



ESPE
UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

Evaluación y selección de frameworks para el desarrollo de aplicaciones móviles según los criterios de calidad de la norma ISO: un enfoque en la calidad de uso, externa e interna.

Chuquitarco Copara, Angie Maricela y Unda Reinoso, Anthony Josue

Departamento de Ciencias de la Computación

Carrera de Ingeniería de Software

Trabajo de Unidad de Integración Curricular, previo a la obtención del Título de Ingeniero de Software

Msc. Montaluisa Yugla, Franklin Javier

01 de marzo del 2024

Latacunga- Ecuador



Tesina_Unda_Chuquitarco VERSION R...

Scan details

Scan time:
February 29th, 2024 at 20:56 UTC

Total Pages:
79

Total Words:
19727

Plagiarism Detection



AI Content Detection



Plagiarism Results: (44)

NORMAS ISO 25000 1.2%
https://iso25000-com.translate.google.com/index.php/normas-iso-25000?x_tr_sl=es&x_tr_tl=en&x_tr_hi=en&x_tr...
 iso25000.com Buscar... ..

Proyecto ra | PDF 1.1%
https://es-slideshare-net.translate.google.com/2609/proyecto-ra?x_tr_sl=es&x_tr_tl=en&x_tr_hi=en&x_tr_pto=sc...
 ...


REALIDAD VIRTUAL | PDF 1%
https://es-slideshare-net.translate.google.com/diegobravo862113/realidad-virtual-255471391?x_tr_sl=es&x_tr_tl=en...
 ...

La familia de normas ISO/IEC 25000 0.9%
<https://iso25000.com/index.php/normas-iso-25000/iso-25010?id=10&start=3>
 iso25000.com ...



About this report
help.copyleaks.com





MSc. Montaluisa Yugla, Franklin Javier
 C.C.: 0502166796




Departamento de Ciencias de la Computación

Carrera de Software

Certificación

Certifico que el trabajo de Unidad de integración curricular: **"Evaluación y selección de frameworks para el desarrollo de aplicaciones móviles según los criterios de calidad de la norma ISO: un enfoque en la calidad de uso, externa e interna"** fue realizado por los estudiantes **Chuquitarco Copara, Angie Maricela y Unda Reinoso, Anthony Josue** los mismo que cumple con los requisitos legales, teóricos, científicos, técnicos y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, además fue revisado y analizada en su totalidad por la herramienta de prevención y/o verificación de similitud de contenidos; razón por la cual me permito acreditar y autorizar para que se lo sustenten públicamente.

Latacunga, 01 de marzo del 2024



MSc. Montalujsa Yugla Franklin Javier
C.C.: 0502166796



Departamento de Ciencias de la Computación

Carrera de Software

Responsabilidad de autoría

Nosotros, **Chuquitarco Copara, Angie Maricela y Unda Reinoso, Anthony Josue**, con cédula de ciudadanía n° 1719652420 y n° 0502888738, declaramos que el contenido, ideas y criterios del trabajo de Unidad de integración curricular: **“Evaluación y selección de frameworks para el desarrollo de aplicaciones móviles según los criterios de calidad de la norma ISO: un enfoque en la calidad de uso, externa e interna”** es de nuestra autoría y responsabilidad, cumpliendo con los requisitos legales, teóricos, científicos, técnicos, y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, respetando los derechos intelectuales de terceros y referenciando las citas bibliográficas.

Latacunga, 01 de marzo del 2024

Chuquitarco Copara, Angie Maricela

C.C.: 1719652420

Unda Reinoso, Anthony Josue

C.C.: 0502888738



Departamento de Ciencias de la Computación

Carrera de Software

Autorización de publicación

Nosotros, Chuquitarco Copara, Angie Maricela y Unda Reinoso, Anthony Josue, con cédula de ciudadanía n° 1719652420 y n° 0502888738, autorizamos a la Universidad de las Fuerzas Armadas ESPE publicar el trabajo de Unidad de integración curricular: **"Evaluación y selección de frameworks para el desarrollo de aplicaciones móviles según los criterios de calidad de la norma ISO: un enfoque en la calidad de uso, eterna e interna"** en el Repositorio Institucional, cuyo contenido, ideas y criterios son de nuestra responsabilidad.

Latacunga, 01 de marzo del 2024

Chuquitarco Copara, Angie Maricela

C.C.: 1719652420

Unda Reinoso, Anthony Josue

C.C.: 0502888738

Dedicatoria

“Agradezco a Dios por ser mi guía constante, brindándome la fuerza necesaria para superar desafíos.

A mi madre, Aurora Copara, pilar fundamental en mi vida, cuyo amor, dedicación y perseverancia han sido constantes. Gracias por inculcarme los valores de la constancia y determinación, los cuales han sido fundamentales para alcanzar metas.

A mi hermano, Andres, cuyo apoyo constante y buenos deseos ha sido mi mejor compañía, siendo un confidente en este arduo recorrido académico.

A mi tía Piedad Tasipanta, cuya constante presencia ha sido un apoyo invaluable y ha sido testigo de mi crecimiento y evolución como persona a lo largo del tiempo.

A mis queridos amigos Stalin, Brandon y Romel: ustedes han sido mis compañeros de viaje en este desafiante recorrido universitario, juntos hemos enfrentado aventuras, superado obstáculos y compartido risas, siendo mi apoyo incondicional en los momentos de mayor dificultad.

A mi futuro compañero de vida, Roberto, quien ha sido mi soporte incluso en la distancia, brindándome su apoyo incondicional en cada paso de este viaje.

Con profunda gratitud y emoción, este logro se transforma en un tributo a la maravillosa red de personas que han iluminado mi sendero. Cada uno de ustedes ha sido un faro de inspiración y apoyo, alimentando mi crecimiento tanto personal como profesional. Esta dedicatoria apenas rasga la superficie de la inmensa gratitud y cariño que albergo hacia cada uno de ustedes”.

Angie Maricela, Chuquitarco Copara

Dedicatoria

A mi querida madre, Maribel Reinoso, cuyo amor incondicional, dedicación y sacrificio han sido la roca sobre la cual se ha construido este proyecto. Tu presencia constante y tu apoyo desinteresado son el faro que ha iluminado mi camino hacia este logro, y por eso te dedico este trabajo con todo mi corazón.

A mi hermana, Debora Morales, quien ha sido mi confidente más cercana y mi compañera inquebrantable en cada etapa de este viaje. Tus palabras de aliento, tu apoyo incondicional y tu amor sincero han sido mi fuente de fortaleza en los momentos más oscuros

A mi hermano Francisco Unda, cuya sabiduría y experiencia han sido un faro de orientación en medio de la tormenta. Tu consejo siempre oportuno y tu apoyo incondicional han sido fundamentales para superar los obstáculos que encontré en mi camino hacia la culminación de este proyecto.

A mi hermano David Unda, cuya sabiduría y perspicacia han sido una guía invaluable en los momentos de incertidumbre y toma de decisiones difíciles. Tu capacidad para ver más allá de las apariencias y tu agudo sentido de la razón han sido una inspiración para mí en mi búsqueda del conocimiento y la excelencia académica. Te dedico este trabajo como un testimonio de nuestro vínculo fraternal y como un homenaje a tu inquebrantable apoyo a lo largo de este viaje.

Este logro no habría sido posible sin el amor, el apoyo y la guía de mi familia.

Anthony Josue, Unda Reinoso

Agradecimiento

Quiero transmitir mi profundo agradecimiento a mis colegas de estudio, cuyo respaldo ha sido vital en la creación de esta tesina. Sus valiosas ideas y enfoques han sido elementos esenciales para dar forma a este documento académico.

Me gustaría expresar mi profundo agradecimiento a los profesores que me brindaron la base sólida y las herramientas esenciales para enfrentar este desafío académico. Su compromiso y pasión por la docencia son ejemplares, y su dedicación incansable ha sido fundamental en mi formación.

Agradezco a la Universidad de las Fuerzas Armadas “ESPE-L” por brindarme la oportunidad de desarrollarme, proporcionándome los conocimientos necesarios para trazar mi camino y por motivarme a disfrutar cada momento de mi carrera en ingeniería de software.

Angie Maricela, Chuquitarco Copara

Agradecimiento

Quiero expresar mi profundo agradecimiento a los distinguidos docentes de mi querida universidad por compartir generosamente su vasto conocimiento y experiencia, que han enriquecido enormemente mi desarrollo académico y personal.

A mi tutor, el Mgtr. Javier Montaluisa, le agradezco sinceramente por su guía experta y constante apoyo durante todo el proceso de desarrollo de este proyecto. Su orientación ha sido fundamental para alcanzar cada uno de los hitos y superar los desafíos encontrados en el camino hacia la culminación de esta tesis.

A mis queridos tíos Katy Reinoso y Elías Muñoz, les agradezco de todo corazón por su confianza inquebrantable y su apoyo incondicional. Sus palabras de aliento y ánimo siempre han sido un faro de luz en los momentos de duda y dificultad, impulsándome a seguir adelante con determinación y perseverancia.

A mis compañeros Klever Mise y Fabricio Tapia, les agradezco por su amistad sincera y lealtad inquebrantable. Su compañerismo y apoyo mutuo han hecho de este viaje académico una experiencia memorable y enriquecedora.

Y a mi querida mascota Osa, mi fiel compañera de todas las horas, gracias por tu amor incondicional y tu presencia reconfortante. Tu compañía ha sido una fuente inagotable de alegría y consuelo durante los momentos de trabajo arduo y estrés.

Este logro no habría sido posible sin el apoyo y la contribución de cada uno de ustedes. Su generosidad y aliento han sido un regalo invaluable que siempre atesoraré en mi corazón.

Anthony Josue, Unda Reinoso

ÍNDICE DE CONTENIDOS

Carátula	1
Reporte de verificación de contenido	2
Certificación	3
Responsabilidad de autoría.....	4
Autorización de publicación.....	5
Dedicatoria.....	6
Dedicatoria.....	7
Agradecimiento.....	8
Agradecimiento.....	9
Índice de contenidos	10
Índice de tablas	14
Índice de figuras	16
Resumen	18
Abstract.....	19
Capítulo I: Introducción	20
Antecedentes	20
Propósito y contexto del proyecto.....	21
Justificación de la evaluación	22
Alcance	22
Objetivos	23
Objetivo General.....	23

<i>Objetivos Específicos</i>	23
Capítulo II: Marco Teórico	24
Norma ISO 25000	24
Realidad aumentada y señales de tránsito	27
<i>Realidad aumentada</i>	27
<i>Definición y características de Realidad Aumentada</i>	29
<i>Señales de tránsito</i>	29
<i>Importancia de las señales de tránsito en Realidad Aumentada</i>	30
Calidad en aplicaciones de RA	31
<i>Relevancia de la evaluación de calidad en RA</i>	31
<i>Norma ISO 25000 y su aplicabilidad a RA</i>	32
Framework de Desarrollo de Aplicaciones Móviles	32
<i>Descripción del frameworks desde cero</i>	33
<i>Flutter</i>	33
<i>React Native</i>	34
<i>Xamarin</i>	34
<i>Ionic</i>	35
Consideraciones de diseño y desarrollo	35
Capítulo III: Definición de metodología y métrica para la evaluación de calidad	38
Métricas de calidad interna, externa y de uso	38
<i>Calidad interna</i>	38

<i>Calidad Externa</i>	39
<i>Calidad de uso</i>	41
Adaptación de Métricas de la Norma ISO 25000 al método BBR	42
Desarrollo de modelo de evaluación BRR aplicado a la ISO	43
<i>Definición de criterios y umbrales de calidad</i>	48
Diseño de la evaluación	55
<i>Selección de escenarios de evaluación</i>	55
<i>Definición de procedimientos y tareas</i>	56
Capítulo IV: Desarrollo e implementación en los Frameworks	59
<i>Arquitectura General en cada Framework</i>	59
<i>Funcionalidades de Reconocimiento de Señales</i>	63
<i>Integración de Información Contextual</i>	64
<i>Consideraciones de Seguridad y Privacidad</i>	65
Capítulo V: Evaluación y resultados	67
Implementación del modelo BRR	67
Pruebas evaluadas	68
<i>Evaluación de frameworks de acuerdo con los criterios establecidos.</i>	68
Análisis de resultados	124
<i>Resultados cuantitativos</i>	124
<i>Resultados cualitativos</i>	129
Resultados de la evaluación	136

Capítulo VI: Conclusiones y Futuras Mejoras.....	138
Cumplimiento de Objetivos.....	138
Conclusiones	139
Recomendaciones.....	141
Bibliografía	143
Anexos	148

ÍNDICE DE TABLAS

Tabla 1 <i>Requisitos de evaluación de frameworks</i>	44
Tabla 2 <i>Características de evaluación</i>	45
Tabla 3 <i>Asignación de importancia</i>	47
Tabla 4 <i>Categorías seleccionadas para la evaluación</i>	48
Tabla 5 <i>Definición de métricas de calidad</i>	48
Tabla 6 <i>Umbrales de calidad interna cuantitativa</i>	49
Tabla 7 <i>Umbrales de calidad externa cuantitativa</i>	51
Tabla 8 <i>Umbrales de calidad interna cualitativa</i>	52
Tabla 9 <i>Umbrales de calidad externa cualitativa</i>	52
Tabla 10 <i>Umbrales de calidad de uso cualitativa</i>	53
Tabla 11 <i>Arquitectura del caso práctico</i>	63
Tabla 12 <i>Frameworks propuestos para evaluación</i>	67
Tabla 13 <i>Evaluación de métrica</i>	69
Tabla 14 <i>Resultado de cohesión y acoplamiento en React Native</i>	73
Tabla 15 <i>Resultado de cohesión y acoplamiento en Flutter</i>	78
Tabla 16 <i>Resultado de cohesión y acoplamiento en Ionic</i>	80
Tabla 17 <i>Resultado de cohesión y acoplamiento en Xamarin</i>	85
Tabla 18 <i>Evaluación de métrica</i>	86
Tabla 19 <i>Tiempos de ejecución por tarea en Flutter</i>	86
Tabla 20 <i>Tiempos de ejecución por tarea en React Native</i>	87
Tabla 21 <i>Tiempos de ejecución por tarea en Ionic</i>	89
Tabla 22 <i>Tiempos de ejecución por tarea en Xamarin</i>	90
Tabla 23 <i>Porcentaje de CPU utilizado Flutter</i>	92
Tabla 24 <i>Porcentaje de CPU utilizado React Native</i>	93
Tabla 25 <i>Porcentaje de CPU utilizado Ionic</i>	94

Tabla 26 <i>Valor de memoria RAM utilizado Flutter</i>	96
Tabla 27 <i>Valor de memoria RAM utilizado React Native</i>	98
Tabla 28 <i>Valor de memoria RAM utilizado Ionic</i>	99
Tabla 29 <i>Valor total sobre extensión de código Flutter</i>	102
Tabla 30 <i>Valor total sobre extensión de código React Native</i>	103
Tabla 31 <i>Valor total sobre extensión de código Ionic</i>	104
Tabla 32 <i>Valor total sobre extensión de código Xamarin</i>	105
Tabla 33 <i>Resultados de evaluación ordinal de frameworks</i>	107
Tabla 34 <i>Resultados de evaluación ordinal frameworks</i>	108
Tabla 35 <i>Resultados de evaluación ordinal frameworks</i>	109
Tabla 36 <i>Resultados de evaluación ordinal frameworks</i>	110
Tabla 37 <i>Resultados de evaluación ordinal frameworks</i>	112
Tabla 38 <i>Resultados de evaluación ordinal frameworks</i>	113
Tabla 39 <i>Información respalda en base a la encuesta</i>	113
Tabla 40 <i>Resultados de evaluación ordinal frameworks</i>	115
Tabla 41 <i>Información respalda en base a la encuesta</i>	116
Tabla 42 <i>Resultados de evaluación ordinal frameworks</i>	117
Tabla 43 <i>Resultados de evaluación ordinal frameworks</i>	122
Tabla 44 <i>Información respalda en base a la encuesta</i>	123
Tabla 45 <i>Interpretación de resultados obtenidos en la evaluación</i>	124
Tabla 46 <i>Interpretación de resultados obtenidos en la evaluación</i>	129

ÍNDICE DE FIGURAS

Figura 1 <i>Familia ISO/IEC 25000</i>	24
Figura 2 <i>Ciclo de vida de la calidad del producto software</i>	26
Figura 3 <i>Arquitectura de Flutter</i>	59
Figura 4 <i>Arquitectura de React Native</i>	60
Figura 5 <i>Arquitectura de Xamarin</i>	61
Figura 6 <i>Arquitectura de Ionic</i>	62
Figura 7 <i>APIs implementadas para reconocimiento de señales</i>	68
Figura 8 <i>APIs implementadas para la obtención de información</i>	69
Figura 9 <i>Detección de señaléticas de tránsito Flutter</i>	91
Figura 10 <i>Presentación de la información Flutter</i>	91
Figura 11 <i>Descarga del archivo Flutter</i>	91
Figura 12 <i>Detección de señaléticas de tránsito React Native</i>	92
Figura 13 <i>Presentación de la información React Native</i>	93
Figura 14 <i>Descarga del archivo React Native</i>	93
Figura 15 <i>Detección de señal Ionic</i>	94
Figura 16 <i>Presentación de información Ionic</i>	94
Figura 17 <i>Descarga de archivo Ionic</i>	94
Figura 18 <i>Detección de señal Xamarin</i>	95
Figura 19 <i>Presentación de información Xamarin</i>	95
Figura 20 <i>Descarga de archivo Xamarin</i>	95
Figura 21 <i>Detección de señaléticas de tránsito Flutter</i>	96
Figura 22 <i>Presentación de la información Flutter</i>	96
Figura 23 <i>Descarga de archivo Flutter</i>	96
Figura 24 <i>Detección de señal React Native</i>	97
Figura 25 <i>Presentación de información React Native</i>	97

Figura 26 <i>Descarga de archivo React Native</i>	98
Figura 27 <i>Detección de señal Ionic</i>	98
Figura 28 <i>Presentación de información Ionic</i>	99
Figura 29 <i>Descarga de archivo Ionic</i>	99
Figura 30 <i>Detección de señal Xamarin</i>	100
Figura 31 <i>Presentación de información Xamarin</i>	100
Figura 32 <i>Descarga de archivo Xamarin</i>	101
Figura 33 <i>Valor de memoria RAM utilizado Xamarin</i>	101
Figura 34 <i>Código fuente evaluado</i>	102
Figura 35 <i>Código fuente evaluado</i>	103
Figura 36 <i>Código fuente evaluado</i>	105
Figura 37 <i>Código fuente evaluado</i>	106
Figura 38 <i>Resultado de encuesta</i>	107
Figura 39 <i>Resultado de encuesta</i>	108
Figura 40 <i>Resultado de encuesta</i>	109
Figura 41 <i>Resultado de encuesta</i>	110
Figura 42 <i>Comparación de Frameworks</i>	111
Figura 43 <i>Resultado de encuesta</i>	112
Figura 44 <i>Resultado de encuesta</i>	113
Figura 45 <i>Resultado de encuesta</i>	115
Figura 46 <i>Resultado de encuesta</i>	117
Figura 47 <i>Información respalda en base a la encuesta</i>	118
Figura 48 <i>Resultado de encuesta</i>	122

Resumen

Este trabajo de investigación explora la necesidad de una metodología sistemática clara para la evaluación y selección de entornos de desarrollo de aplicaciones móviles con realidad aumentada (AR), basándose en la norma ISO 25010 la cual forma parte de la familia ISO/IEC 25000. Los objetivos incluyen identificar y definir indicadores de calidad relevantes que cubran aspectos clave de la calidad interna, externa y de uso del software. La justificación de la evaluación de calidad se basa en el creciente número de aplicaciones móviles y la importancia de la elección del framework open source adecuado para el desarrollo y éxito del proyecto. En este caso, se considera opciones como Flutter, React Native, Ionic y Xamarin. Se proponen métricas de evaluación en base a la norma ISO 25010 para medir la calidad del software, teniendo en cuenta criterios de calidad interna, externa y de uso. Los resultados esperados incluyen una guía práctica y objetiva para tomar decisiones informadas al seleccionar el entorno más apropiado para sus proyectos, y también contribuir al avance del conocimiento en los campos de la realidad aumentada y la ingeniería de software proporcionando conocimientos y recomendaciones sobre las mejores prácticas para el desarrollo de aplicaciones de realidad aumentada.

Palabras clave: framework, aplicación móvil, norma ISO, calidad SW.

Abstract

This research work explores the need for a clear systematic methodology for the evaluation and selection of mobile application development environments with augmented reality (AR), based on the ISO 25010 standard which is part of the ISO/IEC 25000 family. Objectives include identifying and defining relevant quality indicators that cover key aspects of internal, external and usability quality of the software. The justification for the quality assessment is based on the growing number of mobile applications and the importance of choosing the appropriate open source framework for the development and success of the project. In this case, options such as Flutter, React Native, Ionic and Xamarin are considered. Evaluation metrics are proposed based on the ISO 25010 standard to measure the quality of the software, taking into account internal, external and use quality criteria. The expected results include a practical and objective guide to make informed decisions when selecting the most appropriate environment for your projects, and also contribute to the advancement of knowledge in the fields of augmented reality and software engineering by providing insights and recommendations on best practices. for the development of augmented reality applications. Likewise, case studies will be addressed that demonstrate the application of different development environments in specific situations, analyzing their performance and their ability to meet the project requirements.

Keywords: framework, mobile application, ISO standard, SW quality.

Capítulo I

Introducción

Antecedentes

En el mundo digital actual, las aplicaciones móviles de realidad aumentada (AR) se han convertido en poderosas herramientas que transforman muchas experiencias y servicios en nuestra vida diaria. Estas aplicaciones no solo brindan nuevas formas de interactuar con el mundo digital, sino que también aumentan la eficiencia, la seguridad y la competencia en diversas actividades.

La elección del entorno de desarrollo es fundamental para los desarrolladores de aplicaciones de RA porque cada entorno tiene características, capacidades y limitaciones únicas que afectan directamente la calidad y el rendimiento del producto final. Por lo tanto, es importante utilizar un enfoque sistemático y objetivo para evaluar y seleccionar la estructura más adecuada para cada aplicación específica.

Este artículo se centra en satisfacer esta necesidad específica proporcionando un enfoque riguroso y estructurado para evaluar y seleccionar un marco de desarrollo de aplicaciones AR. Para lograrlo, nos basamos en los estándares y principios establecidos por ISO 25000, que proporciona un marco integral para evaluar la calidad del software.

Utilizando el enfoque ISO 25000, nuestro objetivo es identificar y definir indicadores de calidad relevantes que aborden aspectos clave de la calidad interna del software, la calidad externa percibida por el usuario y la calidad de usabilidad real. Al integrar estas métricas en el proceso de evaluación del sistema, nuestro objetivo es brindar a los desarrolladores de aplicaciones AR una guía práctica y objetiva para tomar decisiones informadas al seleccionar el entorno más apropiado para sus proyectos.

Además, esta investigación tiene como objetivo contribuir al avance del conocimiento en los campos de la realidad aumentada y la ingeniería de software proporcionando conocimientos y recomendaciones sobre las mejores prácticas para el desarrollo de aplicaciones de RA. Por tanto, este análisis se centra en la evaluación y selección de entornos de desarrollo de aplicaciones de realidad aumentada para mejorar la calidad y seguridad de dichas aplicaciones utilizando un enfoque basado en ISO 25000.

Propósito y contexto del proyecto

El propósito de este proyecto es abordar la necesidad de crear un método claro y eficaz para evaluar y seleccionar marcos para desarrollar aplicaciones móviles de realidad aumentada (AR). En un contexto en el que la tecnología de realidad aumentada desempeña un papel cada vez más importante en la mejora de la seguridad vial, el proyecto tiene como objetivo proporcionar a los desarrolladores una guía fiable basada en la norma ISO 25000 para evaluar y comparar sistemas con criterios de calidad internos, externos y en uso. Para demostrar la aplicabilidad práctica del enfoque, se llevará a cabo un estudio de caso sobre el desarrollo de una aplicación de detección de señales de tráfico. Esta medida no solo tiene como objetivo proporcionar soluciones concretas para mejorar la seguridad vial, sino que también es un ejemplo concreto de cómo la implementación de métodos de evaluación de la calidad puede promover el desarrollo de aplicaciones de RA de alta calidad.

En el entorno actual, donde la seguridad vial es una prioridad innegable y la tecnología de realidad aumentada sigue evolucionando, el programa busca integrar eficazmente estos dos aspectos. Al proporcionar una base sólida para las opciones de diseño y centrarse en la calidad de las aplicaciones de RA, se espera que el programa haga una contribución significativa a mejorar la seguridad vial y la eficacia de las soluciones de RA en esta importante área.

Justificación de la evaluación

El comparar diferentes frameworks de desarrollo de aplicaciones móviles utilizando criterios internos y externos de calidad y usabilidad, con el objetivo de identificar aquellos frameworks que optimicen y aceleren el proceso de desarrollo. El crecimiento exponencial de las aplicaciones móviles está impulsado por ventajas como menores costos, mayor accesibilidad y flexibilidad en el proceso de desarrollo, lo que resalta la importancia de elegir el entorno de código abierto adecuado para el éxito del proyecto. Esto define características, funcionalidad, rendimiento, capacidad de aprendizaje y una comunidad de desarrolladores accesible.

El esfuerzo de investigación se centra en una evaluación detallada de los marcos de código abierto para el desarrollo de aplicaciones móviles. El marco propuesto será analizado en detalle, teniendo en cuenta sus características, desempeño, comunidad y facilidad de aprendizaje. Además, se propondrán indicadores de evaluación basados en la norma ISO 25000 para medir la calidad del software en base a estándares internos y externos de calidad y usabilidad.

