ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE SISTEMAS

MAESTRIA DE SOFTWARE

ANÁLISIS DE MANTENIBILIDAD Y PORTABILIDAD DEL FRAMEWORK REACT NATIVE APLICANDO LA NORMA ISO/IEC 25010 MEDIANTE UN CASO DE ESTUDIO EN LA APLICACIÓN DE GESTIÓN DE EVENTOS OAQ.

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL GRADO DE MAGÍSTER EN SOFTWARE MENCIÓN CALIDAD

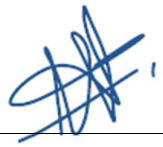
DARWIN ANTONIO MENA ALVARADO
darwin.mena@epn.edu.ec

Director: MARCO OSWALDO SANTÓRUM GAIBOR Ph.D.
marco.santorum@epn.edu.ec

JUNIO 2020

APROBACIÓN DEL DIRECTOR

Como director del trabajo de titulación Análisis de mantenibilidad y portabilidad del Framework React Native aplicando la norma ISO/IEC 25010 mediante un caso de estudio en la aplicación de gestión de eventos OAQ, desarrollado por Darwin Antonio Mena Alvarado, estudiante del programa de Maestría en Software, habiendo supervisado la realización de este trabajo y realizado las correcciones correspondientes, doy por aprobada la redacción final del documento escrito para que prosiga con los trámites correspondientes a la sustentación de la Defensa Oral.



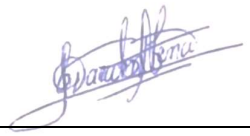
Marco Santórum G. Ph.D.

DIRECTOR

DECLARACIÓN DE AUTORÍA

Yo, Darwin Antonio Mena Alvarado, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y que he consultado las referencias bibliográficas que se incluyen en este documento.

La Escuela Politécnica Nacional puede hacer uso de los derechos correspondientes a este trabajo, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.



Darwin Antonio Mena Alvarado

DEDICATORIA

Dedicado al amor de mi vida, mi esposa Michelle, por ser quien inspira mis pasos. Recuerdo que iniciamos nuestros sueños juntos y hoy somos una realidad, comenzando nuestra aventura como recién casados, los sueños sí se cumplen. Recalco que eres lo mejor que me pasó en la vida; eres quien mueve mi corazón, mi alma y me hace feliz.

A mis padres Raúl y Nancy, mi pedacito de Dios en la Tierra, quienes han guiado todos mis pasos con amor y dedicación; me han enseñado siempre que uno debe luchar por todo superando todas las circunstancias.

A mi hermana Evelyn y sobrinito Martín porque veo el amor de una madre que lucha por su pequeño alcanzando sus sueños, y de un hijo que crece lleno de amor y que cuida desde ya a su madre.

AGRADECIMIENTO

Agradezco a Dios infinitamente pues en medio de la crisis global del Coronavirus hace posible la culminación del presente proyecto, recordándome que la humanidad necesita cambios profundos y sobre todo la auto reflexión de ser mejores personas cada día.

A mis padres Raúl y Nancy por su amor, dedicación y constancia, son los héroes de mi camino a quienes les debo el éxito; gracias por animarme a prepararme y crecer personal y profesionalmente. Son los guerreros de los sueños y la muestra de que todo es posible.

A mi esposa Michelle por su paciencia, dedicación y aporte en el presente proyecto, sin duda es mi regalo del cielo y es hermoso saber que suma su mano y apoyo para culminar este y todos los objetivos que nos plantearemos en la vida, juntos iremos por más metas y sueños, Te Mega Amo.

Al Doctor Marco Santórum, director del presente proyecto, muchas gracias por su guía científico técnica; sin duda es una persona apasionada por su trabajo, con vocación y servicio permite que muchos profesionales de pregrado y postgrado alcancen sus objetivos. Continúe con su excelente labor en favor de la sociedad.

Al Doctor Ericson López, director del Observatorio Astronómico, gracias por permitir que uno de los proyectos de la institución se involucre en el presente trabajo, y por su visión de desarrollo y crecimiento de la investigación al servicio de la sociedad

ÍNDICE DE CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

RESUMEN

Esta tesis de maestría tiene la intención de evaluar la mantenibilidad y portabilidad del framework React Native (RN), al ser un entorno de desarrollo nuevo empresas y desarrolladores se plantean incógnitas como que tan fácil es modificar y transferir un producto de software de un entorno a otro. Para lo cual hace uso de un modelo de calidad estandarizado como el planteado en la norma ISO/IEC 25010 mediante un caso de estudio, de esta manera se evalúan todas las métricas de las subcaracterísticas generando resultados cuantitativos que indirectamente indican que tan portable y mantenible es el framework RN y además la metodología aplicada en este estudio es una herramienta que permite generar buenas prácticas de desarrollo que incrementan notablemente la calidad de producto de software.

Palabras clave: Mantenibilidad, Portabilidad, ISO 25010, React Native, Calidad de Software.

ABSTRACT

This Master Thesis aims to evaluate the maintainability and portability of the React Native (RN) framework, as it is a new development environment companies and developers ask themselves questions about how easy it is to modify and transfer a software product from one environment to other. For which it makes use of a standardized quality model such as the one proposed in the ISO / IEC 25010 standard through a case study, in this way all the metrics of the sub-characteristics are evaluated, generating quantitative results that indirectly indicate how portable and maintainable it is the React Native framework and also the methodology applied in this study is a tool that allows generating good development practices that notably increase the quality of the software product.

Keywords: Maintainability, Portability, ISO 25010, React Native, Quality of Software.

1. INTRODUCCIÓN

Actualmente el desarrollo de aplicaciones móviles presenta un constante crecimiento. Los dispositivos móviles son cada vez más avanzados y eficientes, estimándose que para el año 2020 existan cerca de 3.8 billones de usuarios de teléfonos inteligentes (Greenfield, n.d.) . El desarrollo de apps actualmente debe tener en cuenta las particularidades de cada uno de los sistemas operativos, es decir Android o IOS independientemente del lenguaje de programación (Aguirre, 2018).

Varias tecnologías se ofrecen como opción a la hora de desarrollar aplicaciones móviles. Por ejemplo, el framework de desarrollo React Native lanzado oficialmente por Facebook en el 2016 es una solución en el desarrollo multiplataforma (Charland & LeRoux, 2011). La principal diferencia con otro tipo de plataformas radica en que permite desarrollar aplicaciones nativas utilizando Javascript y React, lo que transforma una aplicación web en móvil en menos tiempo y con menos recursos. Además, existen estudios de laboratorio como el publicado por “BIT Computer Training” (Ahumada,2019), donde se analiza el rendimiento de RN (React Native), alcanzando cuotas muy cercanas a una aplicación nativa desarrollada para IOS. La documentación oficial del Framework indica que compañías como Instagram, Pinterest, Skype, Tesla, Uber, Walmart, SoundCloud, Vogue entre otras utilizan la herramienta de desarrollo (Aguirre, 2018).

La calidad del producto de software demanda la utilización de menos recursos, y uno de los principales problemas que presentan los desarrolladores, y pequeñas empresas de desarrollo al momento de optar por un entorno nuevo como React Native (Johansson & Söderberg, 2018), son los pocos estudios de calidad disponibles hasta el momento. Como se mencionó anteriormente el Framework RN presenta la posibilidad de reutilización de código generando versiones para multiplataformas. Es importante entonces analizar la mantenibilidad, puesto que esta característica según el estándar ISO 25010 define la capacidad que tendría un producto de software a ser modificado de manera efectiva y eficiente.(ISO_25010, 2011) Dentro de las subcaracterísticas se encuentra por ejemplo la capacidad a ser probado, y para los desarrolladores en general es vital que el lenguaje a utilizarse permita establecer criterios de prueba.

Otra de las incógnitas para desarrolladores y equipos de desarrollo es saber que capacidad tiene un lenguaje o Framework de desarrollo al ser transferido sobre otro entorno. Para tal fin la norma ISO/IEC 25010 plantea la característica de portabilidad que

permite analizar adaptabilidad, capacidad para ser instalado y capacidad para ser reemplazado.

En función de lo expuesto, a continuación, se establecen la pregunta de investigación y los objetivos del presente proyecto.

1.1. Pregunta de investigación

¿Es posible evaluar las características de mantenibilidad y portabilidad del Framework React Native, basado en la norma ISO/IEC 25010?

1.2. Objetivo general

Evaluar la mantenibilidad y portabilidad del Framework React Native utilizando la norma ISO 25010 en un caso de estudio.

1.3. Objetivos específicos

- Estudiar el modelo de calidad ISO 25010 para identificar las características de mantenibilidad y portabilidad que permitan evaluar el Framework React Native.
- Evaluar la mantenibilidad y portabilidad del Framework React Native utilizando el modelo ISO 25010 aplicado a una implementación práctica.
- Identificar buenas prácticas de calidad de software, conclusiones y recomendaciones para el framework React Native.

1.4. Marco Teórico

1.4.1 Modelos de calidad

Durante los últimos 40 años se han desarrollado varios modelos de calidad de software para evaluar la calidad de un sistema a través da la interacción de la calidad de todos sus elementos (Dromey, 1995).

Entre los principales modelos desarrollados hasta el momento están los siguientes:

Modelo de McCall

McCall, Richards y Walters en 1977 propusieron un modelo de factores de calidad del software que incluye 11 factores, entre los que se encuentran corrección, confiabilidad, eficiencia, integridad, mantenibilidad, usabilidad, flexibilidad, testeabilidad, portabilidad, reusabilidad e interoperabilidad.

Según los autores, esto permite la medición de la calidad del software en todas las fases del proceso de su desarrollo. Los factores de calidad del modelo se miden con preguntas diatómicas del tipo “SI” o “NO”, las cuales son contestadas por una o varias personas.

Si bien el modelo presenta una manera fácil de evaluar la calidad de producto de software, es subjetivo dado que necesita de personas y de cómo cada una de ellas puede evaluar la calidad de producto de software.

Modelo de Boehm

Barry Boehm en 1978 propone un modelo de factores de calidad del software, donde los factores se representan de forma jerárquica. Los factores de alto nivel son asociados con el uso del software, mientras que los factores de bajo nivel están relacionados con las métricas.

En las características de alto nivel tenemos a la utilidad, mantenimiento, y utilidad en general, mientras que en un nivel intermedio define características como portabilidad, fiabilidad, eficiencia, usabilidad, capacidad de prueba y flexibilidad.

Si bien el modelo introduce métricas cualitativas y tiene más desarrollada la característica de mantenibilidad, la portabilidad aún no presenta la madurez necesaria y actualmente es un modelo desactualizado.

Modelo de FURPS+

Robert Grady en 1987, mientras trabajaba en Hewlett-Packard, presentó el modelo FURPS, cuyo acrónimo indica el conjunto de factores de calidad: funcionalidad, usabilidad, fiabilidad, rendimiento y compatibilidad.

El símbolo "+" se agregó más tarde al nombre del modelo para incluir restricciones relacionadas con el diseño, implementación, interfaces y requisitos físicos.

El modelo toma como bases características de los modelos de Boehm y McCall, sin embargo, no define métricas de evaluación en el caso de un producto de software, por lo cual no es adecuado para el presente proyecto.

Modelo ISO/IEC 25010

Es un estándar que tiene dos modelos de factores de calidad. El primero reúne características de calidad interna y externa, y el segundo, características de calidad en uso.

El modelo de factores de calidad internos y externos describe cómo funciona el producto en su entorno de desarrollo.

En el interno las características de calidad están relacionadas con la medición de resultados intermedios, en términos de aspectos estáticos de artefactos relacionados con el producto.

En la calidad externa las características están relacionadas con aspectos de comportamiento, medidos a partir de la ejecución del código de producto.

Para el presente proyecto se seleccionó el modelo de calidad ISO/IEC 25010 no solo porque es actual, sino porque además presenta métricas bien definidas que pueden utilizarse al momento de elaborar una matriz de prueba.

1.4.2 Estudio del modelo ISO/IEC 25010

La norma ISO/IEC 25010 la define la calidad de software, como el grado en el que un producto satisface las necesidades establecidas bajo condiciones específicas .

La norma antes mencionada se apoya de la norma ISO/IEC 25000, también conocida como SQuaRE (Requisitos y Evaluación de Calidad de Productos de Software), la cual reúne una guía de criterios, requisitos, métricas y evaluación.

El modelo de calidad genérico ISO/IEC 25010 define dos modelos:

- El modelo para la calidad en uso de un producto de software.
- El modelo para la calidad interna y externa de un producto de software.

Modelo de calidad en uso de un producto de software (ISO_25010, 2011)

El modelo define características para analizar la interacción con un sistema o producto de software.

Se definen 5 características: Efectividad, Eficiencia, Satisfacción, Libertad de Riesgo y Cobertura de Contexto, las cuales a su vez se subdividen en subcaracterísticas descritas en la Figura 1.

Figura 1. Modelo de calidad en uso de producto de software.

Fuente: ISO/IEC 25010

Autor: ISO/IEC 25010

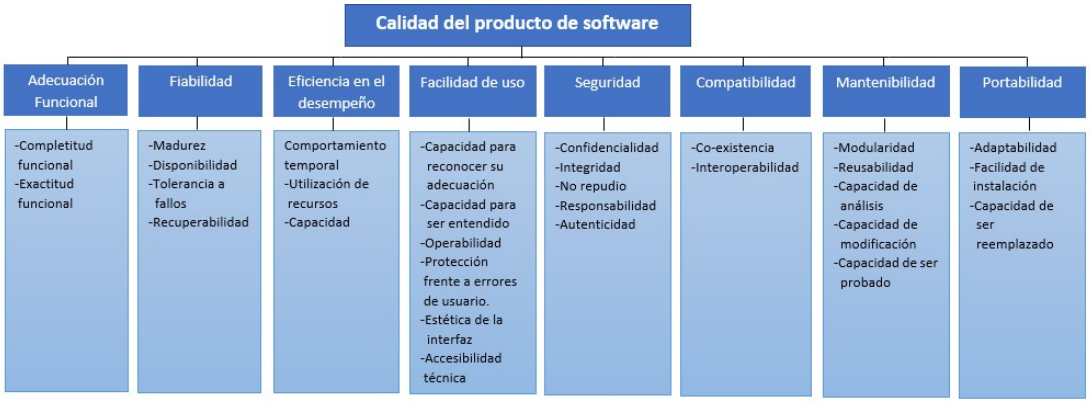
Cabe mencionar que la evaluación de la calidad en uso no es parte de los objetivos de este proyecto.

