Universidad Nacional de Loja

Facultad de la Energía, las Industrias y los de Recursos Naturales No Renovables

Maestría en Ingeniería en Software

Automatización del proceso DevOps de aplicaciones desarrolladas en Flutter. Caso de aplicación: Solución para consulta de establecimientos del catastro turístico de la Ciudad De Loja

Trabajo de Titulación previo a la obtención del título de Magíster en Ingeniería en Software

AUTOR:
Jean Paul Mosquera Arévalo

DIRECTORES:
Ing. Edwin Rene Guamán Quinche, Mg.Sc.
Ing. José Oswaldo Guamán Quinche, Mg.Sc.

Loja - Ecuador
2023

Certificación

Loja, 24 de abril de 2023

Ing. Edwin Rene Guamán Quinche, Mg.Sc.
DIRECTOR DEL TRABAJO DE TITULACIÓN
Ing. José Oswaldo Guamán Quinche, Mg.Sc.
DIRECTOR DEL TRABAJO DE TITULACIÓN

CERTIFICAMOS:

Que hemos revisado y orientado todo proceso de la elaboración del Trabajo de Titulación denominado: **Automatización del proceso DevOps de aplicaciones desarrolladas en Flutter. Caso de aplicación: Solución para consulta de establecimientos del catastro turístico de la Ciudad De Loja**, previo a la obtención del título de **Magíster en Ingeniería en Software**, de autoría del estudiante **Jean Paul Mosquera Arévalo**, con cédula de identidad Nro. **1105082042**, una vez que el trabajo cumple con todos los requisitos exigidos por la Universidad Nacional de Loja para el efecto, autorizamos la presentación para la respectiva sustentación y defensa.

Firmado electrónicamente por:
EDWIN RENE GUAMAN QUINCHE

Ing. Edwin Rene Guamán Quinche, Mg.Sc
DIRECTOR DEL TRABAJO DE TITULACIÓN

Firmado electrónicamente por:
JOSE OSWALDO GUAMAN QUINCHE

Ing. José Oswaldo Guamán Quinche, Mg.Sc.
DIRECTOR DEL TRABAJO DE TITULACIÓN

Autoría

Yo, **Jean Paul Mosquera Arévalo**, declaro ser autor del Trabajo de Titulación y eximo expresamente a la Universidad Nacional de Loja y a sus representantes jurídicos de posibles reclamos y acciones legales, por el contenido del mismo. Adicionalmente acepto y autorizo a la Universidad Nacional de Loja la publicación del Trabajo de Titulación en el Repositorio Digital Institucional – Biblioteca Virtual.

Firma: 

Cédula de Identidad: 1105082042

Fecha: 03/05/2023

Correo electrónico: jean.mosquera@unl.edu.ec

Teléfono: 0978600824

Carta de autorización por parte del autor, para consulta, reproducción parcial o total y/o publicación electrónica de texto completo, del Trabajo de Titulación

Yo, **Jean Paul Mosquera Arévalo**, declaro ser autor del Trabajo de Titulación denominado: **Automatización del proceso DevOps de aplicaciones desarrolladas en Flutter. Caso de aplicación: Solución para consulta de establecimientos del catastro turístico de la Ciudad De Loja** como requisito para optar el título de **Magíster en Ingeniería en Software**, autorizo al sistema Bibliotecario de la Universidad Nacional de Loja para que con fines académicos muestre la producción intelectual de la Universidad, a través de la visibilidad de su contenido de la siguiente manera en el Repositorio Institucional.

Los usuarios pueden consultar el contenido de este trabajo en el Repositorio Institucional, en las redes de información del país y del exterior con las cuales tenga convenio la Universidad.

La Universidad Nacional de Loja, no se responsabiliza por el plagio o copia del Trabajo de Titulación que realice un tercero.

Para constancia de esta autorización, en la ciudad de Loja, a los tres días del mes de mayo de dos mil veintitrés.

Firma: 

Autor: Jean Paul Mosquera Arévalo
Cédula de identidad: 1105082042
Dirección: Loja, Ecuador
Correo electrónico: jean.mosquera@unl.edu.ec
Teléfono: 0978600824

DATOS COMPLEMENTARIOS:

Director del Trabajo de Titulación: Ing. Edwin Rene Guamán Quinche Mg. Sc.