Alcance

El alcance del proyecto se centra en la implementación y aplicación de un modelo de evaluación diseñado para analizar y evaluar entornos de desarrollo de aplicaciones móviles de realidad aumentada (AR). El modelo se desarrollará para identificar, definir y medir criterios de calidad y darles relevancia contextual en el proceso de evaluación.

El modelo de evaluación propuesto se basará en la norma ISO/IEC 25000, que trata de la evaluación de las cualidades internas, externas y de usabilidad del software. A través de este marco regulatorio se establecerán parámetros adecuados para evaluar el marco elegido.

Realizando una evaluación integral de los marcos seleccionados para ayudar a los desarrolladores a tomar decisiones informadas sobre la plataforma más adecuada para el desarrollo de aplicaciones móviles AR. Se espera que esta evaluación proporcione una guía confiable para el desarrollo de aplicaciones de RA de alta calidad, contribuyendo así de manera significativa al avance del conocimiento en el campo de la selección de estructuras.

Objetivos

Objetivo General

- Evaluar y comparar frameworks de desarrollo para aplicaciones móviles con realidad aumentada (RA) con el propósito de seleccionar el que mejor se adapte a las necesidades específicas de una aplicación móvil. El objetivo principal es identificar el framework que ofrezca el mejor equilibrio entre calidad interna, externa y de uso de acuerdo con los estándares de la norma ISO 25000.

Objetivos Específicos

- Asegurar que la aplicación móvil de detección de señales de tránsito cumpla con los criterios de calidad definidos por la norma ISO 25000, lo que implica alcanzar niveles óptimos de eficacia, eficiencia, mantenibilidad y usabilidad. Este objetivo busca garantizar que la aplicación entregue resultados precisos y confiables, sea fácil de utilizar y pueda mantenerse y mejorarse con eficiencia en el futuro.
- Establecer un marco de referencia para la evaluación de calidad en el desarrollo de aplicaciones de RA en el contexto de la seguridad vial, lo que puede servir como guía para futuros proyectos y contribuir al avance en esta área de investigación y desarrollo tecnológico.

Capítulo II

Marco Teórico

Norma ISO 25000

ISO/IEC 25000, conocida como SQuaRE (System and Software Quality Requirements and Evaluation), es una familia de normas que tiene por objetivo la creación de un marco de trabajo común para evaluar la calidad del producto software. Siendo como resultado de la evolución de otras normas anteriores, especialmente de las normas ISO/IEC 9126, que describe las particularidades de un modelo de calidad del producto software, e ISO/IEC 14598, que abordaba el proceso de evaluación de productos software (*Norma ISO/IEC 25000 | Tecnología Investigación y Academia, 2017*).

La ISO/IEC 25000 está compuesta por cinco divisiones las cuales se enfocan en la calidad del producto software:

Figura 1

Familia ISO/IEC 25000



Nota. Tomado de (*NORMAS ISO 25000*, s. f.).

División de gestión de calidad – ISO/IEC 2500n. – Se define modelos, términos y definiciones comunes referenciados por la familia 25000 formada por:

ISO/IEC 25000 (Guide to SQuaRE). - Engloba los modelos de arquitectura, terminologías, así como guías de usuario mediante documentos SQuaRE.

ISO/IEC 25001 (Planning and Management). - Establece orientaciones y requisitos para la gestión tanto en la evolución como especificación de los mismos.

División de modelo de calidad – ISO/IEC 2501n.- Se define modelos que incluyen características detalladas sobre la calidad interna, externa y de uso.

División de medición de calidad – ISO/IEC 2502n.- Se define un modelo de referencia sobre la medición de calidad interna, externa y de uso del producto software, mediante la aplicación práctica de métricas de calidad y definiciones:

ISO/IEC 25020 (Measurement reference model and guide). - Proporciona orientación inicial y modelos de referencia para seleccionar o desarrollar medidas propuestas.

ISO/IEC 25021 (Quality measure elements). - Define un conjunto de métricas básicas y derivadas que se pueden utilizar durante todo el ciclo de vida del software.

ISO/IEC 25022 (Measurement of quality in use). - Define indicadores para medir la calidad en uso.

ISO/IEC 25023 (Measurement of system and software product quality). - Define métricas para medir la calidad de productos y sistemas.

ISO/IEC 25024 (Measurement of data quality). - Define indicadores de calidad de los datos de medición.

División de requisitos de calidad – ISO/IEC 2503n. - Especifica los requisitos de calidad utilizados en el proceso de identificación de requisitos de desarrollo.

División de evaluación de calidad – ISO/IEC 2504n. - Se define modelos, términos y definiciones comunes referenciados por la familia 25000 formada por:

ISO/IEC 25040 (Evaluation reference model and guide). - Propone un modelo que considera los insumos y las limitaciones del proceso de desarrollo de recursos para producir resultados apropiados.

ISO/IEC 25041 (Evaluation guide for developers, acquirers and independent evaluators). - Define requisitos y recomendaciones para la evaluación práctica de productos desde la perspectiva de desarrolladores y evaluadores independientes. Modelo de Calidad ISO/IEC 25010

La ISO/IEC 25010 maneja la calidad del producto software en tres fases (Vaca & Jácome, 2018a):

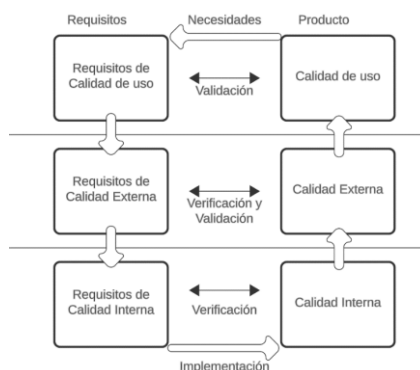
Calidad Interna: Se refiere al “desarrollo” del producto software.

Calidad Externa: Se refiere al “funcionamiento” del producto software.

Calidad en Uso: Se refiere al “uso” del producto software.

Figura 2

Ciclo de vida de la calidad del producto software



Nota. Tomado de (Vaca & Jácome, 2018a).

Realidad aumentada y señales de tránsito

Realidad aumentada

En palabras de (2012) las ideas de sobre Realidad Aumentada no son nuevas, tiene sus inicios a mediados del siglo XX, en 1962 Morton Heilig crea un simulador de moto llamado Sensorama con imágenes, sonido, vibración y olfato que ofrecía recrear el mundo real en un sistema mecánico. Cuatro años después Ivan Sutherland inventa un display de cabeza (HMD) que sugiere una ventana a un mundo virtual, lo que supuso un gran avance de la idea final de la que hoy hablamos. En 1972 surgió un avance importante cuando Myron Krueger crea Videoplace, un sistema que permite a los usuarios interactuar con objetos virtuales por primera vez. Veinte años más tarde, 1992, cuando Steven Feiner, Blair MacIntyre y Doree Seligmann diseñan el primer prototipo importante de un sistema de Realidad Aumentada, KARMA. A partir de este momento el desarrollo de la tecnología crece muy rápidamente ofreciendo en el 2000 ARQuake, el primer juego al aire libre con dispositivos móviles de Realidad Aumentada, lo que supuso el empujón final para el mundo comercial y la llegada de inversores que más tarde permitiría en el 2008 la salida al mercado de AR Wikitude Guía, una aplicación Android que permitía al usuario gracias a la cámara enfocar una imagen y obtener información en tiempo real sobre el lugar de interés (p. 1) .

La historia sobre la evolución de la Realidad Aumentada revela como esta tecnología ha pasado de ser una idea innovadora a convertirse en una herramienta cotidiana con un impacto significativo en nuestra percepción del mundo real.

La realidad aumentada permite potenciar los sentidos con los cuales se percibe la realidad, esto se logra a través de la información que existe en el mundo digital sobre las cosas que están alrededor de las personas, así que la realidad aumentada actúa como un lente con el cual se ve el mundo (Iván Mauricio, Melo Bohórquez, 2018).

Ofreciendo experiencias interactivas al usuario al combinar la dimensión virtual con el mundo físico participando en actividades más dinámicas de esta forma mejorando su comprensión y retención de conceptos al ser autónomos en el uso de esta tecnología.

“La Realidad Aumentada no es una tecnología que necesite de muchos requerimientos técnicos para ponerla en práctica, los requerimientos son mínimos ” (Reinoso 2013).

- Una cámara o webcam que capte la imagen del entorno.
- Software de Realidad Aumentada que permita superponer contenido digital sobre la escena real.
- Microprocesador con capacidad de procesamiento para modificar la señal de vídeo que se entrega a la pantalla.
- Un monitor o pantalla donde visualiza la imagen real tomada por la cámara combinada en tiempo real con el contenido digital.

Según su método de operación la RA se clasifica en diferentes niveles como son:

- **Hiperenlaces en el mundo físico - Nivel 0.-** Se considera como RA inicial o simple enlazando el mundo real con el virtual, un claro ejemplo son los códigos QR que se enlazan con sitios web.
- **Realidad aumentada basada en marcadores - Nivel 1.-** Basados en marcadores de referencia los cuales se usan como punto de referencia o de medida, puesto que ocupan un patrón único que permite que se diferencien unos de otros.
- **Realidad aumentada markerless - Nivel 2.-** Basado en reconocimiento de imágenes, objetos y localizaciones, en los últimos años se ha venido desarrollado aplicaciones móviles con RA para captar imágenes del entorno y superponer información sobre puntos de interés del usuario.

- **Visión aumentada - Nivel 3.-** El propósito es visualizar información si utilizar las manos impartiendo ordenes de voz hacia gafas inteligente.

Definición y características de Realidad Aumentada

“Podemos definir *la Realidad Aumentada* (RA) como una técnica para mostrar información extra sobre el mundo real” (Muñoz-Saavedra et al., 2020) está integrando de manera progresiva en el ámbito profesional, emergiendo como una tecnología con gran potencial futuro. Sin embargo, RA no sustituye el mundo real por uno virtual; en cambio, mantiene al individuo inmerso en el entorno físico y lo enriquece al complementarlo con información virtual.

Según Azuma, un sistema de Realidad Aumentada se caracteriza por lo siguiente:

- Combina elementos reales y virtuales.
- Ofrece interactividad en tiempo real.
- Realiza un registro en tres dimensiones (3D).

Señales de tránsito

Las señales de tránsito desempeñan un papel esencial en la regulación y control del tráfico, actuando como una herramienta de información a través de palabras o símbolos que transmiten información importante, advertencias o instrucciones a todos los peatones y conductores en la vía.

Las señales de tránsito vigentes existen en gran cantidad, en la normativa ecuatoriana son: manuales, luminosas, acústicas, verticales y horizontales. Las señales preventivas se utilizan para ayudar al movimiento seguro y ordenado (Cain Guambo, 2021). Se menciona que las señales de tránsito transmiten un mensaje mediante su diferente forma, color o símbolo tanto para peatones y conductores.

Señales informativas: Las señales guían a los transeúntes mediante la información de lugares, distancia de ciudades, nombres de avenidas, etc. En colores representativos como el verde para el fondo y blanco para brindar la información.

Señales regulatorias: La señal se establece geoméricamente rectangular, verticalmente utilizando color como blanco para el fondo, negro para la información y rojo para la restricción.

Señales preventivas: De acuerdo a la normativa nacional esta señal se establece en forma de rombo en color amarillo y negro, se instalan al lado derecho de la calzada sin obstruir su visibilidad.

Señales de información vial: El propósito de las señales es orientar y guiar a los usuarios a llegar a su destino, siendo representados por el color azul, verde o naranja.

Importancia de las señales de tránsito en Realidad Aumentada

La Realidad Aumentada (RA) mejora notoriamente la visualización de objetos del mundo real para una mejor interacción del usuario (Contreras et al., 2019). En entornos de tráfico, esta tecnología puede desempeñar un papel decisivo al llevar información visualmente rica sobre las señales de tráfico directamente al campo de visión del conductor o del peatón. Un desafío fundamental en este contexto es detectar e identificar rápidamente los diferentes tipos de señales de tránsito presentes en el entorno del usuario. Esto implica un procesamiento flexible de la información visual, especialmente en entornos complejos y cambiantes como las carreteras urbanas.

El uso de la realidad aumentada en la interpretación de las señales de tráfico puede mejorar la conciencia situacional de conductores y peatones, contribuyendo a mejorar la seguridad vial. Al proporcionar información relevante en tiempo real, la RA puede ayudar a reducir el riesgo de accidentes y permitir una conducción más segura y eficiente. Además, la

integración de la RA con las señales de tráfico permite personalizar la información en función de las necesidades específicas del usuario, como adaptarla a diferentes idiomas o habilitar alertas personalizadas.

Calidad en aplicaciones de RA

Relevancia de la evaluación de calidad en RA

En el contexto de la realidad aumentada (RA), la evaluación de la calidad se ha convertido en un tema de gran interés y preocupación. A pesar de los importantes esfuerzos para estandarizar la medición de la calidad de la comunicación audiovisual, la RA todavía tiene características que requieren métodos de evaluación especiales. Los enfoques tradicionales, que a menudo se centran en modalidades únicas como audio y video, a menudo no son adecuados para los sistemas AR porque dependen de la interacción activa del usuario y brindan experiencias visuales y perceptuales únicas.

Jordi Puig, Andrew Perquis, Frank Lindseth y Touraj Ebrahimi abordan esta cuestión señalando los desafíos especiales que plantea la RA en la evaluación de la calidad. A pesar de los avances en otras áreas, como las comunicaciones audiovisuales, todavía no hay consenso sobre cómo evaluar mejor las diversas cualidades visuales y perceptuales de la RA. Esta brecha resalta la necesidad general de desarrollar métodos efectivos y específicos para evaluar la calidad de las experiencias en entornos de Realidad Aumentada (Puig et al., 2012).

La relevancia de la evaluación de la calidad de la RA radica en su potencial para garantizar que las aplicaciones y sistemas desarrollados cumplan con los estándares de calidad esperados, ayudando así a proporcionar una experiencia de usuario satisfactoria y eficaz. Una evaluación rigurosa de la calidad en AR no solo puede mejorar la usabilidad y la usabilidad, sino que también puede tener un impacto positivo en áreas importantes como la

seguridad y la eficacia en aplicaciones que van desde la industria hasta la medicina y la educación.

Norma ISO 25000 y su aplicabilidad a RA

El constante avance de las tecnologías de realidad aumentada ha generado la aparición de diversas aplicaciones y herramientas de desarrollo que se adaptan a las cambiantes necesidades de los usuarios finales. La elección de una herramienta entre las numerosas opciones disponibles se ha convertido en un desafío que implica tiempo y prueba y error.

En el trabajo de Fausto A. Salazar Fierro, Carpio A. Pineda Manosalvas, Nancy N. Cervantes Rodríguez y Pablo Landeta, se aborda la importancia de evaluar la eficiencia de desempeño en aplicaciones de realidad aumentada (RA). La investigación se centra en el uso de la norma ISO/IEC/25010 para analizar dos aplicaciones desarrolladas con herramientas distintas, Vuforia y Wikitude, con el objetivo de identificar la herramienta que ofrece un rendimiento superior en la ejecución de aplicaciones de RA. La eficiencia de desempeño se evalúa considerando aspectos temporales y el uso de recursos, siendo crucial para la selección de la herramienta más adecuada en la creación de productos de realidad aumentada (Salazar et al., 2020).

Analizar las características de calidad del software definidas por la ISO/IEC 25010, como la eficiencia en el desempeño, la usabilidad, la seguridad, entre otras, son relevantes y se aplican a las aplicaciones de realidad aumentada. En el ámbito de la RA, donde la interacción en tiempo real y la experiencia del usuario son fundamentales, comprender cómo la normativa ISO/IEC 25010 se adapta y se utiliza para evaluar la calidad de las aplicaciones puede ser crucial para el desarrollo y la selección efectiva de herramientas y plataformas en este campo tecnológico en constante evolución (Vaca & Jácome, 2018b).

Framework de Desarrollo de Aplicaciones Móviles

Descripción del frameworks desde cero

Flutter

Flutter es un marco de desarrollo de código abierto creado por Google para la creación de aplicaciones móviles nativas y de alta calidad para múltiples plataformas, algunas ventajas de Flutter es el lenguaje de programación Dart que tiene rendimiento sólido, rendimiento que es excepcional debido a su arquitectura de compilación, flexibilidad en widgets personalizables dando flexibilidad en el diseño de la interfaz, soporte de plataformas siendo compatibles con iOS y Android (Smartup, 2023).

Flutter, un framework de Google, te permite crear apps móviles para Android e iOS a la vez. Con un solo código y lenguaje (Dart), desarrollas apps con rendimiento nativo, interfaz personalizada y rápida actualización de cambios. Ideal para apps de negocios, ecommerce, redes sociales o juegos, esta tecnología en auge te ofrece calidad y eficiencia en el desarrollo móvil multiplataforma. framework multiplataforma que te permite crear apps para Android e iOS con un solo código. Ofrece un rendimiento nativo, interfaces personalizadas y desarrollo rápido con Hot Reload. Su lenguaje Dart es moderno y fácil de aprender. Cuenta con una comunidad activa, widgets personalizables, arquitectura reactiva, testing robusto y una amplia gama de plugins. Además, soporta Firebase y es de código abierto. En resumen, Flutter te permite crear apps móviles de alta calidad de forma eficiente y versátil.

Ventajas: Flutter te permite crear apps multiplataforma con rendimiento nativo, interfaces personalizadas y desarrollo rápido. Su lenguaje Dart es moderno y fácil de aprender. Cuenta con una comunidad activa y una amplia gama de recursos.

Desventajas: Es un framework relativamente nuevo, con un tamaño de aplicación potentially mayor que las aplicaciones nativas. Requiere aprender Dart y presenta algunas limitaciones en cuanto a plugins y soluciones (Rambhia et al., 2023).

React Native

React incluye muchos bloques de código JavaScript que se pueden reutilizar para crear elementos de la interfaz de usuario (UI) llamados widgets. Esta biblioteca integra JavaScript en el código HTML y es fácil de usar porque está basada en JavaScript. Además, admite la reutilización de componentes de JavaScript, lo que simplifica la creación de componentes de JavaScript para garantizar un rendimiento óptimo y admite estrategias de optimización de motores de búsqueda para mejorar la eficiencia (A, 2020).

Xamarin

La plataforma Xamarin es reconocida por su enfoque de código abierto, destinado a la creación de aplicaciones altamente eficientes y modernas para iOS, Android y Windows mediante el uso de .NET. Esta herramienta proporciona una capa de abstracción que facilita la comunicación entre el código compartido y el código específico de cada plataforma, simplificando así el proceso de desarrollo y garantizando un rendimiento óptimo en diferentes entornos móviles.

Permite a los desarrolladores compartir un promedio del 90 % de la aplicación entre plataformas. Permite crear una interfaz de usuario nativa en cada plataforma y escribir lógica de negocios en C# que se comparte entre plataformas (profexorgeek, 2023).

Este framework para el desarrollo de aplicaciones móviles multiplataforma gracias a sus diversas características. Con la capacidad de compartir hasta un 90% del código entre plataformas, permite la implementación de aplicaciones nativas para Android, iOS y Windows Phone. Además, garantiza un alto rendimiento, equiparable al de las aplicaciones nativas, y proporciona acceso completo a las APIs nativas de cada plataforma. Al ser un framework de código abierto y gratuito, respaldado por una comunidad activa, ofrece una opción atractiva para los desarrolladores. Su integración con los IDEs oficiales, Visual Studio y Visual Studio para Mac, proporciona un entorno de desarrollo familiar, facilitando la transición para aquellos

que ya están acostumbrados a estas herramientas. Además, Xamarin ofrece una extensa variedad de plugins que permiten la integración de diversas funcionalidades, como geolocalización, cámara, sensores y redes sociales, ampliando significativamente las posibilidades de desarrollo (Vishal & Kushwaha, 2018).

Ionic

Como menciona (Yang et al., 2017) es un marco de desarrollo de código abierto basado en tecnologías HTML5, JavaScript, CSS y otras tecnologías para crear una experiencia nativa de aplicación móvil, crea aplicaciones a través de SASS “Syntactically awesome Stylesheets” siendo su objetivo el simplificar así como mejorar la forma en que se escribe y mantiene el código CSS, utilizando el marco JavaScript MVVM y Angular JS .

Ionic un modelo MVVM (Modelo-Vista-VistaModelo) ofrece enlace de datos bidireccional simplificando la manipulación de datos complejos. Es altamente personalizable y ofrece rendimiento en todas las plataformas móviles, desarrolla páginas front-end móviles con herramientas como WebStorm, lo que facilita la creación rápida de aplicaciones empresariales.

Consideraciones de diseño y desarrollo

El proceso de diseño y desarrollo de aplicaciones móviles de realidad aumentada (RA) implica muchas cuestiones clave que van desde el diseño de la interfaz de usuario hasta el desarrollo funcional. Estos son los puntos clave a considerar durante el proceso:

- **Interfaz de usuario intuitiva.** - El diseño de la interfaz de usuario debe considerar las necesidades y expectativas de los usuarios finales para brindar una experiencia de usuario intuitiva y placentera. La facilidad de uso y la accesibilidad deben ser una prioridad, minimizando la complejidad y proporcionando una navegación clara y consistente (Winkler et al., 2007).

- **Comentarios visuales.** - Comentarios claros. Se deben utilizar elementos visuales como animaciones, transiciones y efectos visuales para proporcionar comentarios claros sobre las acciones del usuario y los cambios en el entorno de RA. Se deben utilizar indicadores visuales para informar a los usuarios sobre el estado de la aplicación, como la carga de contenido o la disponibilidad de funciones (Cheng, 2023).
- **Sentido suficiente de la realidad.** - Integre con el entorno real. Los elementos virtuales deben diseñarse para integrarse de forma natural y coherente con el entorno físico, manteniendo el equilibrio adecuado entre realismo y funcionalidad. Se debe mantener la coherencia visual entre los elementos virtuales para garantizar una experiencia de RA coherente y atractiva (Krause, 2005).
- **Gestión del espacio.** - Es necesario considerar cómo los elementos virtuales ocupan e interactúan con el espacio físico del usuario para evitar interferencias con objetos reales y asegurar una experiencia inmersiva. Los elementos virtuales deben escalarse y posicionarse correctamente en relación con el entorno físico utilizando tecnologías como el reconocimiento de superficies y la detección de profundidad (Lang & Zhuang, 2023).
- **Desarrollo de funciones de realidad aumentada.** - El rendimiento de la aplicación debe optimizarse para garantizar una experiencia de RA fluida e ininterrumpida, minimizar los tiempos de carga y maximizar las velocidades de renderizado. Se deben utilizar sensores de dispositivos como cámaras, acelerómetros y giroscopios para mejorar la precisión y estabilidad del rendimiento de AR. Adicional, se debe implementar una gestión de datos eficaz

para cargar y procesar contenido AR, minimizar el uso de datos y maximizar la velocidad de acceso a recursos externos (Ma et al., 2022).

Capítulo III

Definición de metodología y métrica para la evaluación de calidad

Métricas de calidad interna, externa y de uso

Calidad interna

La calidad interna del software se refiere a una evaluación de su estructura interna y la forma en que está diseñada y desarrollada. Esta medición de la calidad se centra en aspectos técnicos que pueden no ser directamente visibles para el usuario final, pero que son necesarios para garantizar un producto de software confiable, mantenible y eficiente a largo plazo.

Funcionalidad

- **Integrarse con otras tecnologías.** - Esta métrica evalúa la capacidad del software para interactuar de manera efectiva con otras tecnologías, sistemas o API externas. Una buena integración garantiza la interoperabilidad y la compatibilidad con diferentes entornos, lo que hace que el software sea fácil de expandir y ampliar
- **Modular.** - Modularidad se refiere a la capacidad del software de estar compuesto por módulos independientes y bien definidos. El diseño modular hace que el software sea más fácil de entender, mantener y ampliar porque cada módulo se puede desarrollar, probar y actualizar de forma independiente sin afectar al resto del sistema.
- **Acceder a los permisos del dispositivo.** - Esta métrica se centra en cómo el software gestiona y accede a los permisos de los dispositivos, como cámaras, micrófonos, ubicaciones, etc. Es muy importante que el software solicite y utilice permisos de forma correcta y segura, respete las políticas de seguridad y privacidad establecidas por la plataforma y proteja la información del usuario.

- **Complejidad de la implementación funcional.** - evalúa si el software implementa correctamente todas las funciones y capacidades especificadas en los requisitos del sistema y del usuario. Esta métrica garantiza que el software cumpla con las expectativas y necesidades del usuario final y evita la falta de funciones o errores de implementación (Gordieiev & Kharchenko, 2020).

Rendimiento

- **Tiempo de finalización de la operación.** - se refiere al tiempo que tarda el software en completar operaciones clave o ciertos procesos. La ejecución rápida es fundamental para proporcionar una experiencia de usuario fluida y receptiva, especialmente en aplicaciones interactivas o en tiempo real.
- **El uso de recursos.** - Esta métrica evalúa cómo el software utiliza los recursos del sistema, como la CPU, la memoria y la red. La utilización eficiente de los recursos garantiza un rendimiento óptimo y evita problemas de congestión o consumo excesivo de recursos que pueden afectar negativamente la experiencia del usuario y la estabilidad del sistema.

Estas métricas son necesarias para evaluar la calidad interna del software y garantizar que esté bien estructurado, funcione correctamente y funcione de manera óptima. Al prestar atención a estos aspectos durante el proceso de desarrollo, el equipo de desarrollo puede identificar áreas de mejora y tomar medidas proactivas para optimizar la calidad interna del software (Jiménez & Juárez-Ramírez, 2019).