Modelo de calidad interna y externa de un producto de software (ISO_25010, 2011).

El modelo define 8 características para la calidad interna y externa de un producto software: Adecuación, Funcionalidad, Eficiencia de desempeño, Compatibilidad, Usabilidad, Fiabilidad, Seguridad, Mantenibilidad y Portabilidad.

Las cuales a su vez son subdivididas en subcaracterísticas descritas en la figura 2.

Figura 2. Modelo de calidad interna y externa de producto de software.



Fuente: ISO/IEC 25010

Autor: ISO/IEC 25010

A continuación, se utilizará la siguiente codificación para identificar las características, subcaracterísticas y métricas del modelo de calidad del producto de software.

- **A**: Adecuación funcional
- **F**: Fiabilidad
- **E**: Eficiencia en el desempeño
- **FU**: Facilidad de uso
- **S**: Seguridad
- **C**: Compatibilidad
- **M**: Mantenibilidad
- **P**: Portabilidad
- **S1-n:** Identificador de subcaracterísticas numerado del 1 al número total de subcaracterísticas de cada característica.
- **M1-n:** Identificador de métrica numerado del 1 al número total de métricas de cada subcaracterística.

A: Adecuación funcional (ISO_25010, 2011)

Representa la capacidad del producto o sistema software para proporcionar las funciones necesarias para satisfacer al usuario.

Esta característica se divide en las siguientes subcaracterísticas:

- **AS1- Completitud funcional:** capacidad del sistema software para proporcionar un conjunto de funcionalidades apropiadas que cubran todas las tareas y objetivos determinados por el usuario.

- **AS2- Exactitud funcional:** capacidad del sistema software para proporcionar los resultados correctos con el grado necesario de precisión.

F: Fiabilidad (ISO_25010, 2011)

Capacidad del producto o sistema software para realizar las funciones específicas cuando se lo utiliza bajo ciertas condiciones y determinados periodos de tiempo. Esta característica se divide en las siguientes subcaracterísticas:

- **FS1- Madurez:** capacidad del sistema software para satisfacer las necesidades de fiabilidad durante el funcionamiento normal.
- **FS2- Disponibilidad:** capacidad de un sistema software de estar operativo y accesible para su uso cuando se necesite.
- **FS3- Tolerancia a Fallos:** capacidad de un sistema software para operar cuando se presenten fallos.
- **FS4- Recuperabilidad:** capacidad de un sistema software de reestablecer el estado del sistema y recuperar datos que se hayan afectado, en caso de interrupción o fallo.

E: Eficiencia en el desempeño (ISO_25010, 2011)

Capacidad de un producto o sistema software de proporcionar un rendimiento apropiado, respecto a la cantidad recursos utilizados bajo determinadas condiciones. Esta característica se divide en las siguientes subcaracterísticas:

- **ES1- Comportamiento Temporal:** capacidad de un sistema software para proporcionar los tiempos de respuesta y procesamiento apropiados.
- **ES2- Utilización de Recursos:** capacidad en que un sistema software utiliza las cantidades y tipos de recursos adecuados.
- **ES3- Capacidad:** capacidad de un sistema software de cumplir con los requisitos determinados.

FU: Facilidad de uso (ISO_25010, 2011)

Capacidad del producto o sistema software para que sea entendido, aprendido, agradado y usado por el usuario. Esta característica se divide en las siguientes subcaracterísticas:

- **FUS1- Capacidad de reconocer su adecuación:** capacidad del sistema software que permite al usuario entender si el software es adecuado para sus necesidades.

- **FUS2-Capacidad para ser entendido:** capacidad del sistema, que permite al usuario entender si el software es adecuado para alcanzar sus objetivos determinados.
- **FUS3-Operatividad**: capacidad de un sistema software que permite al usuario operarlo y controlarlo con facilidad.
- **FUS4-Protección contra errores del usuario:** capacidad en que el sistema brinda la protección necesaria contra errores que realizan los usuarios.
- **FUS5-Estética de la Interfaz del usuario:** capacidad en que la interfaz de usuario llega a satisfacer y agradar al usuario.
- **FUS6-Accesibilidad técnica:** capacidad del sistema software para que se permita ser utilizado por usuarios con determinadas discapacidades.

S: Seguridad (ISO_25010, 2011)

Capacidad de proteger la información y los datos, de manera que personas o sistemas no autorizados puedan tener acceso para consultas o actualizaciones. Esta característica se divide en las siguientes subcaracterísticas:

- **SS1-Confidencialidad:** capacidad de proteger la información y el acceso a datos no autorizados, ya sea de manera accidental o intencional.
- **SS2-Integridad:** capacidad de un producto, sistema o componente software para evitar accesos no autorizados a datos o programas de computación.
- **SS3-No – repudio:** capacidad para demostrar que los eventos han ocurrido, de manera que dichos eventos no puedan ser refutados posteriormente.
- **SS4-Responsabilidad:** capacidad de dar seguimiento a las acciones que fueron realizadas por una entidad.
- **SS5-Autenticidad:** capacidad de demostrar la identidad de un sujeto o un recurso.

C: Compatibilidad (ISO_25010, 2011)

Capacidad de dos o más sistemas software, para llevar acabo sus funciones intercambiando información mientras comparten el mismo entorno. Esta característica se divide en las siguientes subcaracterísticas:

- **CS1- Co-Existencia:** capacidad de un sistema software para coexistir en un entorno en el cual comparten recursos comunes con otro software independiente.
- **CS2- Interoperatividad:** capacidad de dos o más sistemas software para intercambiar información y utilizar dicha información.

M: Mantenibilidad (ISO_25010, 2011)

La mantenibilidad representa la capacidad del producto de software para ser modificado efectiva y eficientemente. Esta característica se subdivide a su vez en las siguientes subcaracterísticas:

- **MS1-Modularidad:** Capacidad de un sistema o programa de ordenador (compuesto de componentes discretos) que permite que un cambio en un componente tenga un impacto mínimo en los demás.
- **MS2-Reusabilidad:** Capacidad de un activo que permite que sea utilizado en más de un sistema software o en la construcción de otros activos.
- **MS3-Analizabilidad:** Facilidad con la que se puede evaluar el impacto de un determinado cambio sobre el resto del software, diagnosticar las deficiencias o causas de fallos en el software, o identificar las partes a modificar.
- **MS3-Capacidad para ser modificado:** Capacidad del producto que permite que sea modificado de forma efectiva y eficiente sin introducir defectos o degradar el desempeño.
- **MS4-Capacidad para ser probado:** Facilidad con la que se pueden establecer criterios de prueba para un sistema o componente, y con la que se pueden llevar a cabo las pruebas para determinar si se cumplen dichos criterios.

P: Portabilidad (ISO_25010, 2011)**.**

La portabilidad analiza la capacidad del producto o componente a ser transferido de forma efectiva y eficiente de un entorno de hardware, software de utilización a otro. Esta característica se subdivide a su vez en las siguientes subcaracterísticas:

- **PS1-Adaptabilidad:** Capacidad del producto que le permite ser adaptado de forma efectiva y eficiente a diferentes entornos determinados de hardware, software, operacionales o de uso.
- **PS2-Capacidad para ser instalado:** Facilidad con la que el producto se puede instalar y/o desinstalar de forma exitosa en un determinado entorno.
- **PS3-Capacidad para ser reemplazado:** Capacidad del producto para ser utilizado en lugar de otro producto software determinado con el mismo propósito y en el mismo entorno.

Para evaluar las subcaracterísticas antes mencionadas el modelo ISO/IEC 25010 hace uso de métricas que son descritas en la norma ISO/IEC 25022.

1.4.3 Métricas para la calidad interna, externa – ISO/IEC 25022

La norma ISO/IEC 25010 utiliza el conjunto de métricas propuesta en la norma ISO/IEC 25022 para la evaluación de la calidad de producto de software. En la Tabla 1 se describen las métricas de interés para el presente proyecto correspondientes a mantenibilidad y portabilidad.

Tabla 1: Subcaracterísticas y métricas de mantenibilidad y portabilidad (ISO_25022, 2016).

MANTENIBILIDAD		
Código	**SUBCARACTERÍSTICAS**	**MÉTRICAS**
MS1	Modularidad	MS1M1- Capacidad de condensación
		MS1M2- Acoplamiento de clases
MS2	Reusabilidad	MS2M1- Ejecución de reusabilidad
MS3	Capacidad a ser analizado	MS3M1- Capacidad de pistas de auditoria
		MS3M2- Diagnóstico de funciones suficientes
MS4	Capacidad a ser modificado	MS4M1- Complejidad ciclo matica
		MS4M2- Profundidad de herencia
		MS4M3- Grado de localización de corrección de impacto
		MS4M4- Complejidad de modificación
		MS4M5- Índice de éxito de modificación
MS5	Capacidad a ser probado	MS5M1- Complejidad funcional de funciones de prueba
		MS5M2- Capacidad de prueba autónoma
		MS5M3- Capacidad de reinicio de pruebas
PORTABILIDAD		
Código	**SUBCARACTERÍSTICAS**	**MÉTRICAS**
PS1	Adaptabilidad	PS1M1- Adaptabilidad en entorno hardware
		PS1M2- Adaptabilidad en entorno de software
		PS1M3- Adaptabilidad en entorno organizacional
PS2	Capacidad de ser instalado	PS2M1- Eficiencia en el tiempo de instalación
		PSSM2- Facilidad de instalación
PS3	Capacidad de ser reemplazado	PS3M1- Consistencia en la función de soporte al usuario
		PS3M2- Inclusividad funcional
		PS3M3- Uso continuo de datos

Fuente: ISO/IEC 25010

MS1: Modularidad (ISO_25022, 2016)

- **MS1M1- Capacidad de condensación:** métrica de calidad interna, mide que tan fuerte es la relación entre los componentes del sistema; para lo cual cuenta el número de componentes que no son afectados por cambios de otros componentes y el número total de componentes específicos.

- **MS1M2- Acoplamiento de clases:** métrica de calidad interna, mide que tan fuerte es la relación entre una función del sistema con otras clases implementada; para lo cual cuenta el número de relaciones que tiene una función con respecto a otra clase.

MS2: Reusabilidad (ISO_25022, 2016)

- **MS2M1- Ejecución de reusabilidad:** métrica de calidad interna, mide cuantos elementos pueden ser reutilizados; para lo cual cuenta el número de elementos reutilizados y el número total de elementos de la biblioteca reutilizable.

MS3: Capacidad de ser analizado (ISO_25022, 2016)

- **MS3M1- Capacidad de pistas de auditoria:** métrica de calidad interna y externa, mide si los usuarios pueden identificar fácilmente la operación específica que causó el fallo; para lo cual cuenta el número de datos realmente grabadas durante la operación y el número de datos previstos a grabarse para controlar el estado del sistema durante la operación.

- **MS3M2- Diagnóstico de funciones suficientes:** métrica de calidad interna y externa, mide hasta qué punto las funciones de diagnóstico están preparadas o hasta qué punto funcionan para el análisis causal; para lo cual cuenta el número de funciones de diagnóstico implementadas y contar el número de funciones de diagnóstico requeridas en la especificación de requerimientos.

MS4: Capacidad de ser modificado (ISO_25022, 2016)

- **MS4M1- Complejidad ciclomática:** métrica de calidad interna, mide cuál es la complejidad estructural de un código fuente; para lo cual cuenta las instrucciones condicionales, bucles, salidas de métodos y clausulas AND y OR dentro de los condicionales.

- **MS4M2- Profundidad de herencia:** métrica de calidad interna, mide qué tan profunda es la jerarquía de la herencia de las clases involucradas en una determinada función; para lo cual cuenta las jerarquías empleadas en una determinada función o método.

- **MS4M3- Grado de localización de corrección de impacto:** métrica de calidad interna y externa, mide hasta qué punto los problemas causados pueden tener como consecuencia un mantenimiento; para lo cual cuenta el número de fallas aparecidas después que se ha resuelto un fallo y cuenta el número de fallas resultas.

- **MS4M4- Complejidad de modificación:** métrica de calidad externa, mide con qué facilidad el desarrollador puede modificar el software para resolver problema; para lo cual toma el tiempo de trabajo que le toma al desarrollador modificar y contar el número de modificaciones.

- **MS4M5- Índice de éxito de modificación:** métrica de calidad externa, mide hasta qué punto puede el sistema ser operado sin fallas después del mantenimiento; para lo cual cuenta el número de problemas dentro de un determinado período antes de mantenimiento y contar el número de problemas en el mismo período después del mantenimiento.

MS5: Capacidad de ser probado (ISO_25022, 2016)

- **MS5M1- Completitud funcional de funciones de pruebas:** métrica de calidad interna, mide si las funciones de prueba son completas y fáciles de implementa; para lo cual cuenta el número de funciones de prueba implementadas y el número de funciones de prueba requeridas.

- **MS5M2- Capacidad de prueba autónoma:** métrica de calidad interna, mide que tan independiente es el software al ser probado; para lo cual cuenta el número de pruebas que están dependiendo de otros sistemas y el número total de pruebas dependientes con otros sistemas.