Codirector del Trabajo de Titulación: Ing. José Oswaldo Guamán Quinche, Mg.Sc.

Dedicatoria

Este Trabajo de Titulación es dedicado:

A mis padres, Luis y Alicia quienes me han brindado su apoyo en cada etapa de mi vida, quienes me han inculcado valores y me han motivado a superarme. A mi madre quien me ha dado su apoyo incondicional y me ha acompañado siempre, buscando lo mejor para mí, también por inculcarme valores, virtudes y sobre todo motivarme a siempre dar lo mejor de mí y motivarme a superarme.

A mis hermanos, Alexandra, Carmen, Luis, quienes me han apoyado y han estado presentes durante varias etapas de mi vida.

A 3A, por su confianza y permitirme crecer profesionalmente, por brindarme la oportunidad de demostrar mis capacidades y motivarme siempre a dar lo mejor de mí, estoy muy agradecido ya que han sido primordiales para mi crecimiento profesional y personal.

A María de los Ángeles por su apoyo incondicional, su valiosa compañía que aprecio mucho y sobre todo motivarme a seguir con los proyectos.

A mis amigos por su apoyo y compañía.

A Balto, que me ha acompañado durante los trasnoches que han sido necesarias para lograr todo aquello que hasta el momento he obtenido.

A mis abuelos y demás familiares quienes han estado felices por cada logro obtenido durante mi vida.

A mis maestros que me han brindado su sabiduría, conocimiento el cual ha sido primordial para mi formación profesional.

Jean Paul Mosquera Arévalo

Agradecimiento

Quiero rendir mis agradecimientos:

A Edwin Guamán y José Guamán, por su apoyo, guía y consejos en el desarrollo del presente Trabajo de Titulación para que se desarrolle de forma idónea y efectiva.
A todos los docentes de la Maestría de Ingeniería en Software por su dedicación, enseñanzas y aportes en la formación como Magíster en Ingeniería en Software.
A Luis Chamba por su apoyo durante la Maestría para de este modo lograr este gran logro.
A mis padres, hermanos por su apoyo incondicional y preocupación.
A María de los Ángeles, por su motivación, apoyo incondicional y el aliento brindado en todos para concretar mis proyectos.
A mis amigos por su apoyo, y sus palabras de aliento.
A mis compañeros por compartir sus vivencias durante esta etapa y compartir en varios proyectos conocimientos y experiencias.

Sin duda son personas que llevaré siempre en mi corazón y estaré infinitamente agradecido.

Jean Paul Mosquera Arévalo

Índice de contenidos

Índice de tablas:

Índice de figuras:

Índice de anexos:

1. Título

Automatización del proceso DevOps de aplicaciones desarrolladas en Flutter. Caso de aplicación: Solución para consulta de establecimientos del catastro turístico de la Ciudad De Loja.

2. Resumen

La adopción de la integración y despliegue continuo es más recurrente en la industria del software debido a los múltiples beneficios entre los cuales se encuentran la resolución ágil de errores, agilidad del proceso de entrega o puesta a producción de las aplicaciones y reducir de forma notoria fallas. También se destaca la preferencia creciente de parte de la población por dispositivos móviles como smartphones. Esto ha motivado a la industria a construir soluciones multiplataforma siendo en la actualidad Flutter una gran alternativa en caso de requerirse una solución compatible con diversas plataformas. No obstante, si bien este framework permite la construcción de artefactos compatibles con plataformas como Android y iOS, existen limitantes al momento de la construcción principalmente en el caso de la plataforma iOS, esto se debe a que se requiere de entorno basado en el sistema operativo macOS para ejecutar el proceso de construcción y posterior distribución. Esto puede llegar a convertirse en una característica bloqueante en caso de que no se disponga de dicho entorno puesto en el presente caso no podría ejecutarse ni la fase de construcción y posterior distribución.

El presente trabajo busca la automatización del proceso DevOps para de este modo lograr optimizar los tiempos de entrega de soluciones desarrolladas con el Framework Flutter, además de permitir estandarizar el proceso de integración y entrega logrando así un proceso trazable y ordenado. Para ello se ha realizado el análisis de herramientas que permitan o faciliten este proceso y permitan cubrir la limitante previamente definida, además de ejecutar la configuración de las herramientas y finalmente evaluar el proceso de automatización propuesto para de este modo visibilizar la efectividad de la propuesta.