Calidad Externa

La calidad del software externo se centra en la experiencia del usuario final y en cómo el software interactúa con el entorno externo. Estas métricas se centran en aspectos visibles y tangibles para los usuarios, así como en la aceptación y adopción del software en el mercado.

Fácil de usar:

- **Experiencia del usuario final.** - esta métrica mide la satisfacción general del usuario al interactuar con el software. Se basa en aspectos como la facilidad de uso, la eficiencia en la realización de tareas, la comprensibilidad de la interfaz y la capacidad de respuesta del sistema. Una buena experiencia del usuario final es fundamental para la adopción y adopción de software.
- **Requisitos previos de instalación.** - evalúe los requisitos técnicos y de hardware necesarios para instalar y ejecutar el software. Cuantos menos requisitos previos tenga un software, más fácil será que más usuarios lo utilicen.
- **Tiempo de instalación y configuración básica.** - Mide el tiempo necesario para instalar y configurar el software para su uso inicial. La instalación y configuración rápidas y sencillas mejoran la experiencia del usuario y reducen las barreras de entrada.
- **Fácil de aprender.** - Esta métrica evalúa qué tan fácil es para los usuarios aprender a utilizar el software sin una formación exhaustiva. El software con una curva de aprendizaje baja es más atractivo para los usuarios y más fácil de implementar (Jiménez & Juárez-Ramírez, 2019).

Adopción:

- **Popularidad.** - La popularidad de un software significa que es ampliamente aceptado y utilizado en el mercado. Se puede medir a través de métricas como la cantidad de descargas, la cantidad de usuarios activos, reseñas y calificaciones de los usuarios e informes de los medios. Una mayor popularidad suele significar una mayor aceptación y satisfacción del usuario.

Calidad de uso

La usabilidad se refiere a cómo los usuarios interactúan con el software después de ejecutarlo. La atención se centra en el soporte, la formación y la documentación necesarios para garantizar que los usuarios puedan utilizar el software de forma eficaz y eficiente.

Apoyo:

- **Soporte.** - Esta métrica mide la calidad y disponibilidad del soporte técnico brindado a los usuarios. Incluye aspectos como el tiempo de respuesta, la amabilidad y eficiencia del personal de soporte, y la capacidad de resolver problemas de manera rápida y eficiente.
- **Capacitación.** - Mide la efectividad del programa de capacitación brindado a los usuarios para aprender a utilizar el software. Evaluar la comprensibilidad de los materiales de formación, la utilidad de las actividades prácticas y la capacidad de adquirir las habilidades necesarias para utilizar el software.

Documentación:

- **Documentación de referencia.** - La documentación de referencia verifica la exactitud e integridad de la documentación técnica que acompaña al software, desde manuales de usuario hasta documentos de referencia, especificaciones técnicas y otros materiales diseñados para facilitar la comprensión de los usuarios y el uso eficaz del software.
- **Guía de ayuda.** - Mide la utilidad y eficacia de los tutoriales y tutoriales que ayudan a los usuarios a realizar tareas específicas con el software. Las guías de ayuda bien diseñadas y fáciles de entender pueden mejorar enormemente la experiencia del usuario y acelerar el aprendizaje.

Adaptación de Métricas de la Norma ISO 25000 al método BBR

La selección de un framework de desarrollo adecuado es crucial en el proyecto el software a desarrollarse, diversos autores resaltan la importancia de evaluación de frameworks antes de su implementación, mediante un modelo de evaluación estandarizado que permita medir atributos de calidad facilitando la comparación objetiva entre una selección de frameworks, maximizando la productividad y calidad del desarrollo software.

Como menciona (BRR, 2005)," el modelo de evaluación debe incluir requisitos puntuales, que sea completo, simple, adaptable y consistente".

El modelo o metodología propuesto para la evaluación de software debe ser adaptable y fácilmente aplicable para agilizar el proceso de adopción de software. Una propuesta de modelo fue desarrollada conjuntamente por SpikeSource "empresa emergente fundado en 2003 por el ex presidente de Oracle Ray Lane y mutugan Pal dedicado a realizar prueba,certifica e integra componentes de codigo abierto en pila LAMP (Linux – Apache – MySQL-Perl/Python/PHP)" (Goth, 2005) , The Center for Open Source Investigation at Carnegie Mellon West e Intel Corporation siendo una industria para el desempeño e innovación de procesadores especializados en tecnología ,mejorar la educación y ayudar a satisfacer necesidades de la comunidad . El modelo Bussines Readines Rating (BRR) anunciado a principios de 2005 presenta metodología que busca evaluar y agrupar métricas en base a documentos de evaluación estándar como la ISO/IEC 9126 e ISO/IEC 25010, separando el proceso de evaluación en cuatro fases las cuales son:

1. **Evaluación rápida.** – La cual consiste en identificar la lista de frameworks a ser evaluados consecuentemente medir cada componente según criterios de calidad que se proponga.

2. **Evaluación según el objetivo de uso.** – El objetivo es realizar la ponderación de categorías, en este caso el modelo define 12 categorías a evaluar seguidamente clasificarlas según la importancia y orientación funcional del software a 7 o menos. Como segundo paso se debe asignar un valor de importancia a cada métrica de las categorías seleccionadas para la evaluación.
3. **Recolección de datos y el procesamiento.** – Se debe realizar la recopilación de datos para cada métrica utilizada en la evaluación.
4. **Traducción de datos.** – Los datos deben ser interpretados en base a la evaluación realizada.

Desarrollo de modelo de evaluación BRR aplicado a la ISO

Evaluación rápida.

Actualmente el internet se ha constituido como fuente de información principal para buscar información referente a temas específicos, en este caso se ha encontrado una lista amplia de frameworks los cuales poseen diferentes características y servicios propios.

En base a lo mencionado, es conveniente usar como referencia de evolución de frameworks a G2 (Business Software Reviews) que es un software grande y confiable, utilizado para tomar decisiones de software más inteligentes basadas en revisiones auténticas y experiencias personales (G2, 2023). La lista de requisitos propuesta para la evaluación se basa en las reseñas y calificación según la información brindado por los usuarios.

Los requisitos que se propone para la evaluación de los frameworks según G2 son:

Tabla 1*Requisitos de evaluación de frameworks*

Requisitos de evaluación
Facilidad de uso
Facilidad de configuración
Integración de desarrollo móvil
Calidad de soporte
¿Existen ejemplos de aplicaciones exitosas desarrolladas con este framework?
¿Qué tan personalizable (adaptable) es el framework?
Funcionalidad: Integración
Facilidad de administración
Cumple con los requisitos
Dirección del producto (% positivo)

Nota. Requisitos de evaluación de frameworks. Recuperado de (ANGIE MARICELA C. Is Comparing Ionic, React Native, ComponentOne Studio for Xamarin, and Flutter with Android Studio, s. f.).

Consecuentemente, cada requisito propuesto es evaluado en los frameworks seleccionados para su evaluación. Empleando una escala de medición tipo ordinal, la cual ordena entidades o valores de acuerdo con el criterio de ordenamiento. En este caso, se utiliza una escala de uno al cinco, donde uno señala un bajo ajuste y cinco con ajuste fuerte, haciendo referencia al framework evaluado.

Correspondencia de métricas a los atributos de calidad ISO

Evaluación según el objetivo de uso.

Atributos considerados en la evaluación de calidad de frameworks se contextualizarán mediante preguntas que proporcionen una idea clara de cada atributo, facilitando la toma de decisiones en cuanto a la asignación de importancia.

Tabla 2

Características de evaluación

Características de evaluación	Descripción
<i>Calidad interna</i>	
<i>Funcionalidad</i>	¿El framework facilita agregar funciones de realidad aumentada?
<i>Arquitectura</i>	¿El framework es fácil de integrar con otras tecnologías?
<i>Seguridad</i>	¿El framework protege los datos y privacidad de usuarios?
<i>Rendimiento</i>	¿El framework proporciona un rendimiento óptimo en aplicaciones con realidad aumentada?
<i>Calidad externa</i>	
<i>Usabilidad</i>	¿El framework es fácil de usar para el desarrollo de aplicaciones con realidad aumentada?
<i>Calidad de código</i>	¿El código generado por el framework es claro y está adecuadamente documentado?

Características de evaluación	Descripción
<i>Adopción</i>	¿El framework es ampliamente utilizado y aceptado por la comunidad y la industria?
<i>Comunidad</i>	¿La comunidad en base al framework es colaborativa y activa?
<i>Calidad de uso</i>	
<i>Escalabilidad</i>	¿El framework puede adaptarse a diferentes niveles de demanda y tamaño de aplicación, manteniendo un rendimiento óptimo sin importar la carga de trabajo?
<i>Apoyo</i>	¿El framework ofrece buen soporte técnico y actualizaciones regulares?
<i>Documentación</i>	¿La documentación proporcionada del framework es completa y útil para los desarrolladores?
<i>Profesionalismo</i>	¿La organización del framework muestra un alto nivel de compromiso, integridad, mantenimiento y calidad en desarrollo?

Nota. Descripción de categorías de evaluación.

Ponderación de categorías en base a su importancia

De acuerdo con (Chandra et al., 2011), las características de calidad, que pretenden ser aplicadas a cualquier producto software deben ser específicas, puesto que no tienen la misma importancia o prioridad para cada tipo de software. Por lo que, al especificar el modelo de evaluación de calidad, implica elegir características y sub características de calidad según la importancia específica del producto software.

Considerando lo expuesto, las doce categorías sugeridas por el modelo BRR se clasificarán conforme a su importancia, determinada mediante la evaluación del framework estableciendo seis categorías de evaluación.

Tabla 3

Asignación de importancia

Características de evaluación	Clasificación
Calidad interna	
<i>Funcionalidad</i>	1
<i>Arquitectura</i>	12
<i>Seguridad</i>	8
<i>Rendimiento</i>	5
Calidad externa	
<i>Usabilidad</i>	2
<i>Calidad de código</i>	9
<i>Adopción</i>	7
<i>Comunidad</i>	4
Calidad de uso	
<i>Escalabilidad</i>	11
<i>Apoyo</i>	6
<i>Documentación</i>	3
<i>Profesionalismo</i>	10

Nota. Asignación de importancia a las características de evaluación.

Tabla 4*Categorías seleccionadas para la evaluación*

Características de evaluación
Funcionalidad
Usabilidad
Documentación
Rendimiento
Apoyo
Adopción

Nota. Características a evaluar.***Definición de criterios y umbrales de calidad*****Tabla 5***Definición de métricas de calidad*

Características de evaluación	Métricas
Calidad interna	
<i>Funcionalidad</i>	Integración con otras tecnologías
	Modularidad
	Acceso a permisos del dispositivo
	Compleitud de la implementación funcional
<i>Rendimiento</i>	Extensión de código
	Tiempo de ejecución de operaciones
	Utilización de recursos
Calidad externa	
<i>Usabilidad</i>	Experiencia de usuario final

	Requisitos previos de instalación
	Tiempo de Instalación y configuración básica
	Facilidad de aprendizaje
Adopción	Popularidad
Calidad de uso	
Apoyo	Soporte
	Capacitación
Documentación	Documentación de referencia
	Tutorial de referencia

Nota. Definición de métricas de calidad.

Tabla 6

Umbrales de calidad interna cuantitativa

Características	Descripción	Métricas	Propósito	Fórmula	Interpretación del valor
Funcionalidad	¿El framework facilita agregar funciones de realidad aumentada?	Integración con otras tecnologías.	¿Qué tan completa es la integración del sistema con otras tecnologías a través de APIs?	$X = \frac{A}{Ta}$ <p>A = Apis integradas. Ta = Total, de Apis requeridas.</p>	$X \geq 1$ Entre más cercano a 1 mejor.
		Modularidad	¿Qué tan cohesivos y desacoplados están los módulos del sistema?	$X = \frac{(Cm + Am)}{2}$ <p>Cm =Cohesión de módulos Am =Acoplamiento de módulos.</p>	$X \geq 1$ Entre más cercano a 1 es mejor indica alta cohesión y bajo acoplamiento.

	Acceso a los recursos del dispositivo.	¿Qué porcentaje de los recursos del dispositivo utiliza la aplicación?	$X = \frac{R}{Tr}$ <p>R = Recursos utilizados. Tr = Total, de recursos del dispositivo.</p>	$X < 1$ Un valor menor a 1 indica que la aplicación utiliza de manera eficiente los recursos.
	¿El framework proporciona un rendimiento optimo en aplicaciones con realidad aumentada?	Tiempo de ejecución de operaciones.	¿Cuánto se tarda para completar una tarea?	$X = Tt$ Tt = Tiempo de tarea. $X \leq 0$ Entre más corto mejor.
		Utilización de recursos.	¿Qué porcentaje de CPU es usado para realizar una tarea dada?	$X = A$ A = Porcentaje de uso. $1\% \leq X \leq 50\%$ Entre más cercano a 1% es mejor.
			¿Cuánto espacio de memoria RAM ocupa en una tarea dada?	$X = A$ A = Cantidad de memoria RAM. $512 \leq X \leq 5120$ Entre más cercano a 512 es mejor.
Rendimiento	Extensión de código.	Índice de modificación del código.	$X = \frac{Lc}{T}$ <p>Lc = Líneas de código nuevas o modificadas. T = Total, de líneas de código original.</p>	$0 < X < 1$ Un X cercano a 0 sugiere estabilidad en el código, mientras que un valor cercano a 1 indica

una alta actividad
de desarrollo.

Nota. Valores cuantitativos de la calidad interna.

Tabla 7

Umbral de calidad externa cuantitativa

Características	Descripción	Métricas	Propósito	Fórmula	Interpretación del valor
Usabilidad	¿El framework es fácil de usar para el desarrollo de aplicaciones con realidad aumentada?	Experiencia de usuario final	¿Qué tan atractiva es la interfaz?	$X = n$ $n = \text{número de respuestas.}$	Comparación de respuestas.
		Tiempo de Instalación y configuración básica	¿Cuánto tiempo se es requerido para la instalación y configuración?	$X = Tt$ $Tt = \text{Tiempo de tarea.}$	Entre más corto mejor.
Adopción	¿El framework es ampliamente utilizado y aceptado por la comunidad y la industria?	Popularidad	¿Cuál es el nivel de popularidad entre los usuarios respecto a las características del framework?	$X = n$ $n = \text{número de respuestas.}$	Comparar valores previos.

Nota. Valores cualitativos de la calidad externa.

Tabla 8

Umbrales de calidad interna cualitativa

Características	Descripción	Métricas	Propósito	Puntaje				
				1 Inaceptable	2 Pobre	3 Aceptable	4 Muy bueno	5 Excelente
Funcionalidad	¿El framework facilita agregar funciones de realidad aumentada?	Complejidad de la implementación funcional.	¿El framework proporciona todas las características y funcionalidades necesarias para desarrollar una aplicación de realidad aumentada?	No está disponible.	Es limitado e inconsistente.	Disponible para elementos de código relevantes.	Disponible para funciones con un tiempo de respuesta rápido.	Disponible en todos los elementos de código con una respuesta instantánea.

Nota. Asignación cualitativa a la calidad interna.

Tabla 9

Umbrales de calidad externa cualitativa

Características	Descripción	Métricas	Propósito	Puntaje				
				1 Inaceptable	2 Pobre	3 Aceptable	4 Muy bueno	5 Excelente
Usabilidad	¿El framework es fácil de usar para el desarrollo de	Requisitos previos de instalación.	¿Cuán compleja es la instalación?	Instalación compleja requiriendo varios pasos, tiempo y experiencia técnica.	-	Relativamente sencillo aun con pasos y configuraciones técnicas.	-	Instalación sencilla, en poco tiempo incluso para usuarios

Características	Descripción	Métricas	Propósito	Puntaje				
				1 Inaceptable	2 Pobre	3 Aceptable	4 Muy bueno	5 Excelente
	aplicaciones con realidad aumentada?	Facilidad de aprendizaje.	¿Cuánto tiempo le tomo al usuario aprender a desarrollar en el framework?	Requiere tiempo extenso aproximadamente seis meses o más.		Requiere un tiempo moderado aproximadamente entre uno a tres meses.		sin experiencia. Experiencia básica en el desarrollo en un tiempo aproximado de una a dos semanas.

Nota. Asignación cualitativa a la calidad externa.

Tabla 10

Umbral de calidad de uso cualitativa

Características	Descripción	Métricas	Propósito	Puntaje				
				1 Inaceptable	2 Pobre	3 Aceptable	4 Muy bueno	5 Excelente
Apoyo	¿El framework ofrece buen soporte técnico y actualización?	Soporte	¿Cuánto tiempo tarda el equipo de soporte en responder a las consultas?	No responde a las consultas.	-	Responde en un tiempo razonable, aunque existe variabilidad	-	Responde de manera rápida y confiable.

Características	Descripción	Métricas	Propósito	Puntaje				
				1 Inaceptable	2 Pobre	3 Aceptable	4 Muy bueno	5 Excelente
	nes regulares?		de los usuarios?			en el tiempo de respuesta.		
			¿Las respuestas proporciona das por el equipo de soporte son claras, útiles y satisfacen las necesidade s de los usuarios?	Respuesta s confusas, incoherente s o poco claras.		Mayoría de respuestas útiles y claras.		Respuesta s claras y útiles satisfacien do a los usuarios de forma efectiva.
		Capacitaci ón	¿Hay eventos de capacitación o meetups organizadas por la comunidad de usuarios que utilizan el framework?	Ninguna disponibilid ad de capacitacio nes o meetups.		Disponibilid ad de capacitacio nes o meetups frecuentes con variedad de temas.		Capacitacio nes o meetups disponibles frecuentem ente presentado por expertos.
Documentación	¿La documentación proporciona da del	Documentación de referencia	¿Existe documentación completa y fácilmente	Documentación incompleta, confusa, difícil de		Documentación suficiente abordando		Documentación exhaustiva clara y

Características	Descripción	Métricas	Propósito	Puntaje				
				1 Inaceptable	2 Pobre	3 Aceptable	4 Muy bueno	5 Excelente
	framework es completa y útil para los desarrolladores?	Tutorial de referencia	¿La documentación incluye ejemplos claros y guías paso a paso para ayudar a los desarrolladores?	La documentación no incluye tutoriales de referencia.	Tutoriales de referencia de baja calidad difícil de entender.	Los tutoriales están disponibles y cubren características principales del framework.	Tutoriales de referencia amplios y bien elaborados, proporcionando ejemplos prácticos.	Tutoriales de referencia excepcionales en calidad, cobertura y utilidad.

Nota. Asignación cualitativa a la calidad de uso.

Diseño de la evaluación

Selección de escenarios de evaluación

Con el uso del dispositivo móvil Samsung A50 con las siguientes características tales como específicamente el modelo SM-A505G (a50ub), CPU exynos9610, resolución de pantalla de 1080 x 2340 pixeles, RAM de 4 GB y un almacenamiento de 64 GB con el objeto de prueba para los siguientes escenarios utilizando las historias de usuario del caso práctico a evaluar. A continuación, se detallará cada uno de ellos:

1. Inicie la aplicación e inicie el reconocimiento de señales de tráfico en tiempo real:

- **Escenario:** el usuario abre una aplicación en Samsung A50.
- **Resultados esperados:** la aplicación se inicia sin problemas y muestra una interfaz clara compatible con la resolución de pantalla del Samsung A50. Los

usuarios pueden activar fácilmente la función de reconocimiento de señales de tráfico en tiempo real utilizando los controles táctiles del dispositivo.

2. Acceda a páginas de detalles de señales por categoría:

- **Escenario:** el usuario navega por la aplicación para encontrar detalles sobre las señales de tráfico en un Samsung A50.
- **Resultados esperados:** La aplicación presenta una interfaz de usuario optimizada para la pantalla Samsung A50, con un diseño responsivo y fácil de usar. Los usuarios encontrarán una página organizada por categoría de señales de tráfico, donde los detalles de cada categoría se presentan de forma clara y fácil de entender en el Samsung A50.

3. Descargue las normas de tráfico para obtener más información:

- **Escenario:** el usuario desea acceder a las reglas de tráfico a través de la aplicación en Samsung A50 incluso sin conexión a Internet.
- **Resultados esperados:** Las reglas de tráfico se descargan correctamente en el dispositivo Samsung A50 y se pueden ver en la aplicación incluso sin conexión a Internet. Descargar y ver las reglas de tráfico en el Samsung A50 es rápido y eficiente gracias a las capacidades de almacenamiento y procesamiento de datos del dispositivo.

Definición de procedimientos y tareas

1. Calificaciones de evaluadores humanos:

Los evaluadores (que pueden incluir desarrolladores de software) realizarán una revisión exhaustiva del código fuente de la plataforma. Esta revisión se centrará en los aspectos importantes que estén contemplados dentro de la evaluación.

Utilice herramientas de análisis de código estático, como linters y analizadores de código estándar de la industria, para identificar posibles problemas de estilo, convenciones de codificación incumplidas y vulnerabilidades de seguridad.

Los indicadores definidos en la Sección de Métricas se utilizarán para asignar una puntuación cuantitativa y cualitativa a cada aspecto de la evaluación. Estos indicadores proporcionarán una base objetiva para la evaluación y permitirán una comparación precisa con los estándares de calidad establecidos.

2. Evaluación del depurador nativo del Framework:

Se utilizarán depuradores nativos y herramientas de análisis de código dinámico para realizar pruebas integrales del tiempo de ejecución en la plataforma. Estas pruebas se centrarán en identificar uso de recursos, pérdidas de memoria y problemas de rendimiento.

El depurador permitirá a los evaluadores seguir el progreso del código paso a paso, examinar el estado de las variables en diferentes etapas de ejecución y detectar posibles cuellos de botella.

Los resultados de las pruebas realizadas por el depurador se registran cuidadosamente, incluidos los seguimientos de ejecución, los registros de errores y cualquier excepción detectada. Estos registros servirán como evidencia objetiva de la calidad del código y guiarán el proceso de mejora continua.

Debemos especificar que depuradores

3. Pruebas y encuestas de aplicaciones móviles:

Se desarrollarán casos de prueba integrales que cubran todas las funciones de la aplicación móvil, incluidas las historias de usuarios mencionadas en la de Escenarios de

Evaluación. Estos casos de prueba se ejecutarán en múltiples dispositivos y entornos para garantizar la compatibilidad y confiabilidad.

Los usuarios que representan el público objetivo de la aplicación participarán en pruebas de usabilidad estructuradas. Estas pruebas incluirán casos de uso reales en los que se pedirá a los usuarios que realicen determinadas tareas dentro de la aplicación y proporcionen comentarios sobre su experiencia.

Las pruebas de usuario recopilarán datos cuantitativos y cualitativos, incluidos tiempos de respuesta, tasas de error, métricas de rendimiento y comentarios detallados sobre la experiencia del usuario. Estos datos se analizarán cuidadosamente para identificar áreas de mejora y priorizar futuras iteraciones de desarrollo.

Además, se realizarán encuestas y entrevistas estructuradas para recopilar comentarios más amplios sobre la solicitud. Estas encuestas cubrirán áreas como la percepción general de la calidad, la satisfacción del usuario, la facilidad de uso y el cumplimiento de las expectativas.

Capítulo IV

Desarrollo e implementación en los Frameworks

Arquitectura General en cada Framework

La arquitectura de los entornos de desarrollo de aplicaciones móviles de realidad aumentada (AR) es un aspecto clave que determina la estructura y organización de estos entornos de desarrollo. La arquitectura general del marco seleccionado para este estudio se describe a continuación:

Figura 3

Arquitectura de Flutter



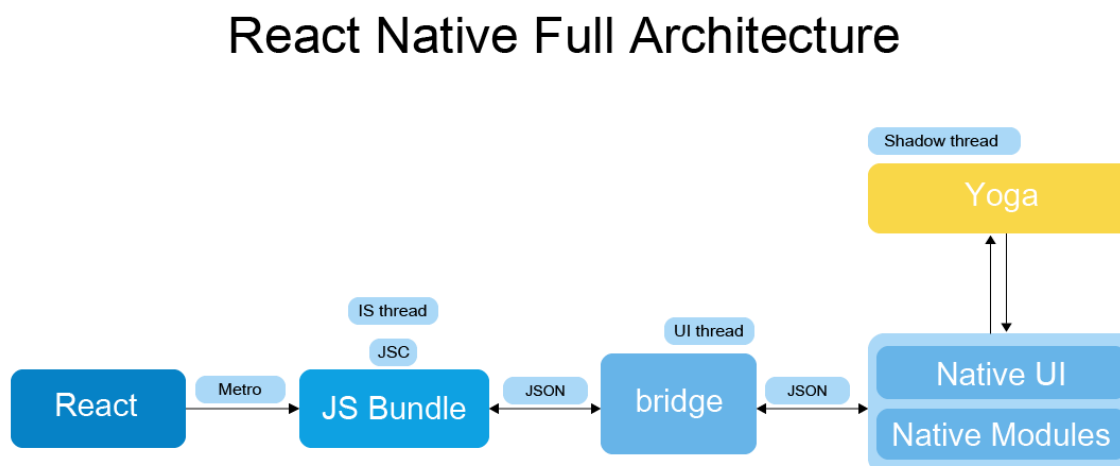
Nota. El gráfico representa la arquitectura de Flutter. Tomado de (Naik, 2022).

- **Componentes básicos.** - Flutter se basa en la arquitectura de widgets, donde cada elemento de la interfaz de usuario es un widget. Utiliza su propio motor de renderizado para crear una interfaz de usuario flexible y personalizable.
- **Capa de abstracción.** - Flutter proporciona una capa de abstracción que le permite crear aplicaciones nativas para múltiples plataformas (iOS y Android) a partir de una única biblioteca de código escrita en Dart (Szczepanik & Kedziora, 2020).