- **MS5M3- Capacidad de reinicio de pruebas:** métrica de calidad externa, mide con qué facilidad se puede llevar a cabo las pruebas nuevamente después del

mantenimiento; para lo cual cuenta el número de casos en los cuales el mantenedor puede pausar y restaurar las pruebas y contar el número de casos de pausa en la ejecución de pruebas.

PS1: Adaptabilidad (ISO_25022, 2016)

- **PS1M1- Adaptabilidad en entorno de hardware:** métrica de calidad interna y externa, mide si el sistema es lo suficientemente capaz de adaptarse al entorno de hardware; para lo cual cuenta el número funciones operativas de las tareas que no se hayan completado durante las pruebas operativas con el entorno hardware y contar el número total de funciones las cuales han sido probadas.

- **PS1M2- Adaptabilidad en entorno de software:** métrica de calidad interna y externa, mide si el sistema es lo suficientemente capaz de adaptarse al entorno del sistema software; para lo cual cuenta el número de funciones operativas de las tareas que no se hayan completado durante las pruebas operativas con el sistema y cuenta el número total de funciones las cuales han sido probadas.

- **PS1M3- Adaptabilidad en entorno empresarial:** métrica de calidad interna y externa, mide si el sistema es capaz de adaptarse al entorno operacional; para lo cual cuenta el número funciones operativas de las tareas que no se hayan completado durante las pruebas operativas con usuarios del entorno empresarial y el número total de funciones las cuales han sido probadas.

PS2: Capacidad de ser instalado (ISO_25022, 2016)

- **PS2M1- Eficiencia en tiempo de instalación:** métrica de calidad externa, mide cuanto tiempo es requerido para realizar una instalación, para lo cual estima el tiempo total transcurrido al instalar el sistema y contar el número de reintentos al instalar el sistema.

- **PS2M2- Facilidad de instalación:** métrica de calidad externa, mide si el usuario o el desarrollador puede instalar el software en un entorno operacional; para lo cual cuenta el número casos en que los usuarios tuvieron éxito al instalar el sistema cambiando proceso de instalación para su conveniencia y cuenta el número total de casos en que los usuarios han intentado cambiar el proceso de instalación para su conveniencia.

PS3: Capacidad de ser reemplazado (ISO_25022, 2016)

- **PS3M1: Consistencia en la función de soporte de usuario:** métrica de calidad interna y externa, mide cuan consistente es el nuevo componente con la interfaz de usuario existente; para lo cual cuenta el número de nuevas funciones que son consideradas como no consistentes por el usuario y el número de nuevas funciones.

- **PS3M2: Inclusividad funcional:** métrica de calidad externa, mide si las funciones pueden ser utilizadas después de ser cambiadas por otras similares; para lo cual cuenta el número de funciones que producen resultados similares con anterioridad y que no se han exigido cambios y el número de funciones probadas que son similares a las funciones proporcionadas por otro software para ser reemplazado.

- **PS3M3: Uso continuo de datos:** métrica de calidad externa, mide si los datos pueden ser fácilmente utilizados después de reemplazar el software por uno similar; para lo cual cuenta el número de datos que son continuamente utilizables por el software a ser reemplazado y el número de datos que son continuamente reutilizables por el software a ser reemplazado.

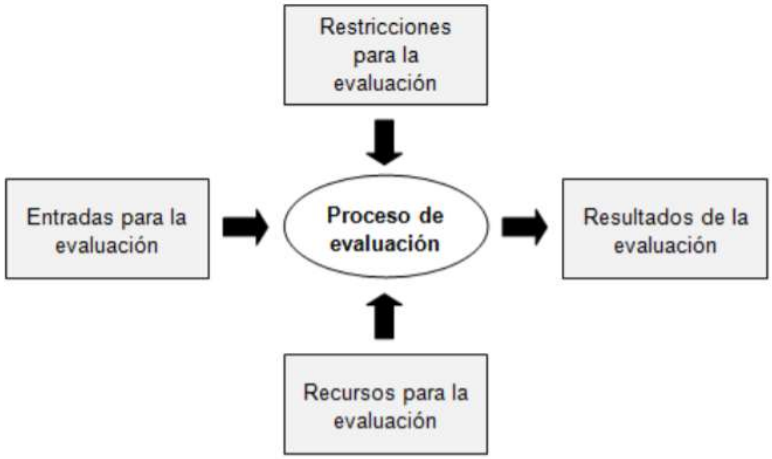
Hasta el momento se ha identificado características, subcaracterísticas y métricas de calidad de producto de software. La norma ISO/IEC 25040 descrita a continuación describe los mecanismos para generar la evaluación de un producto de software.

1.4.4 Modelo de evaluación de calidad de producto de software ISO/IEC 25040

Proporciona una descripción del proceso de evaluación y todos los requisitos involucrados, el proceso se puede utilizar para la evaluación de la calidad interna, externa y en uso (Chisaguano Balseca, 2014).

A continuación, en la Figura 3 se describen las entradas, salidas, recursos y restricciones necesarios a tomar en cuenta en este proceso.

Figura 3 Esquema del proceso de evaluación de la calidad de producto de software (Chisaguano Balseca, 2014)



Fuente: ISO/IEC 25040

Autor: ISO/IEC 25040

- **Entradas para la evaluación**

  Las entradas para la evaluación son todos los requisitos y especificaciones de calidad de producto de software a ser tomados en cuenta.

- **Restricciones para la evaluación**

  Las restricciones prohíben recursos, horarios, costos, entornos, metodología y herramientas y finalmente informes para la evaluación.

- **Recursos para la evaluación**

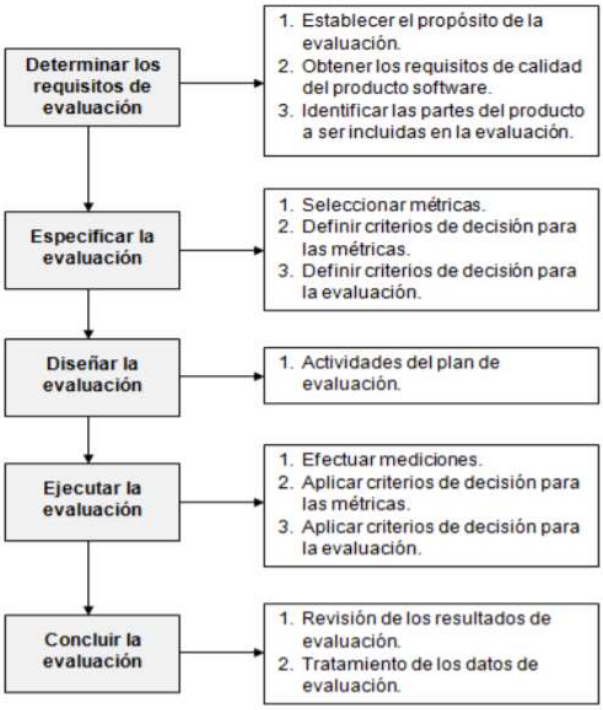
  Los recursos contemplan metodología, herramientas, recursos humanos, recursos económicos y sistemas de información necesarios para la evaluación.

- **Resultados de la evaluación**

  Los resultados generan salidas, tales como reporte de evaluación, plan de evaluación, criterios de decisión definidos por las métricas, planificación de las actividades de evaluación y resultados de las métricas de calidad utilizadas.

Además, el modelo describe los procesos generales y detalla las actividades, tareas, sus propósitos, entradas, resultados e información complementaria para la evaluación de calidad como muestra la figura a continuación.

Figura 4 Procesos generales y actividades para la evaluación de calidad de producto de software (Chisaguano Balseca, 2014).



Fuente: ISO/IEC 25040

Autor: ISO/IEC 25040

En la figura 4 antes descrita se describe cual es el procedimiento general para evaluar la calidad de producto de software.

El proceso es cíclico, es decir un paso a continuación de otro, los procesos generales a seguir son los siguientes:

- **Determinar los requisitos de evaluación:** En esta etapa es importante establecer el propósito de evaluación, definir los requisitos de calidad a estimarse y finalmente identificar la parte del producto de software a utilizarse en la evaluación.

- **Especificar la evaluación:** Es importante seleccionar las métricas y definir los criterios de decisión para la evaluación.

- **Diseñar la evaluación:** En este punto es indispensable definir cuáles son las actividades a realizarse y generar un plan de evaluación.

- **Ejecutar la evaluación:** Es importante efectuar las mediciones basadas en los criterios de decisión.
- **Concluir la evaluación:** Finalmente en este paso se revisan y analizan los resultados.

1.4.5 Software Plato: Análisis estático de código fuente para JavaScript

Plato es una herramienta desarrollada por Jarrod Overson y otros investigadores, que tiene por objetivo generar reportes estadísticos de varios aspectos del código fuente de determinado producto de software, para lo cual utiliza módulos de Node .

Plato se instala por línea de comandos vía npm generando un directorio raíz en el que se coloca todo el proyecto a ser analizado en código fuente. El software permite obtener estadísticas de:

- **Estadísticas de mantenibilidad:** Determina que tan mantenible es el código fuente de un producto de software basado en tres métricas; Volumen de Halstead (HV), Complejidad Ciclomática (CC), Número promedio de Líneas de Código (LOC), y una métrica opcional para el porcentaje de comentarios a LOC en un módulo (COM).

- **Líneas de código:** Identifica cuales archivos tienen mayor cantidad de líneas de código y cuales pueden mejorarse o simplificarse de ser el caso.

- **Recuento estimado de errores:** Se calcula identificando el volumen de errores que entrega la métrica de Halstead y puede ser utilizada para identificar posibles bucles o fallos del producto de software.

- **Contador de errores linter:** Identifica errores linter es decir errores de sintaxis, lógica o cálculos que salgan de rango.

Para el presente proyecto resulta interesante analizar el índice de mantenibilidad que proporciona el software Plato, por lo cual a continuación se muestra la explicación de las métricas que este utiliza.

1.4.6 Índice de mantenibilidad (MI) (Amirhossein Mousavi, 2016).

En 1991, Omán y Hagemeister de la Universidad de Idaho diseñaron el índice de mantenimiento para determinar objetivamente el mantenimiento de un producto basado en el código fuente correspondiente.

El MI se fundamenta en la composición de varias métricas diferentes para un sistema de software. Estas métricas son; Volumen de Halstead (HV), la Complejidad Ciclomática (CC), Número Promedio de Líneas de Código (LOC), y una métrica opcional para el porcentaje de comentarios a LOC en un módulo (COM).

1.4.7 Complejidad Ciclomática (CC) (Amirhossein Mousavi, 2016)

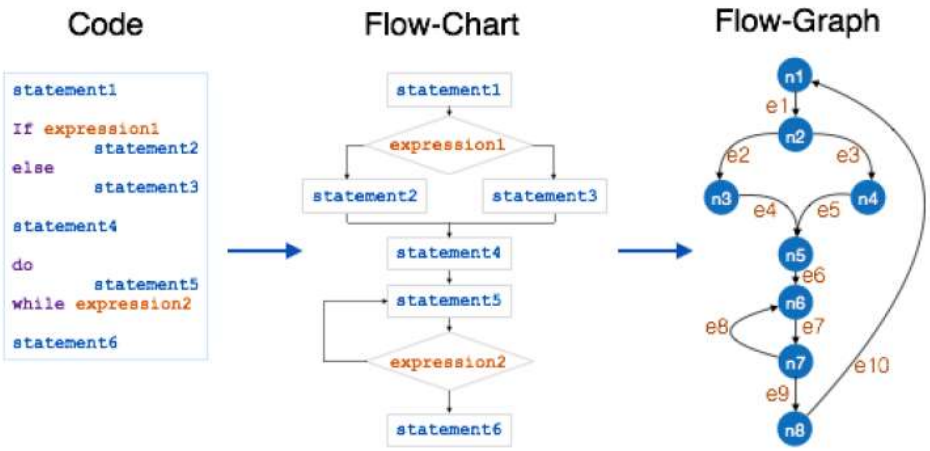
El CC de McCabe propone una ecuación para medir las rutas linealmente independientes a través del código del gráfico de flujo de control (CFG).

La ecuación define el número ciclomático de un gráfico G con n vértices, e bordes, y p componentes conectados como $V\ (G) = e-$ n + p.

Cada componente conectado representa una función o un módulo en el programa. Por ejemplo, en el CFG que se muestra en la Figura 5 el punto de salida regresa a la entrada y crea dos componentes conectados.

Por lo tanto, el CC del gráfico es: $V\ (G) = 10-$ 8 + 2 = 4

Figura 5 Ejemplo de gráfico de flujo (Amirhossein Mousavi, 2016)



Fuente: Amirhossein Seyen et.

Autor: Amirhossein Seyen et.

1.4.8 Volumen de Halstead (Amirhossein Mousavi, 2016)

Halstead introduce una métrica de software estático que no necesita la ejecución del programa. La métrica se calcula directamente de forma estática a partir de operadores y operandos del código fuente.

La métrica de Halstead analiza los algoritmos y programas como secuencias de operadores y sus operandos relacionados. A continuación, la Tabla 2 muestra los parámetros a utilizarse en las métricas y la Tabla 3 muestra las métricas y la evaluación de cada una.

Tabla 2: Parámetros de Halstead

Parámetro	Medida
n1	Número de operadores únicos
n2	Número de operandos únicos
N1	Número total de la ocurrencia de operadores
N2	Número total de ocurrencia de operandos

Fuente: Amirhossein Seyen et.

Tabla 3. Métricas de Halstead

Métrica	Operación matemática
Vocabulario	n1+n2
Tamaño	N1+N2
Volumen	Tamaño * log2Vocabulario
Dificultad	(n1/n2) * (N1/N2)
Esfuerzo	Dificultad*Volumen
Errores	Volumen/3000

Fuente: Amirhossein Seyen et.

El presente capítulo presenta los objetivos del proyecto en base a los que se plantea la revisión de norma ISO/ICE 25010. La norma presenta el modelo de calidad interna y externa de un producto de software, la misma que contiene las características de mantenibilidad y portabilidad con sus respectivas subcaracterísticas.

