo
			Accesibilidad
			Comunicatividad
	Usabilidad	Medida en que es confiable, eficiente y diseñado por humanos.	Robustez / Integridad
			Accesibilidad
			Comunicatividad

Mantenibilidad	Testabilidad	Medida en que facilita el establecer criterios de verificación y apoya la evaluación de su desempeño.	Responsabilidad
			Accesibilidad
			Autodescriptividad
			Estructuración
	Indestabilidad	Medida en que su propósito es claro para el inspector.	Consistencia
			Autodescriptividad
			Estructuración
			Concisión
			Legibilidad
	Flexibilidad	Medida en que facilita la incorporación de cambios, una vez determinada la naturaleza del cambio deseado.	Estructuración
			Aumentabilidad

Fuente: Elaboración propia.

2.3. FURPS

En la tabla 15 se detalla las características y subcaracterísticas presentadas en el modelo FURPS (Yadav & Kishan, 2020).

Tabla 15 Características del Modelo de calidad FURPS

Característica	Definición	Subcaracterística
Funcionalidad	Se evalúan las capacidades y características del programa, así como la generalidad de las funciones entregadas y la seguridad del sistema.	Reutilización
		Capacidad
		Seguridad
Usabilidad	Se evalúan factores humanos, consistencia, estética en general, materiales y documentación.	Factores humanos
		Interfaz de usuario
		Materiales de entrenamiento
		Documentación
Confiabilidad	Se valora la severidad y frecuencia de las fallas, la precisión de los resultados, el tiempo medio de falla (MTTF), la capacidad para predecir fallos y recuperarse de ellos.	Frecuencia de falla
		Recuperabilidad
		Tiempo medio entre fallos
		Precisión
Rendimiento	Se calcula mediante la velocidad del procesamiento, el consumo de recursos, el tiempo de respuesta, la eficacia y rendimiento.	Tiempo de respuesta
		Eficiencia
		Precisión
		Tiempo de recuperación
		El uso de recursos
Compatibilidad	Combina la extensibilidad del programa, la utilidad, la capacidad de servicio y la adaptabilidad.	Adaptabilidad
		Testabilidad
		Compatibilidad
		Utilidad

Fuente: Elaboración propia.

2.4. Dromey

El modelo Dromey incluye cuatro categorías y características para cada una (Nistala et al., 2019). Esto se detalla en la tabla 16.

Tabla 16 Categorías y características del modelo Dromey

Categoría	Definición	Característica
Exactitud	Evalúa si se violan algunos principios básicos.	Funcionalidad
		Fiabilidad
Interno	Mide qué tan bien se ha implementado un componente de acuerdo con su uso previsto	Mantenibilidad
		Eficiencia
		Fiabilidad
Contextual	Se ocupa de las influencias externas por y sobre el uso de un componente.	Mantenibilidad
		Reutilización
		Portabilidad
		Fiabilidad
Descriptivo	Mide el carácter descriptivo de un componente.	Mantenibilidad
		Reutilización
		Portabilidad
		Usabilidad

Fuente: Elaboración propia.

2.5. ISO 25010

La tabla 17 presenta las características y subcaracterísticas que incluye del modelo ISO 25010 (ISO, 2011).

Tabla 17 Características y subcaracterísticas de la ISO/IEC 25010

Característica	Subcaracterística	Definición
Funcionalidad	Completitud funcional	Se refiere al conjunto de funciones que cubre todas las tareas especificadas y los objetivos del usuario.
	Corrección funcional	Se refiere a qué tan bien un producto o sistema proporciona los resultados correctos con el grado de precisión necesario.
	Pertinencia funcional	Se refiere a qué tan bien las funciones pueden lograr tareas y objetivos específicos.
Fiabilidad	Madurez	Se refiere a qué tan bien un sistema, producto o componente puede satisfacer sus necesidades de confiabilidad.
	Disponibilidad	Se refiere a si un sistema, producto o componente está operativo y es accesible.
	Tolerancia a fallos	Se refiere a qué tan bien funciona un sistema, producto o componente a pesar de fallas de hardware y / o software.
	Recuperabilidad	Se refiere a qué tan bien un producto o sistema puede recuperar datos en caso de una interrupción o falla.
Eficiencia	Comportamiento temporal	Se refiere a los tiempos de respuesta y procesamiento, y las tasas de rendimiento de un producto o sistema mientras realiza sus funciones.

	Utilización de recursos	Se refiere a las cantidades y tipos de recursos utilizados por un producto o sistema mientras realiza sus funciones.
	Capacidad	Determina los límites máximos de un producto o parámetro del sistema.
Usabilidad	Idoneidad Reconocimiento	Se refiere a qué tan bien puede reconocer si un producto o sistema es apropiado para sus necesidades.
	Capacidad de aprendizaje	Se refiere a lo fácil que es aprender a usar un producto o sistema.
	Operabilidad	Se refiere a si un producto o sistema tiene atributos que facilitan su operación y control.
	Protección contra errores del usuario	Se refiere a qué tan bien un sistema protege a los usuarios contra cometer errores.
	Estética de la interfaz de usuario	Se refiere a si una interfaz de usuario es agradable.
	Accesibilidad	Se refiere a qué tan bien se puede usar un producto o sistema con la más amplia gama de características y capacidades.
Seguridad	Confidencialidad	Se refiere a qué tan bien un producto o sistema puede garantizar que los datos solo sean accesibles para aquellos que tienen acceso autorizado.
	Integridad	Se refiere a la capacidad de un sistema, producto o componente para evitar el acceso no autorizado y la modificación de programas y / o datos informáticos.
	No repudio	Se refiere a qué tan bien se puede probar que se han llevado a cabo acciones o eventos.
	Responsabilidad	Se refiere a las acciones de un usuario no autorizado que se pueden rastrear hasta ellos.
	Autenticidad	Se refiere a qué tan bien se puede probar la identidad de un sujeto o recurso.
Compatibilidad	Coexistencia	Se refiere a la efectividad de un producto para realizar sus funciones requeridas mientras comparte recursos y entornos comunes con los productos, sin afectar negativamente a ningún otro.
	Interoperabilidad	Se refiere a qué tan bien dos o más sistemas, productos o componentes pueden intercambiar información y utilizar esa información.
Mantenibilidad	Modularidad	Se refiere a si los componentes de un sistema o programa se pueden cambiar con un impacto mínimo en los demás componentes.
	Reutilización	Se refiere a qué tan bien se puede usar un activo en más de un sistema.
	Analizabilidad	Se refiere a la eficacia de una evaluación de impacto sobre los cambios previstos. Además, también se refiere al diagnóstico de deficiencias o causas de fallas, o para identificar piezas a modificar.
	Modificabilidad	Se refiere a qué tan bien se puede modificar un producto o sistema sin introducir defectos o degradar la calidad del producto existente.
	Testabilidad	Se refiere a la eficacia de los criterios de prueba para un sistema, producto o componente. Además, también se refiere a las pruebas que se pueden realizar para determinar si se han cumplido los criterios de prueba.
Portabilidad	Adaptabilidad	Se refiere a qué tan bien se puede adaptar un producto o sistema para hardware, software u otros entornos de uso diferentes o en evolución.
	Instalabilidad	Se refiere a la eficacia con la que se puede instalar y / o desinstalar un producto o sistema.
	Reemplazabilidad	Se refiere a qué tan bien un producto puede reemplazar a otro producto comparable.

Fuente: Elaboración Propia

2.6. EFFORT

El método de evaluación de calidad para proyectos de Software libre EFFORT, presenta tres categorías con sus respectivas características (Kan, 2002). El detalle se observan la tabla 18.

Tabla 18 Características del Modelo de calidad EFFORT

Categoría	Características
Calidad del producto de software	Portabilidad
	Mantenibilidad
	Fiabilidad
	Funcionalidad
	Usabilidad
	Eficiencia
Confiabilidad de la comunidad	Desarrolladores
	Comunidad
	Actividad
	Herramientas de apoyo
	Servicios de apoyo
	Documentación
Atractivo del producto	Adecuación funcional
	Difusión
	Rentabilidad
	Reutilización legal

Fuente: Elaboración propia.

2.7. Bertoa

El modelo de calidad Bertoa posee características y subcaracterísticas basadas en la ISO 9126 (Bertoa & Vallecillo, 2003). Esto se visualiza en la tabla 19.

Tabla 19 Características y subcaracterísticas del Modelo de calidad Bertoa

Característica	Definición	Subcaracterística
Funcionalidad	Capacidad de un componente para proporcionar los servicios y funciones requeridos, cuando se utiliza en las condiciones especificadas.	Precisión
		Seguridad
		Idoneidad
		Interoperabilidad
		Cumplimiento
Fiabilidad	Esta característica es directamente aplicable a los componentes y fundamental para su reutilización.	Madurez
		Idoneidad
Usabilidad	La usabilidad de un componente es la capacidad para ser utilizado por el desarrollador de la aplicación al construir un producto o sistema de software.	Capacidad de aprendizaje
		Comprensibilidad
		Operabilidad
Eficiencia	Define los atributos que influyen en la relación entre la cantidad de recursos utilizados y el nivel de rendimiento del software en las condiciones establecidas.	Comportamiento temporal
		Comportamiento de los recursos
Mantenibilidad	Identifica si un producto de software tiene la capacidad para ser modificado.	Posibilidad de cambiar
		Testabilidad
Portabilidad	Identifica si un producto de software tiene la capacidad para transferirse de un entorno a otro.	Reemplazabilidad

Fuente: Elaboración propia

2.8. Rawashdeh

Rawashdeh asume la definición propuesta por la ISO 9126 (Adnan & Bassem, 2006), las características y subcaracterísticas que el modelo incluye, se presenta en la tabla 20.

Tabla 20 Modelo de calidad Rawashdeh

Características	Subcaracterísticas
Funcionalidad	Precisión
	Seguridad
	Idoneidad
	Interoperabilidad
	Cumplimiento
	Compatibilidad
Fiabilidad	Recuperabilidad
Usabilidad	Madurez
	Capacidad de aprendizaje
	Comprensibilidad
	Operabilidad
	Complejidad
Eficiencia	Comportamiento temporal
	Comportamiento de los recursos
Mantenibilidad	Posibilidad de cambiar
	Testabilidad
Manejabilidad	Gestión de la calidad

Fuente: (Miguel et al., 2014)

2.9. CBQM

El modelo CBQM incluye grupos de participación con sus respectivas características (Dixit, 2013). Esto se visualiza en la tabla 21.

Tabla 21 Modelo de calidad CBQM

Grupos de participación	Características	Definición
Usuario	Certificación	Es la garantía proporcionada para la calidad del software por terceros.
	Sencillez	La simplicidad es la capacidad de modelar problemas del mundo real en estructuras de software intuitivas.
	Rendimiento	Es la capacidad de predecir el tiempo de exposición de una aplicación en función de la arquitectura del software, la cantidad de datos procesados, el hardware y los factores internos.
	Económico	La economía es la relación entre los beneficios financieros y su coste.
	Funcionalidad	Cada uno de ellos puede representar una funcionalidad del sistema que se está desarrollando y cada uno es un requisito funcional de todo el sistema.
	Eficiencia	Expresa la capacidad de un componente para proporcionar un rendimiento adecuado, en relación con la cantidad de recursos utilizados.
	Usabilidad	Capacidad de un componente para ser comprendido, aprendido, utilizado, configurado y ejecutado, cuando se utiliza en condiciones específicas.
Desarrollador	Extensibilidad	Es la facilidad para ampliar los productos de software a nuevos cambios de especificación u otros dominios de problemas.
	Exactitud	Es la representación exacta de un producto de software con respecto a su especificación.
	Fiabilidad	La fiabilidad incluye aspectos como la disponibilidad, la precisión y la capacidad de recuperación
	Portabilidad	Es la capacidad del software para responder adecuadamente a condiciones anormales o modificadas
Integradores	Modularidad	Capacidad de descomponer una aplicación en módulos individuales, que se comunican únicamente a través de una interfaz definida.
	Compatibilidad	Es la facilidad para combinar elementos de software con otros
	Rendimiento	Capacidad de predecir el tiempo de exposición de una aplicación en función de la arquitectura del software, la cantidad de datos procesados y el hardware y en los siguientes factores internos
	Complejidad	La complejidad es una métrica de software que proporciona una medida cuantitativa del programa.
	Mantenibilidad	Capacidad de identificar un fallo dentro de un componente de software. Expresa la capacidad de un componente para ser modificado.
	Portabilidad	Es la capacidad del software para responder adecuadamente a condiciones anormales o modificadas
	Fiabilidad	La confiabilidad se puede definir como la capacidad del componente para mantener un nivel específico de rendimiento, cuando se usa en condiciones específicas.

Fuente: Elaboración propia.

2.10. Comparativa de los métodos de evaluación de calidad.

La tabla 22 resume los factores/atributos/características obtenidas de los modelos de calidad, así también, resalta las coincidencias entre estos modelos de calidad analizados.

Tabla 22 Tabla comparativa de los métodos de evaluación de calidad

Características	25010	MCALL	BOEHM	FURPS	DROMEY	BERTOA	RAWASHDEH	CBQM	EFFORT
Funcionalidad	X			X	X	X	X	X	X
Fiabilidad	X	X	X	X	X	X	X	X	X
Eficiencia	X	X	X	X	X	X	X	X	X
Usabilidad	X	X	X	X		X	X	X	X
Seguridad	X								
Compatibilidad	X			X				X	
Mantenibilidad	X	X			X	X	X	X	X
Exactitud		X						X	
Integridad		X							
Portabilidad		X	X		X	X		X	X
Testabilidad			X						
Comprensibilidad			X						
Flexibilidad			X						
Reutilización					X				
Manejabilidad							X		
Certificación								X	
Sencillez								X	
Económico								X	
Extensibilidad								X	
Modularidad								X	
Complejidad								X	
Ingeniería humana			X						
Modificabilidad			X						
Interoperabilidad		X							

Fuente: Elaboración propia.