Gracias al proceso implementado se observó una reducción en un 85% del tiempo en el que se completa el proceso de entrega. Concluyendo así que la automatización del proceso DevOps además de permitir la estandarización y automatización de tareas, reduce de forma importante el tiempo y permite la inclusión de tareas que permitirían garantizar la calidad del producto y el código, a diferencia de ejecutar de forma manual el proceso.

Palabras claves: *DevOps, Flutter, CI/CD, automatización.*

2.1. Abstract

The adoption of continuous integration and deployment is more common in the software industry due to its multiple benefits, including agile error resolution, faster delivery or production of applications, and a significant reduction in failures. Additionally, there has been a growing preference for mobile devices such as smartphones among the population. This has motivated the industry to build cross-platform solutions, with Flutter currently being a great alternative for creating compatible solutions for various platforms. However, while this framework allows for the creation of artifacts compatible with platforms like Android and iOS, there are limitations when building primarily for the iOS platform. This is because a macOS-based environment is required to execute the build and subsequent distribution process. This can become a blocking feature if this environment is not available as in this case, neither the build nor the subsequent distribution phase could be executed.

The present work aims to automate the DevOps process in order to optimize the delivery time of solutions developed with the Flutter Framework, as well as to standardize the integration and delivery process, achieving a traceable and orderly process. To achieve this, an analysis of tools that allow or facilitate this process has been carried out, in addition to executing the configuration of the tools and finally evaluating the proposed automation process to visualize the proposal's effectiveness.

Thanks to the implemented process, an 85% reduction in the time it takes to complete the delivery process was observed. It can be concluded that automating the DevOps besides allowing the standardization and automation of tasks significantly reduces the time and enables the inclusion of tasks that ensure product and code quality, as opposed to manually executing the process.

Keywords: *DevOps, Flutter, CI/CD, automation.*

3. Introducción

La automatización del proceso DevOps durante los últimos años ha captado el interés de la industria de software y se ha convertido en una necesidad para empresas de este sector que buscan mejorar la calidad y velocidad de entrega de sus soluciones de software entre estos sistemas, aplicaciones, entre otras [1]–[3]. Otra tendencia que ha crecido exponencialmente es el desarrollo de aplicaciones móviles debido al uso masivo de smartphones, de este modo en [4] y [5] se reporta que el número de dispositivos móviles en Ecuador es equivalente al 92.3% de la población total del país, también se destaca que entre 2022 y 2023 el número de dispositivos incrementó un 4.2%, lo que demuestra la preferencia de los usuarios por dispositivos móviles principalmente smartphones. Gracias a la creciente preferencia por los usuarios por el uso de dispositivos móviles debido a su portabilidad y facilidad de uso se ha observado la aparición de varias herramientas o frameworks que permiten acelerar el desarrollo de este tipo de aplicaciones. En la actualidad entre este amplio espectro de opciones para el desarrollo multiplataforma, Flutter se ha constituido en una gran alternativa a la hora de buscar desarrollar aplicaciones para todas las plataformas [6].

Si bien Flutter ha permitido la posibilidad de lograr con un mismo código fuente la construcción de artefactos compatibles con las diversas plataformas móviles como son Android y iOS, existen limitaciones externas al framework que complican ciertos procesos entre ellos la construcción, una de estas limitantes es la dependencia de un entorno macOS para la construcción de artefactos compatibles con dispositivos con el sistema operativo iOS [9], [10]. Otro aspecto importante es el proceso y requerimientos de cada plataforma para generar los artefactos que se emplean para publicar y distribuir las aplicaciones, en el caso de iOS es mandatorio disponer de una cuenta de desarrollador para poder firmar las aplicaciones y generar los artefactos, mientras que en el caso de aplicaciones para dispositivos Android no se presenta esta limitación ya que la cuenta de desarrollador se emplea únicamente para la distribución de la aplicación, mientras que el proceso de firmado de aplicaciones se puede realizar de forma libre [9], [11].