Además, se identifican las métricas y como se evalúa cada subcaracterística en un producto de software. Finalmente se describen los pasos y actividades para ejecutar una evaluación, planteados por la norma ISO/IEC 25040 y la revisión de las métricas que utiliza el software “Plato.js” para calcular el índice de mantenibilidad del código fuente.

En conclusión, el capítulo 1 brinda todo el soporte teórico para evaluar mantenibilidad y portabilidad de un producto de software.

2. METODOLOGÍA

Para el desarrollo de este proyecto se plantean las siguientes etapas:

- **Estudio del modelo de calidad ISO/IEC 25010**
  La norma presenta el modelo de calidad interna y externa de producto de software en el que se identifican las características de portabilidad y mantenibilidad con sus respectivas subcaracterísticas a ser evaluadas.

  Las subcaracterísticas serán evaluadas mediante métricas que plantea la norma ISO/IEC 25022, todo el estudio mencionado se encuentra en capítulo 1.

- **Presentación de un caso de estudio**
  El caso de estudio a utilizarse es una aplicación de gestión de eventos que tiene por objetivo gestionar eventos, conferencias y congresos. Además, facilita al usuario la posibilidad de enviar preguntas al conferencista durante una charla, y genera una agenda con las conferencias de interés.

  La aplicación está desarrollada en React Native framework, que permite el desarrollo multiplataforma tanto para Android como para IOS. En el presente capitulo se detalla el caso de estudio.

- **Elaboración de la matriz de evaluación**
  Con base al modelo de calidad interna y externa de software ISO/IEC 25010 se genera la matriz de prueba que contiene características, subcaracterísticas y métricas y como se evalúan. Más adelante se encuentra la explicación de la matriz y como se generó.

- **Evaluación del caso de estudio**
  Luego de generar la matriz de prueba se procede a evaluar la misma tomando como entrada todos los parámetros que se generan a partir del caso de estudio.

- **Identificar buenas prácticas, conclusiones y recomendaciones**
  Finalmente, luego de analizar los resultados se generan buenas prácticas, conclusiones y recomendaciones que se presentan en los capítulos posteriores.

2.1 Descripción del caso de estudio

La aplicación de gestión de eventos del observatorio a evaluarse se encuentra desarrollada en RN, tiene por objetivo gestionar eventos, conferencias y congresos que se desarrollan en la institución; pero principalmente facilitarle al usuario la posibilidad de enviar preguntas al conferencista y agendar las conferencias de interés.

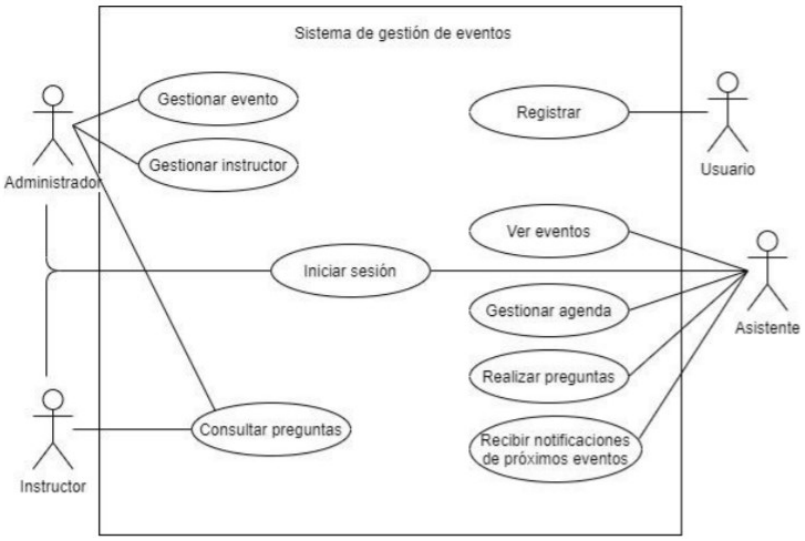
La aplicación en promedio es útil como caso de estudio, puesto que maneja la persistencia de datos y notificaciones en tiempo real, es decir lo que se busca en las aplicaciones móviles actuales.

2.1.1 Diagrama de casos del sistema de gestión de eventos

La figura 6 muestra el diagrama de casos de uso del sistema de gestión de eventos que de manera global tiene los siguientes roles: administrador, instructor, usuario o asistente.

- **Rol administrador:** quien gestiona un evento y un instructor con todos sus campos es decir horarios, lugar, fechas y a su vez agregar, modificar o eliminar un próximo evento o instructor.
- **Rol instructor:** quien podrá leer las preguntas al final de la conferencia, facilitando las respuestas a inquietudes por parte del auditorio, y además agendar eventos de interés.
- **Rol asistente:** quien podrá enviar preguntas al conferencista, agendar eventos de interés y recibir notificaciones a manera de recordatorio.

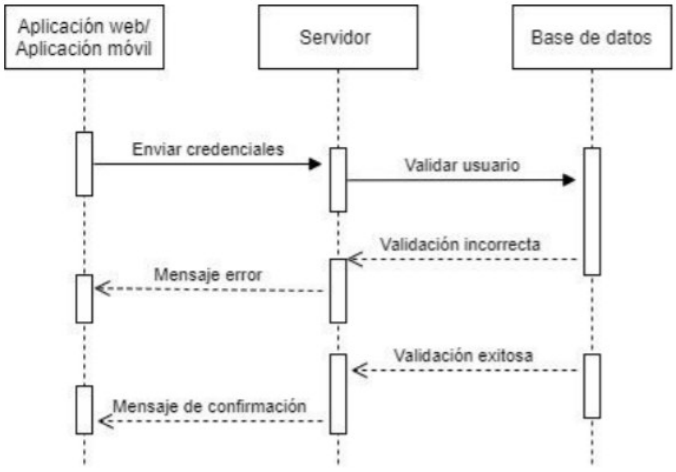
Figura 6 Diagrama de casos de uso del sistema de gestión de eventos.



2.1.2 Diagrama de secuencias de la aplicación de gestión de eventos

La figura 7 a continuación muestra un ejemplo de cuáles son las secuencias de la aplicación, y su correspondiente comunicación con el servidor y la persistencia a la base de datos.

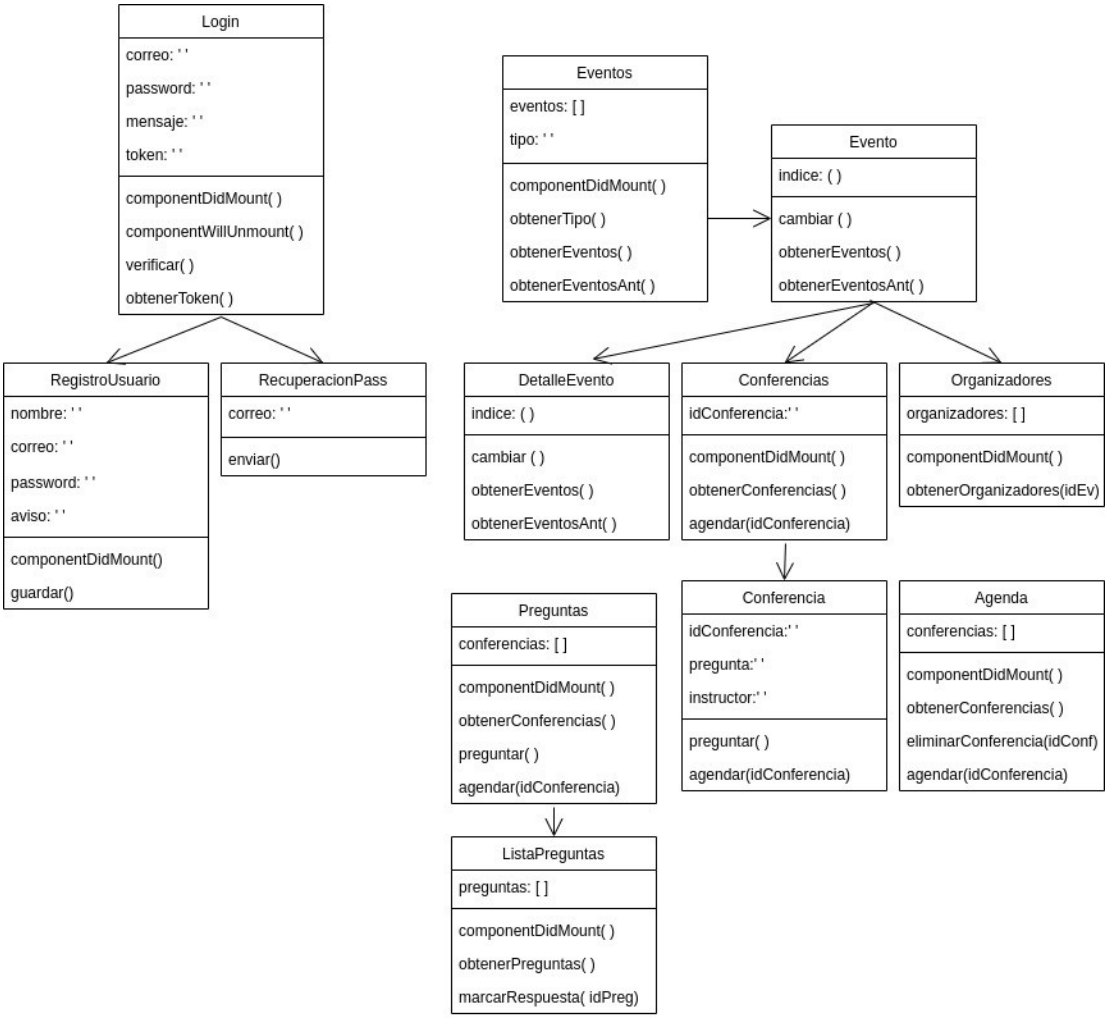
Figura 7 Ejemplo de diagrama de secuencias de la aplicación gestión de eventos.



2.1.3 Diagrama de clases de la aplicación gestión de eventos

La figura 8 a continuación muestra el diagrama de clases de la aplicación del caso de estudio, donde se identifican módulos y funciones a utilizarse posteriormente para realizar la evaluación de mantenibilidad y portabilidad.

Figura 8 Diagrama de clases de la aplicación gestión de eventos



2.2 Elaboración de la matriz de prueba

En base a la revisión de la Norma ISO/IEC 25010 presentada en el capítulo 1, se identificó que para realizar la evaluación de mantenibilidad y portabilidad del framework RN, es necesario la utilización de métricas de calidad interna y externa.

En la sección 1.4.3 del capítulo anterior se describen las métricas y además el cómo se obtienen los parámetros de medición.

A continuación, se presentan las Tablas 4 y 5, que contienen las matrices de evaluación de mantenibilidad y portabilidad; en las que se encuentra las subcaracterísticas con sus métricas a evaluar, el método de aplicación, la fórmula matemática, el rango del valor deseado y el recurso a utilizarse.

- **Subcaracterísticas:** se encuentra el nombre de las subcaracterísticas correspondiente a mantenibilidad y portabilidad.
- **Métricas a evaluar:** se encuentra el nombre de las métricas a evaluar correspondiente a cada una de las subcaracterísticas.
- **Método de aplicación:** se encuentra el cómo medir los parámetros A y B de cada métrica.
- **Fórmula matemática:** muestra la fórmula matemática de cada métrica.
- **Rango del valor deseado:** muestra el rango y el valor deseado que debe tener la respuesta luego de aplicar la fórmula matemática de cada métrica.
- **Recurso a utilizarse:** muestra los recursos a utilizarse, estos pueden ser código fuente, desarrollador o tester.

Por ejemplo; para el caso de mantenibilidad si se va a evaluar la subcaracterística de modularidad, entonces se debe medir dos métricas, la capacidad de condensación y el acoplamiento de clases.

Para el caso de la capacidad de condensación se cuenta el número de componentes que no son afectados por cambios de otros componentes y se lo define como A, y se cuenta el número total de componentes específicos y se lo define como B. Al realizar la evaluación el valor deseado es de 0 a 1, y el mejor valor es el más cercano a 0. Finalmente, el recurso a utilizarse es el código fuente.

2.1.2 Matriz de evaluación de mantenibilidad.

Tabla 4. Matriz de evaluación de mantenibilidad

SUBCARACTERÍSTICAS	MÉTRICAS	MÉTODO DE APLICACIÓN	FORMULA	VALOR DESEADO	TIPO DE MEDIDA	RECURSOS A UTILIZARSE
Modularidad	Capacidad de condensación	Contar el número de componentes que no son afectados por cambios de otros componentes y el número total de componentes específicos	X = A / B ---A = Número de componentes que no son afectados por cambios de otros componentes B = Número total de componentes específicos Dónde: B > 0	0<=X<=1 El más cercano a 0, es el mejor	X= Contable/ Contable A= Contable B= Contable	Código fuente
	Acoplamiento de clases	Contar el número de relaciones que tiene una función con respecto a otras clases	X = A-- A = Número de relaciones que tiene una función con respecto a otras clases	1<=X<=4 El más cercano a 1, es el mejor	X= Contable A= Contable	Código fuente
Reusabilidad	Ejecución de reusabilidad	Contar el número de elementos reutilizados y el número total de elementos de la biblioteca reutilizable	X = A / B A = Número de elementos reutilizados B = Número total de elementos de la biblioteca reutilizable Dónde: B > 0	0<=X<=1 El más cercano a 1, es el mejor	X= Contable/ Contable A= Contable B= Contable	Código fuente

Tabla 4 Matriz de evaluación de mantenibilidad (continuación).