Al comparar las características en los nueve modelos de calidad (tabla 22), se evidencia que, de las 24 características obtenidas, los atributos de *Fiabilidad y Eficiencia*, son comunes en los 9 modelos de calidad. Además, la *Usabilidad* es común en 4 de los 5 modelos. Por otro lado, hay atributos como *Funcionalidad* y *Mantenibilidad* que son comunes en 7 modelos de calidad. La *Portabilidad* es común para 6 modelos de calidad. La *Compatibilidad* es una característica común en 3 modelos de calidad. Además, la *Exactitud* es común para 2 modelos de calidad. Por último, las características restantes *Interoperabilidad*, *Modificabilidad*, *Ingeniería humana*, *Complejidad*, *Modularidad*, *Extensibilidad*, *Económico*, *Sencillez*, *Certificación*, *Manejabilidad*, *Reutilización*,

Flexibilidad, *Comprensibilidad*, *Testabilidad*, *Integridad* y *Seguridad*, se aplican en un solo modelo de calidad.

Del total de características, se observa que la mayoría comparten características de la ISO 25010, lo que permite catalogar a este modelo como uno de los más confiables para métricas de calidad de software, para ser utilizada por desarrolladores.

A continuación, en la tabla 23 se muestra una comparativa detallada de las subcaracterísticas recuperadas de los modelos de calidad con sus respectivas coincidencias. Para este análisis no se consideran los modelos Dromey, CBQM, FURPS y EFFORT por no contar con acceso a la documentación completa de sus subcaracterísticas.

Tabla 23 Tabla comparativa de subcaracterísticas de los modelos de calidad

Característica	Subcaracterística	ISO 25010	MCALL	BOEHM	BERTOA	RAWASHDEH
Funcionalidad	Completitud funcional	X			X	X
	Corrección funcional	X				
	Pertinencia funcional	X				
	Seguridad				X	X
	Precisión				X	X
	Idoneidad				X	X
	Interoperabilidad				X	X
	Compatibilidad					X
Fiabilidad	Madurez	X			X	
	Disponibilidad	X				
	Tolerancia a fallos	X	X			
	Recuperabilidad	X				X
	Consistencia		X	X		
	Precisión		X	X		
	Autocontención			X		
	Completitud			X		
	Robustez / Integridad			X		
	Idoneidad				X	
Eficiencia	Comportamiento temporal	X			X	X
	Utilización de recursos	X			X	X
	Capacidad	X				
	Eficacia de ejecución		X			
	Eficiencia de almacenamiento		X			
	Eficiencia del dispositivo			X		
	Responsabilidad			X		
	Accesibilidad			X		
	Comunicatividad			X		

Usabilidad	Idoneidad Reconocimiento	X				
	Capacidad de aprendizaje	X	X		X	X
	Operabilidad	X	X		X	X
	Factores humanos	X				
	Estética de la interfaz de usuario	X				
	Accesibilidad	X		X		
	Comunicatividad		X	X		
	Robustez / Integridad			X		
	Comprensibilidad				X	X
	Madurez					X
	Complejidad					X
Seguridad	Confidencialidad	X				
	Integridad	X				
	No repudio	X				
	Responsabilidad	X				
	Autenticidad	X				
Compatibilidad	Coexistencia	X				
	Interoperabilidad	X				
Mantenibilidad	Modularidad	X	X			
	Reutilización	X				
	Analizabilidad	X				
	Modificabilidad	X			X	X
	Testabilidad	X			X	X
	Simplicidad		X			
	Concisión		X			
	Auto-descriptividad		X			
Portabilidad	Adaptabilidad	X				
	Instalabilidad	X				
	Reemplazabilidad	X			X	
	Simplicidad		X			
	Independencia del software		X			
	Independencia del dispositivo		X	X		
	Autocontención			X		
Exactitud	Trazabilidad		X			
	Completitud		X			
	Consistencia		X			
Integridad	Control de acceso		X			
	Auditoría de acceso		X			
Testabilidad	Sencillez		X			
	Instrumentación		X			
	Auto-descriptividad		X	X		
	Modularidad		X			
	Responsabilidad			X		
	Accesibilidad			X		
	Estructuración			X		
Flexibilidad	Sencillez		X			

	Expansibilidad		X			
	Generalidad		X			
	Modularidad		X			
	Estructuración			X		
	Aumentabilidad			X		
Reutilización	Sencillez		X			
	Generalidad		X			
	Modularidad		X			
	Independencia del software		X			
	Independencia de la máquina		X			
Interoperabilidad	Modularidad		X			
	Comunicación de comunicaciones		X			
	Datos comunes		X			
Comprensibilidad	Consistencia			X		
	Auto-descriptividad			X		
	Estructuración			X		
	Concisión			X		
	Legibilidad			X		
Manejabilidad	Gestión de la calidad					X

Fuente: Elaboración propia.

Según lo mostrado en la tabla 23, se observa que subcaracterísticas como el *entrenamiento, la operabilidad, la modificabilidad, testabilidad, comportamiento temporal y utilización de recursos*, son las más utilizadas por los modelos de calidad.

Las definiciones proporcionadas por los autores para cada una de las subcaracterísticas están detalladas en el Anexo 1.

3. Resultados

Una vez culminado el análisis y comparación de los métodos de calidad, se realiza la propuesta del modelo de calidad objetivo del presente trabajo de titulación, el mismo que se orienta a ser un recurso de apoyo para evaluar los frameworks que existen dentro del área de desarrollo de software, a fin de identificar el idóneo bajo los requerimientos específicos de un equipo de desarrollo en particular.

3.1. Definición del método de evaluación de calidad propuesto

En la tabla 24 y 25 se especifica el modelo propuesto, en donde se identifican: ***características y subcaracterísticas***. Para evaluar la efectividad, a cada característica y subcaracterística se le asigna un porcentaje: ***porcentaje característica y porcentaje subcaracterística***, cuyo valor es asignado de acuerdo a la concurrencia compartida entre los modelos analizados en el capítulo 3; ***valoración*** corresponde al puntaje asignado a cada criterio por parte del evaluador; ***ponderación*** resultante de aplicar el porcentaje de la subcaracterística a la valoración; y ***observación*** para incluir datos que contribuyan a la evaluación de la característica o subcaracterística por parte del evaluador.

Tabla 24 Plantilla del método de evaluación de calidad propuesto

Característica	Porcentaje Característica	Subcaracterística	Porcentaje Subcaracterística	Valoración (1/5)	Ponderación	Observaciones
Funcionalidad	8,00%	Completitud funcional	20,00%			
		Corrección funcional	5,00%			
		Pertinencia funcional	5,00%			
		Seguridad	20,00%			
		Precisión	10,00%			
		Idoneidad	10,00%			
		Interoperabilidad	10,00%			
		Compatibilidad	5,00%			
		Multiproyecto	5,00%			
		Arquitectura	5,00%			
		Escalabilidad	5,00%			
Fiabilidad	16,00%	Madurez	15,00%			
		Disponibilidad	10,00%			
		Tolerancia a fallos	15,00%			
		Recuperabilidad	10,00%			
		Consistencia	15,00%			

		Precisión	15,00%			
		Autocontención	5,00%			
		Completitud	5,00%			
		Robustez / Integridad	5,00%			
		Idoneidad	5,00%			
Eficiencia	8,00%	Comportamiento temporal	20,00%			
		Utilización de recursos	20,00%			
		Capacidad	10,00%			
		Eficacia de ejecución	10,00%			
		Accesibilidad	10,00%			
		ORM	10,00%			
		Comunicatividad	10,00%			
Usabilidad	13,00%	Idoneidad Reconocimiento	5,00%			
		Capacidad de aprendizaje	25,00%			
		Operabilidad	20,00%			
		Factores humanos	10,00%			
		Accesibilidad	5,00%			
		Comunicatividad	10,00%			
		Robustez / Integridad	5,00%			
		Comprensibilidad	10,00%			
		Madurez	5,00%			
		Complejidad	5,00%			
Seguridad	5,00%	Confidencialidad	30,00%			
		Integridad	40,00%			
		No repudio	30,00%			
Compatibilidad	6,00%	Coexistencia	50,00%			
		Interoperabilidad	50,00%			
Mantenibilidad	13,00%	Modularidad	15,00%			
		Reutilización	5,00%			
		Analizabilidad	5,00%			
		Modificabilidad	30,00%			
		Testabilidad	30,00%			
		Simplicidad	5,00%			
		Concisión	5,00%			
		Auto-descriptividad	5,00%			
Portabilidad	8,00%	Adaptabilidad	10,00%			
		Instalabilidad	10,00%			
		Reemplazabilidad	30,00%			
		Simplicidad	10,00%			
		Internacionalización	30,00%			
		Autocontención	10,00%			
Exactitud	4,00%	Trazabilidad	40,00%			
		Consistencia	60,00%			
Integridad	2,00%	Control de acceso	50,00%			
		Auditoría de acceso	50,00%			
Testabilidad	3,00%	Sencillez	15,00%			

		Instrumentación	20,00%			
		Auto-descriptividad	35,00%			
		Modularidad	20,00%			
		Accesibilidad	10,00%			
Flexibilidad	3,00%	Sencillez	20,00%			
		Expandibilidad	10,00%			
		Generalidad	50,00%			
		Modularidad	20,00%			
Reutilización	3,00%	Sencillez	25,00%			
		Generalidad	40,00%			
		Modularidad	20,00%			
		Independencia del software	15,00%			
Interoperabilidad	5,00%	Modularidad	35,00%			
		Comunicación de comunicaciones	50,00%			
		Datos comunes	15,00%			
Documentación	3,00%	Integridad	30,00%			
		Operabilidad	15,00%			
		Exactitud	15,00%			
		Comunidad	20,00%			
		Apariencia	10,00%			
		Contenido multilingüe	10,00%			

Fuente: Elaboración propia

Tabla 25 Plantilla de totales del método de evaluación de calidad propuesto

Característica	Pct.	∑ ponderación	Valoración Total
Funcionalidad	8.00%		
Fiabilidad	16.00%		
Eficiencia	8.00%		
Usabilidad	13.00%		
Seguridad	5.00%		
Compatibilidad	6.00%		
Mantenibilidad	13.00%		
Portabilidad	8.00%		
Exactitud	4.00%		
Integridad	2.00%		
Testabilidad	3.00%		
Flexibilidad	3.00%		
Reutilización	3.00%		
Interoperabilidad	5.00%		
Documentación	3.00%		
Total			**0**
Total pct.	**100%**		**0.00%**

Fuente: Elaboración propia

A continuación, se detallan los pasos a realizar por parte del evaluador:

1. Valorar cada una de las subcaracterísticas (tabla 24), mediante la escala de Likert (1932) que incluye los siguientes niveles de valoración:

Tabla 26 Valoración con la escala de Likert

Valoración	Nivel
5	Muy importante
4	Importante
3	Neutral
2	Poco importante
1	No es importante

Fuente: Elaboración propia

2. Calcular la ponderación de cada una de las subcaracterísticas (tabla 24), con la fórmula:

   Fórmula 1:

$$Ponderación = \%\ subcaracterística * valoración$$

3. Calcular valoración total de cada característica (tabla 25), para esto realizar la sumatoria de las ponderaciones por característica aplicando la fórmula 2. Este resultado es sobre 5 puntos.

   Fórmula 2:

$$Valoración\ total\ por\ característica = \sum ponderaciones * \%\ característica$$

4. Calcular la sumatoria de las columnas de valoración total (tabla 25), lo cual arroja un resultado sobre 5 puntos.
5. Opcional, se puede calcular el porcentaje de la valoración obtenida (tabla 25).

4. Evaluación del método

Para validar el método de evaluación de herramientas de desarrollo de software libre, se aplica como caso de prueba la reconstrucción del software UDA-ERP, en dónde, el método propuesto servirá de apoyo para la selección del framework adecuado que satisfaga las condiciones necesarias para el desarrollo del software.

4.1. Escenario de Pruebas

Se selecciona el UDA-ERP debido a que es un software empresarial que incluye procesos compartidos, flujo de datos y operaciones transaccionales, lo que demanda una alta calidad de las herramientas de desarrollo de software La primera versión del sistema UDA-ERP está desarrollada en Oracle Apex, una plataforma de desarrollo empresarial bajo código y licencia de Oracle. El sistema está formado por diez módulos: Administración, Contabilidad, SRI, Compras, Ventas, Inventario, Producción, Costos, RRHH y Seguridad. Debido a la pandemia ocasionada por el Covid-19, el equipo de trabajo de este software se ha visto en la necesidad de buscar alternativas económicamente más viables para mantener el proyecto, es por ello que se volcaron hacia una migración completa con herramientas de software libre.

Como punto de partida en la migración del software UDA-ERP, es necesaria la selección de un framework que facilite el desarrollo y a su vez otorgue los estándares de calidad requeridos. Es así que se evalúan tres frameworks de libre distribución: Django, Laravel y .NET Core. Cabe mencionar que esta selección fue realizada por el equipo de docentes que colabora en el proyecto de desarrollo del software UDA-ERP.

4.2. Evaluación del método propuesto

Como se menciona en el aparado previo, se evalúan tres herramientas de desarrollo web, gratuitas y de código abierto: Laravel, .NET Core y Django.

- **Laravel** es un framework de aplicación web de PHP basado en Symfony, creado por Taylor Otwel, tiene una sintaxis expresiva y elegante que facilita el desarrollo de tareas comunes utilizadas en la mayoría de proyectos web (https://laravel.com/).

- **ASP.NET Core** es un framework de desarrollo web popular para crear aplicaciones en la plataforma .NET. Es el sucesor de .NET framework y es desarrollado por empleados de Microsoft a través de la fundación .NET y publicado una licencia MIT (https://dotnet.microsoft.com/).
- **Django** es un framework web escrito en Python que fomenta un diseño limpio y pragmático y un desarrollo rápido. Es mantenido por Django Software Fundation, establecida en los EE. UU. como una organización sin fines de lucro (https://www.djangoproject.com/).