- **Bibliotecas y API.** - incluye un conjunto completo de bibliotecas y API para desarrollar fácilmente aplicaciones móviles con capacidades avanzadas de realidad aumentada, incluida la compatibilidad con gráficos 2D y 3D, interacciones táctiles, sensores de dispositivos y más.

Figura 4

Arquitectura de React Native



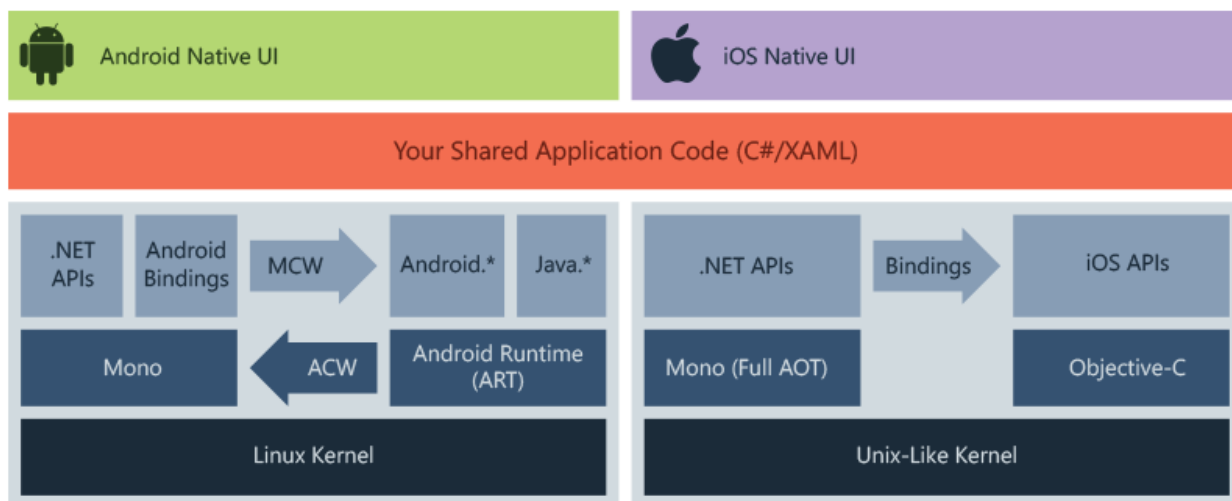
Nota. El gráfico representa la arquitectura de React Native. Tomado de (*About React Native – Welcome – React Native Course*, s. f.).

- **Componentes básicos.** -React Native utiliza una arquitectura basada en componentes, donde los elementos de la interfaz de usuario se definen como componentes reutilizables escritos en JavaScript.
- **Puente nativo.** - Contiene un "bridge" que proporciona comunicación entre el código JavaScript y los componentes nativos de iOS y Android, lo que permite el acceso nativo a la funcionalidad de realidad aumentada.
- **Bibliotecas y API.**- Proporciona un extenso ecosistema de bibliotecas y API para facilitar la integración de capacidades de realidad aumentada, como detección de

marcadores, superposiciones de objetos virtuales e interacción con el entorno físico (Dekkati et al., 2019).

Figura 5

Arquitectura de Xamarin



Nota. El gráfico representa la arquitectura de React Native. Tomado de (*Xamarin Apps with Business Central Integration – P*, s. f.).

- **Componentes básicos.** - Xamarin sigue un enfoque de desarrollo basado en .NET Framework, donde los desarrolladores pueden escribir código en C# y usarlo en múltiples plataformas.
- **Capa de abstracción.** - proporciona una capa de abstracción que facilita la interacción con las API nativas de iOS y Android y proporciona acceso unificado a la funcionalidad AR.
- **Bibliotecas y API.** - proporciona acceso a bibliotecas y API que facilitan la integración de capacidades de realidad aumentada en sus aplicaciones, incluido el seguimiento de objetos, la detección de superficies y las capacidades de representación de gráficos 3D (Vishal & Kushwaha, 2018).

Figura 6*Arquitectura de Ionic*

Nota. El gráfico representa la arquitectura de React Native. Tomado de (*Análisis de capacidades, diferencias, roadmap y futuras posibilidades de Ionic, React Native and NativeScript, s. f.*).

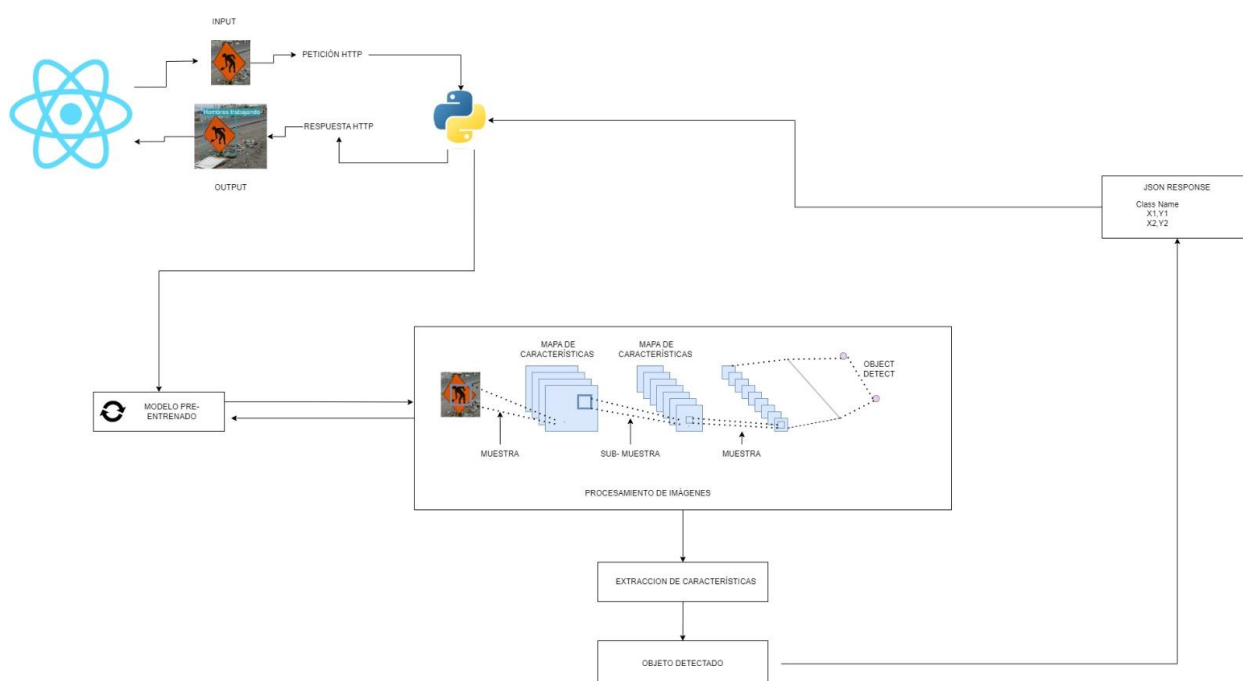
- **Componentes básicos.** - Ionic crea aplicaciones móviles multiplataforma basadas en tecnologías web estándar como HTML, CSS y JavaScript.
- **Capa de abstracción.** - Utilice una capa de abstracción para permitir que las aplicaciones web se empaqueten como aplicaciones móviles nativas para iOS y Android, proporcionando acceso a la funcionalidad de realidad aumentada a través de complementos y extensiones.
- **Bibliotecas y API.** -Proporciona una variedad de complementos y extensiones para integrar fácilmente funciones de AR como detección de marcadores, superposiciones de objetos 3D e interacción de gestos (Chaudhary, 2018).

Funcionalidades de Reconocimiento de Señales

Esta sección detalla las funciones relacionadas con el reconocimiento de señales de tráfico como parte del desarrollo de aplicaciones de realidad aumentada (RA). Estas características son necesarias para que la aplicación reconozca y procese correctamente las señales de tráfico en tiempo real y proporcione información relevante al usuario.

Tabla 11

Arquitectura del caso práctico



Nota. El gráfico presenta la arquitectura de la aplicación móvil, para el reconocimiento de señales de tránsito. Elaborado por Klever Mise.

- Procesamiento y reconocimiento de imágenes.** -La plataforma envía solicitudes HTTP a un backend desarrollado en Python y Flask y envía imágenes codificadas en base64. El backend procesa esta imagen utilizando un modelo previamente entrenado utilizando un conjunto de datos de señales de tráfico. El modelo utiliza técnicas avanzadas de procesamiento de imágenes y aprendizaje

automático para analizar imágenes y detectar la presencia de señales de tráfico en ellas.

- **Identificación y clasificación de signos.** -Una vez que se detectan señales de tráfico en una imagen, el modelo las identifica y clasifica en varias categorías, como señales de alto, señales de límite de velocidad, señales de dirección, etc. Esta clasificación se basa en los patrones y características específicos asociados con cada tipo de señal.
- **Produce una respuesta en formato JSON.** -El backend genera una respuesta con formato JSON que contiene información relevante sobre la señal de tráfico detectada. La respuesta contiene el nombre de la clase a la que pertenece cada señal, y las coordenadas (x1, y1) y (x2, y2) que definen el área de la señal en la imagen.

Integración de Información Contextual

Esta sección describe el proceso de integración de información contextual capturada por la cámara de un dispositivo móvil con elementos de realidad aumentada (AR) que aparecen en vivo en la pantalla del usuario.

Captura de información a través de la cámara. - Al abrir la aplicación, los usuarios pueden acceder al botón de la cámara para comenzar el proceso de captura de información. La aplicación utiliza la cámara del dispositivo para ver el entorno del usuario en tiempo real.

Información de procesamiento en segundo plano. - El backend recibe imágenes del entorno y utiliza visión por computadora y algoritmos de aprendizaje automático para analizar información contextual. Los métodos de procesamiento de imágenes se utilizan para identificar objetos en el campo de visión de la cámara e identificar señales de tráfico.

Generar elementos de realidad aumentada (AR). - Una vez que se reconocen e identifican los objetos y puntos de referencia, el backend genera los elementos de realidad aumentada (AR) correspondientes. Estos elementos pueden incluir modelos 3D de señales de tráfico detectadas, así como información adicional sobre las mismas, como su significado, reglas asociadas, distancias, etc.

Vea las señales de tráfico en la pantalla de su dispositivo. - Finalmente, en la pantalla del dispositivo móvil se muestran elementos de realidad aumentada (AR) correspondientes a las señales de tráfico detectadas. Estos elementos se superponen instantáneamente a la imagen de la cámara, permitiendo a los usuarios ver las señales identificadas y obtener información adicional sobre ellas de una forma intuitiva y práctica.

Consideraciones de Seguridad y Privacidad

Esta sección describe las medidas tomadas para proteger la seguridad y privacidad del usuario cuando utiliza aplicaciones de reconocimiento de señales de tráfico de realidad aumentada (AR).

Control de permisos. - La aplicación sólo requiere los permisos que necesita para funcionar, incluido el acceso a la cámara del dispositivo para detectar señales de tráfico en tiempo real. Además, descargar y guardar el Manual de Tránsito de Ecuador requiere permiso de almacenamiento, lo que garantiza que la aplicación solo acceda a la información requerida para brindar su funcionalidad básica.

Gestión de datos. - Todos los datos recopilados y procesados por la aplicación, incluidas las imágenes capturadas por las cámaras y cualquier información descargada de las guías de carreteras, se gestionan de forma segura y se almacenan cifrados en el dispositivo del usuario. Se implementan medidas de seguridad para proteger los datos del acceso no autorizado y posibles violaciones de seguridad.

Transferencia segura de datos. -Cualquier comunicación entre la aplicación y el backend se produce a través de una conexión segura utilizando protocolos de cifrado estándar como HTTPS. Esto garantiza que los datos transmitidos entre el dispositivo del usuario y los servidores de la aplicación estén protegidos contra la interceptación y manipulación por parte de terceros.

Actualizaciones y mantenimiento. - La aplicación se actualiza y mantiene periódicamente para eliminar posibles vulnerabilidades de seguridad y garantizar el cumplimiento de los últimos estándares de seguridad y privacidad. Implemente parches de seguridad con prontitud para abordar cualquier riesgo potencial que pueda surgir.

Proteger la privacidad del usuario. -Tomamos medidas para proteger la privacidad del usuario y evitar la recopilación y el uso indebido de datos personales. La aplicación no recopila información personal sin el consentimiento del usuario y garantiza que cualquier dato confidencial se trate de forma confidencial y de conformidad con las leyes de privacidad aplicables.

Capítulo V

Evaluación y resultados

Implementación del modelo BRR

Tabla 12

Frameworks propuestos para evaluación

		IONIC	REACT	XAMARIN	FLUTTER
<i>Facilidad de uso</i>	G2	8.9	8.4	7.8	9.2
	Propio	6.0	8.0	6.0	10.0
<i>Facilidad de configuración</i>	G2	8.5	8.6	-	9.7
	Propio	4.0	8.0	6.0	10.0
<i>Integración de desarrollo móvil</i>	G2	-	-	-	-
	Propio	6.0	10.0	6.0	10.0
<i>Calidad de soporte</i>	G2	8.2	8.3	-	-
	Propio	4.0	8.0	4.0	10.0
<i>¿Existen ejemplos de aplicaciones exitosas desarrolladas con este framework?</i>	G2	-	-	-	8.7
	Propio	8.0	10.0	2.0	8.0
<i>¿Qué tan personalizable (adaptable) es el framework?</i>	G2	-	-	-	-
	Propio	6.0	10.0	2.0	10.0
<i>Funcionalidad: Integración</i>	G2	9.3	-	-	-
	Propio	4.0	10.0	4.0	10.0
<i>Facilidad de administración</i>	G2	8.9	8.5	-	10.0
	Propio	6.0	8.0	4.0	8.7
<i>Cumple con los requisitos</i>	G2	8.6	8.7	8.3	9.0
	Propio	6.0	8.0	2.0	10.0
<i>Dirección del producto (% positivo)</i>	G2	9.0	8.4	7.7	9.7
	Propio	6.0	8.0	4.0	10.0
<i>Puntuación general</i>		5.4 / 10	8.6 / 10	4.9 / 10	9.9 / 10

Nota. Esta tabla muestra la evaluación de los frameworks en base a los requerimientos propuestos.

Pruebas evaluadas

Evaluación de frameworks de acuerdo con los criterios establecidos.

Funcionalidad

- **Integración con otras tecnologías**

Para medir la integración del sistema con otras tecnologías a través de APIs, se considera la implementación de dos APIs requeridas que en este caso son: el reconocimiento de señales de tránsito y la obtención de información relacionada con estas señales.

La Figura 7, describe la funcionalidad para el reconocimiento de las señaléticas enviadas por el front end, gestiona las peticiones y realiza la inferencia con el modelo devuelve en formato json el objeto detectado.

Figura 7

APIs implementadas para reconocimiento de señales

```

object_detector.py
object_detector.py > ...
1 from flask import request, Flask, jsonify
2 from PIL import Image
3 import base64
4 from io import BytesIO
5 from ultralytics import YOLO
6
7 app = Flask(__name__)
8
9
10 @app.route("/detect", methods=["POST"])
11 def detect():
12     try:
13         data = request.json
14         image_base64 = data.get("image")
15
16         if not image_base64:
17             return jsonify({"error": "Image data is missing"}), 400
18
19         image_data = base64.b64decode(image_base64)
20         image = Image.open(BytesIO(image_data))
21         image = image.convert('RGB')
22         image = image.resize((640, 480))
23         image = image.convert('RGB')
24
25         model = YOLO('yolov8n.pt')
26         results = model(image)
27         detections = results[0].boxes.data.tolist()
28         response = {
29             "detections": detections,
30             "speed": f"Speed: {results[0].speed}ms preprocess, {results[0].inference}ms inference, {results[0].postprocess}ms postprocess per image at shape {results[0].shape}"
31         }
32         return jsonify(response), 200
33     except Exception as e:
34         return jsonify({"error": str(e)}), 500
35
36 if __name__ == '__main__':
37     app.run(host='0.0.0.0', port=5000)

```

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
0: 640x480 (no detections), 147.1ms
Speed: 8.0ms preprocess, 147.1ms inference, 1.9ms postprocess per image at shape (1, 3, 640, 480)
10.52.84.98 - - [26/Feb/2024 11:44:55] "POST /detect HTTP/1.1" 200 -
0: 640x480 (no detections), 141.3ms
Speed: 5.0ms preprocess, 141.3ms inference, 1.1ms postprocess per image at shape (1, 3, 640, 480)
10.52.84.98 - - [26/Feb/2024 11:45:06] "POST /detect HTTP/1.1" 200 -

```

Nota. API de detección de señales de tránsito.

La Figura 8, describe la respuesta de la API de información, la cual describe el id de la señalética, el nombre, tipo de señal, Id de la clase, y el nombre en base a los manuales de tránsito vigentes proporcionados por la ANT.

Figura 8

APIs implementadas para la obtención de información