SUBCARACTERÍSTICAS	MÉTRICAS	MÉTODO DE APLICACIÓN	FORMULA	VALOR DESEADO	TIPO DE MEDIDA	RECURSOS A UTILIZARSE
Capacidad a ser analizado	Capacidad de pistas de auditoria	Contar el número de datos realmente grabadas durante la operación y el número de datos previstos a grabarse para controlar el estado del sistema durante la operación	X = A / B A = Número de datos realmente grabadas durante la operación B = Número de datos previstos a grabarse para controlar el estado del sistema durante la operación Dónde: B > 0	0<=X<=1 El más cercano a 1, es el mejor	X= Contable/ Contable A= Contable B= Contable	Código fuente
	Diagnóstico de funciones suficientes	Contar el número de funciones de diagnóstico implementadas y el número de funciones de diagnóstico requeridas en la especificación de requerimientos.	X = A/B A = Número de funciones de diagnóstico implementadas B = Número de funciones de diagnóstico requeridas en la especificación de requerimientos Dónde: B > 0	0<=X<=1 El más cercano a 1, es el mejor	X = Contable/ Contable A= Contable B= Contable	Especificación de requerimientos, Código fuente, Desarrollador, Tester

Tabla 4 Matriz de evaluación de mantenibilidad (continuación).

SUBCARACTERÍSTICAS	MÉTRICAS	MÉTODO DE APLICACIÓN	FORMULA	VALOR DESEADO	TIPO DE MEDIDA	RECURSOS A UTILIZARSE
Capacidad a ser modificado	Complejidad ciclomática	Contar las instrucciones condicionales, bucles, salidas de métodos y clausulas AND y OR dentro de los condicionales	X = A+1 A = Número de instrucciones condicionales que tiene una función	1<=X<15 El más cercano a 1, es el mejor	X= Contable A= Contable	Código fuente
	Profundidad de herencia	Contar las jerarquías empleadas en una determinada función o método.	X = A A = Número de jerarquías empleadas para una determinada función.	0<=X<=4 El más cercano a 0 es el mejor	X= Contable A= Contable	Código fuente
	Grado de localización de corrección de impacto	Contar el número de fallas aparecidas después que se ha resuelto un fallo y el número de fallas resultas	X = A/B A = Número de fallas aparecidas después que se ha resuelto un fallo B = Número de fallas resueltas Dónde: B > 0	0<=X<=1 El más cercano a 0, es el mejor	X= Contable A= Contable B = Contable	Especificación de requerimientos, Código fuente, Desarrollador, Tester

Tabla 4 Matriz de evaluación de mantenibilidad (continuación).

SUBCA RACTERÍSTICAS	MÉTRICAS	MÉTODO DE APLICACIÓN	FORMULA	VALOR DESEADO	TIPO DE MEDIDA	RECURSOS A UTILIZARSE
Capacidad a ser modificado	Complejidad de modificación	Tomar el tiempo de trabajo que le toma al desarrollador modificar y contar el número de modificaciones	X = A/T A = Número de modificaciones B = Tiempo de trabajo que le toma al desarrollador modificar Dónde: T > 0	X = A/T El más lejano a 0/t es el mejor	X= Contable/ Tiempo A= Contable B= Tiempo	Desarrollador
	Índice de éxito de modificación	Contar el número de problemas dentro de un determinado período antes de mantenimiento y el número de problemas en el mismo periodo después del mantenimiento	X = A/B A = Número de problemas dentro de un determinado período antes de mantenimiento B = Número de problemas en el mismo período después del mantenimiento (B>0)	0<=X<=1 El más cercano a 0, es el mejor	X= Contable/ Contable A= Contable B= Contable	Desarrollador

Tabla 4 Matriz de evaluación de mantenibilidad (continuación).

SUBCARACTERÍSTICAS	MÉTRICAS	MÉTODO DE APLICACIÓN	FORMULA	VALOR DESEADO	TIPO DE MEDIDA	RECURSOS A UTILIZARSE
Capacidad a ser probado	Complejidad funcional de funciones de prueba	Contar el número de funciones de prueba implementadas y el número de funciones de prueba requeridas	X = A/B A = Número de funciones de prueba implementadas B = Número de funciones de prueba requeridas Dónde: B > 0	0<=X<=1 El más cercano a 1, es el mejor	X= Contable/ Contable A= Contable B= Contable	Código fuente, Tester
	Capacidad de prueba autónoma	Contar el número de pruebas que están dependiendo de otros sistemas y el número total de pruebas dependientes con otros sistemas	X = A/B A = Número de pruebas que están dependiendo de otros sistemas B = Número total de pruebas dependientes con otros sistemas Dónde: B > 0	0<=X<=1 El más cercano a 0, es el mejor	X= Contable/ Contable A= Contable B= Contable	Código fuente, Tester

Tabla 4 Matriz de evaluación de mantenibilidad (continuación).

SUBCARACTERÍSTICAS	MÉTRICAS	MÉTODO DE APLICACIÓN	FORMULA	VALOR DESEADO	TIPO DE MEDIDA	RECURSOS A UTILIZARSE
Capacidad a ser probado	Capacidad de reinicio de pruebas	Contar el número de casos en los cuales el mantenedor puede pausar y restaurar las pruebas; y el número de casos de pausa en la ejecución de pruebas	X = A/B A = Número de casos en los cuales el mantenedor puede pausar y restaurar las pruebas B = Número de casos de pausa en la ejecución de pruebas Dónde: B > 0	0<=X<=1 El más cercano a 1, es el mejor	X= Contable/ Contable A= Contable B= Contable	Desarrollador, tester

2.1.2 Matriz de evaluación de portabilidad.

Tabla 5 Matriz evaluación de portabilidad.

SUBCARACTERÍSTICAS	MÉTRICAS	MÉTODO DE APLICACIÓN	FORMULA	VALOR DESEADO	TIPO DE MEDIDA	RECURSOS A UTILIZARSE
Adaptabilidad	Adaptabilidad en entorno hardware	Contar el número funciones operativas de las tareas que no se hayan completado durante las pruebas operativas con el entorno hardware; y el número total de funciones las cuales han sido probadas	X = A/B --A = Número funciones operativas de las tareas que no se hayan completado durante las pruebas operativas con el entorno hardware B = Número total de funciones que han sido probadas Dónde: B > 0	0<=X<=1 El más cercano a 0, es el mejor	X= Contable/ Contable A= Contable B= Contable	Especificación de requerimientos, Código fuente, Desarrollador, Tester
	Adaptabilidad en entorno de software	Contar el número de funciones operativas de las tareas que no se hayan completado durante las pruebas operativas con el sistema; y el número total de funciones las cuales han sido probadas	X = A/B--- A = Número de funciones operativas de las tareas que no hayan completado durante las pruebas operativas con el sistema--B=Número total de funciones que han sido probadas (Donde B>0)	0<=X<=1 El más cercano a 0, es el mejor	X= Contable/ Contable A= Contable B=Contable	Especificación de requerimientos, Código fuente, Desarrollador, Tester

Tabla 5 Matriz de evaluación de portabilidad (continuación).

SUBCARACTERÍSTICAS	MÉTRICAS	MÉTODO DE APLICACIÓN	FORMULA	VALOR DESEADO	TIPO DE MEDIDA	RECURSOS A UTILIZARSE
Adaptabilidad	Adaptabilidad en entorno organizacional	Contar el número funciones operativas de las tareas que no se hayan completado durante las pruebas operativas con usuarios del entorno empresarial; y el número total de funciones las cuales han sido probadas	X = A/B ---A = Número de funciones operativas de las tareas que no se hayan completado durante las pruebas operativas con usuarios del entorno empresarial B = Número total de funciones que han sido probadas Dónde: B > 0	0<=X<=1 El más cercano a 0, es el mejor	X= Contable/ Contable A= Contable B= Contable	Especificación de requerimientos, Código fuente, Desarrollador, Tester
Capacidad de ser instalado	Eficiencia en el tiempo de instalación	Contar el tiempo total transcurrido al instalar el sistema; y el número de reintentos al instalar el sistema	X = A/T-- A = Número de reintentos al instalar el sistema T = Tiempo total transcurrido al instalar el sistema Dónde: T > 0	X = A/T El más lejano a 0/t es el mejor	X= Contable/ Contable A= Contable B= Contable	Desarrollador, Tester

Tabla 5 Matriz de evaluación de portabilidad (continuación).

SUBCARACTERÍSTICAS	MÉTRICAS	MÉTODO DE APLICACIÓN	FORMULA	VALOR DESEADO	TIPO DE MEDIDA	RECURSOS A UTILIZARSE
Capacidad de ser instalado	Facilidad de instalación	Contar el número casos en que los usuarios tuvieron éxito al instalar el sistema cambiando proceso de instalación para su conveniencia; y el número total de casos en que los usuarios han intentado cambiar el proceso de instalación para su conveniencia	X = A/B A = Número casos en que los usuarios tuvieron éxito al instalar el sistema cambiando proceso de instalación para su conveniencia B = Número total de casos en que los usuarios han intentado cambiar el proceso de instalación para su conveniencia. (Donde B>0)	0<=X<=1 El más cercano a 1, es el mejor	X= Contable/ Contable A= Contable B= Contable	Desarrollador, Tester
Capacidad de ser reemplazado	Consistencia en la función de soporte al usuario	Contar el número de nuevas funciones que son consideradas como no consistentes por el usuario; y el número de nuevas funciones	X = A/B --A = Número de nuevas funciones que son consideradas como no consistentes por el usuario B = Número de nuevas funciones Dónde: B > 0	0<=X<=1 El más cercano a 0, es el mejor	X= Contable/ Contable A= Contable B= Contable	Especificación de requerimientos, Código fuente, Desarrollador, Tester

Tabla 5 Matriz de evaluación de portabilidad (continuación).

SUBCARACTERÍSTICAS	MÉTRICAS	MÉTODO DE APLICACIÓN	FORMULA	VALOR DESEADO	TIPO DE MEDIDA	RECURSOS A UTILIZARSE
Capacidad de ser reemplazado	Inclusividad funcional	Contar el número de funciones que producen resultados similares con anterioridad y que no se han exigido cambios y contar el número de funciones probadas que son similares a las funciones proporcionadas por otro software para ser reemplazado	X = A/B A = Número de funciones que producen resultados similares con anterioridad y que no se han exigido cambios B = Número de funciones probadas que son similares a las funciones proporcionadas por otro software para ser reemplazado. (Donde B>0)	0<=X<=1 El más cercano a 1, es el mejor	X= Contable/ Contable A= Contable B= Contable	Desarrollador, Tester
	Uso continuo de datos	Contar el número de datos que son continuamente utilizables por el software a ser reemplazado; y el número de datos que son continuamente reutilizables por el software a ser reemplazado	X = A/B A = número de datos que son continuamente solo utilizables por el software a ser reemplazado B = Número de datos que son reutilizables por el software a ser reemplazado Dónde: B > 0	0<=X<=1 El más cercano a 1, es el mejor	X= Contable/Contable A= Contable B= Contable	Desarrollador, Tester

2.3 Evaluación de mantenibilidad y portabilidad en el caso de estudio

2.3.1 Identificación de parámetros para la evaluación de mantenibilidad

MS1M1: Capacidad de condensación.

Para el caso de la capacidad de condensación, se identificaron 10 componentes *Acerca.js, App.js, Home.js, Informacion.js, Login.js, MenuEntrada.js, Preguntas.js, RegistroUsuario.js y RecuperarContraseña.js* no afectados por cambios de otros componentes (A=10).

Para el caso de B que corresponde al número total de componentes específicos se tiene que B=19 como indica la Tabla 6 a continuación.

Tabla 6 Lista de componentes de la aplicación caso de estudio

Número	Componentes	Funciones
1	Acerca.js	render()
2	Agenda.js	componentDidMount()
		obtenerConferencias(idusuario)
		mostrarConferencias()
		eliminarConferencias(idusuario)
		render()
3	App.js	componente para iniciar la aplicación
4	Conferencias.js	obtenerConferencias(idevento)
		agendar(idconferencia)
		componentDidMount()
		render()
5	Conferencia.js	componentDidMount()
		preguntar()
		render()
6	Detallesevento.js	componentDidMount()
		obtenerDetalles()
		render()
7	Eventos.js	componentDidMount()
		obtenerlista(fechaactual)
		obtenerlistaanteriores(fechaactual)
		render()
8	Evento.js	render()
9	Home.js	componentDidMount()
		render()

Tabla 6 Lista de componentes de la aplicación caso de estudio (continuación).

Número	Componentes	Funciones
10	HomeAs.js	componentDidMount()
		render()
11	Informacion.js	abrirLink()
12	ListaPreguntas.js	componentDidMount()
		obtenerPreguntas(id_conferencia)
		render()
13	Login.js	componentDidMount()
		registroNotificaciones()
		login()
		guardarInfo()
		render()
14	MenuEntrada.js	componentDidMount()
		render()
15	MenuEvento.js	render()
16	Organizadores.js	componentDidMount()
		obtenerOrganizadores()
		obtiene el id del evento por props
		render()
17	Preguntas.js	componentDidMount()
		obtenerConferencias(idInstructor)
		render()
18	RegistroUsuario.js	componentDidMount()
		guardar()
		render()
19	RecuperarContrasenia.js	enviardatos()

MS1M2: Acoplamiento de clases.

Se identificó una relación de una función con respecto a otras clases es decir A=1, puesto que la función del sistema *ComponentDidMount* se relaciona con todas las clases.

MS2M1: Ejecución de reusabilidad

La aplicación evaluada contiene 12 elementos reutilizables tanto para Android como para IOS que permiten manejar toda la aplicación, como se muestra en la Tabla 7 a continuación, es decir A=12.

Tabla 7 Elementos reutilizables en Android e IOS.

Num	Elementos
1	Flatlist
2	Image
3	DrawerNavigator
4	StyleSheet
5	Text
6	View
7	ImageBackground
8	TextInput
9	TouchableOpacity
10	Modal
11	Alert
12	AsyncStorage

La Tabla 8 a continuación muestra los 53 elementos de la biblioteca reutilizable de React Native, es decir B=53.