La tabla 27 detalla los resultados al aplicar el método propuesto en el análisis de los tres frameworks mencionados. Los puntajes asignados a cada subcategoría corresponden a los criterios del equipo evaluador, además, el método calcula el valor de ponderación que servirá como soporte en la selección de la herramienta mejor calificada. También se adjunta las observaciones apuntadas en algunas subcategorías.

Tabla 27: Resultados del método de evaluación propuesto

			Django			.NET Core			Laravel		
Característica	**Subcaracterística**	**%**	**Valoración 1-5**	**Ponderación**	**Observación**	**Valoración 1-5**	**Ponderación**	**Observación**	**Valoración 1-5**	**Ponderación**	**Observación**
Funcionalidad	Completitud funcional	20%	5	1		5	1		2	0,4	Orientado a desarrollo web
	Corrección funcional	5%	5	0,25		5	0,25		3	0,15	
	Pertinencia funcional	5%	4	0,2	Pruebas con un Maestro-detalle	3	0,15	Pruebas con un Maestro-detalle	3	0,15	Pruebas con un Maestro-detalle
	Seguridad	20%	5	1	XSS, CSRF, Clickjaking protection, SQL injection, SSL/HTTPS enforcement, Host Header validation, Session security	4	0,8	Session security, XSRF/CSRF, CORS, XSS, HTTPS enforcement	3	0,6	CSRF, Session security,Authentication Drivers
	Precisión	10%	4	0,4	Evaluación con un mantenimiento simple	5	0,5	Evaluación con un mantenimiento simple	5	0,5	Evaluación con un mantenimiento simple
	Idoneidad	10%	4	0,4	Rápida implementación, menor rendimiento	4	0,4	Rápido rendimiento, lenta implementación	1	0,1	
	Interoperabilidad	10%	5	0,5		5	0,5		5	0,5	
	Compatibilidad	5%	5	0,25		5	0,25	.net Standard	5	0,25	
	Multiproyecto	5%	5	0,25	Nativo	4	0,2	Lógico	3	0,15	Solo con AWS
	Arquitectura	5%	5	0,25	MVT	4	0,2	MVC	4	0,2	MVC

	Escalabilidad	5%	5	0,25	vertical y horizontal	5	0,25	vertical y horizontal	1	0,05	No es escalable
Fiabilidad	Madurez	15%	5	0,75	Evaluación con un mantenimiento simple	5	0,75	Evaluación con un mantenimiento simple	5	0,75	Evaluación con un mantenimiento simple
	Disponibilidad	10%	5	0,5		5	0,5		5	0,5	
	Tolerancia a fallos	15%	3	0,45		4	0,6		2	0,3	
	Recuperabilidad	10%	4	0,4		5	0,5		3	0,3	
	Consistencia	15%	5	0,75		5	0,75		4	0,6	
	Precisión	15%	4	0,6	Pruebas con un Maestro-detalle	5	0,75	Pruebas con un Maestro-detalle	3	0,45	Pruebas con un Maestro-detalle
	Autocontención	5%	5	0,25		5	0,25		5	0,25	
	Completidud	5%	5	0,25		5	0,25		5	0,25	
	Robustez / Integridad	5%	5	0,25		5	0,25		5	0,25	
	Idoneidad	5%	5	0,25		4	0,2		3	0,15	Limimantes al no ser escalable
Eficiencia	Comportamiento temporal	20%	4	0,8	Pruebas con QPS	5	1	Pruebas con QPS	3	0,6	Pruebas con QPS
	Utilización de recursos	20%	4	0,8	Pruebas con varias solicitudes al mismo tiempo	5	1	Pruebas con varias solicitudes al mismo tiempo	2	0,4	Pruebas con varias solicitudes al mismo tiempo
	Capacidad	10%	5	0,5		5	0,5		5	0,5	
	Eficacia de ejecución	10%	4	0,4	Pruebas con varias solicitudes al mismo tiempo	5	0,5	Pruebas con varias solicitudes al mismo tiempo	2	0,2	

	Accesibilidad	10%	5	0,5		5	0,5		5	0,5	
	ORM	10%	5	0,5	Django ORM	5	0,5	Entity Framework, Dapper ORM, Nhibernate	5	0,5	Eloquent ORM
	Comunicatividad	10%	5	0,5		5	0,5		5	0,5	
Usabilidad	Idoneidad Reconocimiento	5%	5	0,25		5	0,25		3	0,15	
	Capacidad de aprendizaje	25%	3	0,75	Nivel de abstracción	4	1		3	0,75	
	Operabilidad	20%	5	1		5	1		4	0,8	
	Factores humanos	10%	3	0,3	Propenso a errores de tabulación	5	0,5		4	0,4	
	Accesibilidad	5%	5	0,25		5	0,25		5	0,25	
	Comunicatividad	10%	5	0,5		5	0,5		5	0,5	
	Robustez / Integridad	5%	5	0,25	Pruebas con rollback ORM	5	0,25	Pruebas con rollback ORM	5	0,25	Pruebas con rollback ORM
	Comprensibilidad	10%	4	0,4		4	0,4		4	0,4	
	Madurez	5%	5	0,25	Django 1.0 (2008) - Django 3.2 (2021)	3	0,15	.net Core 1.0 (2019) - .net Core 6.0.2 (2022)	4	0,2	Laravel 1 (2011) - Laravel 9 LTS (2022)
	Complejidad	5%	4	0,2	Pruebas con despliegue	4	0,2	Pruebas con despliegue	5	0,25	Pruebas con despliegue
Seguridad	Confidencialidad	30%	1	0,3	No aplica	1	0,3	No aplica	1	0,3	No aplica
	Integridad	40%	5	2	ORM de Django abstrae toda la seguridad	4	1,6		4	1,6	
	No repudio	30%	1	0,3	No aplica	1	0,3	No aplica	1	0,3	No aplica

Compatibilidad	Coexistencia	50%	5	2,5	Pruebas implementando microservicios	5	2,5	Pruebas implementando microservicios	3	1,5	Pruebas implementando microservicios (solo con docker)
	Interoperabilidad	50%	5	2,5		5	2,5		5	2,5	
Mantenibilidad	Modularidad	15%	5	0,75		3	0,45		3	0,45	
	Reutilización	5%	5	0,25		5	0,25		4	0,2	
	Analizabilidad	5%	5	0,25		5	0,25		5	0,25	
	Modificabilidad	30%	4	1,2		4	1,2		3	0,9	
	Testabilidad	30%	5	1,5		5	1,5		5	1,5	
	Simplicidad	5%	5	0,25	Pruebas con Prototipado rápido	3	0,15	Pruebas con Prototipado rápido	3	0,15	Pruebas con Prototipado rápido
	Concisión	5%	5	0,25		4	0,2		4	0,2	
	Auto-descriptividad	5%	5	0,25		5	0,25		5	0,25	
Portabilidad	Adaptabilidad	10%	5	0,5		5	0,5		5	0,5	
	Instalabilidad	10%	5	0,5		4	0,4		5	0,5	
	Reemplazabilidad	30%	1	0,3	Web2py	1	0,3	Asp.net	1	0,3	Symfony
	Simplicidad	10%	5	0,5		5	0,5		5	0,5	
	Internacionalización	30%	5	1,5	i18n, i10n	5	1,5	i18n, i10n	5	1,5	i18n, i10n
	Autocontención	10%	5	0,5		5	0,5		5	0,5	
Exactitud	Trazabilidad	40%	3	1,2	Mediante plugins	5	2	Nativo	3	1,2	Mediante Plugins
	Consistencia	60%	5	3		5	3		4	2,4	
Integridad	Control de acceso	50%	5	2,5		5	2,5		5	2,5	

	Auditoría de acceso	50%	5	2,5	Integrado con el ORM	5	2,5	Integrado con el ORM	3	1,5	Mediante plugins
Testabilidad	Sencillez	15%	5	0,75		5	0,75		5	0,75	
	Instrumentación	20%	3	0,6		5	1		4	0,8	
	Auto-descriptividad	35%	5	1,75		5	1,75		5	1,75	
	Modularidad	20%	5	1		5	1		3	0,6	
	Accesibilidad	10%	5	0,5		5	0,5		5	0,5	
Flexibilidad	Sencillez	20%	4	0,8		5	1		5	1	
	Expandibilidad	10%	5	0,5		5	0,5		3	0,3	
	Generalidad	50%	5	2,5		5	2,5		5	2,5	
	Modularidad	20%	5	1		5	1		5	1	
Reutilización	Sencillez	25%	5	1,25		5	1,25		5	1,25	
	Generalidad	40%	5	2		5	2		5	2	
	Modularidad	20%	5	1		5	1		5	1	
	Independencia del software	15%	5	0,75		5	0,75		5	0,75	
Interoperabilidad	Modularidad	35%	5	1,75	Nativo	4	1,4	Lógico	3	1,05	Solo con AWS
	Comunicación de comunicaciones	50%	4	2	Plugins externos	5	2,5	Bibliotecas propias de .net Core	3	1,5	Plugins externos
	Datos comunes	15%	5	0,75		5	0,75		5	0,75	
Documentación	Integridad	30%	3	0,9		4	1,2		3	0,9	
	Operabilidad	15%	5	0,75		3	0,45		3	0,45	
	Exactitud	15%	5	0,75		5	0,75		5	0,75	
	Comunidad	20%	5	1		5	1		3	0,6	
	Apariencia	10%	4	0,4		5	0,5		5	0,5	
	Contenido multilingue	10%	5	0,5		5	0,5		2	0,2	Solo inglés

Siguiendo los pasos del método propuesto, en los resultados obtenidos de las subcaracterísticas, se aplica la sumatoria de las ponderaciones por característica de cada framework y se obtiene la valoración total de cada característica al aplicar la fórmula 2. Finalmente, se suman las columnas de valoración total, arrojando un resultado sobre 5 puntos para cada framework y de manera opcional, se puede calcular el porcentaje de la valoración obtenida. Los resultados se pueden observar en la tabla 28.

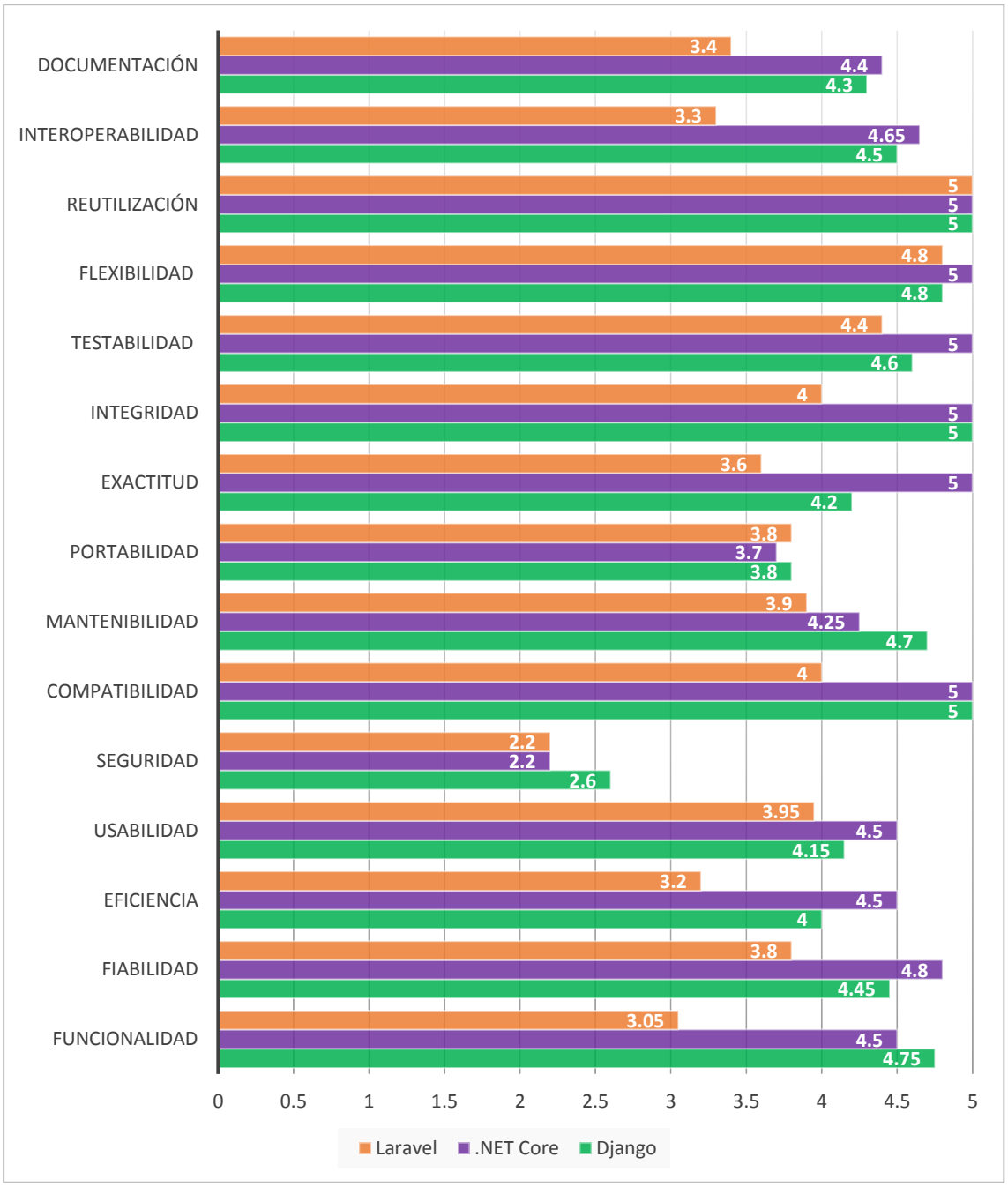
Tabla 28 Resultados de la evaluación de las características

Característica	Pct.	∑ ponderación Django	Valoración Total	∑ ponderación .NET Core	Valoración Total	∑ ponderación Laravel	Valoración Total
Funcionalidad	8.00%	**4.75**	**0.38**	**4.5**	**0.36**	**3.05**	**0.24**
Fiabilidad	16.00%	**4.45**	**0.71**	**4.8**	**0.768**	**3.8**	**0.61**
Eficiencia	8.00%	**4**	**0.32**	**4.5**	**0.36**	**3.2**	**0.26**
Usabilidad	13.00%	**4.15**	**0.54**	**4.5**	**0.585**	**3.95**	**0.51**
Seguridad	5.00%	**2.6**	**0.13**	**2.2**	**0.11**	**2.2**	**0.11**
Compatibilidad	6.00%	**5**	**0.30**	**5**	**0.3**	**4**	**0.24**
Mantenibilidad	13.00%	**4.7**	**0.61**	**4.25**	**0.5525**	**3.9**	**0.51**
Portabilidad	8.00%	**3.8**	**0.30**	**3.7**	**0.296**	**3.8**	**0.30**
Exactitud	4.00%	**4.2**	**0.17**	**5**	**0.2**	**3.6**	**0.14**
Integridad	2.00%	**5**	**0.10**	**5**	**0.1**	**4**	**0.08**
Testabilidad	3.00%	**4.6**	**0.14**	**5**	**0.15**	**4.4**	**0.13**
Flexibilidad	3.00%	**4.8**	**0.14**	**5**	**0.15**	**4.8**	**0.14**
Reutilización	3.00%	**5**	**0.15**	**5**	**0.15**	**5**	**0.15**
Interoperabilidad	5.00%	**4.5**	**0.23**	**4.65**	**0.2325**	**3.3**	**0.17**
Documentación	3.00%	**4.3**	**0.13**	**4.4**	**0.132**	**3.4**	**0.10**
Total			**4.35**		**4.45**		**3.70**
Total pct.	**100%**		**87.01%**		**88.92%**		**73.99%**

Fuente: Elaboración propia

En la figura 5 se observa en el gráfico de barras los puntajes obtenidos en las características por cada framework analizado.