El interés por la aplicación de DevOps en la industria ha logrado que se concentren los esfuerzos por la automatización de los procesos de construcción y entrega continua. Previo a ejecutar procesos de automatización deben considerarse diversos aspectos propios de la organización o solución que podrían convertirse relevantes al momento de

definir una herramienta para ejecutar este proceso de automatización y sobre todo las fases que formarán parte de este proceso, un ejemplo es la disponibilidad de la aplicación para dispositivos iOS. Así, la principal interrogante es la definición de una herramienta óptima que permita incluir las diversas fases que permitan la construcción y entrega continua que aborde la distribución a las principales plataformas móviles como son Android y iOS. Además de también considerar la inclusión de herramientas para garantizar la calidad del código, como son las herramientas de análisis estático de código.

Son múltiples los beneficios que trae la adopción de estas prácticas de integración y despliegue, entre los más relevantes se encuentran el aumento de satisfacción del cliente al poder resolver de forma ágil cualquier problema que presente la solución, además de agilizar las puestas a producción de las soluciones y reducir en gran medida grandes fallas que puedan presentarse al ejecutar varias tareas de forma manual que con la integración y despliegue continuo se realizan de forma automática [3], [7], [8].

En el presente Trabajo de Titulación (TT) se propone la automatización del proceso DevOps mediante el uso de herramientas cloud que sean compatibles con aplicaciones desarrolladas con el Framework Flutter. Para lograr el objetivo descrito se propone inicialmente analizar herramientas que puedan emplearse para la automatización del proceso de integración y entrega continua (CI/CD) de aplicaciones desarrolladas con Flutter, para lo cual se ha considerado establecer un cuadro comparativo. Una vez seleccionadas las herramientas se propone ejecutar la automatización del proceso CI/CD, para finalmente evaluar la propuesta implementada por medio el empleo del caso de aplicación propuesto el cual busca emplear la información del catastro turístico del Ministerio de Turismo del Ecuador para el diseño y desarrollo de una aplicación de consulta.

Si bien existen aproximaciones que abordan el proceso de automatización del proceso CI/CD de aplicaciones desarrolladas con Flutter, se destaca [12], el cual aborda la automatización del proceso CI/CD de una solución desarrollada con dicho framework, no obstante el proceso propuesto no aborda la construcción para plataformas móviles, la propuesta del autor es una aplicación web desarrollada con Flutter.

El alcance del presente TT se concentra en definir la automatización del proceso DevOps de aplicaciones desarrolladas con el Framework Flutter como se describen en los

objetivos propuestos, entre varias limitaciones que podrían presentarse se encuentra el acceso a un entorno macOS, aunque esta limitación es fácilmente mitigable.

Para finalizar, el documento se estructura en 5 secciones. La sección de Marco teórico detalla trabajos relacionados con el TT y también aborda conceptos que permitirán la comprensión de DevOps, Flutter y herramientas para la automatización CI/CD compatibles con Flutter. Seguidamente en la sección de Metodología se presenta el área de estudio y los procedimientos que permitirán el cumplimiento de los objetivos propuestos. En la sección resultados se presentan los hallazgos que surgieron de la ejecución de los procedimientos. Mientras tanto en la sección discusión se propone el análisis de varios puntos definidos en los resultados. Finalmente, en las conclusiones se presentan varias ideas que surgieron al completar el presente proyecto.

4. Marco teórico

4.1. **Estado del arte**

Son varios los trabajos que están alineados con el proceso de automatización del proceso DevOps o CI/CD de aplicaciones Flutter, mediante la ejecución de procesos de búsqueda en bases de datos científicas y motores de búsqueda académicos se han seleccionado varios trabajos, de los cuales se han analizado los que se encontraban alineados al ámbito de conocimiento del presente TT, así se encuentran los siguientes trabajos:

[12], en su trabajo proponen una solución que integre la construcción, prueba y despliegue automatizados para equipos que trabajan con metodologías ágiles. El caso de estudio que abordan se trata de una solución implementada en Node.js y Flutter. Para el proceso de automatización hacen uso de Gitlab y herramientas de Docker, los resultados obtenidos mencionados por los autores son sustanciales y concluyen que fue considerable el tiempo que se redujo al realizar la automatización. Destacando así que, de un proceso manual de varias horas, gracias a la automatización se redujo significativamente a un promedio de 3 a 4 minutos.