Tabla 8 Lista de elementos de la biblioteca reutilizable de React Native

Num	Elementos
1	AccessibilityInfo
2	ActivityIndicator
3	Alert
4	Animated
5	AppRegistry
6	AppState
7	AsyncStorage
8	BackHandler
9	Button
10	Clipboard
11	Dimensions
12	Easing
13	FlatList
14	Image

Tabla 8 Lista de elementos de la biblioteca reutilizable de React Native (continuación).

Num	Elementos
15	Image Style Props
16	ImageBackground
17	ImageEditor
18	InputAccesoryView
19	InteractionManager
20	Keyboard
21	KeyboardAvoidingView
22	Layout Props
23	LayoutAnimation
24	Linking
25	Modal
26	PanResponder
27	Picker
28	PixelRatio
29	RefreshControl
30	SafeAreaView
31	ScrollView
32	SectionList
33	Settings
34	Shadow Props
35	Share
36	Slider
37	StatusBar
38	StyleSheet
39	Switch
40	Systrace
41	Text
42	Text Style Props
43	TextInput
44	TimePickerAndroid
45	TouchableHighlight
46	TouchableNavigateFeedback
47	TouchableOpacity
48	TouchableWithoutFeedback
49	Transforms
50	Vibration
51	View
52	View Style Props
53	VirtualizedList

Fuente: React Native información oficial

MS3M1: Capacidad de pistas de auditoria

La Tabla 9 a continuación muestra los parámetros para evaluar la capacidad de pistas de auditoria.

Tabla 9. Lista de datos para evaluar la capacidad de pistas de auditoria

A	**B**
Número de datos realmente grabados durante la operación	**Número de datos previstos a grabarse para controlar el estado del sistema durante la operación**
Nombre	Nombre
Tipo	Correo
Id	Contraseña
	Tipo
	Descripción
	Conferencia
	Id
A=3	**B=7**

MS3M2: Diagnóstico de funciones suficientes

La Tabla 10 a continuación presenta las funciones de diagnóstico para satisfacer el cumplimiento de los requerimientos de la aplicación.

Tabla 10. Lista de funciones de diagnóstico de la aplicación “oaqeventos”

Detalle	**Núm. de funciones de diagnóstico implementadas**	**Núm. de funciones de diagnóstico requeridas en la especificación de requerimientos**
Funciones de servicios web	7	10
Funciones para mostrar datos	12	12
Funciones de navegación entre interfaces	4	4
Funciones de login	4	4
Total	27	30

MS4M1: Complejidad ciclomática

En la aplicación caso de estudio, se identificaron 12 instrucciones condicionales en una función.

MS4M2: Profundidad de herencia

Para el caso de React Native las funciones y métodos se heredan directamente de la clase *component,* es decir solo se presenta una jerarquía.

MS4M3: Grado de localización de corrección de impacto

En este caso se pudo identificar 10 fallas aparecidas después de que se ha resuelto un fallo es decir A=10, y un total de 25 fallas resueltas es decir B=25.

MS4M4: Complejidad de modificación

Se identificó un total de 15 modificaciones es decir A=15, y el tiempo que le toma al desarrollador realizar una modificación es de 30 minutos, es decir B=30.

MS4M5: Índice de éxito de modificación

Se identificaron 2 problemas antes del periodo de mantenimiento es decir A=2 y 3 problemas después del periodo de mantenimiento, es decir B=3.

Los problemas tienen que ver con la acumulación de notificaciones en la nube que maneja la persistencia de aplicación.

MS5M1: Complejidad funcional de funciones de prueba

Para esto se debe identificar el número de funciones de prueba implementadas (A) y el número de funciones de prueba requeridas (B). La Tabla 11 muestra el número de funciones de prueba implementadas.

Tabla 11 Lista de funciones de prueba de la aplicación “oaqeventos”.

Función	Detalle
obtenerConferencias()	Permite probar la obtención de conferencias desde el servidor
elminarConferencias()	Recibe un positivo al eliminar datos del servidor
agendar()	Maneja token para el manejo de notificaciones
preguntar()	Maneja el correcto envío de mensajes
obtenerlista()	Permite probar la obtención de eventos desde el servidor
componentDidMount()	Maneja el asyncStorage para recuperar datos
login()	Maneja estados de navegación
obtenerOrganizadores()	Permite probar la obtención de organizadores desde el servidor
A=8	

Sin embargo, las funciones de prueba requeridas por la aplicación se muestran en la Tabla 12 a continuación.

Tabla 12. Lista de pruebas requeridas por la aplicación caso de estudio.

Detalle	Número de funciones de prueba requeridas.
Pruebas de comunicación con los servicios web	7
Pruebas CRUD de datos	6
Pruebas navegación de usuario	4
Pruebas rendimiento	2
Pruebas resolución de pantalla	3
Total	22

MS5M2: Capacidad de prueba autónoma

Para esto se debe identificar el número de pruebas que están dependiendo de otros sistemas (A) y el número total de pruebas dependientes con otros sistemas (B). La Tabla 13 muestra las pruebas totales que dependen de otros sistemas.

Tabla 13 Lista de pruebas totales que dependen de otros sistemas.

Detalle	Núm. de pruebas dependientes de otros sistemas	Núm. total de pruebas dependientes de otros sistemas
Pruebas persistencia de servicios web de datos	7	7
Pruebas CRUD de datos	4	6
Pruebas navegación de usuario	0	0
Pruebas rendimiento	0	0
Pruebas resolución de pantalla	0	0
Total	A=11	B=13

MS5M3: Capacidad de reinicio de pruebas

Finalmente, la Tabla 14 muestra el número de casos para pausar y restaurar una prueba (A) y el número de casos de pausa durante la ejecución de pruebas (B).

Tabla 14 Lista de casos de reinicio de pruebas en la aplicación.

Detalle	Num de casos para pausar y reiniciar una prueba	Num de casos de pausa en la ejecución
Pruebas servicios web de datos	15	24
Pruebas CRUD de datos	15	26
Pruebas navegación de usuario	0	0
Pruebas rendimiento	0	0
Pruebas resolución de pantalla	0	0
Total	A=30	B=50

Con base a la información antes mencionada se procede a evaluar la mantenibilidad del caso de estudio, como se indica en la Tabla 15 la misma que contiene los siguientes datos:

- **Subcaracterísticas:** contiene las subcaracterísticas correspondientes a mantenibilidad.
- **Métricas:** contiene el nombre da cada subcaracterística a ser evaluada.
- **Fórmula:** se explica la fórmula a utilizarse dependiendo de cada métrica.
- **A:** parámetro que varía de acuerdo con la fórmula de cada métrica y que por lo general se encuentra en el numerador de la ecuación.
- **B:** parámetro que varía de acuerdo con la fórmula de cada métrica y que por lo general se encuentra en el denominador de la ecuación.

- **X:** resultado de la evaluación de cada métrica de mantenibilidad.

Por ejemplo, para el caso de la subcaracterística de modularidad para evaluar la capacidad de condensación, la fórmula indica que se debe contar el número de componentes que no son afectados por cambios de otros componentes y el valor obtenido corresponde al parámetro A.

Para el caso del parámetro B se cuenta el número total de componentes específicos, los datos de los parámetros A y B se encuentran en la sección 2.3.1 en la Tabla 6.

Todos los datos para evaluar las subcaracterísticas y sus correspondientes métricas se encuentran en la sección 2.3.1.

Tabla 15. Evaluación de mantenibilidad de la aplicación caso de estudio

SUBCARACTERISTICAS	MÉTRICAS	FÓRMULA	A	B	X
Modularidad	Capacidad de condensación	X = A / B ---A = Número de componentes que no son afectados por cambios de otros componentes B = Número total de componentes específicos Dónde: B > 0	10	19	0,52631579
	Acoplamiento de clases	X = A-- A = Número de relaciones que tiene una función con respecto a otras clases	1		1
Reusabilidad	Ejecución de reusabilidad	X = A / B A = Número de elementos reutilizados B = Número total de elementos de la biblioteca reutilizable Dónde: B > 0	12	53	0,22641509

Tabla 15 Evaluación de mantenibilidad de la aplicación caso de estudio (continuación).

SUBCARACTERISTICAS	MÉTRICAS	FÓRMULA	A	B	X
Capacidad a ser analizado	Capacidad de pistas de auditoria	X = A / B A = Número de datos realmente grabados durante la operación B = Número de datos previstos a grabarse para controlar el estado del sistema durante la operación Dónde: B > 0	4	8	0,5

Tabla 15 Evaluación de mantenibilidad de la aplicación caso de estudio (continuación).

SUBCARACTERISTICAS	MÉTRICA	FÓRMULA	A	B	X
Capacidad a ser analizado	Diagnóstico de funciones suficientes	X = A/B A = Número de funciones de diagnóstico implementadas B = Número de funciones de diagnóstico requeridas en la especificación de requerimientos Dónde: B > 0	27	30	0,9
Capacidad a ser modificado	Complejidad ciclomática	X = A+1 A = Número de instrucciones condicionales que tiene una función	12		13

Tabla 15 Evaluación de mantenibilidad de la aplicación caso de estudio (continuación).

SUBCARACTERISTICAS	MÉTRICA	FÓRMULA	A	B	X
Capacidad a ser modificado	Profundidad de herencia	X = A A = Número de jerarquías empleadas para una determinada función.	1		1
	Grado de localización de corrección de impacto	X = A/B A = Número de fallas aparecidas después que se ha resuelto un fallo B = Número de fallas resueltas Dónde: B > 0	10	25	0,4

Tabla 15 Evaluación de mantenibilidad de la aplicación caso de estudio (continuación).

SUBCARACTERISTICAS	MÉTRICA	FÓRMULA	A	B	X
Capacidad a ser modificado	Complejidad de modificación	X = A/T A = Número de modificaciones B = Tiempo de trabajo que le toma al desarrollador modificar Dónde: T > 0	15	30	0,5
	Índice de éxito de modificación	X = A/B A = Número de problemas dentro de un determinado período antes de mantenimiento B = Número de problemas en el mismo período después del mantenimiento (B>0)	2	3	

Tabla 15 Evaluación de mantenibilidad de la aplicación caso de estudio (continuación).

SUBCARACTERISTICAS	MÉTRICA	FÓRMULA	A	B	X
Capacidad a ser probado	Complejidad funcional de funciones de prueba	X = A/B A = Número de funciones de prueba implementadas B = Número de funciones de prueba requeridas Dónde: B > 0	8	22	0,3636364
	Capacidad de prueba autónoma	X = A/B A = Número de pruebas que están dependiendo de otros sistemas B = Número total de pruebas dependientes con otros sistemas Dónde: B > 0	11	13	0,84615385

Tabla 15 Evaluación de mantenibilidad de la aplicación caso de estudio (continuación).

SUBCARACTERISTICAS	MÉTRICA	FÓRMULA	A	B	X
Capacidad a ser probado	Capacidad de reinicio de pruebas	X = A/B A = Número de casos en los cuales el mantenedor puede pausar y restaurar las pruebas B = Número de casos de pausa en la ejecución de pruebas Dónde: B > 0	30	50	0,6

2.3.2 Identificación de los parámetros a utilizarse en las métricas de portabilidad.

PS1M1: Adaptabilidad en entorno hardware

La siguiente Tabla 16 muestra el número de funciones operativas de las tareas no completadas (A) y el número total de funciones probadas en hardware (B).

Tabla 16. Lista de tareas no completadas en un entorno de hardware.

Detalle	**Num de funciones operativas de las tareas no completadas**	**Num total de funciones que han sido probadas**
Función Acerca de	0	1
Función Agenda	0	1
Función Conferencia	0	1
Función Conferencias	0	1
Función Detalles evento	0	1
Función eventos	0	1
Función evento	0	1
Función Lista preguntas	0	1
Función login	0	1
Función notificaciones	1	1
TOTAL	A=1	B=10

PS1M2: Adaptabilidad en entorno de software

La siguiente Tabla 17 muestra el número de funciones operativas no completadas (A) y el número total de funciones que han sido probadas (B) pero en un entorno de software.

Tabla 17 Lista de tareas no completadas en un entorno de software.

Detalle	**Num de funciones operativas de las tareas no completadas**	**Num total de funciones que han sido probadas**
Función Acerca de	0	1
Función Agenda	0	1
Función Conferencia	0	1
Función Conferencias	0	1
Función Detalles evento	0	1
Función eventos	0	1
Función evento	1	1
Función Lista preguntas	0	1
Función login	0	1
Función notificaciones	1	1
TOTAL	A=2	B=10

PS1M3: Adaptabilidad en entorno organizacional

La Tabla 18 a continuación muestra el número de funciones operativas de las tareas que no se hayan completado con usuarios (A) y el número total de funciones que han sido probadas (B) al entorno organizacional.

Tabla 18. Lista de tareas no completadas de acuerdo a los usuarios.

Tipo de Usuario	**Usuario Visitante**		**Usuario instructor**		**Núm total de funciones probadas**
Completo	**SI**	**NO**	**SI**	**NO**	
Función Acerca de	1	0	1	0	1
Función Agenda	1	0	1	0	1
Función Conferencia	1	0	1	0	1
Función Conferencias	1	0	1	0	1
Función Detalles evento	1	0	1	0	1
Función eventos	1	0	1	0	1
Función evento	1	0	1	0	1
Función Lista preguntas	1	0	1	0	1
Función login	1	0	1	0	1
Función notificaciones	1	0	0	1	1
TOTAL	10	0	9	A=1	B=10

PS2M1: Eficiencia en el tiempo de instalación

Se identificó un reintento al instalar la aplicación móvil (A = 1) en un tiempo total de instalación de 14 segundos (B=14).

PS2M2: Facilidad de instalación

Se identificaron dos casos de uso en el que los usuarios tuvieron éxito al instalar la aplicación cambiando el proceso de instalación a su conveniencia (A = 2) y seis casos en los que los usuarios cambiaron la configuración de instalación del sistema.