Figura 5 Resultado de la evaluación de las subcaracterísticas



Fuente: Elaboración propia

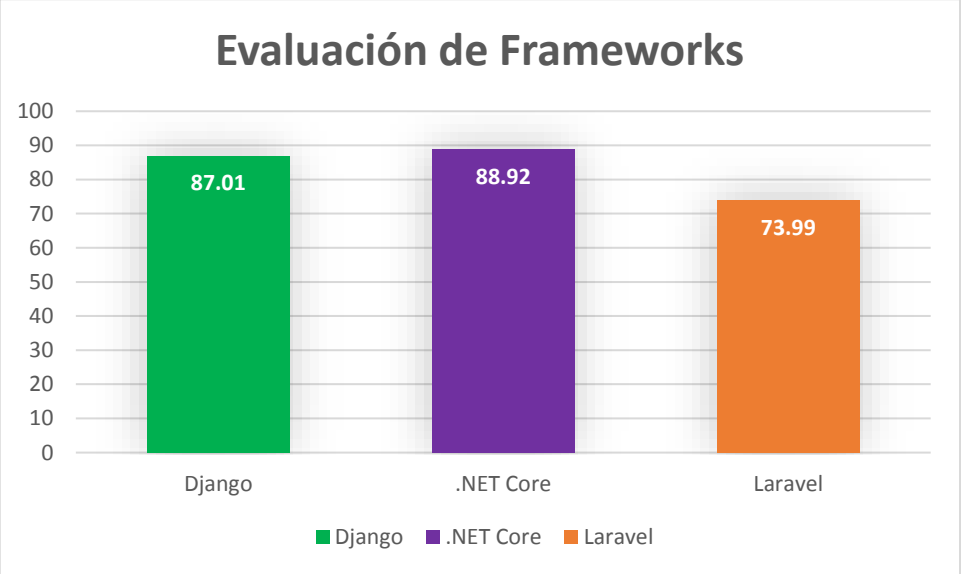
Se puede observar que los frameworks Django y .Net Core tienen una mayor puntuación, que se traduce en una posible aceptación en comparación a Laravel. Entre las valoraciones que destacan para Laravel se evidencia que es un framework no escalable, posee pocas

funcionalidades de seguridad y una menor eficacia de ejecución, además, su documentación no es multilingüe.

4.3. Análisis de evaluación del método propuesto

Los resultados de la aplicación del método de evaluación propuesto, arrojan que los frameworks Django y Laravel son superiores a Laravel en cuanto a la calidad de sus características y subcaracterísticas. Como se puede ver en la figura 6 Django y Laravel comparten un porcentaje mayor al de Laravel, por lo que se puede mencionar que tanto Django como Laravel cumplen con los requisitos para el desarrollo del software UDA-ERP.

Figura 6 Gráfico evaluación de frameworks



Fuente: Elaboración propia

En base a los resultados, también se puede mencionar que Django es un framework que facilita el desarrollo de aplicaciones web de manera rápida y segura, por otro lado, ASP .NET Core proporciona en el desarrollo aplicaciones web con características de eficacia y fiabilidad. En miras a reducir el tiempo de desarrollo del sistema UDA-ERP, el equipo se decide por Django.

5. Conclusiones

Los métodos de evaluación de calidad de software son importantes en el desarrollo de aplicaciones de alto nivel, ya que facilitan a los arquitectos de software a realizar un proceso de desarrollo que refleje el cumplimiento de los requisitos o especificaciones funcionales.

A través del presente trabajo se ha definido un método de evaluación de calidad de herramientas de desarrollo de software libre, basado en diferentes modelos de calidad existentes y puesto a prueba en el proceso de selección del framework adecuado para la realización del software UDA-ERP.

El método propuesto, ayudó a evaluar 3 frameworks de libre distribución (Django, .NET Core, Laravel), facilitando la selección de Django como el adecuado para migración del software UDA-ERP, cumpliendo con los requerimientos propuestos.

Con el método de evaluación de calidad de herramientas de desarrollo de software libre propuesto, el desarrollador obtiene información y valoración de cada herramienta de acuerdo a sus características y subcaracterísticas, permitiendo al arquitecto del software el identificar la herramienta que se adapte a sus necesitades y objetivos.

Como trabajo futuro, se plantea la evaluación del método propuesto en nuevos proyectos de desarrollo como apoyo para el análisis de las herramientas de desarrollo de software libre, basados en criterios propios de evaluación, como podrían ser frameworks de análisis de datos o desarrollo móvil.

6. Bibliografía

Abu Bakar, H. K. M., & Razali, R. (2013). A preliminary review of legacy information systems evaluation models. *International Conference on Research and Innovation in Information Systems, ICRIIS*, 314–318. https://doi.org/10.1109/ICRIIS.2013.6716728

Addo-Tenkorang, R., & Helo, P. (2011). Enterprise Resource Planning (ERP): A Review Literature Report. San Francisco, USA. https://doi.org/10.13140/2.1.3254.7844

Adnan, R., & Bassem, M. (2006). A New Software Quality Model for Evaluating COTS Components. *Journal of Computer Science*, *2*. https://doi.org/10.3844/jcssp.2006.373.381

Alchimowicz, B., & Nawrocki, J. R. (2016). The COCA quality model for user documentation. *Software Quality Journal*, *24*(2), 205–230. https://doi.org/10.1007/S11219-014-9252-4/TABLES/9

Anand, A., Krishna, A., Tiwari, R., & Sharma, R. (2018). Comparative analysis between proprietary software VS. Open-source software VS. free software. In *PDGC 2018 - 2018 5th International Conference on Parallel, Distributed and Grid Computing* (pp. 144–147). Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/PDGC.2018.8745951

Astudillo-Rodríguez, C., Crespo-Martínez, E., & Andrade-Dueñas, I. (2018). UDA - ERP: Emprendimiento y Gestión de recursos empresariales. La llave para la vinculación empresarial | Memorias y Boletines de la Universidad del Azuay. *Memorias Y Boletines De La Universidad Del Azuay*, *1(XIV)*, 92–106. Retrieved from https://revistas.uazuay.edu.ec/index.php/memorias/article/view/182

Aversano, L., & Tortorella, M. (2010). Evaluating the Quality of Free/Open Source Systems: A Case Study. *Lecture Notes in Business Information Processing*, *73 LNBIP*, 119–134. https://doi.org/10.1007/978-3-642-19802-1_9

Bertoa, M., & Vallecillo, A. (2003). Quality Attributes for COTS Components.

Camilo, J. R. M., L'Erario, A., Pagotto, T., & Fabri, J. A. (2018). A process for distributed software evolution: A proprietary software case study. In *Proceedings - International Conference on Software Engineering* (pp. 44–53). IEEE Computer Society. https://doi.org/10.1145/3196369.3196376

Curie, D. H., Jaison, J., Yadav, J., & Fiona, J. R. (2019). Analysis on Web Frameworks. *Journal of Physics: Conference Series*, *1362*(1), 012114. https://doi.org/10.1088/1742-6596/1362/1/012114

Dhir, S., & Dhir, S. (2017). Adoption of open-source software versus proprietary software: An exploratory study. *Strategic Change*, *26*(4), 363–371. https://doi.org/10.1002/jsc.2137

Dixit, D. A. (2013). CBQM: Component Based Quality Model. In *Reliability, Infocom Technologies and Optimization (ICRITO)(Trends and Future Directions), 2015 4th International Conference on* (pp. 1–5). Citeseer.

Django overview | Django. (n.d.). Retrieved March 3, 2022, from https://www.djangoproject.com/start/overview/

Dreyfus, P. (1998). The second wave netscape on usability in the services-based Internet. *IEEE Internet Computing*, *2*(2), 36–40. https://doi.org/10.1109/4236.670681

Economides, N., & Katsamakas, E. (2006). Two-sided competition of proprietary vs. open source technology platforms and the implications for the software industry. *Management Science*, *52*(7), 1057–1071. https://doi.org/10.1287/MNSC.1060.0549

Franco-Bedoya, O., Ameller, D., Costal, D., & Franch, X. (2017). Open source software ecosystems: A Systematic mapping. *Information and Software Technology*, *91*, 160–185. https://doi.org/10.1016/j.infsof.2017.07.007

Gomez-Diaz, T. (2015). Software libre, software de código abierto, licencias. Donde se propone un procedimiento de distribución de software y datos de investigación. https://doi.org/10.5281/ZENODO.31547

ISO. (2011). *ISO - ISO/IEC 25010:2011 - Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models* (1st ed.). Retrieved from https://www.iso.org/standard/35733.html

Kaluža, M., Kalanj, M., & Vukelić, B. (2019). A COMPARISON OF BACK-END FRAMEWORKS FOR WEB APPLICATION DEVELOPMENT. *Zbornik Veleučilišta u Rijeci*, *7*(1), 317–332. https://doi.org/10.31784/ZVR.7.1.10

Kan, S. H. (2002). *Metrics and Models in Software Quality Engineering* (2nd ed.). USA: Addison-Wesley Longman Publishing Co., Inc.

Kountouridou, N., Antoniou, P., & Stamelos, I. (2016). A Comprehensive Approach for Implementing an Open Source ERP in a Greek Industry. In *Proceedings of the 20th Pan-Hellenic Conference on Informatics* (pp. 1–5). New York, NY, USA: ACM. https://doi.org/10.1145/3003733.3003744

Laaziri, M., Benmoussa, K., Khoulji, S., & Kerkeb, M. L. (2019). A Comparative study of PHP frameworks performance. *Procedia Manufacturing*, *32*, 864–871. https://doi.org/10.1016/J.PROMFG.2019.02.295

Laaziri, M., Benmoussa, K., Khoulji, S., Larbi, K., & Yamami, A. (2019). A comparative study of laravel and symfony PHP frameworks. *International Journal of Electrical and Computer Engineering*, *9*, 704–712. https://doi.org/10.11591/ijece.v9i1.pp704-712

Laravel - The PHP Framework For Web Artisans. (n.d.). Retrieved March 3, 2022, from https://laravel.com/

Levine, S. S., & Prietula, M. (2013). Open Collaboration for Innovation: Principles and Performance. *SSRN Electronic Journal*. https://doi.org/10.2139/SSRN.1096442

Likert, R. (1932). A technique for the measurement of attitudes. *Archives of Psychology*, *22 140*, 55.

McCall, J. A., Richards, P. K., & Walters, G. F. (1977). *Factors in software quality. volume i. concepts and definitions of software quality*. GENERAL ELECTRIC CO SUNNYVALE CA.

Miguel, J. P., Mauricio, D., & Rodriguez, G. (2014). A Review of Software Quality Models for the Evaluation of Software Products. *International Journal of Software Engineering & Applications*, *5*(6), 31–53. https://doi.org/10.5121/ijsea.2014.5603

Nistala, P., Nori, K. V., & Reddy, R. (2019). Software quality models: A systematic mapping study. *Proceedings - 2019 IEEE/ACM International Conference on Software and System Processes, ICSSP 2019*, 125–134. https://doi.org/10.1109/ICSSP.2019.00025

Panduwiyasa, H., Saputra, M., Azzahra, Z. F., & Aniko, A. R. (2021). Accounting and Smart System: Functional Evaluation of ISO/IEC 25010:2011 Quality Model (a Case Study). *IOP Conference Series: Materials Science and Engineering*, *1092*(1), 012065. https://doi.org/10.1088/1757-899X/1092/1/012065

PYPL PopularitY of Programming Language index. (2021, July). Retrieved July 28, 2021, from https://pypl.github.io/PYPL.html

Rappaport, J. (2018). Is Proprietary Software Unjust? Examining the Ethical Foundations of Free Software. *Philosophy and Technology*, *31*(3), 437–453. https://doi.org/10.1007/s13347-017-0294-y

Sheoran, K., & Sangwan, O. P. (2015). An Insight of software quality models applied in predicting software quality attributes: A comparative analysis. *2015 4th International Conference on Reliability, Infocom Technologies and Optimization: Trends and Future Directions, ICRITO 2015*. https://doi.org/10.1109/ICRITO.2015.7359355

Singh, A., Bansal, R. ., & Jha, N. (2015). Open Source Software vs Proprietary Software. *International Journal of Computer Applications*, *114*(18), 26–31. https://doi.org/10.5120/20080-2132

Sirshar, M., Ali, A., & Ibrahim, S. (2019). A Comparitive Analysis Between Open Source And Closed Source Software in Terms of Complexity and Quality Factors. https://doi.org/10.20944/preprints201912.0063.v1

Stack Overflow Developer Survey 2021. (2021, May). Retrieved August 11, 2021, from https://insights.stackoverflow.com/survey/2021#technology

Stallman, R. (1986). What is free software? - GNU Project - Free Software Foundation. Retrieved June 11, 2021, from https://www.gnu.org/philosophy/free-sw.html

Subhani, S. (2020). Is Open Source Software the Future of Software Development? *International Journal of Business*, *10*(1). https://doi.org/10.30845/ijbht.v10n1p2

Suradi, N. R. M., Kahar, S., & Jamaluddin, N. A. A. (2018). Identification of software quality characteristics on academic application in higher education institution (HEI). *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, *10*(2–7), 133–136.