En un análisis comparativo entre dos frameworks populares para el desarrollo Cross Platform como son Flutter y React Native en [10] analizan varias características entre las cuales se tienen la eficiencia, efectividad, compatibilidad, crecimiento de la comunidad, documentación, arquitectura, productividad del desarrollador, soporte para la automatización del testing y CI/CD. En el último punto destacan que Flutter a comparación de React Native, posee más documentación relacionada al soporte de CI/CD, adicional de destacar que Flutter permite ejecutar operaciones mediante comandos, lo cual lo vuelve más flexible a lograr una integración CI/CD, de este modo se concluye que Flutter al momento del desarrollo de aplicaciones tiene más ventajas en comparación a React Native.

Del mismo modo [13] realiza el análisis comparativo de Frameworks para el desarrollo de aplicaciones cross platform, para ello se efectúa el análisis de diversas métricas entre ellas el soporte CI/CD, destacando al igual que los anteriores autores la facilidad de establecer CI/CD gracias a los comandos que provee Flutter para ejecutar diversas operaciones como son la construcción, pruebas, análisis de código, entre otras. El autor

concluye que Flutter es una gran elección para el desarrollo de aplicaciones cross platform.

También [14], realiza una comparativa de frameworks para el desarrollo de aplicaciones cross platform a diferencia de los anteriores autores, en este trabajo se aborda los siguiente frameworks como son React Native, Flutter, Xamarin y Ionic. El autor destaca que Flutter a diferencia de los otros Frameworks analizados permite configurar CI/CD con mínima configuración, adicional de las bondades provistas por la interfaz de línea de comandos, concluyendo al igual que los otros trabajos previos que hacen un análisis comparativo que Flutter se destaca entre los frameworks de desarrollo de aplicaciones cross platform.

[15] en cambio propone una canalización CI/CD para la entrega continua de aplicaciones móviles en la nube de Amazon Web Services. Para ello la solución móvil propuesta por el autor para el estudio propuesto se encontraba desarrollada con el framework Flutter. Para el proceso de automatización emplea los servicios de AWS, para el origen destaca AWS CodeCommit, para el proceso de compilación AWS CodeBuild, para el proceso de almacenamiento emplea un bucket de S3, para la ejecución de pruebas AWS DeviceFarm, AWS Lambda para la publicación, el autor destaca que el trabajar con los servicios IaaS de AWS es una excelente alternativa para empresas pequeñas y medianas puedan realizar este tipo de canalizaciones.

4.2. **Bases Teóricas**

En el presente apartado se analizan diversos conceptos para lograr comprender el ámbito de estudio del presente TT. Para ello se ha considerado abordar los conceptos presentados a continuación.

4.2.1. DevOps

DevOps puede concebirse como un enfoque de desarrollo de software que busca la integración, colaboración entre los equipos de desarrollo y operaciones de TI para de este modo lograr acelerar la entrega de software, además de buscar alta calidad y confiabilidad [16], [17]. Según [3] DevOps puede aplicar a múltiples escenarios de negocio entre los más relevantes se destacan:

- Automatización del ciclo de desarrollo, en donde el aplicar DevOps permitirá satisfacer las necesidades del negocio con mínima intervención manual, así se busca que los procesos de construcción y lanzamiento se puedan ejecutar de forma automática, rápida y con mínima intervención humana.
- Eficiente gestión de código fuente, esto se puede lograr mediante el empleo de sistemas de versionamiento en los que se gestionan comentarios, documentación, código, entre otros insumos.
- Gestión de configuración consistente, para lo cual se busca que el proceso de desarrollo, prueba y construcción se ejecutan sobre una misma plataforma, es decir los despliegues funcionan de forma homogénea tanto en ambientes de desarrollo como productivos.
- Preparación del producto para el mercado, al ejecutar muchos procesos manuales, tener compromisos y fechas de entrega impredecibles el riesgo de incumplimiento aumenta. Al aplicar DevOps en este ámbito se lograrían entregas periódicas y a tiempo gracias a la automatización de varias tareas manuales, además de dar garantías en determinados procesos controlados los cuales se llevarían a cabo mediante el empleo de herramientas, entre ellos las pruebas y análisis de código, por ejemplo.
- Automatización de procesos manuales, cuando se ejecutan procesos manuales es muy probable que se produzcan errores durante su ejecución, lo cual dé lugar a que se produzcan cuellos de botella y así el proceso de distribución se vea truncado. Al aplicar DevOps se lograría mayor confidencia y rapidez al ejecutar procesos de forma automática apoyados con herramientas.