```

1 [
2   {
3     "id": "0DKj4fh4NaQyKkkRDvV8",
4     "descripcion": "Advierte a los conductores y peatones sobre la proximidad a áreas donde
5     puede ocurrir actividad volcánica y, por lo tanto, la posibilidad de flujos piroclásticos que se
6     desplazan rápidamente desde un volcán durante una erupción.",
7     "tipo_señal": "Señales de zona de amenazas",
8     "imagen":
9     "https://res.cloudinary.com/dsd4yqa4w/image/upload/v1701489585/samples/Se%C3%B1ales/piroplasticos_f
10    crajr.png",
11    "id_clase": "SGR1",
12    "nombre": "Zona de flujos piroclásticos"
13  },
14  {
15    "id": "0nJkTSfREEdZtcD7fNAS",
16    "descripcion": "Previene al conductor de la aproximación a un cruce de vías en la curva.",
17    "tipo_señal": "Serie de intersecciones y empalmes",
18    "imagen": "https://res.cloudinary.com/turismo-
19    mashca/image/upload/v1701389463/sw_tesis_ar/debadu9hwuyfd3ket0pj.png",
20    "id_clase": "P2",
21    "nombre": "Cruce de Vías en curvas"
22  },
23 ]

```

Nota. API de información de señaléticas.

Tabla 13

Evaluación de métrica

Métrica	Frameworks			
	Flutter	React	Ionic	Xamarin
Integración del sistema con otras tecnologías.	$\frac{2 \text{ APIs Utilizadas}}{2 \text{ APIs Totales}} = 1$	$\frac{2 \text{ APIs Utilizadas}}{2 \text{ APIs Totales}} = 1$	$\frac{2 \text{ APIs Utilizadas}}{2 \text{ APIs Totales}} = 1$	$\frac{2 \text{ APIs Utilizadas}}{2 \text{ APIs Totales}} = 1$

Nota. Evaluación sobre la métrica propuesta.

Modularidad

Análisis de código React Native

Pantalla Main

- **Cohesión.** – Hace referencia a que tan relacionados están los elementos dentro de un módulo. En este caso de evaluación, el componente Main tiene una cohesión alta, puesto que realiza una tarea principal: muestra una pantalla inicial con logo haciendo referencia el contexto del sistema y un botón para iniciar la aplicación. Todos los elementos dentro del componente están relacionados con esta tarea principal. Dado que el componente tiene una funcionalidad específica y todos los elementos están relacionados con dicha funcionalidad, por lo cual se asigna el valor de cohesión alta, $C_m=0.7$.
- **Acoplamiento.** - Hace referencia a la dependencia entre módulos y componentes. En el presente código, se puede observar algunas dependencias como: el componente Main que usa componentes y recursos de React Native. Los recursos como View, Image, Text, Button, StyleSheet, hook useNavigation etc. También depende del hook useNavigation de React Navigation para la navegación. Dado que el componente depende de varios módulos de React Native y React Navigation, se asigna el valor alto-moderado de acoplamiento, $A_m=0.8$.

Pantalla MainMenu

- **Cohesión.** - La relación de elementos dentro del módulo en el componente MainMenu indica una alta cohesión, puesto que todas las partes del componente están relacionadas con la presentación del menú principal y la gestión de la navegación entre las diferentes opciones del menú. Dado que el componente tiene una funcionalidad específica y todos los elementos están la funcionalidad, se asigna el valor alto de cohesión, $C_m=0.8$.

- **Acoplamiento.** - La dependencia entre módulos y componentes en este caso, se puede observar ciertas dependencias en el código como: El componente MainMenu usa componentes de React Native como View, Text, TouchableOpacity, Modal, Image, así como de React Navigation el componente useNavigation. De igual manera depende del componente Header, que es un componente externo. Dado que el componente depende de varios módulos de React Native y React Navigation, así como de un componente externo, se asigna el valor alto-moderado de acoplamiento, $A_m=0.7$.

Pantalla DeteccionSenialesScreen

- **Cohesión.** – En este caso el componente DetectionScreen realiza múltiples tareas como: inicialización de la cámara, procesamiento de fotogramas, gestión de eventos de interfaz de usuario, lógica para mostrar un modal, por lo cual se asigna en valor de cohesión moderado, $C_m=0.5$.
- **Acoplamiento.** – Como el componente usa varios módulos de React Native, depende de recursos externos como: el servidor de detección de objetos y los datos de objetos, teniendo dependencias internas siendo el caso del componente Camera, se asigna el valor alto-moderado de acoplamiento, $A_m=0.8$.

Pantalla TrafficSign

- **Cohesión.** – La cohesión en el componente TransitoPage es alto puesto que, todas las partes del componente están relacionadas con la presentación de señales de tránsito, así como la lógica asociada para cargar y mostrar las imágenes. Como el componente tiene una funcionalidad específica y todos los

elementos están relacionados con esta funcionalidad, se asigna el valor alto-moderado de acoplamiento, $C_m=0.6$.

- **Acoplamiento.** – El código presenta algunas dependencias en este caso: El componente `TransitoPage` usa de React Native componentes como `View`, `Text`, `Image`, `TouchableOpacity`, `ActivityIndicator`, `ScrollView`, `StyleSheet`, `Modal`, así como el hook `useNavigation` de React Navigation. También depende de `axios` para realizar solicitudes HTTP y `react-native-vector-icons` para los iconos. Como el componente depende de varios módulos de React Native, así como de módulos externos, se asigna el valor alto-moderado al acoplamiento, $A_m=0.7$.

Pantalla LawPage

- **Cohesión.** -El componente `LawPage` parece estar bien cohesionado ya que se encarga de mostrar información sobre las leyes de tránsito y proporcionar un botón para descargar el PDF de las leyes. Todos los elementos del componente están relacionados con esta funcionalidad específica. Dado que el componente tiene una funcionalidad claramente definida y todos los elementos están relacionados con esta funcionalidad, asignaremos un valor alto a la cohesión, $C_m = 0.8$.
- **Acoplamiento.** - El acoplamiento se refiere a la dependencia entre módulos y componentes. En este código, podemos ver algunas dependencias: El componente `LawPage` utiliza componentes de React Native como `View`, `Text`, `Image`, `TouchableOpacity`, `StyleSheet`, `Modal`, así como el icono de `Ionicons`. Además, depende del hook `useNavigation` de React Navigation. También depende de `Linking` para abrir el enlace al PDF de las leyes. Dado que el componente depende de varios módulos de React Native y un módulo externo (`Linking`), asignaremos un valor moderado-alto al acoplamiento, $A_m=0.7$.

Pantalla ImageDetailPage

- Cohesión.** - El componente ImageDetailPage se encarga de mostrar detalles de una imagen, incluido su nombre, imagen y descripción. Todos los elementos del componente están relacionados con esta funcionalidad específica. No hay funcionalidades adicionales o características que no estén directamente relacionadas con la visualización de detalles de la imagen. Dado que el componente tiene una funcionalidad claramente definida y todos los elementos están relacionados con esta funcionalidad, asignaremos un valor alto a la cohesión, $C_m = 0.9$.
- Acoplamiento.** -El acoplamiento se refiere a la dependencia entre módulos y componentes. En este código, podemos ver que: El componente ImageDetailPage depende de componentes de React Native como View, Text, Image, TouchableOpacity, ScrollView. Además, depende del icono de MaterialIcons y del hook useNavigation de React Navigation. Dado que el componente depende de varios módulos de React Native y un módulo externo (useNavigation), asignaremos un valor moderado-alto al acoplamiento, $A_m = 0.9$.

Tabla 14

Resultado de cohesión y acoplamiento en React Native

React Native				
	Cohesión	Acoplamiento	Aplicación Fórmula	Total
Main	0.7	0.8	$x = \frac{(0.7 + 0.8)}{2}$	0.75
MainMenu	0.8	0.7	$x = \frac{(0.8 + 0.7)}{2}$	0.75

React Native				
	Cohesión	Acoplamiento	Aplicación Fórmula	Total
<i>DeteccionSenial</i>	0.5	0.8	$x = \frac{(0.5 + 0.8)}{2}$	0.65
<i>esScreen</i>				
<i>TrafficSign</i>	0.6	0.7	$x = \frac{(0.6 + 0.8)}{2}$	0.7
<i>LawPage</i>	0.8	0.8	$x = \frac{(0.8 + 0.8)}{2}$	0.8
<i>ImageDatailPage</i>	0.9	0.9	$x = \frac{(0.9 + 0.9)}{2}$	0.9
Total				0.7583

Nota. Valores cuantitativos sobre cohesión y acoplamiento de código.

Análisis de código Flutter

Pantalla Home

- Cohesión de módulos (Cm).** -En este caso, la clase `_MyHomePageState` muestra alta cohesión, ya que maneja la lógica relacionada con la construcción de la página de inicio (`MyHomePage`), incluida la presentación de la imagen, el texto informativo y el botón de inicio.

Cohesión: El valor asignado es media ya que el widget contiene elementos relacionados con la presentación de información y un botón de inicio, que se pueden considerar cohesivos. $Cm = 0.6$.
- Acoplamiento de módulos (Am).** -En este caso, presenta un acoplamiento entre `_MyHomePageState` y `MenuPage`, ya que al presionar el botón de inicio se navega a la página del menú (`MenuPage`). Además, existe una dependencia con `MaterialPageRoute` y `BuildContext` para la navegación.

Acoplamiento: El valor asignado es bajo ya que no presenta dependencia fuertemente de otras clases fuera de las bibliotecas estándar de Flutter. $A_m = 0.3$.

Pantalla Menu

- **Cohesión de módulos (Cm).** -La cohesión se puede observar en la clase `_MenuPageState`, que maneja la lógica relacionada con la página de menú de la aplicación. Esta clase se encarga de mostrar los botones de las diferentes opciones de la aplicación y de manejar la navegación a las páginas correspondientes.

Cohesión: El valor asignado es medio. El widget contiene elementos relacionados con la presentación de opciones de menú y la interacción con la cámara y las leyes de tránsito. $C_m = 0.5$.

- **Acoplamiento de módulos (Am).** - El acoplamiento se puede observar en la dependencia de `_MenuPageState` con otras partes del código, como las importaciones de las clases `CameraApp`, `TransitoPage` y `LeyesPage`. Además, `_MenuPageState` depende de la clase `CameraController` para manejar la cámara en la aplicación.

Acoplamiento: El valor asignado es medio. Dependencia de la clase `CameraApp` y otras clases para la funcionalidad de la cámara y la presentación de señales de tránsito. $A_m = 0.5$.

Pantalla Camera

- **Cohesión de módulos (Cm).** -En este caso los principales módulos: `main()`: El cual inicializa la aplicación y configura la cámara. `CameraApp`: Widget principal que contiene la lógica de la aplicación de la cámara. `_CameraAppState`: Estado

del widget CameraApp que gestiona la lógica interna, como la inicialización de la cámara, el procesamiento de imágenes y la gestión de la interfaz de usuario.

Cohesión: El valor asignado es alto. El widget se enfoca en la funcionalidad de la cámara y la captura de imágenes. $C_m = 0.8$.

- **Acoplamiento de módulos (A_m):** El acoplamiento bajo indica que los módulos son independientes entre sí y pueden ser modificados sin afectar a otros módulos. En este caso el código presenta acoplamiento entre los módulos debido a la dependencia de la cámara y la comunicación con la API para el procesamiento de imágenes.

Acoplamiento: El valor asignado es medio, por la dependencia de la clase CameraController para controlar la cámara y de otras clases para la presentación de la interfaz de usuario y la navegación. $A_m = 0.5$.

Pantalla Transito

- **Cohesión de módulos (C_m).** - La cohesión se puede observar en la clase `_TransitoPageState`, que maneja la lógica relacionada con la página de señales de tránsito de la aplicación. Esta clase se encarga de mostrar las diferentes categorías de señales de tránsito y las imágenes correspondientes.

Cohesión: El valor asignado es alto. El widget se enfoca en presentar y organizar señales de tránsito por categoría. $C_m = 0.9$.

- **Acoplamiento de módulos (A_m).** -El acoplamiento se puede observar en la dependencia de `_TransitoPageState` con otras partes del código, como las importaciones de las clases `http`, `json`, `DescripcionPage`, y `CachedNetworkImage`. Además, `_TransitoPageState` depende de la clase `CameraController` para manejar la cámara en la aplicación.

Acoplamiento: El valor asignado es medio. Dependencia de la clase DescripcionPage para mostrar la descripción de una señal al hacer clic en ella.
Am= 0.5.

Pantalla Descripcion

- **Cohesión de módulos (Cm).** -La cohesión se puede observar en la clase DescripcionPage, que maneja la lógica relacionada con la página de descripción de una señal de tránsito. Esta clase se encarga de mostrar la información detallada de una señal específica, incluyendo su nombre, tipo, descripción e imagen.

Cohesión: El valor asignado es alto. El widget se enfoca en presentar la información de una señal de tránsito específica.

- **Acoplamiento de módulos (Am).** -El acoplamiento se puede observar en la dependencia de DescripcionPage con otras partes del código, como la importación de flutter/material.dart para widgets de Flutter y la utilización de la clase Navigator para la navegación entre páginas.

Acoplamiento: El valor asignado es bajo. No parece depender fuertemente de otras clases fuera de las bibliotecas estándar de Flutter. Am= 0.4.

Pantalla Leyes

- **Cohesión de módulos (Cm).** -La cohesión se puede observar en la clase LeyesPage, que maneja la lógica relacionada con la página de visualización y descarga de la Ley de Tránsito. Esta clase se encarga de mostrar el título, la imagen representativa y el botón de descarga de la ley.

Cohesión: El valor asignado es alto. El widget se enfoca en presentar información y permitir la descarga de la Ley de Tránsito. Cm= 0.8.

- **Acoplamiento de módulos (Am).** -El acoplamiento se puede observar en la dependencia de LeyesPage con otras partes del código, como la importación de flutter/material.dart para widgets de Flutter y url_launcher para abrir un enlace web para descargar la ley.

Acoplamiento: El valor asignado es bajo. La dependencia de url_launcher para la descarga es una dependencia externa estándar. $A_m=0.4$.

Tabla 15

Resultado de cohesión y acoplamiento en Flutter

Flutter				
	Cohesión	Acoplamiento	Aplicación Formula	Total
Home	0.5	0.3	$x = \frac{(0.6 + 0.3)}{2}$	0.45
Menú	0.5	0.3	$x = \frac{(0.5 - 0.5)}{2}$	0.5
Camera	0.8	0.5	$x = \frac{(0.8 + 0.5)}{2}$	0.65
Transito	0.9	0.5	$x = \frac{(0.9 + 0.5)}{2}$	0.7
Descripción	0.8	0.4	$x = \frac{(0.8 + 0.4)}{2}$	0.6
Leyes	0.8	0.4	$x = \frac{(0.8 + 0.4)}{2}$	0.6
Total				0.5

Nota. Valores cuantitativos sobre cohesión y acoplamiento de código.

Análisis de código Ionic

Pantalla Home:

- **Cohesión:** El módulo Home tiene una alta cohesión (0.8) ya que todas las partes están relacionadas con la presentación de la pantalla de inicio.

- **Acoplamiento:** Tiene un bajo acoplamiento (0.2), lo que indica una dependencia limitada de otros módulos.

Pantalla Page 1- Menu:

- **Cohesión:** Page 1- Menu también presenta una alta cohesión (0.8) al estar enfocado en la presentación del menú principal.
- **Acoplamiento:** Al igual que Home, tiene un bajo acoplamiento (0.2), indicando independencia de otros módulos.

Pantalla Page2 - Captura:

- **Cohesión:** Este módulo muestra una cohesión moderada (0.6) ya que realiza diversas tareas, como la inicialización de la cámara y la captura de imágenes.
- **Acoplamiento:** Presenta un acoplamiento moderado-alto (0.8) debido a su dependencia de varios módulos de Ionic y servicios externos.

Pantalla FlaskApiService:

- **Cohesión:** FlaskApiService tiene una alta cohesión (0.9) al centrarse en la comunicación con un servidor Flask para la detección de objetos.
- **Acoplamiento:** Tiene un acoplamiento moderado (0.4), ya que su dependencia se limita a la comunicación con el servidor.

Pantalla Page3 – Tipos de Señales:

- **Cohesión:** Page3 muestra una alta cohesión (0.8) al presentar información sobre tipos de señales de tránsito.
- **Acoplamiento:** Tiene un acoplamiento moderado-alto (0.7) al depender de módulos de Ionic y servicios externos para cargar datos.

Pantalla Image Detail:

- **Cohesión:** Image Detail tiene una cohesión moderada (0.5), ya que se enfoca en mostrar detalles de una imagen específica.
- **Acoplamiento:** Presenta un acoplamiento moderado (0.6) debido a su dependencia de módulos de Ionic y servicios externos.

Pantalla Page4 – Manual de Transito:

- **Cohesión:** Page4 muestra una alta cohesión (0.8) al proporcionar información sobre leyes de tránsito y permitir la descarga de un PDF.
- **Acoplamiento:** Presenta un acoplamiento moderado (0.7) al depender de módulos de Ionic y servicios externos para la presentación de datos y funcionalidad de descarga.

Tabla 16

Resultado de cohesión y acoplamiento en Ionic

Ionic				
	Cohesión	Acoplamiento	Aplicación Formula	Total
Home	0.8	0.2	$x = \frac{(0.8 + 0.2)}{2}$	0.5
Page 1- Menu	0.8	0.2	$x = \frac{(0.8 + 0.2)}{2}$	0.5
Page2 – Captura	0.6	0.8	$x = \frac{(0.6 + 0.8)}{2}$	0.7
FlaskApiService	0.9	0.4	$x = \frac{(0.9 + 0.4)}{2}$	0.65
Page3 – Tipos de Señales	0.8	0.7	$x = \frac{(0.8 + 0.7)}{2}$	0.75
Image Detail	0.5	0.6	$x = \frac{(0.5 + 0.6)}{2}$	0.55

Ionic				
	Cohesión	Acoplamiento	Aplicación Formula	Total
<i>Page4 – Manual de Transito</i>	0.8	0.7	$x = \frac{(0.8 + 0.7)}{2}$	0.75
Total				0.7333

Nota. Valores cuantitativos sobre cohesión y acoplamiento de código .

Análisis de código Xamarin

Pantalla Captura

- **Cohesión de módulos (Cm).** -Al analizar el código de observa que hay dos clases, Captura y VMCaptura, cada una con sus métodos y funcionalidades específicas.

La clase Captura parece estar relacionada con la interfaz de usuario y la lógica relacionada con la captura de imágenes, detección de objetos y comunicación con un servidor para obtener y mostrar información sobre objetos detectados.

La clase VMCaptura, por otro lado, parece estar relacionada con la lógica de vista-modelo (ViewModel) para la captura de imágenes y el manejo de la comunicación con un servidor para la detección de objetos.

Ambas clases tienen funcionalidades distintas y no comparten mucho código entre sí, lo que sugiere que están bastante cohesivas en términos de sus responsabilidades.

- **Acoplamiento de módulos (Am).** – Se observa algunas dependencias como: La clase Captura utiliza la clase HttpClient para comunicarse con un servidor para la detección de objetos.

La clase VMCaptura también utiliza la clase HttpClient para comunicarse con un servidor para enviar imágenes y recibir respuestas.

Ambas clases dependen de la clase HttpClient para realizar operaciones de red.

Sin embargo, no hay una dependencia directa entre las clases Captura y VMCaptura.

Pantalla DetalleSerial

- **Cohesión de módulos (Cm).** - En este caso, la página DetalleSerial.xaml y su clase VMDetalleSerial están estrechamente relacionadas, ya que la vista (ContentPage) muestra detalles de una señal y la clase ViewModel (VMDetalleSerial) proporciona los datos necesarios para la vista.
- **Acoplamiento de módulos (Am).** -En este caso, la clase VMDetalleSerial depende de la clase SerialModel para obtener los datos de la señal que se mostrarán en la vista. Además, la clase DetalleSerial.xaml.cs depende de la clase VMDetalleSerial para establecer su contexto de datos.

Dado que hay una dependencia directa entre la clase VMDetalleSerial y SerialModel, y entre DetalleSerial.xaml.cs y VMDetalleSerial, podemos considerar que hay un nivel de acoplamiento. Sin embargo, la dependencia es necesaria para el funcionamiento adecuado de la página y su lógica asociada.

Pantalla Inicio

- **Cohesión de módulos (Cm).** - En este caso, la página Inicio.xaml y su clase VMInicio están estrechamente relacionadas, ya que la vista muestra la interfaz de inicio de la aplicación y la clase ViewModel proporciona la lógica necesaria para manejar las interacciones del usuario en esta vista.

- **Acoplamiento de módulos (Am).** -En este caso, la clase VMInicio depende de la interfaz de navegación (INavigation) para poder navegar a otras páginas, como la página Opciones, utilizando el método PushAsync. Sin embargo, no hay otras dependencias significativas en esta página y su código detrás.
Dado que hay una dependencia directa entre la clase VMInicio y INavigation, podemos considerar que hay un nivel de acoplamiento. Sin embargo, la dependencia es necesaria para la navegación dentro de la aplicación y no hay otras dependencias directas significativas.

Pantalla Leyes

- **Cohesión de módulos (Cm).** -En este caso, la página Leyes.xaml y su clase VMLeys están estrechamente relacionadas, ya que la vista muestra información sobre la ley de tránsito y la clase ViewModel proporciona la lógica necesaria para manejar las interacciones del usuario en esta vista.
- **Acoplamiento de módulos (Am).** -En este caso, la clase VMLeys depende de la interfaz de navegación (INavigation) para poder navegar a otras páginas utilizando el método PopAsync. Además, la clase Leyes.xaml.cs depende de la clase VMLeys para establecer su contexto de datos. Sin embargo, estas dependencias son necesarias para el funcionamiento adecuado de la página y su lógica asociada.
Dado que hay una dependencia directa entre la clase VMLeys y INavigation, y entre Leyes.xaml.cs y VMLeys, podemos considerar que hay un nivel de acoplamiento. Sin embargo, las dependencias son necesarias y no hay otras dependencias directas significativas.

Pantalla Opciones

- **Cohesión de módulos (Cm).** - En este caso, la página Opciones.xaml y su clase VM OPCIONES están estrechamente relacionadas, ya que la vista muestra las opciones disponibles para el usuario y la clase ViewModel proporciona la lógica necesaria para manejar las interacciones del usuario en esta vista.
- **Acoplamiento de módulos (Am).** - En este caso, la clase VM OPCIONES depende de la interfaz de navegación (INavigation) para poder navegar a otras páginas utilizando los métodos PushAsync y PopAsync. Además, la clase Opciones.xaml.cs depende de la clase VM OPCIONES para establecer su contexto de datos. Sin embargo, estas dependencias son necesarias para el funcionamiento adecuado de la página y su lógica asociada.
Dado que hay una dependencia directa entre la clase VM OPCIONES y INavigation, y entre Opciones.xaml.cs y VM OPCIONES, podemos considerar que hay un nivel de acoplamiento. Sin embargo, las dependencias son necesarias y no hay otras dependencias directas significativas.

Pantalla Tipos

- **Cohesión de módulos (Cm).** - En este caso, la página Tipos.xaml y su clase VM Tipos están estrechamente relacionadas. La vista muestra una colección de señales de tráfico agrupadas por tipo, y la clase ViewModel proporciona la lógica necesaria para cargar las señales desde una API, manejar la selección de una señal y mostrar los detalles de la misma.
- **Acoplamiento de módulos (Am).** - En este caso, la clase VM Tipos depende de la interfaz de navegación (INavigation) para poder navegar a la página de detalles cuando el usuario selecciona una señal. Además, la clase Tipos.xaml.cs depende de la clase VM Tipos para establecer su contexto de datos. Sin

embargo, estas dependencias son necesarias para el funcionamiento adecuado de la página y su lógica asociada.

Dado que hay una dependencia directa entre la clase VMTipos y INavigation, y entre Tipos.xaml.cs y VMTipos, podemos considerar que hay un nivel de acoplamiento. Sin embargo, las dependencias son necesarias y no hay otras dependencias directas significativas.

Tabla 17

Resultado de cohesión y acoplamiento en Xamarin

Xamarin				
	Cohesión	Acoplamiento	Aplicación Formula	Total
Captura	0.9	0.7	$x = \frac{(0.9 + 0.7)}{2}$	0.8
DetalleSenial	0.9	0.8	$x = \frac{(0.9 + 0.9)}{2}$	0.85
Inicio	0.9	0.7	$x = \frac{(0.9 + 0.7)}{2}$	0.8
Leyes	0.9	0.8	$x = \frac{(0.9 + 0.8)}{2}$	0.85
Opciones	0.9	0.8	$x = \frac{(0.9 + 0.8)}{2}$	0.85
Tipos	0.9	0.8	$x = \frac{(0.9 + 0.8)}{2}$	0.85
Total				0.8333

Nota. Valores cuantitativos sobre cohesión y acoplamiento de código .

- **Acceso a los recursos del dispositivo**

En el sistema propuesto, se consideran los permisos a integrar en una aplicación Android, estos permisos se dividen en nueve categorías: sensores, calendario, cámara, contactos, ubicación, micrófono, teléfono, SMS y almacenamiento.

Tabla 18*Evaluación de métrica*

Métrica	Frameworks			
	Flutter	React Native	Ionic	Xamarin
Recursos del dispositivo utilizados para el sistema.	$\frac{2}{9} = 0.222$	$\frac{2}{9} = 0.222$	$\frac{2}{9} = 0.222$	$\frac{3}{9} = 0.333$

Nota. Comparación de resultados de cohesión y acoplamiento sobre los frameworks.

Rendimiento

- **Tiempo de ejecución de operaciones**

Para la medición del tiempo requerido por los frameworks es realizar las tareas designadas como: reconocimiento, muestra de información y descarga de manual sobre las señales de tránsito, consecuentemente registrar los resultados obtenidos, luego realizar la sumatoria y obtener un resultado total de cada una de las tareas.

En la siguiente las tablas siguientes se presentan la evaluación de tiempos de cada frameworks mencionados anteriormente:

Tabla 19*Tiempos de ejecución por tarea en Flutter*

Tiempo de respuesta con el framework Flutter					
N.º Prueba	Tarea 1 Resultado	Tarea 2 Resultado	Tarea 3 Resultado	Resultado Total	Resultado total / Nº tareas
Prueba 1	5,9	0,2487	3,412	9,5607 seg	3,1869 seg
Prueba 2	4,99	0,2327	4,322	9,5447 seg	3,1815 seg

Tiempo de respuesta con el framework Flutter					
N.º Prueba	Tarea 1	Tarea 2	Tarea 3	Resultado Total	Resultado total / Nº tareas
	Resultado	Resultado	Resultado		
Prueba 3	4,69	0,2852	6,457	6,457 seg	2,1523 seg
Prueba 4	5,07	0,2188	7,564	12,8528 seg	4,2842 seg
Prueba 5	5,24	0,234	5,342	10,816 seg	3,6053 seg
Prueba 6	4,47	0,3126	8,122	12,9046 seg	4,3015 seg
Prueba 7	4,69	0,2154	6,432	11,3374 seg	3,7791 seg
Prueba 8	3,87	0,2096	7,241	11,3206 seg	3,7735 seg
Prueba 9	5,95	0,2062	3,994	10,1502 seg	3,3834 seg
Prueba 10	5,92	0,2395	9,432	15,5915 seg	5,1971 seg
Prueba 11	6,53	0,1947	6,341	13,0657 seg	4,3552 seg
Prueba 12	5,24	0,1961	7,098	12,5341 seg	4,1780 seg
Prueba 13	3,61	0,1963	4,562	8,3683 seg	2,7894 seg
Prueba 14	5,97	0,2095	10,643	16,8225 seg	5,6075 seg
Prueba 15	3,81	0,199	12,767	16,776 seg	5,592 seg
Total					59,3673 seg

Nota. Resultado de ejecución de pruebas por tarea en Flutter.

Tabla 20

Tiempos de ejecución por tarea en React Native

Tiempo de respuesta con el framework React Native					
Nº Prueba	Tarea 1	Tarea 2	Tarea 3	Resultado Total	Resultado total / Nº tareas
	Resultado	Resultado	Resultado	Total	
Prueba 1	31	28	15	74 seg	24,6666 seg
Prueba 2	23	19	13	55 seg	18,3333 seg
Prueba 3	21	19	13	53 seg	17,6666 seg

Tiempo de respuesta con el framework React Native

N° Prueba	Tarea 1	Tarea 2	Tarea 3	Resultado	Resultado
	Resultado	Resultado	Resultado	Total	total / N° tareas
Prueba 4	20	20	13	53 seg	17,6666 seg
Prueba 5	35	16	12	63 seg	21 seg
Prueba 6	32	13	12	57 seg	19 seg
Prueba 7	15	19	11	34 seg	11,3333 seg
Prueba 8	23	20	11	54 seg	18 seg
Prueba 9	24	18	10	52 seg	17,3333 seg
Prueba 10	30	14	12	56 seg	18,6666 seg
Prueba 11	45	18	13	76 seg	25,3333 seg
Prueba 12	20	18	11	49 seg	16,3333 seg
Prueba 13	30	19	12	61 seg	20,3333 seg
Prueba 14	25	20	13	58 seg	19,3333 seg
Prueba 15	30	18	12	60 seg	20 seg
Total					248,3329 seg

Nota. Resultado de ejecución de pruebas por tarea en React Native.

Tabla 21

Tiempos de ejecución por tarea en Ionic

Tiempo de respuesta con el framework Ionic					
N° Prueba	Tarea 1	Tarea 2	Tarea 3	Resultado	Resultado
	Resultado	Resultado	Resultado	Total	total / N° tareas
Prueba 1	32	27	15	74 seg	24,6666 seg
Prueba 2	24	20	13	57 seg	19 seg
Prueba 3	23	19	13	55 seg	18,3333 seg
Prueba 4	22	18	13	53 seg	17,6666 seg
Prueba 5	26	15	12	53 seg	17,6666 seg
Prueba 6	25	16	12	53 seg	17,6666 seg
Prueba 7	23	17	11	51 seg	17 seg
Prueba 8	23	19	11	53 seg	17,6666 seg
Prueba 9	24	18	10	52 seg	17,3333 seg
Prueba 10	26	15	12	53 seg	17,6666 seg
Prueba 11	25	18	13	13 seg	4,33333 seg
Prueba 12	24	16	11	51 seg	17 seg
Prueba 13	23	14	12	49 seg	16,3333 seg
Prueba 14	22	15	13	50 seg	16,6666 seg
Prueba 15	23	16	12	51 seg	17 seg
Total					255,9994 seg

Nota. Resultado de ejecución de pruebas por tarea en Ionic.

Tabla 22

Tiempos de ejecución por tarea en Xamarin

Tiempo de respuesta con el framework Xamarin					
N° Prueba	Tarea 1	Tarea 2	Tarea 3	Resultado	Resultado total / N° tareas
	Resultado	Resultado	Resultado	Total	
Prueba 1	18	10	7	35 seg	11,6666 seg
Prueba 2	14	10	7	31 seg	10,3333 seg
Prueba 3	14	10	8	32 seg	10,6666 seg
Prueba 4	14	10	7	31 seg	10,3333 seg
Prueba 5	20	10	7	37 seg	12,3333 seg
Prueba 6	13	9	8	30 seg	10 seg
Prueba 7	13	10	7	30 seg	10 seg
Prueba 8	13	10	7	30 seg	10 seg
Prueba 9	13	10	7	30 seg	10 seg
Prueba 10	14	10	7	31 seg	10,3333 seg
Prueba 11	13	11	8	32 seg	10,6666 seg
Prueba 12	14	10	8	32 seg	10,6666 seg
Prueba 13	13	10	7	30 seg	10 seg
Prueba 14	13	10	7	30 seg	10 seg
Prueba 15	13	10	7	30 seg	10 seg
Total					156,9996 seg

Nota. Resultado de ejecución de pruebas por tarea en Xamarin.

- **Utilización de recursos**

Utilización de CPU. -La información proporcionada sobre la cantidad de CPU usada por el sistema realizado en cada framework se presenta a continuación.

Flutter:

Esta vista es útil para identificar métodos costosos en un perfil de CPU. Cuando el nodo raíz de la tabla tiene un tiempo intrínseco grande, significa que, en muchos ejemplos de CPU en este perfil, el método termina en la parte superior de la pila de llamadas.

Figura 9

Detección de señaléticas de tránsito Flutter

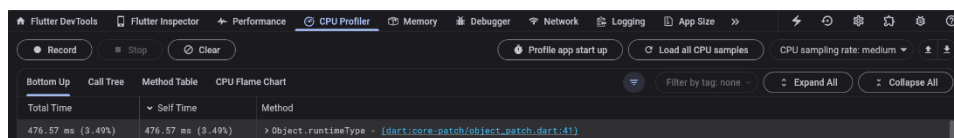


Figura 10

Presentación de la información Flutter

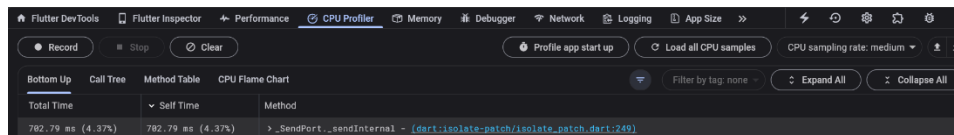


Figura 11

Descarga del archivo Flutter

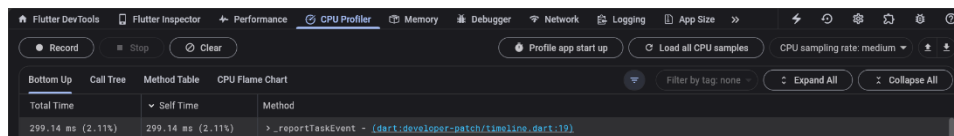


Tabla 23*Porcentaje de CPU utilizado Flutter*

	Flutter
Tarea 1	3,49 %
Tarea 2	4,37 %
Tarea 3	2,11 %
Total	3.3233 %

Nota. Resultado de porcentaje utilizado en CPU.

React Native:

La información presentada detalla la utilización de recursos de CPU, los cuales reflejan los valores que alcanzan al momento de ejecutar una acción dentro de la aplicación móvil.

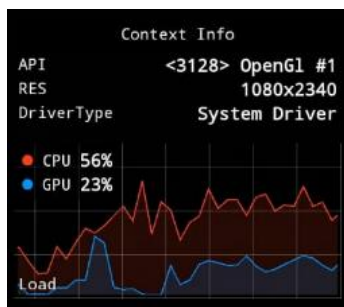
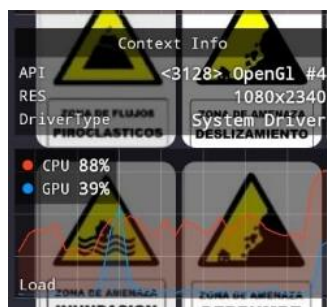
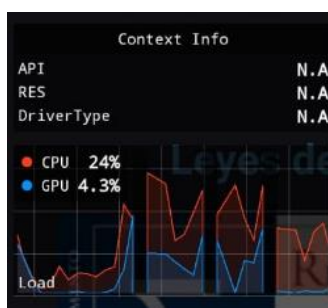
Figura 12*Detección de señaléticas de tránsito React Native*

Figura 13

Presentación de la información React Native

**Figura 14**

Descarga del archivo React Native

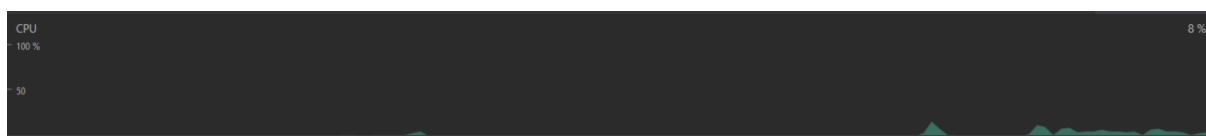
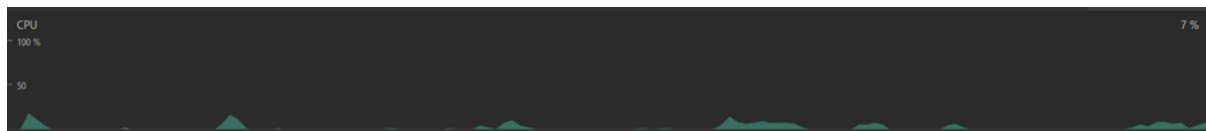
**Tabla 24**

Porcentaje de CPU utilizado React Native

React Native	
Tarea 1	56%
Tarea 2	88 %
Tarea 3	24 %
Total	56 %

Nota. Resultado de porcentaje utilizado en CPU.

Ionic:

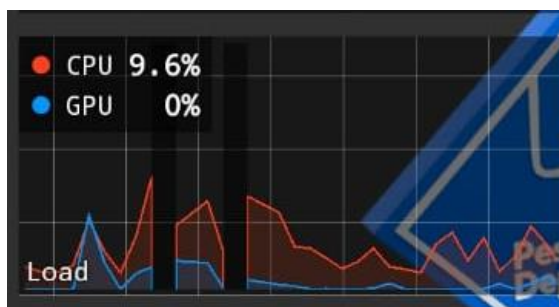
Figura 15*Detección de señal Ionic***Figura 16***Presentación de información Ionic***Figura 17***Descarga de archivo Ionic***Tabla 25***Porcentaje de CPU utilizado Ionic*

Ionic	
Tarea 1	13%
Tarea 2	8 %
Tarea 3	7 %
Total	28 %

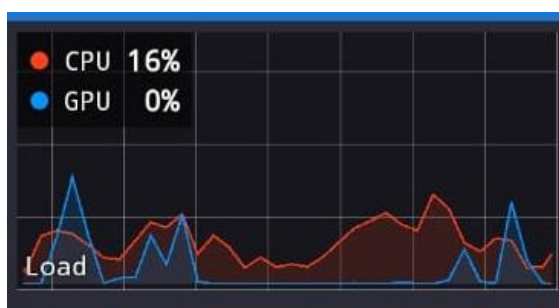
Nota. Resultado de porcentaje utilizado en CPU.**Xamarin:**

Figura 18

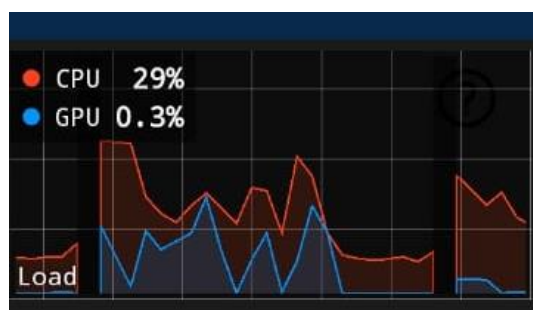
Detección de señal Xamarin

**Figura 19**

Presentación de información Xamarin

**Figura 20**

Descarga de archivo Xamarin



Utilización de memoria RAM. – Los gráficos proporcionados una descripción gráfica del estado del montón de los frameworks y de la memoria de cada uno a lo largo del tiempo.

Flutter:

Figura 21

Detección de señaléticas de tránsito Flutter

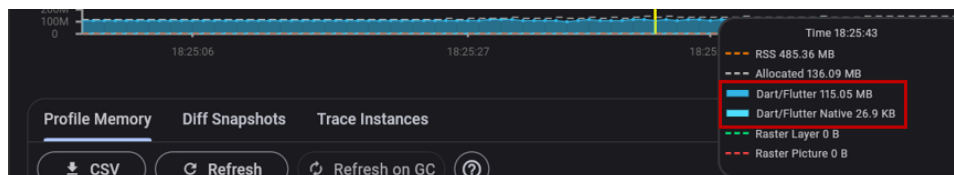


Figura 22

Presentación de la información Flutter

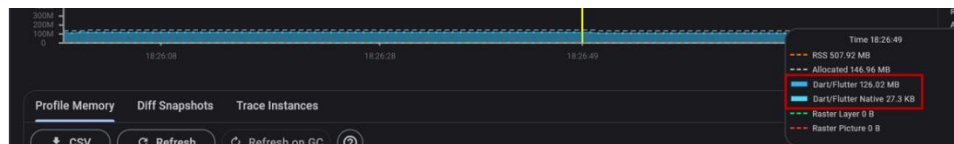


Figura 23

Descarga de archivo Flutter

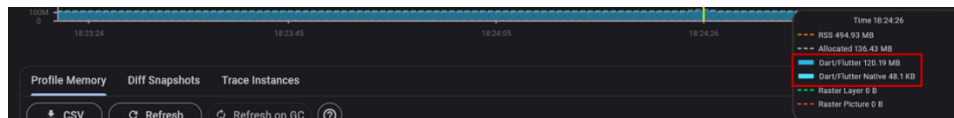


Tabla 26

Valor de memoria RAM utilizado Flutter

Flutter	
Tarea 1	115.05 MB
Tarea 2	126.02 MB
Tarea 3	120.19 MB

Flutter

Total 120.42 MB

Nota. Resultado de memoria RAM utilizado en la ejecución de tareas asignadas.

React Native:

Figura 24

Detección de señal React Native

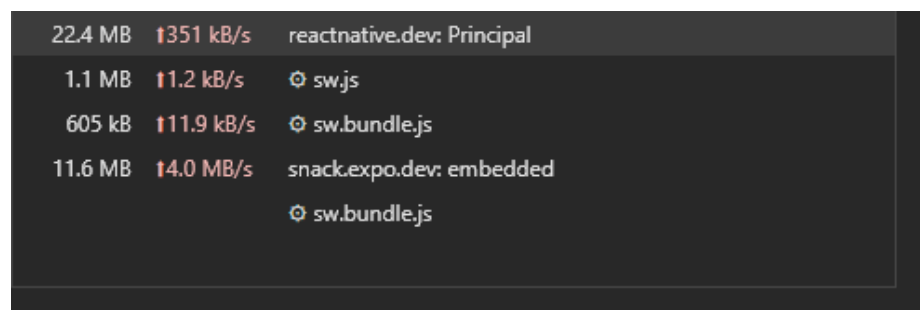


Figura 25

Presentación de información React Native

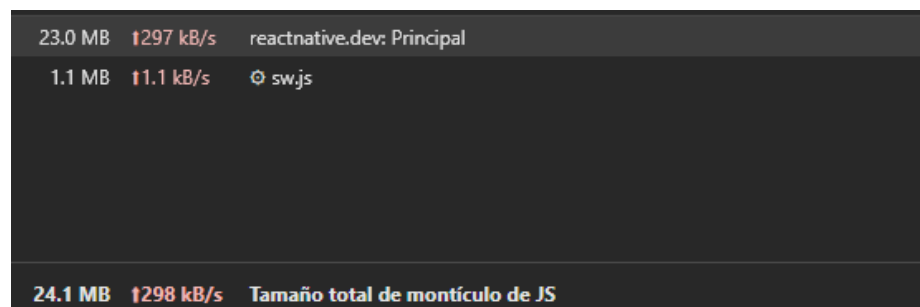
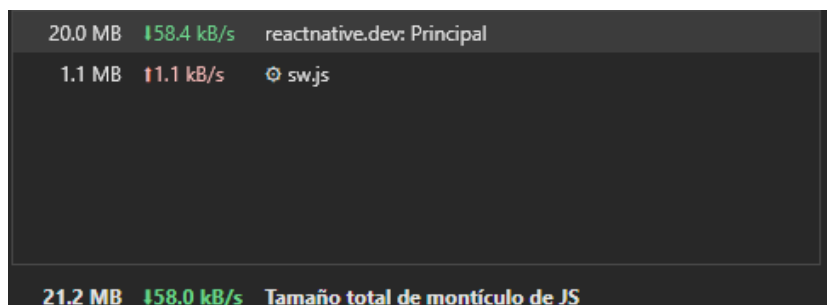
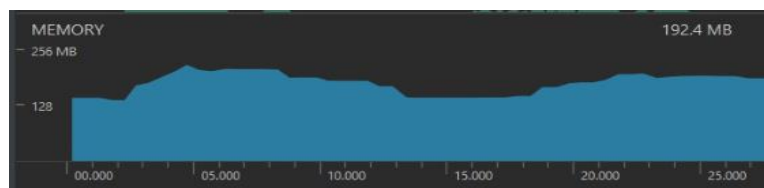


Figura 26*Descarga de archivo React Native***Tabla 27***Valor de memoria RAM utilizado React Native*

React Native	
Tarea 1	22.4 MB
Tarea 2	23 MB
Tarea 3	20 MB
Total	21,8 MB

Nota. Resultado de memoria RAM utilizado en la ejecución de tareas asignadas.**Ionic:****Figura 27***Detección de señal Ionic*

Figura 28*Presentación de información Ionic***Figura 29***Descarga de archivo Ionic***Tabla 28***Valor de memoria RAM utilizado Ionic*

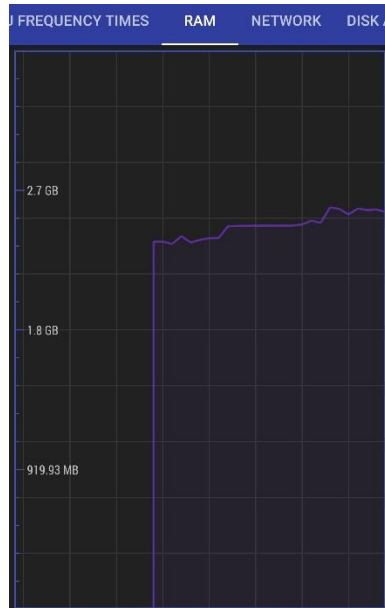
	Ionic
Tarea 1	163 MB
Tarea 2	192.4 MB
Tarea 3	177.1 MB
Total	177.5 MB

Nota. Resultado de memoria RAM utilizado en la ejecución de tareas asignadas.

Xamarin:

Figura 30

Detección de señal Xamarin

**Figura 31**

Presentación de información Xamarin

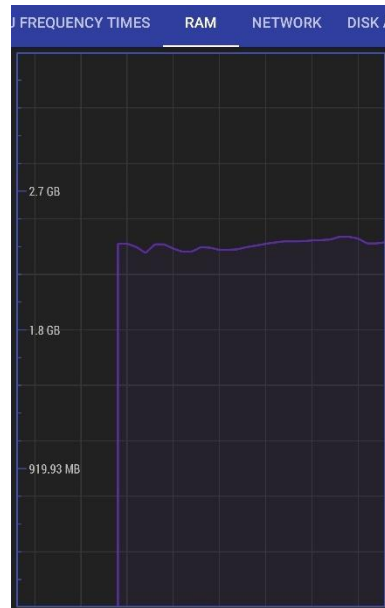
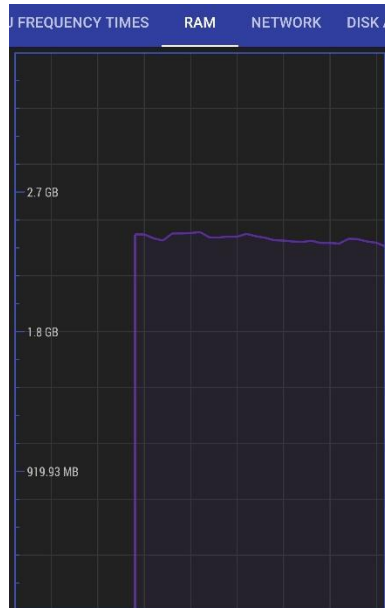


Figura 32*Descarga de archivo Xamarin***Figura 33***Valor de memoria RAM utilizado Xamarin*

Xamarin	
Tarea 1	2,3 GB
Tarea 2	2,5 GB
Tarea 3	2,5 GB
Total	2,43 GB

Nota. Resultado de memoria RAM utilizado en la ejecución de tareas asignadas.

- **Extensión de código**

El índice de modificación de código mide la proporción de líneas de código que han sido modificadas o agregadas con respecto al tamaño total del código original.

Flutter:

Tabla 29

Valor total sobre extensión de código Flutter

Flutter	Líneas de Código Nuevas o Modificadas	Líneas de código originales	Aplicación Formula	Total
Home	64	97	$x = \frac{64}{97}$	0.659
Menú	77	83	$x = \frac{77}{83}$	0.927
Camera	221	232	$x = \frac{221}{232}$	0.952
Transito	350	304	$x = \frac{304}{305}$	0.868
Descripción	67	83	$x = \frac{67}{83}$	0.807
Leyes	75	150	$x = \frac{75}{150}$	0.5
Total				0.7855

Nota. Resultado de extensión de código evaluado en Flutter.

Figura 34

Código fuente evaluado