Talai, A., & Bouras, Z. E. (2017). Software evolution based activity diagrams. In *ICIT 2017 - 8th International Conference on Information Technology, Proceedings* (pp. 82–88). Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/ICITECH.2017.8079949

Tyagi, S., Kumar, D., & Kumar, S. (2019). Open source software: analysis of available

reliability models keeping security in the forefront. *International Journal of Information Technology (Singapore)*. https://doi.org/10.1007/s41870-019-00293-y

What is ASP.NET Core? | .NET. (n.d.). Retrieved March 3, 2022, from https://dotnet.microsoft.com/en-us/learn/aspnet/what-is-aspnet-core

Yadav, S., & Kishan, B. (2020). Analysis and assessment of existing software quality models to predict the reliability of component-based software. *International Journal of Emerging Trends in Engineering Research*, *8*(6), 2824–2840. https://doi.org/10.30534/IJETER/2020/96862020

Yamami, A. El, Laaziri, M., Benmoussa, K., Khoulji, S., Larbi, K. M., & Majida, L. (2019). Elaboration of an Intelligent Decision Support System (IDSS) for the improvement of quality and pedagogical management university courses View project strategic alignment View project A comparative study of laravel and symfony PHP frameworks. *International Journal of Electrical and Computer Engineering (IJECE)*, *9*(1), 704–712. https://doi.org/10.11591/ijece.v9i1.pp704-712

Zuria, N. (2016). User Centric Software Quality Model For Sustainability: A Review. https://doi.org/10.18178/lnse.2016.4.3.250

7. Anexos

Anexo 1: Tabla descriptiva de características y subcaracterísticas de los modelos de calidad

Característica	Subcaracterística	Descripción
Funcionalidad	Completitud funcional	Grado en el cual el conjunto de funcionalidades cubre todas las tareas y los objetivos especificados por el usuario.
	Corrección funcional	Capacidad del producto para proveer resultados correctos con el nivel de precisión requerido.
	Pertinencia funcional	Capacidad del producto para proporcionar un conjunto apropiado de funciones para tareas y objetivos especificados por el usuario.
	Seguridad	Capacidad el producto de manejar un cifrado para de proteger los datos, interfaces con funcionalidad para autenticación de usuarios y un mecanismo de auditoría.
	Precisión	Evalúa la exactitud de resultados obtenidos, de acuerdo a las especificaciones de los requisitos de usuario.
	Idoneidad	Especifica qué tan bien el componente se ajusta a los requisitos del usuario.
	Interoperabilidad	indica si el formato de los datos manejados por el producto cumple con algún estándar.
	Compatibilidad	Refleja el grado en que un componente puede usarse y funcionar correctamente en diferentes entornos.
Fiabilidad	Madurez	Capacidad del sistema para satisfacer las necesidades de fiabilidad en condiciones normales.
	Disponibilidad	Capacidad del componente de estar operativo y accesible cuando se requiera.
	Tolerancia a fallos	Capacidad del componente para operar correctamente en presencia de fallos de hardware o software.
	Recuperabilidad	Capacidad del componente para recuperar datos afectados y reestablecer el estado del sistema en caso de fallos.
	Consistencia	Indica si el componente contiene notación, terminología y simbología uniformes dentro de sí mismo
	Precisión	Otorga la precisión requerida en cálculos y resultados.
	Autocontención	Indica si el componente realiza todas sus funciones explícitas e implícitas dentro de sí mismo.
	Completitud	Especifica si todas las partes del componente están presentes y cada parte está completamente desarrollada.
	Robustez / Integridad	Indica si el componente continúa funcionando en circunstancias que no fueron especificadas en los requerimientos.
	Idoneidad	Especifica qué tan bien el componente se ajusta a los requisitos del usuario.
Eficiencia	Comportamiento temporal	Mide los tiempos de respuesta y procesamiento de un sistema, al desarrollar sus funciones en determinadas condiciones.
	Utilización de recursos	Indica la cantidad y recursos utilizados, cuando el software desempeña su función bajo condiciones determinadas.
	Capacidad	Grado en que los límites máximos de un producto cumplen con los requisitos.
	Eficacia de ejecución	Proporciona un tiempo de procesamiento mínimo por ejecución.
	Eficiencia de almacenamiento	Proporciona requisitos mínimos de almacenamiento durante el funcionamiento.
	Eficiencia del dispositivo	Indica si el componente cumple su propósito sin desperdiciar recursos.
	Responsabilidad	Indica si los segmentos críticos del código, pueden instrumentarse mediante conductos para medir el tiempo, ramas específicas, etc.

	Accesibilidad	Indica si el componente facilita el uso selectivo de sus partes.
	Comunicatividad	Facilita la especificación de entradas y proporciona salidas cuya forma y contenido son fáciles de asimilar.
Usabilidad	Idoneidad Reconocimiento	Capacidad del producto que apoya al usuario a entender si el software es adecuado para sus necesidades.
	Capacidad de aprendizaje	Capacidad del componente que facilita al usuario el aprendizaje de la aplicación.
	Operabilidad	Capacidad del producto que permite al usuario operarlo y controlarlo con facilidad.
	Factores humanos	Capacidad del sistema para proteger al usuario de cometer errores.
	Estética de la interfaz de usuario	Capacidad de la interfaz para satisfacer la interacción con el usuario.
	Accesibilidad	Capacidad del producto para ser utilizado por usuarios con determinadas características y/o discapacidades.
	Comunicatividad	Facilita la especificación de entradas y proporciona salidas cuya forma y contenido son fáciles de asimilar.
	Robustez / Integridad	Indica si el componente continúa funcionando en circunstancias que no fueron especificadas en los requerimientos.
	Comprensibilidad	Determina el impacto directo en la comprensibilidad del componente.
	Madurez	Indica la cantidad de versiones comerciales del componente y el intervalo entre esas versiones.
	Complejidad	Indica la complejidad de los componentes al integrarlos y usarlos dentro de un producto o sistema de software.
Seguridad	Confidencialidad	Garantiza en qué medida la información es accesible únicamente para usuarios con acceso autorizado.
	Integridad	Indica la capacidad de un componente para evitar accesos no autorizados y la modificación de programas y/o datos.
	No repudio	Indica en qué medida se puede verificar la efectividad de acciones o eventos.
	Responsabilidad	Indica las acciones para rastreo de un usuario no autorizado.
	Autenticidad	Indica en qué medida se puede probar la identidad de un sujeto o recurso.
Compatibilidad	Coexistencia	Indica si un producto puede realizar sus funciones de manera eficiente mientras comparte un entorno y recursos comunes con otros productos, sin afectar negativamente a ninguno.
	Interoperabilidad	Indica si 2 o más componentes pueden intercambiar información y utilizarla.
Mantenibilidad	Modularidad	Indica si los componentes de un sistema se pueden modificar con un impacto mínimo en los demás componentes.
	Reutilización	Indica si un componente puede ser implementado en más de un sistema.
	Analizabilidad	Diagnóstico de deficiencias o causas de fallas para identificar componentes a modificar.
	Modificabilidad	Indica si es factible el modificar un producto sin introducir defectos o degradar la calidad del producto existente.
	Testabilidad	Indica las pruebas que se pueden realizar para determinar si se han cumplido los criterios de prueba.
	Simplicidad	Se refiere a una implementación de funciones de la manera más comprensible.
	Concisión	Se refiere a la implementación de funciones con una cantidad mínima de código.
	Auto-descriptividad	Indica los atributos del software que proporcionan una explicación de la implementación de una función.
Portabilidad	Adaptabilidad	Indica si el producto se puede adaptar en diferentes entornos de uso o en evolución.
	Instalabilidad	Indica la eficacia con la que se puede instalar y/o desinstalar un producto o sistema.
	Reemplazabilidad	Indica si un producto puede reemplazar a otro producto comparable.

	Simplicidad	Señala la implementación de funciones de la manera más comprensible.
	Independencia del software	Indica los atributos del software que determinan su dependencia del entorno del software.
	Independencia del dispositivo	Indica los atributos del software que determinan su dependencia del sistema de hardware.
	Autocontención	Indica si el componente realiza todas sus funciones explícitas e implícitas dentro de sí mismo.
Exactitud	Trazabilidad	Indica los atributos del software que proporcionan un hilo conductor (desde los requisitos, hasta la implementación), con respecto al entorno operativo y de desarrollo específico.
	Completitud	Se refiere a la implementación completa de las funciones requeridas.
	Consistencia	Indica si el componente contiene notación, terminología y simbología uniformes dentro de sí mismo
Integridad	Control de acceso	Indica el control de acceso al software y los datos.
	Auditoría de acceso	Especifica la auditoría del acceso al software y a los datos.
Testabilidad	Sencillez	Atributo del software orientado a implementar funciones comprensibles. (Por lo general, evita prácticas que aumentan la complejidad)
	Instrumentación	Atributo del software para medir el uso o identificar errores.
	Auto-descriptividad	Atributo del software que define la implementación de una función.
	Modularidad	Atributo del software que proporciona una estructura de módulos altamente independientes.
	Responsabilidad	Señala que los segmentos críticos de código pueden instrumentarse con sondas para medir el tiempo, ramas específicas, etc.
	Accesibilidad	Indica que el código es accesible en la medida en que facilita el uso selectivo de sus partes.
	Estructuración	Indica que el código posee la característica de estructuración, en la medida en que posee un patrón definido de organización de sus partes interdependientes.
Flexibilidad	Sencillez	Atributo del software que facilita la implementación de funciones de un modo comprensible. (Por lo general, evita prácticas que aumentan la complejidad)
	Expandibilidad	Atributo del software que permite ampliar los requisitos de almacenamiento de datos o las funciones computacionales.
	Generalidad	Atributo del software que brinda amplitud en las funciones realizadas.
	Modularidad	Atributo del software que facilita la estructura de módulos altamente independientes.
	Estructuración	El código posee la característica de estructuración, en la medida en que posee un patrón definido de organización de sus partes interdependientes.
	Aumentabilidad	El código posee la característica de capacidad de aumento, en la medida en que puede adaptarse fácilmente a la expansión de las funciones computacionales de los componentes o los requisitos de almacenamiento de datos.
Reutilización	Sencillez	Atributo del software que facilita la implementación de funciones de un modo comprensible. (Por lo general, evita prácticas que aumentan la complejidad)
	Generalidad	Atributo del software que brinda amplitud en las funciones realizadas.
	Modularidad	Atributo del software que facilita la estructura de módulos altamente independientes.
	Independencia del software	Atributo del software que determinan su dependencia del entorno del software.
	Independencia de la máquina	Atributo del software que determinan su dependencia del sistema de hardware.
Interoperabilidad	Modularidad	Atributo del software que facilita la estructura de módulos altamente independientes.

	Comunicación de comunicaciones	Atributo del software que proporcionan el uso de representaciones de datos estándar.
	Datos comunes	Atributo del software que permiten la implementación de una función con un mínimo de código.
Comprensibilidad	Consistencia	El código posee notación, terminología y simbología uniformes dentro de sí mismo, y consistencia externa en la medida en que el contenido es rastreable a los requisitos.
	Auto-descriptividad	El código posee información para que un lector determine o verifique sus objetivos, suposiciones, restricciones, entradas, salidas, componentes y estado de revisión.
	Estructuración	El código posee la característica de estructuración en la medida en que posee un patrón definido de organización de sus partes interdependientes.
	Concisión	El código posee la característica de concisión en la medida en que no haya información excesiva.
	Legibilidad	El código posee la característica de legibilidad en la medida en que su función se percibe fácilmente leyendo el código.
Manejabilidad	Gestión de la calidad	Indica quienes están monitoreando constantemente lo que hacen para identificar posibles mejoras en la calidad de operación, producto, presupuestos, cronograma, servicios y todo lo relacionado a la empresa.

Fuente: Elaboración propia.

DECANO DE LA FACULTAD DE CIENCIAS DE LA ADMINISTRACIÓN.- Cuenca 2 de diciembre de 2021.- En atención a la solicitud que antecede, presentada por el señor LUIS FERNANDO GUERRERO ORTEGA (cód 74953) estudiante de la CARRERA DE INGENIERIA DE SISTEMAS Y TELEMÁTICA, solicita primera prórroga del trabajo de titulación "Método de evaluación de calidad de herramientas de desarrollo de software libre para construir ERP", que no pudo ser concluido a tiempo debido a situaciones personales de índole laboral. El protocolo fue aprobado por este organismo el 25 de mayo de 2021, mismo que debía ser entregado el 25 de noviembre de 2021. El Consejo de Facultad considerando las razones expuestas por el estudiante, así como el informe de la directora Ingeniera Catalina Astudillo Rodríguez, en el que señala que existe un avance del 70% del trabajo, resuelve conceder la **primera prórroga** para la presentación del trabajo final culminado, esto es máximo hasta el **25 de mayo de 2022.**

Ing. Oswaldo Merchán Manzano.
Decano de la Facultad de Ciencias de la Administración.