A. Ciclo de vida de software en DevOps

El ciclo de vida del software en DevOps se caracteriza por ser un proceso continuo el cual se focaliza en la entrega continua de software de calidad y confiabilidad. Para ello se tienen presentes las fases de planificación, desarrollo, integración, pruebas, implementación y monitoreo, las cuales se describen a continuación [18] :

- Planificación, esta etapa se definen los objetivos del proyecto, se establece la hoja de ruta y actividades. En la fase de planificación es primordial la colaboración de los equipos en el establecimiento de los aspectos previamente descritos.

- Desarrollo, en la presente etapa el equipo de desarrollo crea el software a partir de los requisitos y actividades establecidas en la etapa previa. En esta fase los equipos buscan que el software cumpla con los criterios establecidos y esté alineado a los objetivos del proyecto.
- Integración, en esta etapa se integran las diversas partes desarrolladas previamente, se valida que se integren de forma idónea y no se produzcan errores, en la presente fase los equipos trabajan de forma alineada para que el software sea compatible en todos los ambientes.
- Pruebas, en esta etapa se ejecutan pruebas automatizadas para asegurarse que el software funciona de forma idónea, en esta fase la colaboración de los equipos se concentra en detectar y corregir los errores de forma eficiente.
- Implementación, durante esta etapa el software se implementa en el entorno de producción, donde se hace uso de herramientas de automatización para desplegar de forma rápida y eficiente el software.
- Monitoreo, una vez desplegado el software es importante ejecutar el monitoreo en el entorno de producción con el objetivo de detectar errores y corregirlos antes de que generen un gran impacto, para lo cual se emplea herramientas para capturar errores y monitorización de la aplicación. Durante esta fase se busca asegurar que el software funcione normalmente.

B. Prácticas y herramientas de DevOps

Para [8] las prácticas y herramientas de DevOps se constituyen en un pilar fundamental para implementar de forma exitosa este enfoque de desarrollo de software, no obstante su aplicación estará ligada a la naturaleza de cada proyecto, entre varias prácticas y herramientas se encuentran. :

- Infraestructura como código (IaC): Esta práctica consiste en definir la infraestructura de TI mediante el empleo de código, entre ejemplos de infraestructura tenemos servidores, ambientes, redes, almacenamiento. El empleo de esta práctica permite que los equipos puedan automatizar la creación, configuración y gestión de la infraestructura, logrando de este modo acelerar el proceso de entrega de software. Un ejemplo de este tipo de herramientas es Terraform, una herramienta que permite definir la

infraestructura de forma declarativa y automatizada mediante código tanto en ambientes cloud, como en infraestructura local [19].

- Automatización de pruebas y despliegues: Mediante el empleo de herramientas de automatización de pruebas y despliegues se optimiza el tiempo que lleva la ejecución manual de estos procesos, así también se reduce de forma significativa la aparición de errores y fallas.

  Entre las herramientas que permiten ejecutar estos procesos se encuentran: Jenkins, esta es una herramienta de automatización de código abierto que permite ejecutar el proceso de pruebas y despliegue de software [20]. Otra herramienta cloud popular que permite automatizar estos procesos es Codemagic, que se caracteriza por ser servicio cloud que permite automatizar los procesos de construcción, prueba y despliegue de aplicaciones móviles, además de proveer la capacidad de ejecutar los procesos mediante entornos específicos por ejemplo un entorno macOS [21].
- Monitoreo y métricas: El empleo de herramientas de monitoreo y métricas permite a los equipos el monitoreo de rendimiento, disponibilidad, aparición de errores en producción, buscando así la detección, corrección de fallas se ejecute de forma proactiva y rápida en caso de que se detecten en el entorno productivo. Existen varias herramientas que permiten ejecutar el monitoreo, entre ellas tenemos a Prometheus que es una herramienta de código abierto que permite recopilar métricas [22]. Otra herramienta para el monitoreo es Sentry, el cual es un servicio cloud que busca detectar problemas en tiempo real, entre otras herramientas similares a Sentry tenemos Firebase Crashalytics [23] .
- Gestión de configuración y versionamiento: Mediante el empleo de herramientas de gestión de la configuración y versionamiento los equipos pueden gestionar el código fuente, configuraciones y cambios de forma controlada y automatizada. Varias son las herramientas que permiten la configuración y versionamiento entre ellas tenemos a Git, el cual es una herramienta de control de versiones que permite el trabajo colaborativo y llevar un registro de cambios [24]. En lo que concierne a la configuración tenemos a Ansible que es una herramienta que automatiza el despliegue y configuración de aplicaciones y sistema.