```

lib > src > pages > home.dart u > MyHomePageState
1 import 'package:app_reconocimiento_texto/src/pages/menu.dart';
2 import 'package:flutter/material.dart';
3
4 class MyHomePage extends StatefulWidget {
5   @override
6   MyHomePageState createState() => MyHomePageState();
7
8
9
10 class MyHomePageState extends State<MyHomePage> {
11   @override
12   Widget build(BuildContext context) {
13     return Scaffold(
14       backgroundColor: Color.fromRGB(44, 44, 56, 1), // Fondo de color #000
15       body: Container(
16         padding: EdgeInsets.all(16.0),
17         child: Column(
18           mainAxisAlignment: MainAxisAlignment.center,
19           children: [
20             // Imagen del escudo
21             Image.asset("assets/shield.png", height:
22               361.0), // Image.asset
23             SizedBox(height: 20.0), // Espacio entre la imagen y el texto
24             // Texto de información
25             Container(
26               margin: EdgeInsets.symmetric(horizontal: 32.0),
27               decoration: BoxDecoration(
28                 border: Border.all(color: Colors.lightBlueAccent),
29                 borderRadius: BorderRadius.circular(10.0),
30               ), // BoxDecoration
31               child: Padding(
32                 padding: const EdgeInsets.all(16.0),
33                 child: Column(
34                   mainAxisAlignment: MainAxisAlignment.spaceAround,
35                   crossAxisAlignment: CrossAxisAlignment.center,
36                   children: [
37                     // Icono de exclamación
38                     Row(
39                       mainAxisAlignment: MainAxisAlignment.center,
40                       crossAxisAlignment: CrossAxisAlignment.center,
41                       children: [
42                         Icon(
43                           Icons.info_outline,
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60

```

React Native:

Tabla 30

Valor total sobre extensión de código React Native

React Native	Líneas de Código	Líneas de código	Aplicación Formula	Total
	Nuevas o Modificadas	originales		
Main	76	64	$x = \frac{76}{64}$	1.1875
MainMenu	127	149	$x = \frac{127}{149}$	0.8523
DeteccionSenialesScreen	242	117	$x = \frac{242}{117}$	2.0683
TrafficSign	198	212	$x = \frac{198}{212}$	0.9339
LawPage	165	170	$x = \frac{165}{170}$	0.9705
ImageDatailPage	43	71	$x = \frac{43}{71}$	0.6056
Total				1.1030

Nota. Resultado evaluación sobre la extensión de código.

Figura 35

Código fuente evaluado

```

src > components > Modification.js >
  9  const Det > const
165
166   <Image // 150
167     source={require('./images/example.gif')} // 151
168     style={styles.modalImage} // 152
169   /> // 153
170 </View> // 154
171 <Text>Approach any traffic sign and observe "Aumente
172 <TouchableOpacity style={styles.modalButton} onPress=
173   <Text style={styles.modalButtonText}>Close</Text> /
174 </TouchableOpacity> // 158
175 </View> // 159
176 </Modal> // 160
177 </View> // 161
178 }; // 162
179 }; // 163
180
181 const styles = StyleSheet.create({ // 164
182   detectedObjectContainer: { // 165
183     position: 'absolute', // 166
184     backgroundColor: 'rgba(0, 255, 0, 0.7)', // 167
185     padding: 10, // 168
186     borderRadius: 8, // 169
187   }, // 170
188   detectedObjectLabelText: { // 171
189     color: 'white', // 172
190     fontSize: 16, // 173
191   }, // 174
192   floatingButton: { // 175
193     position: 'absolute', // 176
194     top: 20, // 177
195     left: 20, // 178
196     borderRadius: 30, // 179
197     padding: 20, // 180
198   }, // 181
199   descriptionContainer: { // 182
200     position: 'absolute', // 183
201     backgroundColor: '#3498db', // 184
202     top: 750, // 185
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225   modalContent: { // 208
226     backgroundColor: '#fff', // 209
227     padding: 20, // 210
228     borderRadius: 10, // 211
229     marginTop: 150, // 212
230     marginLeft: 20, // 213
231     marginRight: 20, // 214
232   }, // 215
233   modalTitle: { // 216
234     fontSize: 18, // 217
235     fontWeight: 'bold', // 218
236     marginBottom: 10, // 219
237   }, // 220
238   modalButton: { // 221
239     marginTop: 20, // 222
240     backgroundColor: '#63B5E5', // 223
241     padding: 10, // 224
242     borderRadius: 5, // 225
243     alignItems: 'center', // 226
244   }, // 227
245   modalButtonText: { // 228
246     color: '#fff', // 229
247     fontSize: 16, // 230
248   }, // 231
249   modalImageContainer: { // 232
250     alignItems: 'center', // 233
251     marginBottom: 10, // 234
252   }, // 235
253   modalImage: { // 236
254     width: 200, // 237
255     height: 200, // 238
256     resizeMode: 'cover', // 239
257   }, // 240
258 }); // 241
259
260 export default DetectionScreen; // 242
261

```

Ionic:

Tabla 31

Valor total sobre extensión de código Ionic

Ionic	Líneas de Código Nuevas o Modificadas	Líneas de código originales	Aplicación Formula	Total
Home	15	18	$x = \frac{15}{18}$	0.833
Page 1- Menu	16	31	$x = \frac{16}{31}$	0.516
Page2 - Captura	20	28	$x = \frac{20}{28}$	0.714
Page3 – Tipos de Señales	15	44	$x = \frac{15}{44}$	0.34
Image Detail	10	22	$x = \frac{10}{22}$	0.45
Page4 – Manual de Transito	13	17	$x = \frac{13}{17}$	0.76
Total				0.6021

Nota. Resultado evaluación sobre la extensión de código.

Figura 36

Código fuente evaluado

```

1 Click here to ask Blackbox to help you code faster
2 <ion-header>
3   <ion-toolbar>
4     <ion-title>{{ name }}</ion-title>
5     <ion-buttons slot="end">
6       <ion-button (click)="closeModal()">
7         <ion-icon slot="icon-only" name="close"></ion-icon>
8       </ion-button>
9     </ion-buttons>
10  </ion-toolbar>
11 </ion-header>
12 <ion-content>
13   <ion-img [src]="image" class="full-size-image"></ion-img>
14 </ion-content>
15 <ion-card>
16   <ion-card-content>
17     <p>{{ description }}</p>
18   </ion-card-content>
19 </ion-card>
20 </ion-content>
21
src > app > image-detail > image-detail.page.html > ...
3   <ion-title>{{ name }}</ion-title>
4   <ion-buttons slot="end">
5     <ion-button (click)="closeModal()">
6       <ion-icon slot="icon-only" name="close"></ion-icon>
7     </ion-button>
8   </ion-buttons>
9 </ion-toolbar>
10 </ion-header>
11 <ion-content>
12   <ion-img [src]="image" class="full-size-image"></ion-img>
13 </ion-content>
14 <ion-card>
15   <ion-card-header>
16     <ion-card-title>{{ name }}</ion-card-title>
17   </ion-card-header>
18   <ion-card-content>
19     <p>{{ description }}</p>
20   </ion-card-content>
21 </ion-card>
22 </ion-content>
23

```

Xamarin:

Tabla 32

Valor total sobre extensión de código Xamarin

Xamarin	Líneas de Código Nuevas o Modificadas	Líneas de código originales	Aplicación Formula	Total
Captura	20	28	$x = \frac{20}{28}$	0.71
DetalleSerial	40	44	$x = \frac{40}{44}$	0.90
Inicio	19	21	$x = \frac{21}{29}$	1.10
Leyes	32	37	$x = \frac{37}{32}$	1.15
Opciones	39	45	$x = \frac{45}{39}$	1.15

Xamarin

	Líneas de Código Nuevas o Modificadas	Líneas de código originales	Aplicación Formula	Total
Tipos	43	47	$x = \frac{47}{43}$	1.09
Total				1.0166

Nota. Resultado evaluación sobre la extensión de código.

Figura 37

Código fuente evaluado

```

1 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
2 xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
3 xmlns:local="clr-namespace:FormsTestApp.iOS"
4 NavigationPage.IsNavigationBarVisible="false"
5 BackgroundColor="White"
6 >
7     <StackLayout>
8         <StackLayout Orientation="Horizontal"
9             DepthCue="Ascending"
10            HeightRequest="70"
11            Margin="15,0,0,0"
12            >
13             <Image GestureRecognizers="
14                 {Binding Recognizer1, Converter={}}
15                 {Binding Recognizer2, Converter={}}
16                 {Binding Recognizer3, Converter={}}
17             >
18             </Image>
19         </StackLayout>
20         <StackLayout Grid.Row="1" Margin="0,0,0,0"
21             Grid.Column="0" Grid.ColumnSpan="2"
22             Grid.RowSpan="2"
23             >
24             <Label Text="{Binding Nombre}" FontSize="Large" TextColor="White" FontAttributes="Bold" Margin="0,15,0,0" HorizontalTextAlignment="Center" />
25             </Label>
26             <StackLayout Grid.Row="1" Grid.Column="1"
27                 Grid.ColumnSpan="2"
28                 >
29                 <StackLayout Orientation="Horizontal"
30                     DepthCue="Ascending"
31                     >
32                     <Image Source="{Binding Image}" WidthRequest="200" HeightRequest="200" Aspect="AspectFit" Margin="0,10,0,0" />
33                     </Image>
34                     <Label Text="{Binding Descripción}" TextColor="White" Margin="10,10,0,0" FontSize="20" HorizontalTextAlignment="Center" />
35                     </Label>
36                 </StackLayout>
37             </StackLayout>
38         </StackLayout>
39     </ContentPage>

```