PS3M1: Consistencia en la función de soporte al usuario

Para el caso de la aplicación caso de estudio se implementó la función nuevas preguntas es decir que el número de nuevas funciones que son consideradas como no consistentes por el usuario es igual a 1 (A=1).

De igual forma el número de nuevas funciones es 4 (B=4), se identificó la función agenda, notificaciones, lista preguntas y organizadores.

PS3M2: Inclusividad funcional

La inclusividad funcional mide si se pueden utilizar las funciones después de ser cambiadas por otras similares. Para lo cual se identificó que el número de funciones que producen resultados similares con anterioridad que no han exigido cambios es siete (A=7), y el número de funciones probadas que son similares a las funciones proporcionadas por otro software para ser reemplazado es 10 (B=10) tal como muestra la Tabla 19.

Tabla 19. Lista del número de funciones que no han exigido cambios

Detalle	Num de funciones que producen resultados similares (A)	Num de funciones probadas que son similares a las funciones proporcionadas por otro software (B)
Función Acerca de	1	1
Función Agenda	1	1
Función Conferencia	1	1
Función Conferencias	1	1
Función Detalles evento	1	1
Función eventos	1	1
Función evento	1	1
Función Lista preguntas		1
Función login		1
Función notificaciones		1
TOTAL	7	10

PS3M3: Uso continuo de datos

La métrica identifica el número de datos fácilmente utilizados después de reemplazar el software por uno similar, para lo cual se identificó el número de datos continuamente utilizados (A) y el número de datos continuamente reutilizados (B), como se indica en la Tabla 20.

Tabla 20. Lista de datos utilizados y reutilizados en el reemplazo de software

Datos	Num de datos que son continuamente utilizables por el software (A)	Num de datos que son continuamente reutilizables por el software a ser reemplazado (B)
Nombre	-	1
Token	1	-
Correo	-	1
Contraseña	-	1
Perfil	-	1
Id_evento	1	1
Id_conferencia	1	1
Id_instructor	-	1
Id_usuario	1	1
TOTAL	4	8

La Tabla 21 contiene los siguientes datos:

- **Subcaracterísticas:** contiene las subcaracterísticas correspondientes a mantenibilidad.
- **Métricas:** contiene el nombre de cada subcaracterística a ser evaluada.
- **Fórmula:** se explica la fórmula a utilizarse dependiendo de cada métrica.
- **A:** parámetro que varía de acuerdo con la fórmula de cada métrica y que por lo general se encuentra en el numerador de la ecuación.
- **B:** parámetro que varía de acuerdo con la fórmula de cada métrica y que por lo general se encuentra en el denominador de la ecuación.
- **X:** resultado de la evaluación de cada métrica de portabilidad.

Por ejemplo; para el caso de la subcaracterística de adaptabilidad, para evaluar la adaptabilidad de entorno de hardware, la fórmula indica que se debe contar número de funciones operativas de las tareas que no se hayan completado durante las pruebas, el valor obtenido corresponde al parámetro A.

Para el caso del parámetro B, se cuenta el número total de funciones que han sido probadas, los datos de los parámetros A y B se encuentran en la sección 2.3.2 en la Tabla 16. Todos los datos para evaluar portabilidad con las subcaracterísticas y métricas se encuentran en la sección 2.3.2.

Tabla 21. Evaluación de portabilidad de la aplicación caso de estudio.

SUBCARACTERISTICAS	MÉTRICAS	FÓRMULA	A	B	X
Adaptabilidad	Adaptabilidad en entorno hardware	X = A/B, A = Número funciones operativas de las tareas que no se hayan completado durante las pruebas operativas con el entorno hardware, B = Número total de funciones que han sido probadas (Dónde B > 0)	1	10	0,1
	Adaptabilidad en entorno de software	X = A/B, A = Número de funciones operativas de las tareas que no se hayan completado durante las pruebas operativas con el sistema, B=Número total de funciones que han sido probadas (Donde B > 0)	2	10	0,2
	Adaptabilidad en entorno organizacional	X = A/B, A = Número de funciones operativas de las tareas que no se hayan completado durante las pruebas operativas con usuarios del entorno empresarial, B = Número total de funciones que han sido probadas (Dónde B > 0)	1	10	0,1

Tabla 21 Evaluación de portabilidad de la aplicación caso de estudio (continuación).

SUBCARACTERISTICAS	MÉTRICAS	FÓRMULA	A	B	X
Capacidad a ser instalado	Eficiencia en el tiempo de instalación	X = A/T, A = Número de reintentos al instalar el sistema, T = Tiempo total transcurrido al instalar el sistema (Dónde T > 0)	1	14	0,07
	Facilidad de instalación	X = A/B A = Número casos en que los usuarios tuvieron éxito al instalar el sistema cambiando proceso de instalación para su conveniencia, B = Número total de casos en que los usuarios han intentado cambiar el proceso de instalación para su conveniencia (Donde B > 0)	4	6	0,67

Tabla 21 Evaluación de portabilidad de la aplicación caso de estudio (continuación).

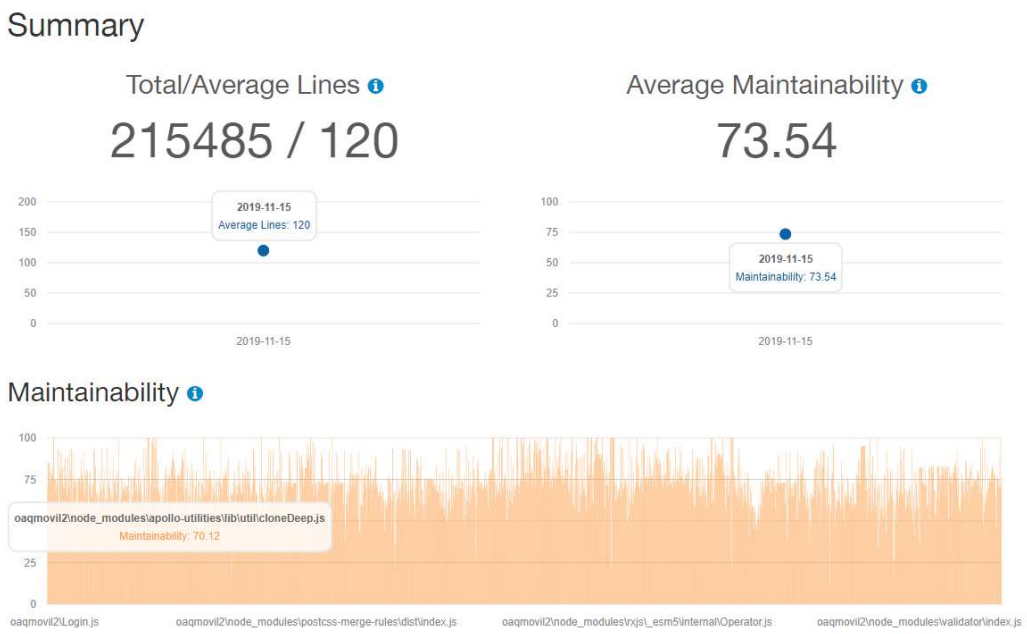
SUBCARACTERISTICAS	MÉTRICAS	FÓRMULA	A	B	X
Capacidad a ser reemplazado	Consistencia en la función de soporte al usuario	X = A/B --A = Número de nuevas funciones que son consideradas como no consistentes por el usuario, B = Número de nuevas funciones (Donde B > 0)	1	4	0,25
	Inclusividad funcional	X = A/B A = Número de funciones que producen resultados similares con anterioridad y que no se han exigido cambios, B = Número de funciones probadas que son similares a las funciones proporcionadas por otro software para ser reemplazado (Donde B > 0)	4	10	0,4
	Uso continuo de datos	X = A/B A = número de datos que son continuamente solo utilizables por el software a ser reemplazado, B = Número de datos que son reutilizables por el software a ser reemplazado (Donde B > 0)	4	8	0,5

2.3.4 Índice de mantenibilidad calculado con el software plato.js.

Cabe mencionar que el índice de mantenibilidad se basa únicamente en el código fuente, es decir que para este estudio se presenta como un complemento para el caso de mantenibilidad.

La figura a continuación muestra el porcentaje de mantenibilidad del framework RN de la aplicación caso de estudio. El resultado del software Plato.js se genera con base a métricas de Volumen de Halstead (HV), Complejidad Ciclomática (CC), Número Promedio de Líneas de Código (LOC) y una métrica opcional, Porcentaje de comentarios a LOC en un Módulo (COM).

Figura 9 Resultados del índice de mantenibilidad obtenidos del software Plato.js



Fuente: Software Plato.js

El presente capítulo describe la metodología a utilizarse en el proyecto, presenta el caso de estudio a evaluar. Con base en la información del capítulo 1 también se presenta la matriz de evaluación a aplicar.

Se detalla cada uno de los parámetros que necesita la matriz de evaluación para evaluar las métricas de las subcaracterísticas de mantenibilidad y portabilidad.

Finalmente, el capítulo presenta los resultados preliminares al aplicar la matriz de prueba de mantenibilidad y portabilidad.

Se concluye que los parámetros y la evaluación requieren de la revisión de cada parámetro como lo indica la matriz de prueba, y en algunos casos de la revisión de conceptos, módulos y funciones.

2.4 Buenas prácticas del desarrollo de software en React Native

Con base a los resultados antes presentados se generan las siguientes buenas prácticas de desarrollo de software:

- Comentar el código de acuerdo con las normas y reglas de desarrollo.

- En lo posible desarrollar la aplicación considerando que todos los componentes a implementarse sean independientes a cambios.
- Es importante al momento de desarrollar una aplicación multiplataforma, utilizar la mayor cantidad de elementos que proporciona React Native para que funcionen de manera simultánea en Android e IOS.

- Es importante implementar funciones que permitan detectar posibles errores en la aplicación.

- Es importante reducir el número de condicionales dentro de una función para garantizar a futuro que el código sea modificable.

- Codificar la aplicación lo más legible posible para que al efectuar cambios se realicen de manera rápida y práctica.

- Generar el número suficiente de funciones de prueba para garantizar que producto final se mantenga sin fallas.

- Generar funciones de prueba autónomas es decir que no dependan de todo el sistema sino solo de partes que facilite al tester realizar las pruebas preliminares.

- Probar la aplicación con los usuarios del sistema para corregir posibles errores de funcionalidad.

- Al momento de generar el “.apk” o “.ipa” generarlos a manera que no necesite mayores permisos para ser instalado.

- Garantizar que los datos y persistencia de la aplicación no generen información parásita dentro del dispositivo.

3. RESULTADOS Y DISCUSIÓN

3.1. Resultados

Luego de concluir la evaluación con los resultados preliminares presentados, se procede a tabular los valores para las métricas de mantenibilidad y portabilidad respectivamente.

Los resultados se estimaron tomando como base el valor deseado en algunos casos el mejor valor era el más cercano a cero, en otros el más cercano a 1 y finalmente en otros casos el valor se encontró dentro de un rango.

El valor obtenido está en la escala de 0 a 1, donde 1 es la máxima ponderación. Como todas las fórmulas y valores son diferentes de acuerdo a cada métrica, se calculó de manera individual las Tablas 22 y 23. A continuación, la Tabla 22 muestra los resultados de mantenibilidad.

Tabla 22 Resultados de mantenibilidad.

Subcaracterísticas	**Métricas**	**Resultado obtenido**	**Valor deseado**	**Valor ponderado**
Modularidad	Capacidad de condensación	0,5	0<=X<=1 El más cercano a 0, es el mejor	0,5
	Acoplamiento de clases	1,0	1<=X<=4 El más cercano a 1, es el mejor	1
Reusabilidad	Ejecución de reusabilidad	0,2	0<=X<=1 El más cercano a 1, es el mejor	0,2264
Capacidad a ser analizado	Capacidad de pistas de auditoria	0,5	0<=X<=1 El más cercano a 1, es el mejor	0,5
	Diagnóstico de funciones suficientes	0,9	0<=X<=1 El más cercano a 1, es el mejor	0,9

Tabla 22 Resultados de mantenibilidad (continuación).

SUBCARACTERÍSTICAS	MÉTRICAS	Resultado obtenido	Valor deseado	Valor ponderado
Capacidad a ser modificado	Complejidad ciclomática	13,0	1<=X<15 El más cercano a 1, es el mejor	0,12
	Profundidad de herencia	1,0	0<=X<=4 El más cercano a 0 es el mejor	0,75
	Grado de localización de corrección de impacto	0,4	0<=X<=1 El más cercano a 0, es el mejor	0,6
	Complejidad de modificación	0,5	X = A/T El más lejano a 0/t es el mejor	0,5
	Índice de éxito de modificación	0,7	0<=X<=1 El más cercano a 0, es el mejor	0,3
Capacidad a ser probado	Complejidad funcional de funciones de prueba	0,4	0<=X<=1 El más cercano a 1, es el mejor	0,4
	Capacidad de prueba autónoma	0,8	0<=X<=1 El más cercano a 0, es el mejor	0,2
	Capacidad de reinicio de pruebas	0,6	0<=X<=1 El más cercano a 1, es el mejo	0,6

De igual forma que la mantenibilidad, se procede con los valores obtenidos para portabilidad, la Tabla 23 muestra los resultados de portabilidad.

Tabla 23 Resultados de portabilidad.