UNIVERSIDAD DEL AZUAY

Abogada Alexandra López Villacís, Secretaria de la Facultad de Ciencias de la Administración de la Universidad del Azuay.

CERTIFICA:

Que, el Consejo de Facultad en sesión del **25 de mayo de 2021**, conoció y aprobó la solicitud para la realización del trabajo de titulación, presentado por:

Estudiante: Guerrero Ortega Luis Fernando
Código: 74953
Tema: "MÉTODO DE EVALUACIÓN DE CALIDAD DE HERRAMIENTAS DE DESARROLLO DE SOFTWARE LIBRE PARA CONSTRUIR ERP"
Título: Ingeniero de Sistemas y Telemática
Director: Ing. Catalina Astudillo Rodríguez
Tribunal: Ing. Oswaldo Merchán Manzano e Ing. Priscila Cedillo Orellana

Plazo de presentación del trabajo de titulación: Se fijó como plazo para la entrega del trabajo de titulación, un periodo académico contado desde la fecha de aprobación del diseño del trabajo, esto es hasta el **25 de noviembre de 2021**.

Cuenca, 26 de mayo de 2021.

Abg. Alexandra López Villacís.
Secretaria de la Facultad de
Ciencias de la Administración

UNIVERSIDAD DEL AZUAY
Facultad de Ciencias de la Administración
SECRETARÍA

UNIVERSIDAD DEL AZUAY

CONVOCATORIA

Señores miembros del tribunal examinador,
Ing. Catalina Astudillo Rodríguez,
Ing. Oswaldo Merchán Manzano,
Ing. Priscila Cedillo Orellana,

Por disposición de la Junta Académica de la escuela de Ingeniería de Sistemas, se les convoca, a la sustentación de manera virtual del Protocolo del Trabajo de Titulación: **Método de evaluación de calidad de herramientas de desarrollo de software libre para construir ERP,** presentado por el estudiante Guerrero Ortega Luis Fernando con código 74953 previa a la obtención del título Ingeniero de Sistemas y Telemática, para el día**: Miércoles, 12 de mayo de 2021 a las 07h40.**

Tomar en cuenta que posterior a la sustentación del Diseño del Trabajo de Titulación, por ningún concepto se puede realizar modificaciones ni cambios en los documentos; únicamente, en caso de diseño aprobado con modificación, el Director adjuntará al esquema un oficio indicando que se procede con los cambios sugeridos.

Abg. Alexandra López Villacís
Secretaria de la Facultad

SECRETARÍA

UNIVERSIDAD DEL AZUAY

ACTA

SUSTENTACIÓN DE PROTOCOLO/DENUNCIA DEL TRABAJO DE TITULACIÓN

1. **Nombre del estudiante**: Guerrero Ortega Luis Fernando
2. **Código:** 74953
3. **Director sugerido:** Ing. Catalina Astudillo Rodríguez
4. **Codirector (opcional)**: Ing. Esteban Crespo Martínez
5. **Tribunal**: Ing. Oswaldo Merchán Manzano y Ing. Priscila Cedillo Orellana
6. **Título propuesto: Método de evaluación de calidad de herramientas de desarrollo de software libre para construir ERP**
7. Aceptado sin modificaciones: __X__
8. Aceptado con las siguientes modificaciones: _____

   _____

   _____

   _____
9. No aceptado : _____
10. Justificación:

    _____

    _____

    _____

Ing. Catalina Astudillo Rodríguez
Director sugerido

Abg. Alexandra López Villacís
Secretaria de la Facultad

UNIVERSIDAD DEL AZUAY
Facultad de Ciencias de la Administración
SECRETARÍA

1. **FECHA DE RECEPCIÓN DE PROTOCOLO:** 04-05-2021 **FIRMA: VIVIANA CALLE**

2. **REVISIÓN DE ESTADO ACADÉMICO DEL ALUMNO:**

NOMBRE: Guerrero Ortega Luis Fernando

CODIGO: 74953

CARRERA: Ingeniería de Sistemas

FECHA DE INICIO DE ESTUDIOS: 17 de marzo de 2014

FECHA DE CULMINACIÓN DE ESTUDIOS: _____

HOMOLOGACIONES: NO **CARRERA PROCEDENTE: _____**

CONVALIDACIONES: NO **UNIVERSIDAD PROCEDENTE: _____**

FECHA DE REVISIÓN: 04 de mayo de 2021 **FIRMA: MARÍA JOSÉ MOSCOSO**

Observaciones: Falta aprobar Idioma Extranjero III

Falta realizar y aprobar Prácticas Preprofesionales

---

3. **DE:** ABG. ALEXANDRA LÓPEZ, SECRETARIA

ASUNTO: ENVÍO DE PROTOCOLO DE TRABAJO DE TITULACIÓN

PARA: JUNTA ACADÉMICA DE LA CARRERA DE INGENIERÍA DE SISTEMAS

TÍTULO A OTORGARSE: INGENIERO DE SISTEMAS Y TELEMÁTICA

Observación: Falta aprobar Idioma Extranjero III

Falta realizar y aprobar Prácticas Preprofesionales

Le falta aprobar las asignaturas que está matriculado en este ciclo.

Fecha de revisión: 04 de mayo de 2021

FIRMA: **Abg. ALEXANDRA LÓPEZ VILLACÍS**

4.TITULO DEL TRABAJO:Método de evaluación de calidad de herramientas de desarrollo de software libre para construir ERP

REALIZADO EN EL CURSO DE METODOLOGÍA: ___x____ SI _____ NO

FECHA DE APROBACIÓN DEL CONSEJO DE FACULTAD: de mayo de 2021

DIRECTOR: Ing. Catalina Astudillo Rodrìguez

TRIBUNAL: Ing. Oswaldo Merchàn Manzano e Ing. Priscila Cedillo Orellana

Firma: RUTH CABRERA R.

Oficio Nro. 046-2021-DIST-UDA

Cuenca, 4 de mayo de 2021

Ingeniero,
Oswaldo Merchán Manzano
DECANO DE LA FACULTAD DE CIENCIAS DE LA ADMINISTRACIÓN
UNIVERSIDAD DEL AZUAY

De nuestras consideraciones,

La Junta Académica de la Escuela de Ingeniería de Sistemas y Telemática, reunida el día 4 de mayo del 2021, revisó la documentación del trabajo de titulación denominado **"MÉTODO DE EVALUACIÓN DE CALIDAD DE HERRAMIENTAS DE DESARROLLO DE SOFTWARE LIBRE PARA CONSTRUIR ERP"**, presentada por el estudiante **GUERRERO ORTEGA LUIS FERNANDO**, con código **74953**, estudiante de la Escuela de Ingeniería de Sistemas y Telemática, y revisado por **CATALINA ASTUDILLO RODRÍGUEZ**, previo a la obtención del título de Ingeniero de Sistemas y Telemática.

La Junta Académica considera que la documentación cumple con las normas legales y reglamentarias de la Universidad y de la Facultad de Ciencias de la Administración y designa como miembros del tribunal a **OSWALDO MERCHÁN** y **PRISCILA CEDILLO**, así por su digno intermedio, el conocimiento y aprobación por parte del Consejo de Facultad.

Atentamente,

0102668209 MARCOS PATRICIO ORELLANA CORDERO

Digitally signed by 0102668209 MARCOS PATRICIO ORELLANA CORDERO
Date: 2021.05.04 08:13:50 -05'00'

Ing. Marcos Orellana
Coordinador Escuela de Ingeniería de Sistemas y Telemática
Universidad del Azuay

Ing. Esteban Crespo
Miembro de Junta Académica
Universidad del Azuay

Ing. Andrés Patiño
Miembro de Junta Académica
Universidad del Azuay

Oficio Estudiante: Solicitud aprobación de Protocolo de Trabajo de Titulación

Cuenca, 03 de mayo de 2021

Ingeniero
Oswaldo Merchán Manzano
DECANO DE LA FACULTAD DE CIENCIAS DE LA ADMINISTRACIÓN
UNIVERSIDAD DEL AZUAY

De mi/ nuestra consideración,

Estimado Señor Decano, yo **Luis Fernando Guerrero Ortega** con C.I. **0106457419**, código estudiantil 74953; estudiante de la Carrera de Sistemas y Telemática, solicito muy comedidamente a usted y por su intermedio al Consejo de Facultad, la aprobación del protocolo de trabajo de titulación con el tema **"MÉTODO DE EVALUACIÓN DE CALIDAD DE HERRAMIENTAS DE DESARROLLO DE SOFTWARE LIBRE PARA CONSTRUIR ERP"** previo a la obtención del título de Ingeniero en Sistemas y Telemática, para lo cual adjunto la documentación respectiva.

Por la favorable acogida que brinde a la presente, anticipo mi agradecimiento.

Atentamente:

_____

Luis Fernando Guerrero Ortega

Estudiante de la Carrera de Sistemas y Telemática

Oficio Director: Revisión protocolo

Cuenca, 3 de mayo de 2021

Ingeniero
Oswaldo Merchán Manzano
DECANO DE LA FACULTAD DE CIENCIAS DE LA ADMINISTRACIÓN
UNIVERSIDAD DEL AZUAY

De mi consideración:

Yo, **Catalina Verónica Astudillo Rodríguez** informo que he revisado el protocolo de trabajo de titulación previo a la obtención del título de Ingeniero en Sistemas y Telemática, denominado **"MÉTODO DE EVALUACIÓN DE CALIDAD DE HERRAMIENTAS DE DESARROLLO DE SOFTWARE LIBRE PARA CONSTRUIR ERP"**, realizado por el estudiante **Luis Fernando Guerrero Ortega**, con código estudiantil 74953, protocolo que a mi criterio, cumple con los lineamientos y requerimientos establecidos por la carrera.

Por lo expuesto, me permito sugerir que sea considerado para la revisión y sustentación del mismo,

Sin otro particular, suscribo.

Atentamente

_____

Ing. Catalina Astudillo Rodríguez

Oficio Nro. 046-2021-DIST-UDA

Cuenca, 4 de mayo de 2021

Ingeniero,
Oswaldo Merchán Manzano
DECANO DE LA FACULTAD DE CIENCIAS DE LA ADMINISTRACIÓN
UNIVERSIDAD DEL AZUAY

De nuestras consideraciones,

La Junta Académica de la Escuela de Ingeniería de Sistemas y Telemática, reunida el día 4 de mayo del 2021, revisó la documentación del trabajo de titulación denominado **"MÉTODO DE EVALUACIÓN DE CALIDAD DE HERRAMIENTAS DE DESARROLLO DE SOFTWARE LIBRE PARA CONSTRUIR ERP"**, presentada por el estudiante **GUERRERO ORTEGA LUIS FERNANDO**, con código **74953**, estudiante de la Escuela de Ingeniería de Sistemas y Telemática, y revisado por **CATALINA ASTUDILLO RODRÍGUEZ**, previo a la obtención del título de Ingeniero de Sistemas y Telemática.

La Junta Académica considera que la documentación cumple con las normas legales y reglamentarias de la Universidad y de la Facultad de Ciencias de la Administración y designa como miembros del tribunal a **OSWALDO MERCHÁN** y **PRISCILA CEDILLO**, así por su digno intermedio, el conocimiento y aprobación por parte del Consejo de Facultad.

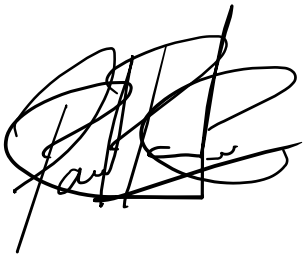
Atentamente,

0102668209
MARCOS PATRICIO
ORELLANA
CORDERO

Digitally signed by
0102668209 MARCOS
PATRICIO ORELLANA
CORDERO
Date: 2021.05.04
08:13:50 -05'00'

Ing. Marcos Orellana
Coordinador Escuela de Ingeniería de Sistemas y Telemática
Universidad del Azuay

Ing. Esteban Crespo
Miembro de Junta Académica
Universidad del Azuay



Ing. Andrés Patiño
Miembro de Junta Académica
Universidad del Azuay

UNIVERSIDAD DEL AZUAY

Rúbrica para evaluación del Protocolo de Trabajo de Titulación (Metodólogo y Director)

1.1. Nombre del Estudiante: Luis Fernando Guerrero Ortega / 74953

1.1.1. **Código:** 74953

1.2. Director sugerido: Ing. Astudillo Rodríguez Catalina Verónica

1.3. Docente metodólogo: Ing. Cedillo Orellana Irene Priscila

1.4. Codirector (opcional): Ing. Paúl Esteban Crespo Martínez

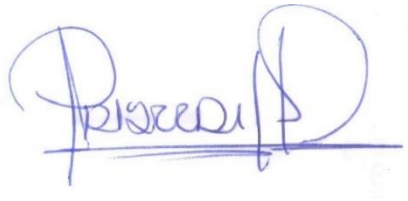
1.5. Título propuesto: Método de evaluación de calidad de herramientas de desarrollo de software libre para construir ERP