```

1 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
2 xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
3 xmlns:local="clr-namespace:FormsTestApp.Droid"
4 NavigationPage.IsNavigationBarVisible="false"
5 BackgroundColor="White"
6 >
7     <StackLayout>
8         <StackLayout Orientation="Horizontal"
9             DepthCue="Ascending"
10            HeightRequest="70"
11            Margin="15,0,0,0"
12            >
13             <Image Source="{Binding Image}" WidthRequest="200" HeightRequest="200" Aspect="AspectFit" Margin="0,10,0,0" />
14             <Label Text="{Binding Descripción}" TextColor="White" Margin="10,10,0,0" FontSize="20" HorizontalTextAlignment="Center" />
15             </Image>
16         </StackLayout>
17         <StackLayout Grid.Row="1" Margin="0,0,0,0"
18             Grid.Column="0" Grid.ColumnSpan="2"
19             Grid.RowSpan="2"
20             >
21             <Label Text="{Binding Nombre}" FontSize="Large" TextColor="White" FontAttributes="Bold" Margin="0,15,0,0" HorizontalTextAlignment="Center" />
22             </Label>
23             <StackLayout Grid.Row="1" Grid.Column="1"
24                 Grid.ColumnSpan="2"
25                 >
26                 <StackLayout Orientation="Horizontal"
27                     DepthCue="Ascending"
28                     >
29                     <Image Source="{Binding Image}" WidthRequest="200" HeightRequest="200" Aspect="AspectFit" Margin="0,10,0,0" />
30                     <Label Text="{Binding Descripción}" TextColor="White" Margin="10,10,0,0" FontSize="20" HorizontalTextAlignment="Center" />
31                     </Image>
32                 </StackLayout>
33             </StackLayout>
34         </StackLayout>
35     </ContentPage>

```

Usabilidad

- Experiencia de usuario final

Tabla 33

Resultados de evaluación ordinal de frameworks

¿Qué tan atractiva es la interfaz? (facilidad de uso terminal, la estética de código, la claridad de la documentación y la eficiencia en la realización de tareas)

React Native	Medio
Xamarin	Bajo
Flutter	Alto
Ionic	Medio

Nota. Resultado ordinal de evaluación de frameworks.

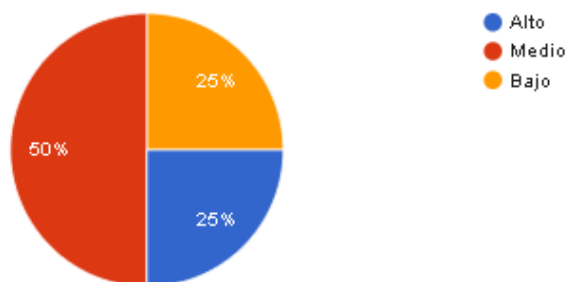
Figura 38

Resultado de encuesta

Usabilidad

¿Qué tan atractiva es la interfaz? (facilidad de uso terminal, la estética de código, la claridad de la documentación y la eficiencia en la realización de tareas)

4 respuestas



- Tiempo de instalación y configuración básica

Tabla 34

Resultados de evaluación ordinal frameworks

¿Cuánto tiempo se es requerido para la instalación y configuración?	
React Native	10 a 30
Xamarin	5 a 10 minutos
Flutter	5 a 10 minutos
Ionic	más de 30

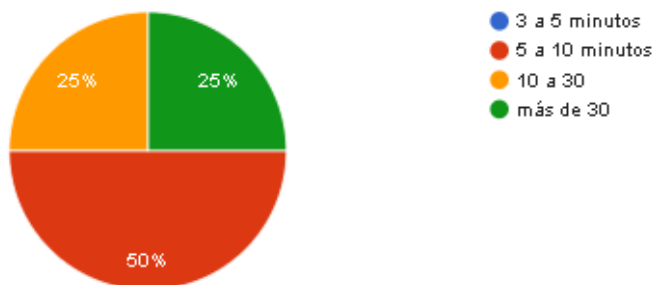
Nota. Resultado ordinal de evaluación de frameworks.

Figura 39

Resultado de encuesta

¿Cuánto tiempo se es requerido para la instalación y configuración?

4 respuestas



- **Requisitos previos de instalación**

En base a la experiencia del usuario en el caso de estudio propuesto con respecto a cuan compleja es la instalación del framework. La Figura 41 describe la pregunta y las opciones que el usuario respondió en base a su criterio.

Tabla 35

Resultados de evaluación ordinal frameworks

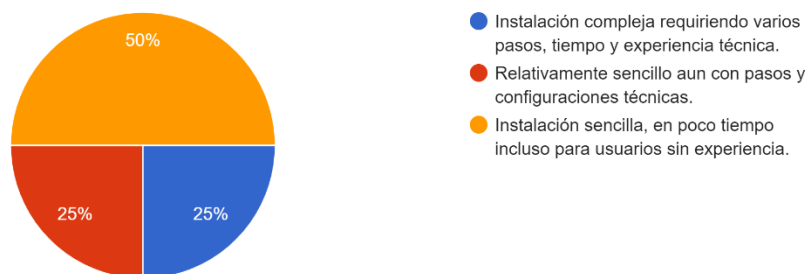
¿Cuán compleja es la Instalación?	
React Native	Instalación sencilla, en poco tiempo incluso para usuarios sin experiencia.
Xamarin	Relativamente sencillo aun con pasos y configuraciones técnicas.
Flutter	Instalación sencilla, en poco tiempo incluso para usuarios sin experiencia.
Ionic	Instalación compleja requiriendo varios pasos, tiempo y experiencia técnica.

Nota. Resultado ordinal de evaluación de frameworks.

Figura 40

Resultado de encuesta

¿Cuán compleja es la Instalación?
4 respuestas



- **Facilidad de aprendizaje**

En base a la experiencia del usuario en el caso de estudio propuesto con respecto a cuánto tiempo le tomo aprender a desarrollar en el framework. Tabla 36 describe la pregunta y las opciones que el usuario respondió en base a su criterio.

Tabla 36

Resultados de evaluación ordinal frameworks

¿Cuánto tiempo le tomo al usuario aprender a desarrollar en el framework?	
React Native	Experiencia básica en el desarrollo en un tiempo aproximado de una a dos semanas.
Xamarin	Requiere un tiempo moderado aproximadamente entre uno a tres meses.
Flutter	Requiere un tiempo moderado aproximadamente entre uno a tres meses.
Ionic	Requiere un tiempo moderado aproximadamente entre uno a tres meses.

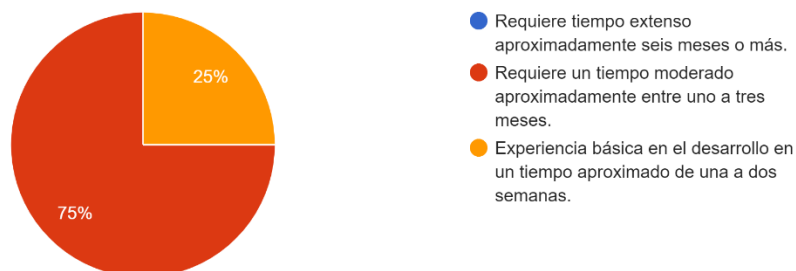
Nota. Resultado ordinal de evaluación de frameworks.

Figura 41

Resultado de encuesta

¿Cuánto tiempo le tomo al usuario aprender a desarrollar en el framework?

4 respuestas







Adopción

- **Popularidad**

Basándose en la cantidad de opiniones y calificación de estrellas de cada framework, se puede realizar un análisis comparativo de la popularidad relativa de los frameworks. La Figura 42 muestra la comparativa de los frameworks.

Figura 42

Comparación de Frameworks

	Flutter	React Native	Ionic	ComponentOne...
	 Flutter	 React Native	 Ionic Try for free	 ComponentOne...
De un vistazo				
Calificación de estrellas	★★★★☆ 63 opiniones	★★★★☆ 117 opiniones	★★★★☆ 148 opiniones	★★★★☆ 11 opiniones
Segmentos de mercado	Pequeñas empresas (70,7% de las reseñas) ⓘ	Pequeñas empresas (60,2% de las reseñas) ⓘ	Pequeñas empresas (71,7% de las reseñas) ⓘ	Pequeñas empresas (81,8% de las reseñas) ⓘ

Nota. La presente figura muestra la comparativa de frameworks proporcionado de (Compare Flutter vs. React Native vs. Ionic vs. ComponentOne Studio for Xamarin, s. f.).

Ionic. - Con 148 opiniones, Ionic es el framework con la mayor cantidad de opiniones en la lista que indica una popularidad significativa entre los desarrolladores.

React. - Con 117 opiniones, React es el framework con segunda mayor cantidad de opiniones en la lista, sugiriendo una alta popularidad y una amplia adopción por parte de la comunidad de desarrollo.

Flutter: A pesar de tener solo 63 opiniones, Flutter en la actualidad ha ganado una gran tracción en la comunidad de desarrollo debido a su rendimiento, facilidad de uso y capacidad para crear aplicaciones móviles de alta calidad.

Xamarin: Con solo 11 opiniones, Xamarin tiene la menor cantidad de opiniones en la lista indicando una menor popularidad en comparación con los otros frameworks mencionados.

Apoyo

- **Soporte**

Según la experiencia de usuario en el caso de estudio propuesto, se analiza el tiempo que tarda el equipo de soporte en responder las consultas de usuarios, y también si el soporte es claro y útil para satisfacer las necesidades del usuario. La Figura 43 y la Figura 44 describen la pregunta planteada y la opción seleccionada por los usuarios en función a su criterio. Además, se incluye la Tabla #, que da información de respaldo a la respuesta proporcionada.

Tabla 37

Resultados de evaluación ordinal frameworks

¿Cuánto tiempo tarda el equipo de soporte en responder a las consultas de los usuarios?	
React Native	Responde de manera rápida y confiable.
Xamarin	No responde a las consultas.
Flutter	Responde de manera rápida y confiable.
Ionic	Responde en un tiempo razonable, aunque existe variabilidad en el tiempo de respuesta.

Nota. Resultado ordinal de evaluación de frameworks.

Figura 43

Resultado de encuesta

¿Cuánto tiempo tarda el equipo de soporte en responder a las consultas de los usuarios?

4 respuestas

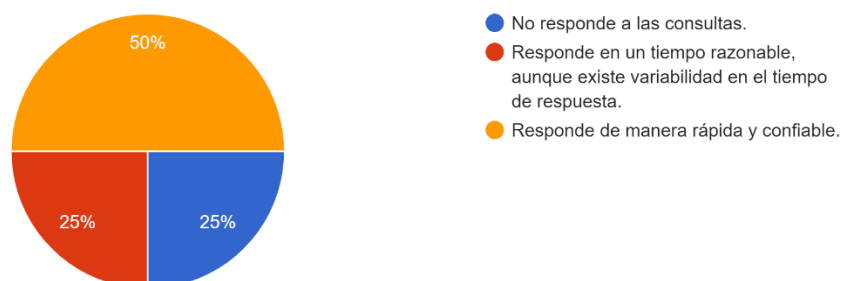


Tabla 38

Resultados de evaluación ordinal frameworks

¿Las respuestas proporcionadas por el equipo de soporte son claras, útiles y satisfacen las necesidades de los usuarios?

React Native	Respuestas claras y útiles satisfaciendo a los usuarios de forma efectiva.
Xamarin	Respuestas confusas, incoherentes o poco claras.
Flutter	Respuestas claras y útiles satisfaciendo a los usuarios de forma efectiva.
Ionic	Mayoría de respuestas útiles y claras.

Nota. Resultado ordinal de evaluación de frameworks.

Figura 44

Resultado de encuesta

¿Las respuestas proporcionadas por el equipo de soporte son claras, útiles y satisfacen las necesidades de los usuarios?

4 respuestas

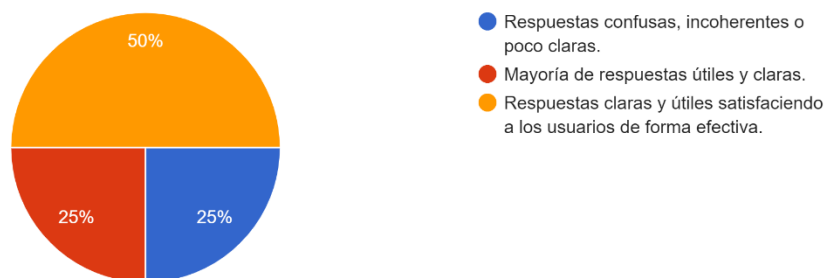


Tabla 39

Información respalda en base a la encuesta

Elementos	Framework	Descripción presentada
Soporte y solución de problemas	Flutter	El soporte brindado por flutter es rápido puesto que en el

Elementos	Framework	Descripción presentada
	React Native	<p>repositorio ayudan resolver los problemas presentados eficientemente.</p> <p>La solución de problemas y soporte se hace mediante la comunidad de React Native en el cual se evidencia la respuesta rápida y actualizada.</p>
	Ionic	<p>El último soporte presentado fue en 29/03/2023 y está activo para actualizaciones.</p>
	Xamarin	<p>El soporte presentado por Xamarin no están recurrente la última actualización de componentes para Android fue hace 8 meses y el soporte para Xamarin finalizara el 01/05/2024 (davidbritch, 2024).</p>

Nota. Respaldo de información generado a partir de la encuesta realizada.

- **Capacitación**

Según la experiencia de usuario en el caso de estudio propuesto, se analiza si existen eventos de capacitación o meetups organizados por la comunidad del framework. La Figura 45 describen la pregunta planteada y la opción seleccionada por los usuarios en función a su

criterio. Además, se incluye la Tabla #, que da información de respaldo a la respuesta proporcionada.

Tabla 40

Resultados de evaluación ordinal frameworks

¿Hay eventos de capacitación o meetups organizadas por la comunidad de usuarios que utilizan el framework?

React Native	Capacitaciones o meetups disponibles frecuentemente presentado por expertos.
Xamarin	Ninguna disponibilidad de capacitaciones o meetups.
Flutter	Capacitaciones o meetups disponibles frecuentemente presentado por expertos.
Ionic	Ninguna disponibilidad de capacitaciones o meetups.

Nota. Resultado ordinal de evaluación de frameworks.

Figura 45

Resultado de encuesta

¿Hay eventos de capacitación o meetups organizadas por la comunidad de usuarios que utilizan el framework?

4 respuestas

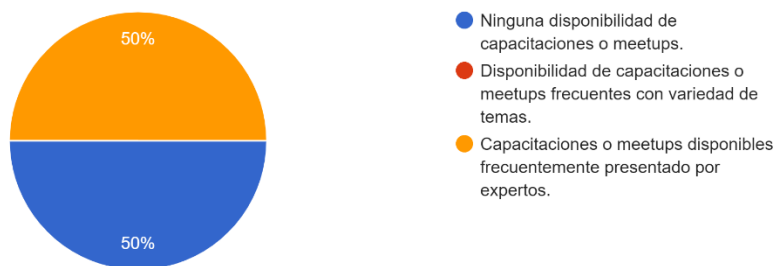


Tabla 41*Información respalda en base a la encuesta*

Framework	Elementos	Descripción presentada
Eventos o meetups de capacitación	Flutter	Los eventos para el framework publicados en meetup plataforma mundial en la cual publica eventos o capacitaciones del framework.
	React Native	Los eventos para el framework son publicados a través de reactiflux y discord.
	Ionic	Los eventos para el framework son relativamente frecuentes ya que presenta en directo en la comunidad de Ionic Utah.
	Xamarin	Los eventos para el framework es escasa ya que no presentan eventos recientes.

Nota. Respaldo de información generado a partir de la encuesta realizada.

- **Documentación de referencia**

En base a la experiencia del usuario en el caso de estudio propuesto con respecto a si existe documentación completa y accesible del framework. La Figura 46 describe la pregunta y

las opciones que el usuario respondió en base a su criterio. Además, se incluye la Tabla 42, que da información de respaldo a la respuesta proporcionada.

Tabla 42

Resultados de evaluación ordinal frameworks

¿Existe documentación completa y fácilmente accesible para el framework?	
React Native	Documentación exhaustiva clara y altamente accesible.
Xamarin	Documentación suficiente abordando aspectos relevantes.
Flutter	Documentación exhaustiva clara y altamente accesible.
Ionic	Documentación suficiente abordando aspectos relevantes.

Nota. Resultado ordinal de evaluación de frameworks.

Figura 46

Resultado de encuesta

¿Existe documentación completa y fácilmente accesible para el framework?
4 respuestas

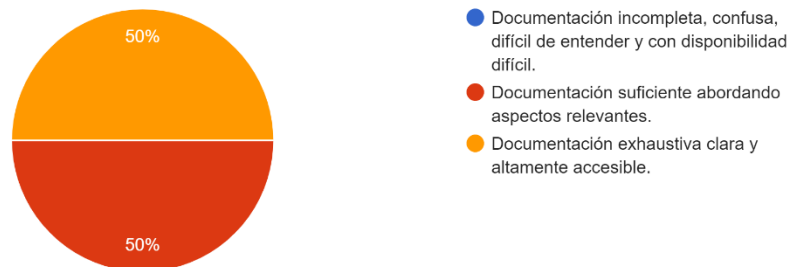


Figura 47

Información respalda en base a la encuesta

Elementos	Framework	Descripción presentada
Ultima actualización	Flutter	La última actualización y de implementación de plugins se realizó el 21/02/2023.
	React Native	La última actualización de React Native se realizó en 08/12/2023.
	Ionic	La última actualización de Ionic 6 en la plataforma oficial del framework fue el 29/03/2024.
Introducción clara	Xamarin	La última actualización en la plataforma oficial del framework fue el 20/09/2022 Introducción clara.
	Flutter	Flutter crea un marco moderno mediante la implementación de widgets que, lo cual la plataforma de Flutter indica que realizar y como realizar su implementación (<i>Flutter documentation</i> <i>Flutter</i> , s. f.).
	React Native	React Native ofrece la capacidad de desarrollar

Elementos	Framework	Descripción presentada
	Ionic	<p>interfaces de usuario dinámicas y responsivas mediante la composición de componentes individuales, los cuales pueden ser ensamblados para formar pantallas, páginas web y aplicaciones completas.</p> <p>Conjunto de herramientas de código abierto, para crear aplicaciones móviles de alto rendimiento, usando HTML, CSS y JavaScript con interacciones como Angular, React y Vue (<i>Introduction to Ionic Ionic Documentation, s. f.</i>).</p>
	Xamarin	<p>Plataforma de código abierto, para crear aplicaciones para iOS, Android y Windows con .NET.Xamarin, con facilidad de compilarse en paquetes de aplicaciones nativas, archivo apk o ipa (<i>What is Xamarin? - Xamarin Microsoft Learn, s. f.</i>).</p>
Instalación y configuración	Flutter	<p>La información se presenta de forma detallada dependiendo</p>

Elementos	Framework	Descripción presentada
	React Native	<p>del sistema operativo que seleccione presentando una guía completa de la instalación y configuración del sistema.</p> <p>La información de instalación proporciona una guía para iniciar un proyecto, configurar el editor y uso de mecanografía, en los cuales se menciona pasos con código o comandos a implementar (<i>React Native · Learn once, write anywhere</i>, s. f.).</p>
	Ionic	<p>La información proporcionada es actualizada y da una guía, para instalar y configurar la versión de Ionic necesaria para el desarrollo.</p>
	Xamarin	<p>La información proporcionada es especificada ya sea para Windows y Mac, además de proporcionar la configuración de firewall de Xamarin.</p>
Guía de inicio rápido	Flutter	<p>En este caso realiza la implementación de widgets</p>

Elementos	Framework	Descripción presentada
	React Native	básicos para indicar el desarrollo paso a paso del ejemplo práctico proporcionado. En este caso presenta de forma clara una guía de cómo crear, agregar, mostrar, renderizar listas, actualizar, compartir datos.
	Ionic	Como inicio rápido se presenta paso a paso la creación de un nuevo proyecto para tomar, guardar y cargar fotos en la implementación de dispositivos móviles.
	Xamarin	Como inicio rápido se presenta paso a paso la creación de un nuevo proyecto, navegación, base de datos, estilo, etc.

Nota. Respaldo de información generado a partir de la encuesta realizada.

- **Tutorial de referencia**

Según la experiencia del usuario en el caso de estudio propuesto sobre si la documentación incluye ejemplos o guías claras paso a paso para los usuarios, la Figura 48

describe la pregunta y las opciones que el usuario respondió según su criterio. Además, se incluye la Tabla 43, que da información de respaldo a la respuesta proporcionada.

Tabla 43

Resultados de evaluación ordinal frameworks

¿La documentación incluye ejemplos claros y guías paso a paso para ayudar a los desarrolladores?

React Native	Tutoriales de referencia amplios y bien elaborados, proporcionando ejemplos prácticos.
Xamarin	Los tutoriales están disponibles y cubren características principales del framework.
Flutter	Tutoriales de referencia amplios y bien elaborados, proporcionando ejemplos prácticos.
Ionic	Los tutoriales están disponibles y cubren características principales del framework.

Nota. Resultado ordinal de evaluación de frameworks.

Figura 48

Resultado de encuesta

¿La documentación incluye ejemplos claros y guías paso a paso para ayudar a los desarrollador
4 respuestas

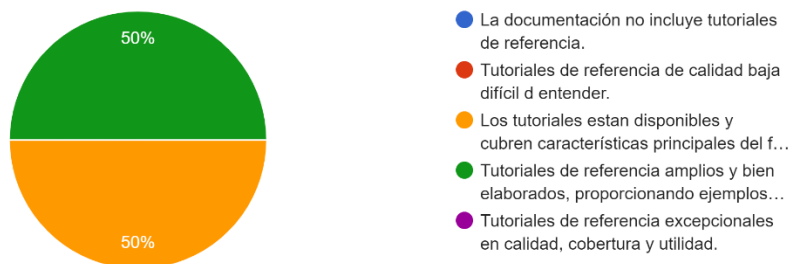


Tabla 44*Información respalda en base a la encuesta*

Elementos	Framework	Descripción presentada
	Flutter	<p>La información proporcionada por la página oficial de flutter muestra código en los tutoriales a seguir, por lo cual es mayormente comprensible.</p>
	React Native	<p>La información proporcionada en el repositorio de react native es actualizada presentando la descripción junto a los pasos y código de muestra para implementarlos.</p>
Ejemplos de código	Ionic	<p>La información detalla conceptos básicos sobre la creación de aplicación, ciclo de vida, navegación / ruta, desplazamiento virtual, migrando de ion-slides a swiper.js, y plataforma</p>
	Xamarin	<p>La información detalla conceptos básicos de creaciones sobre aplicaciones móviles con Xamarin.Forms</p>

Elementos	Framework	Descripción presentada
		como: layouts, control, fundamentos de la aplicación la cual contiene indicaciones, ejemplos y código.

Nota. Respaldo de información generado a partir de la encuesta realizada.

Análisis de resultados

Resultados cuantitativos

Tabla 45

Interpretación de resultados obtenidos en la evaluación

Categoría	Métricas	Fórmula por prueba	Fórmula por todas	Framework	Valor Obtenido	Interpretación del valor
CALIDAD INTERNA						
Funcionalidad	Integración con otras tecnologías	$X = \frac{A}{Ta}$ <i>A</i> =Apis integradas <i>Ta</i> =Total, de Apis requeridas	N/A	React Native	1	1>=1 Entre más cercano a 1 mejor
				Xamarin	1	1>=1 Entre más cercano a 1 mejor (Teniendo en cuenta que no se pudo implementar de forma correcta la AR)
				Flutter	1	1>=1 Entre más cercano a 1 mejor
				Ionic	1	1>=1 Entre más cercano a 1 mejor

Categoría	Métricas	Fórmula por prueba	Fórmula por todas	Framework	Valor Obtenido	Interpretación del valor
				React Native	0.7589	0.7589, React Native muestra una buena cohesión y acoplamiento moderado.
				Xamarin	0.8333	0.8333, lo que sugiere una alta cohesión y bajo acoplamiento (Teniendo en cuenta que no se pudo implementar de forma correcta la AR)
	Modularidad	$X = \frac{Cm + Am}{2}$ Cm =Cohesión de módulos Am =Acoplamiento de módulos.	$\frac{\sum X}{n}$ n =número de pruebas	Flutter	0.5	0.5, Flutter muestra el valor más bajo tiene una cohesión más débil y un acoplamiento más alto
				Ionic	0.7333	0.7333, Ionic se ubica en un nivel razonable de cohesión y acoplamiento
				React Native	0.222	0.222, sugiere que utiliza los recursos de manera eficiente
	Acceso a los recursos del dispositivo	$X = \frac{R}{Tr}$ R = Recursos utilizados Tr = Total, de recursos del dispositivo	N/A	Xamarin	0.333	0.333, todavía muestra un uso eficiente de los recursos, aunque un poco menos en relación a los otros frameworks (Teniendo en cuenta que no se pudo implementar de forma correcta la AR)

Categoría	Métricas	Fórmula por prueba	Fórmula por todas	Framework	Valor Obtenido	Interpretación del valor			
Rendimiento	Tiempo de ejecución de operaciones	$X = Tt$ Tt = Tiempo de tarea	$\Sigma X / n$ n=número de pruebas	Flutter	0.222	0.222, sugiere que utiliza los recursos de manera eficiente			
				Ionic	0.222	0.222, sugiere que utiliza los recursos de manera eficiente			
				React Native	284.3329 seg	284.3329 segundos, muestra un tiempo de carga más largo			
				Xamarin	156.9996 seg	156.9996 segundos, muestra un tiempo de carga moderado (Teniendo en cuenta que no se pudo implementar de forma correcta la AR)			
				Flutter	59.3673 seg	59.3673 segundos, muestra el valor más bajo tiende a cargar más rápido			
				Ionic	255.9994 seg	255.9994 segundos, muestra el tiempo de carga más largo entre todas			
				React Native	56 %	56%, muestra el consumo muy alto entre todas			
				Xamarin	18.2 %	18.2%, muestra un nivel moderado de consumo de recursos (Teniendo en cuenta que no se pudo			
				Utilización de recursos	$X = A$	$\frac{\Sigma X}{n}$			
				Porcentaje de CPU y espacio de RAM	A = Porcentaje de uso de CPU.	n=número de actividades			

Categoría	Métricas	Fórmula por prueba	Fórmula por todas	Framework	Valor Obtenido	Interpretación del valor
						implementar de forma correcta la AR)
				Flutter	3.3233 %	3.3233%, muestra el valor más bajo las aplicaciones tienden a consumir menos
				Ionic	28 %	28%, muestra un nivel más alto de consumo de recursos
				React Native	21,8 MB	21.8 MB, muestra un tamaño bastante bajo
				Xamarin	2,43 GB	2.43 GB, muestra un tamaño significativamente más grande (Teniendo en cuenta que no se pudo implementar de forma correcta la AR)
				Flutter	120.42 MB	120.42 MB, también muestra un tamaño relativamente bajo
				Ionic	177.5 MB	177.5 MB, muestra un tamaño alto
				React Native	1.1030	1.1030, muestra el valor más alto
				Xamarin	1.0166	1.0166, muestra un valor cercano a 1, lo que sugiere una actividad de desarrollo más alta en comparación (Teniendo en cuenta que no se
	Extensión de código	$X = \frac{Lc}{Tr}$ <p>Lc = Líneas de código nuevas o modificadas. Tr = Total, de líneas de código original.</p>	$\frac{\sum X}{n}$ <p>n=número de archivos de código</p>			

Categoría	Métricas	Fórmula por prueba	Fórmula por todas	Framework	Valor Obtenido	Interpretación del valor
						pudo implementar de forma correcta la AR)
				Flutter	0.7855	0.7855, muestra un valor cercano a 0, lo que indica una cierta estabilidad en su código
				Ionic	0.6021	0.6021, muestra el valor más cercano a 0
CALIDAD EXTERNA						
Usabilidad	Experiencia de usuario final	X = conteo de respuestas	N/A	React Native	Medio	"Medio", indica una evaluación intermedia
				Xamarin	Bajo	"Bajo", esta tecnología tiene una evaluación menos favorable en comparación con las otras (Teniendo en cuenta que no se pudo implementar de forma correcta la AR)
				Flutter	Alto	"Alto", lo que sugiere que esta tecnología tiene una evaluación positiva
				Ionic	Medio	"Medio", indica una evaluación intermedia
	Tiempo de Instalación y configuración básica	$X = Tt$ Tt = Tiempo de tarea	N/A	React Native	10 a 30	Requiere un tiempo moderado a largo para completar la actividad
				Xamarin	5 a 10 minutos	Requieren un tiempo moderado para completar la actividad

Categoría	Métricas	Fórmula por prueba	Fórmula por todas	Framework	Valor Obtenido	Interpretación del valor
						(Teniendo en cuenta que no se pudo implementar de forma correcta la AR)
				Flutter	5 a 10 minutos	Requieren un tiempo moderado para completar la actividad
				Ionic	más de 30	Requiere significativamente más tiempo en comparación con las otras

Nota. Resultado cuantitativo de evaluación sobre los diferentes frameworks.

Resultados cualitativos

Tabla 46

Interpretación de resultados obtenidos en la evaluación

Categoría	Métricas	Propósito	Framework	Puntaje					Interpretación del valor
				1 Inaceptable	2 Pobre	3 Aceptable	4 Muy Bueno	5 Excelente	