SUBCARACTERÍSTICAS	MÉTRICAS	Resultado obtenido	Valor deseado	Valor ponderado
Adaptabilidad	Adaptabilidad en entorno de hardware	0,1	0<=X<=1 El más cercano a 0, es el mejor	0,9
	Adaptabilidad en entorno de software	0,2	0<=X<=1 El más cercano a 0, es el mejo	0,8
	Adaptabilidad en entorno organizacional	0,1	0<=X<=1 El más cercano a 0, es el mejor	0,9
Capacidad de ser instalado	Eficiencia en el tiempo de instalación	0,07	X = A/T El más lejano a 0/t es el mejor	0,93
	Facilidad de instalación	0,67	0<=X<=1 El más cercano a 1, es el mejor	0,67
Capacidad de ser reemplazado	Consistencia en la función de soporte al usuario	0,3	0<=X<=1 El más cercano a 0, es el mejor	0,7
	Inclusividad funcional	0,4	0<=X<=1 El más cercano a 1, es el mejo	0,4
	Uso continuo de datos	0,5	0<=X<=1 El más cercano a 1, es el mejor	0,5

Para obtener el porcentaje final de portabilidad y mantenibilidad; se procedió a sumar todos los valores obtenidos y los valores deseados correspondientes de todas las subcaracterísticas, como lo indican las Tablas 24 y 25 a continuación. La Tabla 25 muestra el resultado final correspondiente a mantenibilidad.

Tabla 24. Ponderación final de la evaluación de mantenibilidad.

MANTENIBILIDAD			
SUBCARACTERÍSTICAS	**MÉTRICAS**	**Valor ponderado**	**Máximo valor**
Modularidad	Capacidad de condensación	0,5	1
	Acoplamiento de clases	1	1
Reusabilidad	Ejecución de reusabilidad	0,2	1
Capacidad a ser analizado	Capacidad de pistas de auditoria	0,5	1
	Diagnóstico de funciones suficientes	0,9	1
Capacidad a ser modificado	Complejidad ciclomática	0,12	1
	Profundidad de herencia	0,75	1
	Grado de localización de corrección de impacto	0,6	1
	Complejidad de modificación	0,5	1
	Índice de éxito de modificación	0,3	1
Capacidad a ser probado	Complejidad funcional de funciones de prueba	0,4	1
	Capacidad de prueba autónoma	0,2	1
	Capacidad de reinicio de pruebas	0,6	1
	Sumatoria	6,57	13
	Porcentaje Final	**50,53%**	**100%**

La Tabla 25 muestra el resultado final correspondiente a portabilidad.

Tabla 25. Ponderación final de la evaluación de portabilidad.

PORTABILIDAD			
SUBCARACTERÍSTICAS	**MÉTRICAS**	**Valor ponderado**	**Máximo valor**
Adaptabilidad	Adaptabilidad en entorno de hardware	0,9	1
	Adaptabilidad en entorno de software	0,8	1
	Adaptabilidad en entorno organizacional	0,9	1
Capacidad de ser instalado	Eficiencia en el tiempo de instalación	0,93	1
	Facilidad de instalación	0,67	1
Capacidad de ser reemplazado	Consistencia en la función de soporte al usuario	0,75	1
	Inclusividad funcional	0,4	1
	Uso continuo de datos	0,5	1
	Sumatoria	5,85	8
	Porcentaje Final	**73.13%**	**100%**

3.2. Discusión y análisis de resultados

Medir el grado de calidad para mantenibilidad y portabilidad del software para el framework React Native resulta complejo, por cuanto es un entorno de desarrollo nuevo que basa todo su funcionamiento en encapsular el conocido lenguaje javascript a un ambiente nativo.

La mantenibilidad representa la capacidad del producto de software para ser modificado efectiva y eficientemente a lo largo su evolución, corrección o perfección, e involucra a varios actores en específico al desarrollador y al tester.

Para el caso de portabilidad el enfoque es diferente puesto que se analiza la capacidad del producto de software a ser transferido de un entorno de software o hardware a otro.
El presente trabajo presenta una alternativa de evaluación para mantenibilidad y portabilidad basado en el modelo de calidad ISO 25010, mediante la utilización de métricas de calidad interna y externa planteadas por la norma ISO/IEC 25022.

Las matrices de evaluación aquí utilizadas facilitan la evaluación para este tipo de producto de software, dando un resultado positivo en cuanto a la metodología de evaluación empleada, puesto que se pude identificar claramente requisitos y datos a utilizarse en la evaluación.

A continuación, se analizarán y discutirán los resultados métrica por métrica:

MS1M1: Capacidad de condensación
El valor obtenido es de 0,5, la métrica depende directamente del número de componentes que no son afectados por cambios en otros componentes, el resultado es bueno por cuanto 10 de los 19 componentes muestran independencia a cambios en la aplicación.

MS1M2: Acoplamiento de clases
El resultado es excelente por cuanto el valor obtenido fue el máximo es decir 1, esto implica una ventaja puesto que en React Native existe una función llamada “ComponentDidMount”, encargada de facilitar el manejo el ciclo de vida de todas las clases.

MS2M1: Ejecución de reusabilidad

Esta métrica es importante por cuanto responde a la incógnita que muchas empresas y desarrolladores en general se plantean. Es decir que tan reutilizables son los elementos cuando hablamos de dos sistemas operativos como Android e IOS.

El valor que se obtuvo es de 0,2, alrededor de un 20 % de elementos reutilizables, esto implica que fácilmente se puede desarrollar una aplicación promedio con similares características y funcionalidad en los dos sistemas operativos. Es decir, donde la navegación, el uso de menús y la persistencia a base de datos es prácticamente lo mismo.

MS3M1: Capacidad de pistas de auditoria

La métrica tiene que ver con los datos grabados durante la operación de la aplicación. El valor que se obtuvo fue de 0,5, lo cual es bueno por cuanto de 8 datos a guardarse en desarrollo se decidió guardar 4 es decir la mitad, y más bien trasladar todo el trabajo a los servicios en la nube y no en el dispositivo.

MS3M2: Diagnóstico de funciones suficientes

La métrica analiza el número de funciones de diagnóstico a implementarse para detectar algún error, sea por el desarrollador o el equipo de pruebas.
El valor obtenido fue de 0,9 de 1, es decir un valor alto que implica que React Native es versátil a la hora de crear funciones de prueba.

MS4M1: Complejidad ciclomática

La métrica analiza el número de condicionales dentro de una función y es importante tomarlo en cuenta por cuanto si una aplicación contiene muchos condicionales es más difícil modificarla.

El valor obtenido es 0,12, puesto que se identificaron un total de 12 condicionales, esto significa que la aplicación caso de estudio no es modificable con facilidad.

MS4M2: Profundidad a herencia

La métrica analiza el nivel jerárquico de las funciones y mientras menos jerarquía exista es mejor.

El valor obtenido es de 0,75 lo cual es bastante bueno por cuanto en React Native todas las clases se heredan de la clase Component. Es una ventaja significativa e implica un excelente manejo de eventos, datos y estados.

MS4M3: Grado de localización de corrección de impacto

La métrica analiza que tan fácil es la localización y corrección de errores; el valor obtenido es de 0,6, es aceptable e implica que React Native si maneja y permite corregir las fallas de manera accesible para el desarrollador.

MS4M4: Complejidad de modificación

La métrica analiza el tiempo que le toma al desarrollador modificar la aplicación para el caso de estudio; se obtuvo un valor de 0,5, es moderado porque no le resulta ni rápido ni lento al desarrollador modificar código en React Native, tomando en cuenta que el framework es nuevo el resultado es aceptable.

MS4M5: Índice de éxito de modificación

La métrica analiza el número de problemas dentro de un determinado periodo antes del mantenimiento, el valor obtenido es de 0,3 lo que implica que, si se presentaron errores durante la modificación, lo errores presentados se deben principalmente al manejo de notificaciones.

MS5M1: Complejidad funcional de funciones de prueba

La métrica analiza el número de funciones de prueba implementadas; el valor obtenido es de 0,4, puesto que se implementaron 8 funciones de prueba de 22 requeridas.

Es decir que depende directamente del tester y del desarrollador, sin embargo, React Native si permite la implementación de ambientes de prueba sin problema.

MS5M2: Capacidad de prueba autónoma

La métrica analiza el número de pruebas que dependen de otros sistemas; el valor obtenido es de 0,2, es un valor bajo por cuanto al ser una aplicación cliente servidor la mayoría de las pruebas dependen de otros sistemas.

MS5M3: Capacidad de reinicio de pruebas

La métrica analiza la capacidad que tiene el producto de software en pausar y reiniciar las pruebas. El valor obtenido es de 0,6, es bastante bueno por cuanto React Native es flexible al momento de generar pruebas por cuanto utiliza JavaScript.

PS1M1: Adaptabilidad en entorno de hardware

Es una de las primeras métricas de portabilidad, y mide que tan adaptables son las funciones a un entorno de hardware. El valor obtenido fue de 0,9, es alto por cuanto la mayoría de las funciones se adapta sin problemas al entorno de hardware

PS1M2: Adaptabilidad en entorno de software

La métrica mide que tan adaptable es una función a un entorno de software, en el presente proyecto se analizó dos ambientes básicamente Android e IOS, dando como resultado un valor igual a 0,8 por cuanto el manejo de eventos y notificaciones de la aplicación difiere un poco en los sistemas operativos antes mencionados.

PS1M3: Adaptabilidad en entorno organizacional

La métrica analiza si las funciones operativas se completaron con usuarios del sistema. El valor obtenido fue de 0,9, es decir excelente por cuanto todos los requerimientos del usuario se completaron en la mayoría.

PS2M1: Eficiencia en tiempo de instalación

La métrica analiza el tiempo de instalación del sistema, el valor obtenido es de 0,93, con lo que se comprueba que la aplicación en React Native tiene una alta eficiencia en instalación.

PS2M2: Facilidad de instalación

La métrica analiza a los usuarios que tuvieron éxito al instalar la aplicación. El valor obtenido fue de 0,67, esto garantiza que los usuarios no tendrían ningún inconveniente en instalar la aplicación móvil.

PS3M1: Consistencia de la función de soporte al usuario

La métrica analiza el número de funciones consideradas como no consistentes por el usuario. El valor obtenido fue de 0,75 por cuanto se identificó una función que los usuarios consideran como no consistente.

PS3M2: Inclusividad funcional

La métrica analiza si se pueden utilizar las funciones después de ser cambiadas por otras similares. El valor obtenido fue de 0,4, por cuento 7 de 10 funciones si se pueden volver a utilizar. Esta métrica depende más de cómo se desarrolló el software.

PS3M3: Uso continuo de datos

La métrica analiza el número de datos fácilmente utilizados después de reemplazar el software por uno similar. El valor obtenido fue de 0,5 lo cual es bueno y indica que React Native si puede reutilizar los datos.

Finalmente, también se presentó el índice de mantenibilidad que proporciona el software Plato.js como complemento al presente proyecto, el mismo que tiene un valor de 73.54. El índice generado indica un resultado favorable por cuanto garantiza que sí se puede mantener la aplicación en React Native dado que se desarrolló con funciones, métodos y un número significativo de comentarios al código que garantizan futuros cambios.

4. CONCLUSIONES

- En el presente proyecto se evalúan las características de mantenibilidad y portabilidad del framework React Native utilizando la norma ISO 25010.

- Se estudiaron claramente las subcaracterísticas de mantenibilidad y portabilidad identificando las diferentes métricas descritas en la norma ISO/IEC 25022.

- Se identificaron buenas prácticas para mejorar la calidad de software del caso de estudio desarrollado en React Native.

- React Native si garantiza la reusabilidad por cuanto contiene una amplia librería de elementos reutilizables.

- El uso de buenas prácticas en el desarrollo de software puede incrementar notablemente la mantenibilidad y portabilidad del software.

- La mantenibilidad y portabilidad depende directamente del desarrollador, código fuente y tester por lo que es indispensable que se desarrolle con uso adecuado de todos los requerimientos.

- Los resultados generales mantenibilidad y portabilidad garantizan que React Native es una excelente alternativa para las empresas y desarrolladores que quieren implementar aplicaciones multiplataforma.

REFERENCIAS BIBLIOGRÁFICAS

Aguirre, G. (2018). *Por qué React Native es una excelente alternativa en América Latina*. https://medium.com/underscopeio/por-qué-React-Native-es-la-mejor-alternativa-para-américa-latina-97cf3041d6ae

Ahumada, R. (n.d.). *React Native: Una nueva arquitectura para la generación de Apps - Bit*. Retrieved April 24, 2019, from https://www.bit.es/knowledge-center/React-Native-una-nueva-arquitectura-para-la-generacion-de-apps-moviles/

Amirhossein Mousavi, S. (2016). *Maintainability Evaluation of Single Page Application Frameworks- Angular2 vs. React*. 39. http://www.diva-portal.org/smash/get/diva2:1076563/FULLTEXT01.pdf

Charland, A., & LeRoux, B. (2011). Mobile Application Development: Web vs. Native. *Queue*, *9*(4), 20:20--20:28. https://doi.org/10.1145/1966989.1968203

Chisaguano Balseca, E. A. (2014). *Evaluación de Calidad de Productos Software en Empresas de desarrollo de Software aplicando la Norma ISO/IEC 25000*. 8–11. http://bibdigital.epn.edu.ec/handle/15000/9113

Dromey, R. G. (1995). A Model for Software Product Quality. *IEEE Trans. Softw. Eng.*, *21*(2), 146–162. https://doi.org/10.1109/32.345830

Greenfield, M. (n.d.). • *Number of smartphone users worldwide 2014-2020* | *Statista*. Retrieved March 23, 2020, from https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/

ISO_25010. (2011). ISO/IEC 25010:2011: Systems and Software Engineering - Systems and Software Product Quality Requirements and Evaluation - System and Software Quality Models, 4.1.3.2. In *ISO* (p. 4.). https://www.iso.org/obp/ui/#iso:std:iso-iec:25010:ed-1:v1:en

ISO_25022. (2016). *ISO/IEC 25022:2016 [ISO/IEC 25022:2016] Systems and software engineering — Systems and software quality requirements and evaluation (SQuaRE) — Measurement of quality in use* (p. 41).

Johansson, E., & Söderberg, J. (2018). *Evaluating performance of a React Native feature set*.