	DIRECTOR		METODÓLOGO	
	Cumple	**No cumple**	**Cumple**	**No cumple**
Línea de investigación				
1. ¿El contenido se enmarca en la línea de investigación seleccionada?	X		X	
Título Propuesto				
2. ¿Es informativo?	X		X	
3. ¿Es conciso?	X		X	
Estado del arte				
4. ¿Identifica claramente el contexto histórico, científico, global y regional del tema del trabajo?	X		X	
5. ¿Describe la teoría en la que se enmarca el trabajo	X		X	
6. ¿Describe los trabajos relacionados más relevantes?	X		X	
7. ¿Utiliza citas bibliográficas?	X		X	
Problemática				
8. ¿Presenta una descripción precisa y clara?	X		X	
9. ¿Tiene relevancia profesional y social?	X		X	
Pregunta de investigación				
10. ¿Presenta una descripción precisa y clara?	X		X	
11. ¿Tiene relevancia profesional y social?	X		X	
Hipótesis (opcional)				
12. ¿Se expresa de forma clara?	X		X	
13. ¿Es factible de verificación?	X		X	
Objetivo general				
14. ¿Concuerda con el problema formulado?	X		X	
15. ¿Se encuentra redactado en tiempo verbal infinitivo?	X		X	
Objetivos específicos				
16. ¿Permiten cumplir con el objetivo general?	X		X	
17. ¿Son comprobables cualitativa o cuantitativamente?	X		X	
Metodología				
18. ¿Se encuentran disponibles los datos y materiales mencionados?	X		X	
19. ¿Las actividades se presentan siguiendo una secuencia lógica?	X		X	

20. ¿Las actividades permitirán la consecución de los objetivos específicos planteados?	X		X	
21. ¿Las técnicas planteadas están de acuerdo con el tipo de investigación?	X		X	
Resultados esperados				
22. ¿Son relevantes para resolver o contribuir con el problema formulado?	X		X	
23. ¿Concuerdan con los objetivos específicos?	X		X	
24. ¿Se detalla la forma de presentación de los resultados?	X		X	
25. ¿Los resultados esperados son consecuencia, en todos los casos, de las actividades mencionadas?	X		X	
Supuestos y riesgos				
26. ¿Se mencionan los supuestos y riesgos más relevantes, en caso de existir?	X		X	
27. ¿Es conveniente llevar a cabo el trabajo dado los supuestos y riesgos mencionados?	X		X	
Presupuesto				
28. ¿El presupuesto es razonable?	X		X	
29. ¿Se consideran los rubros más relevantes?	X		X	
Cronograma				
30. ¿Los plazos para las actividades están de acuerdo con el reglamento?	X		X	
Citas y Referencias del documento				
31. ¿Se siguen las recomendaciones de normas internacionales para citar?	X		X	
Expresión escrita				
32. ¿La redacción es clara y fácilmente comprensible?	X		X	
33. ¿El texto se encuentra libre de faltas ortográficas?	X		X	

OBSERVACIONES METODÓLOGO:

OBSERVACIONES DIRECTOR:



…… ……..……

METODÓLOGO

Ing. Cedillo Orellana Irene Priscila

DIRECTOR

Ing. Astudillo Rodríguez Catalina Verónica

UNIVERSIDAD DEL AZUAY

Facultad de Ciencias de la Administración

Escuela de Ingeniería de Sistemas y Telemática

Método de evaluación de calidad de herramientas de desarrollo de software libre para construir ERP

Nombre del Estudiante(s):

Guerrero Ortega Luis Fernando

Director(a) sugerido(a):

Ing. Astudillo Rodríguez Catalina Verónica

Cuenca - Ecuador

2021

1. Datos Generales

1.1. Nombre del Estudiante

Guerrero Ortega Luis Fernando

1.1.1. Código

074953

1.1.2. Contacto

Teléfono:
Celular: +593 98 481 6102
Correo Electrónico: luixg22@es.uazuay.edu.ec

1.2. Director Sugerido: Ing. Astudillo Rodríguez Catalina Verónica

1.2.1. Contacto:

Celular: +593 98 710 9923
Correo Electrónico: cvastudillo@uazuay.edu.ec

1.3. Co-director sugerido: Ing. Paúl Esteban Crespo Martínez

1.3.1. Contacto: +593 99 680 4562

1.4. Asesor Metodológico: Ing. Cedillo Orellana Irene Priscila

1.5. Tribunal designado:

1.6. Aprobación:

1.7. Línea de Investigación de la Carrera:

1.7.1. Código LIDI: 0613 - Software, desarrollo y análisis de aplicaciones

1.7.2. Tipo de trabajo:

a) Proyecto de investigación
b) Investigación científica

1.8. Área de Estudio:

Calidad de Software

1.9. Título Propuesto:

Método de evaluación de calidad de herramientas de desarrollo de software libre para construir ERP.

1.10. Subtítulo:

1.11. Estado del proyecto

Proyecto nuevo

2. Contenido

2.1. Motivo de la Investigación:

Ante la inminente crisis económica ocasionada por el Covid-19, los desarrolladores ven la necesidad de buscar alternativas para mantener en continuidad sus proyectos de software, por lo que han optado por adoptar herramientas de desarrollo de software libre para evitar costos de licenciamiento, a partir de esto, los desarrolladores tienen varias opciones al momento de seleccionar una herramienta de desarrollo de software libre, sin embargo, la calidad de las herramientas es su principal preocupación, puesto que los proyectos tienen como objetivo producir software de la mejor calidad, superando las expectativas de los usuarios(Andrade et al., 2020). Las preocupaciones sobre la calidad se manifiestan como iniciativas para implementar métodos de evaluación que cumplan los estándares mínimos para lograr una mejora general en la calidad de las herramientas de desarrollo de software libre.

2.2. Problemática

El software libre conlleva toda una serie de ventajas sobre el software propietario por los derechos que otorga a sus usuarios. Algunas de estas ventajas pueden ser más apreciadas por los desarrolladores, otras ventajas por las empresas, y otras por las administraciones públicas (I. Hernández & J. M.,2019).

Las medidas sanitarias adoptadas por el gobierno del Ecuador, para combatir la pandemia, han provocado un cambio radical al que las instituciones deben adaptarse mediante acciones y estrategias ágiles e innovadoras para mantener su estabilidad económica (G. Navarro, 2020).

El presente trabajo de investigación, surge de la necesidad de seleccionar una herramienta de desarrollo de software libre, que mejor se ajuste a las necesidades del UDA-ERP. Uno de los principales problemas definidos del UDA-ERP se refiere a la

falta de recursos económicos debido a la pandemia ocasionada por el Covid-19, para cubrir este requerimiento es necesario utilizar herramientas de desarrollo de software libre, sin embargo, estas herramientas de software deben cumplir con estándares de calidad que permita evaluar y seleccionar la herramienta adecuada.

2.3. Pregunta de Investigación

¿Cuáles son las características y atributos que se deben considerar en un método para evaluar la calidad de herramientas de desarrollo de software libre, orientadas a la construcción de un software ERP?

2.4. Resumen

Un ERP (Enterprise Resource Planning) es un tipo de software que facilita la organización de las pequeñas y medianas empresas (PYMES) y que les permite optimizar sus procesos y recursos de manera que éstas puedan aumentar su productividad y competitividad en el mercado. Hoy en día, los sistemas ERP son fundamentales para gestionar miles de empresas en todo el mundo. UDA-ERP es un proyecto que vincula a la institución universitaria con la sociedad, debido a la pandemia ocasionada por el Covid-19, muchos recursos económicos se han limitado, siendo necesario buscar mecanismos que bajen los costos de licenciamiento, ya que el UDA-ERP está desarrollado en Oracle APEX. Las herramientas de código abierto son desarrolladas y distribuidas con una licencia que permite ver y utilizar el código libremente y sin restricciones, estas herramientas son perfectas para el desarrollo de software empresarial, evitando así cubrir gastos adicionales. El presente trabajo, propone un método para evaluar la calidad de herramientas de software libre para la construcción de un software ERP. Mediante un análisis y comparativa con base a criterios de madurez y documentación, se implementará este método para seleccionar la herramienta que cumpla las exigencias para el desarrollo de un software ERP.

2.5. Estado del Arte y marco teórico

Según el estándar ISO 8402 (1986), un modelo de calidad puede definirse como, el conjunto de factores de calidad, y de relaciones entre ellos, que proporciona una base para la especificación de requisitos de calidad y para la evaluación de la calidad de los componentes software.

Durante las últimas décadas, se han realizado estudios que analizan herramientas de desarrollo de software, mediante métricas basadas en metodologías de calidad de software; muy pocas enfocadas a herramientas de desarrollo de código abierto.

Para Python se han aplicado modelos de evaluación basados en las características de calidad propuestas en la norma ISO/IEC 9126, analizando los resultados obtenidos sobre las herramientas de desarrollo de software (Pyramid, Turbogear, Django y Web2PY) mostraron las fortalezas y debilidades de cada framework. En el caso de (M. Ríos et al., 2016) aplica esta norma para determinar que Django es la mejor herramienta para la implementación de desarrollo de sistemas web.

Otra metodología para evaluar la calidad de herramientas de desarrollo de software es mediante factores como Rapidez, Flexibilidad, Seguridad, Soporte, Multihilo y Control de datos. Se han aplicado estas métricas para un análisis comparativo entre Asp.Net y Php, para el cual Php resultó tener muchas más características útiles para el desarrollo, sin embargo, una de las razones se debe a que los desarrolladores se inclinan más hacia los desarrollos con herramientas de código abierto (E. Mina & S. Cedeño, 2018).

La norma ISO/IEC 25010 es una parte fundamental durante las definiciones de métodos de calidad de desarrollo de software. Según el estándar ISO 25010 (2011) el modelo de calidad determina qué características de calidad se tendrán en cuenta al evaluar las propiedades de un producto de software. Está compuesta por ocho características (Funcionalidad, desempeño, compatibilidad, usabilidad, fiabilidad, seguridad, mantenibilidad, portabilidad) que se encuentran representados en el modelo de calidad, el cual categoriza la calidad del producto en características y subcaracterísticas.

Un estudio realizado por A. Matute et al. (2020) utiliza las métricas de eficiencia con respecto al tiempo de respuesta definido en la norma ISO/IEC 25010 para un caso de estudio basado en los frameworks Laravel y Vue.js con un enfoque hacia el desarrollo de sistemas web.

La familia ISO/IEC 25000 es la más común al momento de evaluar la calidad de un producto de software (Calabrese & Muñoz, 2018); sin embargo, existen otros modelos de calidad de software implementados que permiten certificar y garantizar la calidad de un producto y sus procesos, tales como:

- TSP (Team Software Process): Modelo enfocado para equipos de desarrollo de software autodirigidos, orientado al desarrollo de productos con el mínimo defecto en tiempo y costos.
- FURPS: Modelo que se enfoca en funcionalidad, usabilidad, confiabilidad, desempeño y soportabilidad al momento desarrollar un producto de software.
- LSP: Método cuantitativo que aplica una evaluación, comparación y selección de sistemas de distintos tipos.

(Callejas Cuervo et al., 2017)

La combinación de estos modelos de calidad, permite establecer criterios para la óptima evaluación de calidad para herramientas de desarrollo de software libre.

2.6. Hipótesis

No aplica

2.7. Objetivo General

Definir un método de evaluación de calidad de herramientas de desarrollo de software libre para la construcción de software ERP.

2.8. Objetivos Específicos

1. Analizar los métodos de evaluación de calidad de herramientas de desarrollo de software existentes.
2. Definir el método de evaluación de calidad de herramientas de desarrollo de software libre.
3. Analizar el método de calidad de herramientas de desarrollo de software libre propuesto.

2.9. Metodología

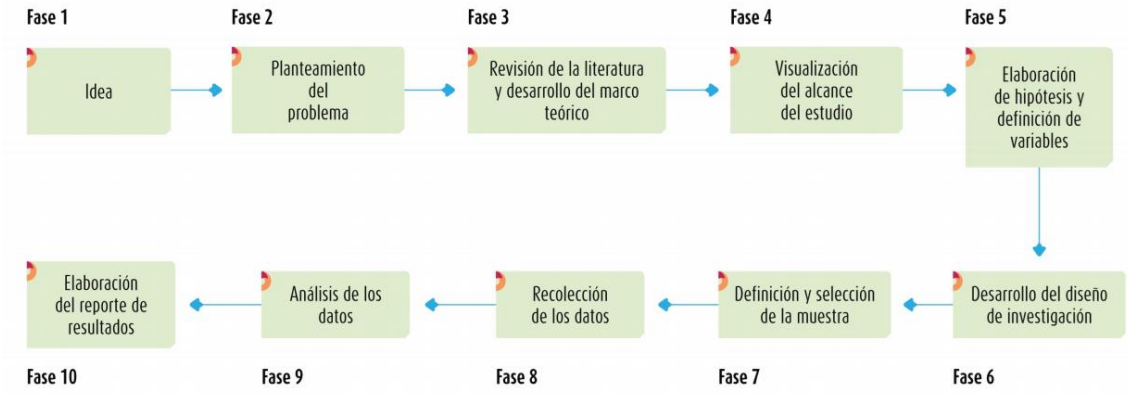


Figura 2.1: Modelo de investigación cuantitativo. (Hernández et al., 2014)

1. **Idea:** En la primera fase se originan las ideas de investigación que son planteadas, en este caso, la definición de un método de evaluación de calidad para herramientas de desarrollo de software libre.
2. **Planteamiento del problema:** Consiste en identificar el problema que da origen al propósito investigativo, en este caso, el conflicto que enfrenta el equipo de programadores al momento de seleccionar la herramienta de desarrollo de software libre que cumpla con estándares de calidad requeridos.
3. **Revisión de literatura y desarrollo del marco teórico:** En esta etapa, se procede a levantar la sustentación teórica con el fin de presentar un trabajo sostenible y posteriormente conocer los estudios que se han llevado a cabo en el campo del problema planteado. Para ello se realizará una revisión bibliográfica siguiendo el procedimiento definido por Kitchenham (2007).
4. **Visualización del alcance del estudio:** En esta fase se define el alcance que el estudio va a tener. En este caso, se pretende llegar a definir un método de calidad para herramientas de desarrollo de software libre.
5. **Elaboración de hipótesis y definición de variables:** Se definen las variables dependientes e independientes que serán observadas y medidas. Las variables planteadas como punto de partida serán obtenidas de la literatura revisada.
6. **Desarrollo del diseño de la investigación:** Esta etapa se refiere al plan que se hace con el fin de mostrar de manera práctica las preguntas de investigación, donde

se debe buscar cubrir los objetivos planteados. Para esta fase, con las variables específicas, se propone definir el método de evaluación de calidad para herramientas de desarrollo de software libre.