CALIDAD INTERNA									
Funcionalidad	Completitud de la implementación funcional	¿El framework proporciona todas las características y funcionalidades necesarias para desarrollar una aplicación de	React Native						Disponible en todos los elementos de código con una respuesta instantánea.
			Xamarin		x				Es limitado e inconsistente (Teniendo en cuenta que no

Categoría	Métricas	Propósito	Puntaje					Interpretación del valor
			Framework	1	2	3	4	
				Inaceptable	Pobre	Aceptable	Muy Bueno	Excelente
		realidad aumentada?						se pudo implementar de forma correcta la AR)
			Flutter				x	Disponible en todos los elementos de código con una respuesta instantánea.
			Ionic		x			Es limitado e inconsistente
CALIDAD EXTERNA								
			React Native				x	Instalación sencilla, en poco tiempo incluso para usuarios sin experiencia
Usabilidad	Requisitos previos de instalación	¿Cuán compleja es la Instalación	Xamarin			x		Relativamente sencillo aun con pasos y configuraciones técnicas (Teniendo en cuenta que no se pudo implementar de forma correcta la AR)
			Flutter				x	Instalación sencilla, en

Categoría	Métricas	Propósito	Puntaje					Interpretación del valor	
			Framework	1 Inaceptable	2 Pobre	3 Aceptable	4 Muy Bueno		5 Excelente
								poco tiempo incluso para usuarios sin experiencia	
			Ionic					x	Instalación sencilla, en poco tiempo incluso para usuarios sin experiencia
			React Native					x	Experiencia básica en el desarrollo en un tiempo aproximado de una a dos semanas.
	Facilidad de aprendizaje	¿Cuánto tiempo le tomo al usuario aprender a desarrollar en el framework?	Xamarin	x					Requiere tiempo extenso aproximadamente seis meses o más (Teniendo en cuenta que no se pudo implementar de forma correcta la AR)
			Flutter	x					Requiere tiempo extenso aproximadamente seis meses o más.
			Ionic	x					Requiere tiempo extenso

Categoría	Métricas	Propósito	Puntaje					Interpretación del valor	
			Framework	1	2	3	4		5
				Inaceptable	Pobre	Aceptable	Muy Bueno	Excelente	
									aproximadamente seis meses o más.
CALIDAD DE USO									
			React Native					x	Responde de manera rápida y confiable
		¿Cuánto tiempo tarda el equipo de soporte en responder a las consultas de los usuarios?	Xamarin	x					No responde a las consultas (Teniendo en cuenta que no se pudo implementar de forma correcta la AR)
			Flutter					x	Responde de manera rápida y confiable.
Apoyo	Soporte		Ionic			x			Responde en un tiempo razonable, aunque existe variabilidad en el tiempo de respuesta.
		¿Las respuestas proporcionadas por el equipo de soporte son claras, útiles y satisfacen las necesidades	React Native					x	Respuestas claras y útiles satisfaciendo a los usuarios de forma efectiva.
			Xamarin	x					Respuestas confusas, incoherentes o

Categoría	Métricas	Propósito	Puntaje					Interpretación del valor	
			Framework	1	2	3	4		5
				Inaceptable	Pobre	Aceptable	Muy Bueno	Excelente	
		de los usuarios?							poco claras (Teniendo en cuenta que no se pudo implementar de forma correcta la AR)
			Flutter					x	Respuestas claras y útiles satisfaciendo a los usuarios de forma efectiva.
			Ionic			x			Mayoría de respuestas útiles y claras.
			React Native					x	Capacitaciones o meetups disponibles frecuentemente presentado por expertos.
	Capacitación	¿Hay eventos de capacitación o meetups organizadas por la comunidad de usuarios que utilizan el framework?	Xamarin	x					Ninguna disponibilidad de capacitaciones o meetups (Teniendo en cuenta que no se pudo implementar de forma correcta la AR)

Categoría	Métricas	Propósito	Framework	Puntaje					Interpretación del valor
				1 Inaceptable	2 Pobre	3 Aceptable	4 Muy Bueno	5 Excelente	
			Flutter					x	Capacitaciones o meetups disponibles frecuentemente presentado por expertos.
			Ionic	x					Ninguna disponibilidad de capacitaciones o meetups.
			React Native					x	Documentación exhaustiva clara y altamente accesible.
Documentación	Documentación de referencia	¿Existe documentación completa y fácilmente accesible para el framework?	Xamarin			x			Documentación suficiente abordando aspectos relevantes (Teniendo en cuenta que no se pudo implementar de forma correcta la AR)
			Flutter					x	Documentación exhaustiva clara y altamente accesible.
			Ionic			x			Documentación suficiente abordando

Categoría	Métricas	Propósito	Puntaje					Interpretación del valor	
			Framework	1	2	3	4		5
				Inaceptable	Pobre	Aceptable	Muy Bueno	Excelente	aspectos relevantes.
			React Native				x		Tutoriales de referencia amplios y bien elaborados, proporcionando ejemplos prácticos.
		¿La documentación incluye ejemplos claros y guías paso a paso para ayudar a los desarrolladores?	Xamarin			x			Los tutoriales están disponibles y cubren características principales del framework (Teniendo en cuenta que no se pudo implementar de forma correcta la AR)
			Flutter				x		Tutoriales de referencia amplios y bien elaborados, proporcionando ejemplos prácticos.
			Ionic			x			Los tutoriales están disponibles y cubren características

Categoría	Métricas	Propósito	Framework	Puntaje					Interpretación del valor
				1	2	3	4	5	
				Inaceptable	Pobre	Aceptable	Muy Bueno	Excelente	principales del framework.

Nota. Resultado ordinal de evaluación sobre los diferentes frameworks.

Resultados de la evaluación

Después de un análisis exhaustivo de métricas cuantitativas y cualitativas relacionadas con entornos de desarrollo de aplicaciones móviles relacionados con la realidad aumentada (AR), llegamos a las conclusiones básicas:

Calidad interna:

React Native y Flutter se destacan por su capacidad para proporcionar una implementación integral de la funcionalidad AR y una respuesta instantánea a todos los elementos del código, como lo demuestran los datos cuantitativos recopilados. Por el contrario, la tabla cualitativa muestra limitaciones e inconsistencias entre Xamarin e Ionic en su capacidad para integrar y ejecutar de manera efectiva la funcionalidad AR.

Cualidades externas:

En términos de accesibilidad y facilidad de aprendizaje, React Native, Flutter e Ionic se destacan como opciones viables ya que son relativamente fáciles de instalar y usar. Sin embargo, es importante tener en cuenta que adquirir habilidades de Xamarin puede llevar una cantidad de tiempo significativa debido a la pronunciada curva de aprendizaje que se muestra en el análisis cualitativo.

Calidad de uso:

En términos de soporte técnico, React Native y Flutter fueron elogiados por brindar respuestas rápidas y claras a las preguntas de los desarrolladores, lo que hizo que el proceso de desarrollo fuera más fluido. Xamarin e Ionic, por otro lado, tienen tiempos de respuesta variables y a veces confusos, lo que puede obstaculizar el progreso del desarrollo.

Existen claras diferencias entre entornos en lo que respecta a documentación y formación. React Native y Flutter proporcionan recursos completos y maduros con tutoriales completos y documentación detallada para facilitar el proceso de aprendizaje y desarrollo. En comparación, tanto Xamarin como Ionic carecen de la formación adecuada y su documentación no es muy completa, lo que puede crear desafíos adicionales para los desarrolladores.

Teniendo en cuenta estos resultados, Flutter demuestra ser el entorno más favorable para el desarrollo de aplicaciones AR. No solo proporciona una implementación completa de la funcionalidad AR con respuesta inmediata a todos los elementos del código, sino que también presenta una instalación sencilla, una curva de aprendizaje asequible, soporte técnico confiable y documentación completa. Estas cualidades hacen de Flutter la mejor opción para las personas que desean desarrollar aplicaciones AR de manera eficiente y efectiva.

Capítulo VI

Conclusiones y Futuras Mejoras

Cumplimiento de Objetivos

Para lograr nuestros objetivos declarados, evaluamos y comparamos cuidadosamente entornos de desarrollo de realidad aumentada (AR) móvil:

Como parte del objetivo general, es posible evaluar y comparar las plataformas de aplicaciones móviles AR disponibles en el mercado, centrándose en el equilibrio entre la calidad interna, externa y la calidad de uso, de conformidad con los estándares establecidos por la norma ISO 25000 mediante la comparación cuantitativa y Indicadores cualitativos: Se ha realizado un análisis exhaustivo y se ha identificado el marco que mejor se adapta a las necesidades específicas de las aplicaciones móviles de detección de señales de tráfico para un propósito específico

Garantía de Calidad ISO 25000: Garantizar que las aplicaciones móviles cumplan con los estándares de calidad definidos por la norma ISO 25010. Esto implica alcanzar niveles óptimos de eficiencia, economía, mantenibilidad y facilidad de uso. Al seleccionar la plataforma más adecuada, hemos hecho el trabajo de garantizar que la aplicación entregue resultados precisos y confiables, sea fácil de usar y pueda mantenerse y mejorarse de manera efectiva en el futuro.

Establecimiento de un sistema de referencia: se ha establecido un sistema de referencia sólido para evaluar la calidad del desarrollo de aplicaciones de RA en el contexto de la seguridad vial. El marco proporciona una guía clara y detallada para futuros proyectos en esta área, estimulando así la investigación y el desarrollo tecnológico relacionados con la seguridad vial y la realidad aumentada.

Conclusiones

- La evaluación de entornos de desarrollo de aplicaciones móviles de realidad aumentada (AR) no sólo ayuda a seleccionar el entorno más adecuado para las aplicaciones móviles de detección de señales de tráfico, sino que también aporta varias contribuciones importantes:
- Elegir la mejor estructura: Las aplicaciones móviles para la detección de señales de tráfico se beneficiarán de elegir la estructura más adecuada basándose en una evaluación exhaustiva y cuidadosa. Esto garantiza que las aplicaciones estén respaldadas por tecnologías potentes y confiables que optimicen su rendimiento y funcionalidad desde una perspectiva de realidad aumentada para cumplir con requisitos específicos de calidad y rendimiento establecidos.
- Experiencia de usuario mejorada: la aplicación móvil resultante brindará a los usuarios una experiencia mejorada y agradable al interactuar con la realidad aumentada al detectar señales de tráfico. Al elegir un diseño que garantice una alta usabilidad y eficiencia, puede asegurarse de que los usuarios puedan usar su aplicación de manera intuitiva y efectiva, mejorando así su experiencia y aumentando la aceptación y adopción de la tecnología.
- Contribución al desarrollo tecnológico: Esta evaluación sienta un precedente importante para el desarrollo de aplicaciones de realidad aumentada en seguridad vial. Al proporcionar una orientación clara y detallada sobre la evaluación de la calidad en esta área específica, puede estimular el avance de la investigación y el desarrollo tecnológico, estimular proyectos futuros y contribuir al progreso general del campo de la realidad aumentada aplicada a la seguridad vial.

- Impacto social potencial: la aplicación móvil resultante podría tener un impacto positivo en la seguridad vial al proporcionar a los usuarios una herramienta eficaz para la detección y comprensión instantáneas de las señales de tráfico. Al aumentar la conciencia y el cumplimiento de las leyes de tránsito, puede ayudar a reducir los accidentes y mejorar la seguridad de los usuarios de la vía.

El aporte de la aplicación y evaluación del marco de desarrollo de aplicaciones móviles AR no solo tiene implicaciones técnicas y técnicas, sino también sociales y de seguridad vial, ayudando a desarrollar soluciones innovadoras que tengan un impacto positivo en la sociedad.

Recomendaciones

A pesar de los resultados positivos obtenidos al evaluar entornos de desarrollo de aplicaciones móviles de realidad aumentada (AR), es importante reconocer ciertas limitaciones e identificar áreas que podrían beneficiarse de futuras mejoras:

- **Limitaciones técnicas.** El proceso de evaluación identificó ciertas limitaciones técnicas inherentes a determinadas plataformas, como capacidades de realidad aumentada, rendimiento subóptimo en determinados dispositivos o dificultades de integración con otras tecnologías. Estas limitaciones pueden afectar la disponibilidad y eficacia de la Aplicación en determinadas circunstancias.
- **Complejidad de implementación.** Aunque se ha seleccionado un diseño óptimo para una aplicación móvil de detección de señales de tráfico, es importante darse cuenta de que la implementación eficaz de la realidad aumentada puede ser compleja y requerir habilidades técnicas específicas. Esto puede ser un desafío para los desarrolladores, especialmente aquellos con experiencia limitada en el campo.
- **Requiere actualizaciones constantes.** Dado el rápido desarrollo de la tecnología, es importante reconocer la necesidad de actualizar y mejorar constantemente las aplicaciones móviles y su infraestructura. Esto incluye agregar nuevas funciones, optimizar el rendimiento, corregir errores y adaptarse a los cambios en el entorno técnico y las necesidades del usuario.
- **Prueba empírica.** Aunque se han realizado evaluaciones rigurosas utilizando medidas tanto cuantitativas como cualitativas, es importante complementar estos resultados con estudios empíricos y pruebas de usuarios en entornos del mundo real. Estas pruebas adicionales brindan una comprensión más profunda de cómo

se desempeñan la aplicación y su diseño en situaciones del mundo real y ayudan a identificar áreas específicas que requieren atención.

- **Habilite funciones avanzadas.** En el futuro, sería beneficioso explorar y desarrollar funciones avanzadas de RA que puedan mejorar aún más la experiencia del usuario y la usabilidad de las aplicaciones. Esto podría incluir funciones como el reconocimiento de objetos 3D, la navegación AR en tiempo real o la integración con sistemas de asistencia al conductor.

Aunque la evaluación arrojó resultados positivos y proporcionó una buena base para el desarrollo de aplicaciones móviles de detección de símbolos de tráfico, es importante reconocer las limitaciones actuales y trabajar en áreas de mejora futuras para garantizar la eficacia continua de la solución y la correlación.

Bibliografía

- A, D. (2020, febrero 4). Qué es React: Definición, características y funcionamiento. *Tutoriales Hostinger*. <https://www.hostinger.es/tutoriales/que-es-react>
- About React Native – Welcome – React Native Course*. (s. f.). Recuperado 27 de febrero de 2024, de <https://hendrixer.github.io/nextjs-course/about-react-native>
- Análisis de capacidades, diferencias, roadmap y futuras posibilidades de Ionic, React Native and NativeScript*. (s. f.). Recuperado 27 de febrero de 2024, de <https://blog-es.mimacom.com/roadmap-ionic-react-ns/>
- ANGIE MARICELA C. is comparing Ionic, React Native, ComponentOne Studio for Xamarin, and Flutter with Android Studio*. (s. f.). G2. Recuperado 31 de enero de 2024, de <https://www.g2.com/assistant/53cab747-a7ea-4525-8d01-377c947a9d00>
- BRR. (2005). *A Proposed Open Standard to Facilitate Assessment and Adoption of Open Source Software*. Spike source. https://www.immagic.com/eLibrary/ARCHIVES/GENERAL/CMU_US/C050728W.pdf
- Cain Guambo, V. E. (2021). *Implementación de un sistema de alerta para la prevención de accidentes mediante reconocimiento de señales de tránsito y visión artificial* [bachelorThesis, Riobamba Universidad Nacional de Chimborazo]. <http://dspace.unach.edu.ec/handle/51000/7846>
- Chandra, E., Francis Xavier Christopher, D., & Vijaykumar, S. D. (2011). Study of CMMI based process framework for quality models. *3rd International Conference on Trendz in Information Sciences & Computing (TISC2011)*, 173-176. <https://doi.org/10.1109/TISC.2011.6169109>
- Chaudhary, P. (2018). Ionic framework. *Int. Res. J. Eng. Technol*, 5(05), 3181-3185.
- Cheng, G. (2023). Image Generation and Feedback Based on Deep Learning in Visual Communication Design. *2023 International Conference on Evolutionary Algorithms and*

Soft Computing Techniques (EASCT), 1-5.

<https://doi.org/10.1109/EASCT59475.2023.10393578>

Compare Flutter vs. React Native vs. Ionic vs. ComponentOne Studio for Xamarin. (s. f.). G2.

Recuperado 27 de febrero de 2024, de <https://www.g2.com/compare/componentone-studio-for-xamarin-vs-flutter-vs-ionic-vs-react-native>

Contreras, M. S., Barrios, E. M., & Rodriguez, J. C. (2019). Reconocimiento y rastreo de

imágenes en aplicaciones de Realidad Aumentada. *Revista ESPACIOS*, 40(34).

<https://www.revistaespacios.com/a19v40n34/19403407.html>

David Britch. (2024, febrero 13). *Actualización de Xamarin a .NET - .NET MAUI.*

<https://learn.microsoft.com/es-es/dotnet/maui/migration/?view=net-maui-8.0>

Dekkati, S., Lal, K., & Desamsetti, H. (2019). React Native for Android: Cross-Platform Mobile

Application Development. *Global Disclosure of Economics and Business*, 8, 153-164.

<https://doi.org/10.18034/gdeb.v8i2.696>

Flutter documentation | Flutter. (s. f.). Recuperado 28 de febrero de 2024, de

<https://docs.flutter.dev/>

G2. (2023, agosto 2). *About | G2 Culture.* G2 Business Software Reviews.

<https://company.g2.com/about>

Gordieiev, O., & Kharchenko, V. (2020). PROFILE-ORIENTED ASSESSMENT OF SOFTWARE

REQUIREMENTS QUALITY: MODELS, METRICS, CASE STUDY. *International Journal*

of Computing, 656-665. <https://doi.org/10.47839/ijc.19.4.2001>

Goth, G. (2005). Open source business models: Ready for prime time. *IEEE Software*, 22(6),

98-100. <https://doi.org/10.1109/MS.2005.157>

Introduction to Ionic | Ionic Documentation. (s. f.). Ionic Framework Docs. Recuperado 28 de

febrero de 2024, de <https://ionicframework.com/docs>

Iván Mauricio, Melo Bohórquez. (2018). *Realidad aumentada y aplicaciones | Tecnología*

Investigación y Academia. Vol. 6(Núm. 1).

<https://revistas.udistrital.edu.co/index.php/tia/article/view/11281>

Jiménez, S., & Juárez-Ramírez, R. (2019). A Quality Framework for Evaluating Grammatical Structure of User Stories to Improve External Quality. *2019 7th International Conference in Software Engineering Research and Innovation (CONISOFT)*, 147-153.

<https://doi.org/10.1109/CONISOFT.2019.00029>

Krause, D. (2005). Structures and Structural Realism. *Logic Journal of the IGPL*, 13(1), 113-126. <https://doi.org/10.1093/jigpal/jzi007>

Lang, H., & Zhuang, P. (2023). Design of Interactive Virtual System of Architectural Space Based on Multi-Objective Optimization Algorithm. *2023 IEEE International Conference on Image Processing and Computer Applications (ICIPCA)*, 1610-1614.

<https://doi.org/10.1109/ICIPCA59209.2023.10257849>

Ma, H., Liu, D., Yan, N., Li, H., & Wu, F. (2022). End-to-End Optimized Versatile Image Compression With Wavelet-Like Transform. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(3), 1247-1263. <https://doi.org/10.1109/TPAMI.2020.3026003>

Muñoz-Saavedra, L., Miró-Amarante, L., & Domínguez-Morales, M. (2020). Augmented and Virtual Reality Evolution and Future Tendency. *Applied Sciences*, 10(1), Article 1.

<https://doi.org/10.3390/app10010322>

Naik, N. (2022, abril 26). Flutter Architecture. *Medium*.

<https://medium.com/@niranjanikaik267/flutter-architecture-fb99d57e0a89>

Norma ISO/IEC 25000 | Tecnología Investigación y Academia. (2017).

<https://geox.udistrital.edu.co/index.php/tia/article/view/8373>

NORMAS ISO 25000. (s. f.). Recuperado 8 de noviembre de 2023, de

<https://iso25000.com/index.php/normas-iso-25000>

profexorgeek. (2023, julio 13). *¿Qué es Xamarin? - Xamarin*. <https://learn.microsoft.com/es-es/xamarin/get-started/what-is-xamarin>

Puig, J., Perkis, A., Lindseth, F., & Ebrahimi, T. (2012). Towards an efficient methodology for evaluation of quality of experience in Augmented Reality. *2012 Fourth International Workshop on Quality of Multimedia Experience*, 188-193.
<https://doi.org/10.1109/QoMEX.2012.6263864>

Rambhia, P., Shinde, P., & Bamane, K. (2023). Securing Flutter Applications: A Comprehensive Study. *2023 7th International Conference On Computing, Communication, Control And Automation (ICCUBEA)*, 1-5. <https://doi.org/10.1109/ICCUBEA58933.2023.10392001>

React Native · Learn once, write anywhere. (s. f.). Recuperado 28 de febrero de 2024, de <https://reactnative.dev/>

Redondo, D. A. (2012). Realidad aumentada. *España: Universidad Carlos III de Madrid*.

Reinoso, R. (2013). Introducción a la realidad aumentada. *Simo Network*.

Salazar, F., Manosalvas, C., Rodríguez, N., & Landeta-López, P. (2020). *Análisis de la eficiencia de desempeño en aplicaciones de Realidad Aumentada utilizando la normativa ISO/IEC/25010*.

Smartup, A. (2023, octubre 30). React Native, Flutter y Xamarin, tres frameworks de garantía. *Arangoya Centro Educativo*. <https://arangoya.org/frameworks-desarrollo-flutter-vs-react-native-vs-xamarin/>

Szczepanik, M., & Kedziora, M. (2020). *State Management and Software Architecture Approaches in Cross-platform Flutter Applications* (p. 414).
<https://doi.org/10.5220/0009411604070414>

Vaca, T., & Jácome, A. (2018a). *Calidad de software del módulo de talento humano del sistema informático de la Universidad Técnica del Norte bajo la norma ISO/IEC 25000*.

Vaca, T., & Jácome, A. (2018b). *Calidad de software del módulo de talento humano del sistema informático de la Universidad Técnica del Norte bajo la norma ISO/IEC 25000*.

- Vishal, K., & Kushwaha, A. S. (2018). Mobile Application Development Research Based on Xamarin Platform. *2018 4th International Conference on Computing Sciences (ICCS)*, 115-118. <https://doi.org/10.1109/ICCS.2018.00027>
- What is Xamarin? - Xamarin | Microsoft Learn*. (s. f.). Recuperado 28 de febrero de 2024, de <https://learn.microsoft.com/en-us/xamarin/get-started/what-is-xamarin>
- Winkler, S., Rangaswamy, K., Tedjokusumo, J., & Zhou, Z. (2007). Intuitive application-specific user interfaces for mobile devices. *Proceedings of the 4th international conference on mobile technology, applications, and systems and the 1st international symposium on Computer human interaction in mobile technology*, 576-582. <https://doi.org/10.1145/1378063.1378158>
- Xamarin Apps with Business Central integration – P.* (s. f.). Recuperado 27 de febrero de 2024, de <https://community.dynamics.com/blogs/post/?postid=a451f420-4656-4e7f-9cd4-6ea69d1314df>
- Yang, Y., Zhang, Y., Xia, P., Li, B., & Ren, Z. (2017). Mobile Terminal Development Plan of Cross-Platform Mobile Application Service Platform Based on Ionic and Cordova. *2017 International Conference on Industrial Informatics - Computing Technology, Intelligent Technology, Industrial Information Integration (ICIICII)*, 100-103. <https://doi.org/10.1109/ICIICII.2017.28>

Anexos