7. **Definición y selección de la muestra:** De acuerdo al alcance del estudio se definen los participantes, objetos, sucesos o colectividades de estudio. En esta fase, mediante un caso de aplicación, se seleccionan herramientas de desarrollo de software libre para evaluar el método definido.
8. **Recolección de los datos:** Se trata de la elaboración de un plan detallado de procedimientos que conduzcan a reunir los datos con un propósito específico, en este caso, se obtendrán los resultados del método de evaluación de calidad de herramientas de desarrollo de software libre del caso de aplicación.
9. **Análisis de los datos:** Esta fase permite conocer e interpretar la información recolectada con la finalidad de identificar puntos de valor. En esta etapa se analiza el resultado de la aplicación del método de calidad de evaluación sobre las herramientas de desarrollo de software libre definidas anteriormente.
10. **Elaboración del reporte de resultados:** En la fase final del método de investigación, se analizan los resultados y se elaboran los reportes. Para esta etapa, se presentan los resultados obtenidos del método de evaluación de calidad de herramientas de desarrollo de software libre en base al análisis de los datos obtenidos del caso de aplicación.

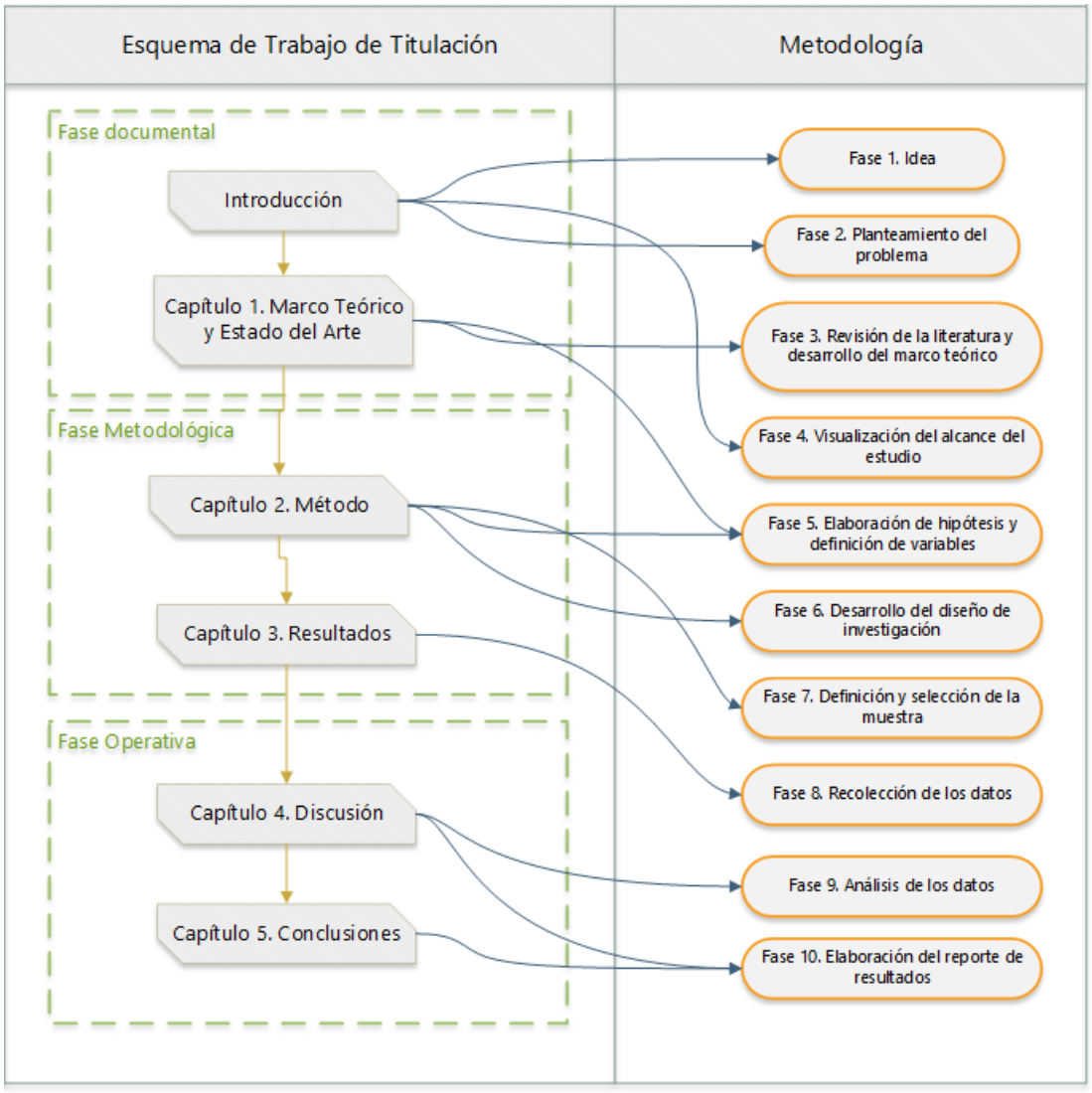


Figura 2.2: Proceso Cuantitativo. Fuente: Elaboración Propia

2.10. Alcances y resultados esperados

Mediante este trabajo de investigación se pretende definir un nuevo método de evaluación de calidad de herramientas de desarrollo de software libre que integre atributos basados en los análisis de los métodos existentes, de manera que permita seleccionar la que mejor se adapte a los requerimientos de un software ERP.

2.11. Supuestos y riesgos

Riesgos	**Probabilidad**	**Alternativas de solución**
La definición del método incluye atributos innecesarios.	Media	Realizar una breve revisión de la literatura de estándares de evaluación de calidad y determinar las métricas apropiadas para la investigación.
Incorrecta evaluación del método definido en las herramientas de desarrollo de software.	Media	Definir los pasos para evaluar una herramienta según el método de calidad definido.
Falta de conocimiento de las herramientas de desarrollo de software.	Media	Realizar una breve inducción acerca del funcionamiento de las herramientas de desarrollo de software a utilizar.
Más tiempo de desarrollo del previsto.	Alta	Determinar el alcance y establecer el tiempo necesario. Entregas parciales apropiadas y a tiempo.
Indisponibilidad de las personas claves para el alcance del proyecto.	Baja	Establecer un cronograma de actividades, que involucre tareas, responsabilidades y revisiones periódicas.
La falta de acceso a ciertos estándares de pago.	Media	Buscar alternativas de acceso a través de otros trabajos científicos o de herramientas y/o soluciones de libre acceso.

2.12. Presupuestos

Rubro	**Costo (USD)**	**Justificación**
Computadora	200	Realización del trabajo de investigación
Transporte	100	Traslados que haya que cumplir con el afán del desarrollo adecuado de la investigación
Internet	150	Consultas e investigación del estudio
Artículos de oficina	180	Copias, impresiones, anillados, hojas, carpetas, empastados, entre otros.
Costos salariales del Personal	3200	Tiempo de desarrollo e investigación en base a los costos salariales
Otros	100	Para solventar cualquier pago de carácter urgente
TOTAL	**3930**	

2.13. Financiamiento

Propio

2.14. Esquema tentativo

Introducción

Capítulo 1. Marco Teórico y Estado del Arte

1.1.1. Herramientas de desarrollo de software

1.1.2. Software libre

1.1.3. Software privativo

1.1.4. Software libre vs software privativo

1.1.5. Software ERP

1.1.6. Revisión de literatura de herramientas de desarrollo de software libre

1.1.7. Revisión de literatura de estándares de evaluación de calidad de software

Capítulo 2. Método

2.1. Análisis y comparación de los métodos de evaluación de calidad de herramientas de desarrollo de software

2.1.1 Métodos de evaluación de calidad.

2.1.2. Comparativa de los métodos de evaluación de calidad.

2.2. Definición del nuevo método de evaluación de calidad

2.2.1. Selección de las métricas de evaluación de calidad

2.2.2. Documentación

2.3. Materiales

2.3.1. Escenario de Pruebas

Capítulo 3. Resultados

3.1. Fases del método de evaluación de calidad propuesto

Capítulo 4. Discusión

4.1. Escenarios de evaluación del método propuesto

4.2. Análisis de evaluación del método propuesto

Capítulo 5. Conclusiones

2.15. Cronograma

Objetivo Específico	Actividad	Resultados Esperados	Semana																
			1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Analizar los métodos de evaluación de calidad de herramientas de desarrollo de software existentes.	Revisar la literatura para la definición del marco teórico	Marco teórico.																
	Revisar la literatura de herramientas de desarrollo de software libre.	Estado del arte.																
	Revisar la literatura de estándares de evaluación de calidad de software.																	
Definir el método de evaluación de calidad de herramientas de desarrollo de software libre.	Analizar y comparar los métodos de evaluación de calidad de herramientas de desarrollo de software existentes.	Análisis y comparación de los métodos de evaluación de calidad																
	Definir el método de evaluación de calidad para herramientas de desarrollo de software libre	Método de evaluación de calidad para herramientas de desarrollo de software libre																
	Definir el escenario de pruebas	Escenario de pruebas																
Analizar el método de calidad de herramientas de desarrollo de software libre propuesto.	Definir las fases del método de evaluación de calidad de desarrollo de software	Fases del método de evaluación de calidad de desarrollo de software libre																
	Ejecutar el escenario de pruebas	Escenarios de evaluación																
	Evaluar el método propuesto	Análisis de evaluación del método propuesto																

2.16. Referencias

Funes, A., & Dasso, A. (2014). Evaluación de Frameworks para Aplicaciones Web. Workshop de Investigadores en Ciencias de la Computación.

Rodríguez, C., & Enríquez, H. (2014). Características del desarrollo en Frameworks multiplataforma para móviles. Ingenium.

Haefliger, S., von Krogh, G., & Spaeth, S. (2008). Code Reuse in Open Source Software. Management Science , 54 (1), 180 - 193.

J. Molina, N. Loja, M. Ordóñez & Loaiza (2016). Evaluación de Frameworks en el Desarrollo de Aplicaciones Web con Python. https://doi.org/10.18294/relais.2016. 201-207.

Giles, M. (2020). Free Software That Businesses, Schools And Others Can Use During The COVID-19 Crisis. FORBES. https://www.forbes.com/sites/martingiles/2020/03/19/free-software-for-businesses-and-schools-covid19/?sh=2029630c752d

INTERNATIONAL STANDARDS ORGANIZATION (1986). ISO International Standard 8402: Quality Management and Quality AssuranceVocabulary.

Molina Ríos, J. R., Loja Mora, N. M., Zea Ordóñez, M. P., & Loaiza Sojos, E. L. (2016). Evaluación de los Frameworks en el Desarrollo de Aplicaciones Web con Python. *Revista Latinoamericana de Ingenieria de Software*, *4*(4), 201. https://doi.org/10.18294/relais.2016.201-207

Sierra, F., Acosta, J., Ariza, J., & Salas, M. (2017). Estudio y análisis de los framework en php basados en el modelo vista controlador para el desarrollo de software orientado a la web. *Investigación y Desarrollo En TIC*, *4*(2), 14–26. http://sourceforge.net/projects/wasp/files/

Espinoza Mina, M. A., & Sierra Cedeño, A. Y. (2018). Análisis comparativo entre ASP.NETy PHP. *INNOVA Research Journal*, *3*(4), 25–43. https://doi.org/10.33890/innova.v3.n4.2018.474

ISO/IEC 25010:2011, Software Engineering — Software product Quality Requirements and Evaluation (SQuaRE) — Guide to SQuaRE

Callejas-Cuervo, M., Alarcón-Aldana, A. C., & María Álvarez-Carreño, A. (2017). Modelos de calidad del software, un estado del arte. *SciELO*, *13*(1). https://doi.org/10.18041/entramado.2017v13n1.25125

Calabrese, J., & Muñoz, R. (2018). *ASISTENTE PARA LA EVALUACIÓN DE CALIDAD DE PRODUCTO DE SOFTWARE SEGÚN LA FAMILIA DE NORMAS ISO/IEC 25000 UTILIZANDO EL ENFOQUE GQM*.

Argote PuetamanI. M., Jiménez ToledoR. A., Enrí-quez CastroD. M., & Ceballos RoseroD. (2016). Desarrollo de Apps: un estudio comparativo entre frameworks libres y privativos. Boletí-n Informativo CEI, 3(2). http://editorial.umariana.edu.co/revistas/index.php/BoletinInformativoCEI/article/view/1083

Giles-Navarro. C. (2020). Recomendaciones para las MIPyME ¿Qué hacer para sobrevivir a la pandemia del Covid-19? [Recommendations for MSMEs What to do to survive the Covid-19 pandemic?]. Instituto Belisario Domínguez Senado de la República. https://n9.cl/gi4h

Kitchenham, B. A. & Charters, S. (2007). Guidelines for performing Systematic Literature Reviews in Software Engineering (EBSE 2007-001). Keele University and Durham University Joint Report

I. Hernàndez, J. M. (2019). Software libre: técnicamente viable, económicamente sostenible y socialmente justo.

Andrade, E. C., Barraza, E. U., Enrique Cuan Durón, & Agundis, D. U. (2020). *Sistema de Gestión Académica-Administrativa (SIGAA) Evaluado por la Norma ISO / IEC 25000 para Productos de Software (SQuaRE - System and Software Quality Requirements and Evaluation)*. *8*, 156–161.

2.17. Anexos

2.18. Firma de responsabilidad (estudiante)

Luis Fernando Guerrero Ortega

2.19. Firma de responsabilidad director (luego de aplicación de rúbrica)

Ing. Astudillo Rodríguez Catalina Verónica

2.20. Firma de responsabilidad profesor metodólogo (luego de aplicación de rúbrica)

Ing. Cedillo Orellana Irene Priscila

2.21. Fecha de entrega

Lunes, 3 de mayo de 2021