- Comunicación y colaboración: El empleo de herramientas de comunicación y colaboración entre los equipos es fundamental para lograr el éxito de DevOps, ya que mediante estas herramientas se logra que los equipos trabajen de forma coordinada, eficiente y colaborativa. En lo relacionado a herramientas de comunicación y comunicación tenemos a Slack, la cual es una herramienta de comunicación que permite la comunicación y colaboración en tiempo real, otras herramientas similares son Google Chat/Meet, Microsoft Teams, además de canales de notificación vía correo electrónico [7].

C. Casos de éxito y tendencia

- Casos de éxito en la industria:

Las grandes empresas se han constituido como promotores en la aplicación de este enfoque de desarrollo, entre los casos más sobresaliente según [25] se encuentran:

- Amazon, gracias a la implementación de prácticas DevOps Amazon implementa código en promedio de 11,7 segundos. También gracias al enfoque ágil las interrupciones se vieron reducidas, así como el tiempo que estas se producían permitiendo así mayor disponibilidad de servicio e ingresos.
- Netflix, mediante la implementación de DevOps en la actualidad los ingenieros implementan código miles de veces al día, destacando la adopción de nuevas tecnologías y la implementación del enfoque DevOps.
- Walmart, el equipo de tecnología de Walmart llamado WalmartLabs implementó en la nube llamada OneOps, la automatización del proceso DevOps logrando así acelerar la implementación de aplicaciones. OneOps está formado por 100.000 núcleos de OpenStack, siendo esta una iniciativa de computación en la nube para proporcionar una infraestructura como código.
- Nordstrom, la aplicación de DevOps en esta empresa constituyó en una reducción de errores y aumento del rendimiento de las soluciones, adicionalmente los despliegues que antes se realizaban dos veces al año pasaron a ser mensuales.

- Tendencias de DevOps:

El futuro de DevOps es prometedor, se prevé para el 2025 que el tamaño del mercado de DevOps alcance los 12.850 millones de dólares anuales de ingresos, esto gracias a la creciente aceptación y adopción de tecnologías Cloud, metodologías ágiles y digitalización esto según destaca la consultora Grand View Research [7], [26], [27]. Varias son las tendencias que se auguran para DevOps entre ellas tenemos:

- Aprendizaje Automático e inteligencia artificial dentro del marco de DevOps, la adopción de inteligencia artificial dentro de las canalizaciones de DevOps puede habilitar a que estas se ejecuten de una forma más óptima, así han surgido nuevos enfoques entre los cuales se encuentran AiOps y DataOps los cuales concentran sus esfuerzos en la utilización de Machine Learning e Inteligencia artificial para obtener valor de los registros y métricas para de este modo impulsar el proceso DevOps de forma idónea.
- Nuevas herramientas para la automatización, en la actualidad la aplicación de la automatización de varios procesos como parte del marco de DevOps ha traído lugar al desarrollo de soluciones enfocadas en la automatización de varios procesos, así como a una configuración más simple y rápida, de este modo se ha visto la aparición de plataformas como servicio (PaaS) enfocadas en la automatización DevOps.
- GitOps, esta práctica en la actualidad está ganando gran relevancia, la cual consiste en combinar dos tecnologías populares como son Git y Kubernetes con el objetivo de gestionar la infraestructura y la entrega continua [28].
- DevSecOps, es una enfoque de desarrollo que incorpora la seguridad desde el principio del ciclo de vida del software, en otros términos vendría a convertirse en un enfoque DevOps que integra la seguridad, con el objetivo de desarrollar software seguro [2].
- Computación sin servidor, es un modelo de computación en la nube el cual permite a los desarrolladores la creación y ejecución de aplicaciones sin necesidad de la gestión de infraestructura. En la actualidad existen varios proveedores de servicios en la nube entre los cuales se encuentra: Amazon Web Services, Microsoft Azure, Google Cloud Platform, servicios que escalan y administran de forma automática la infraestructura acorde a las necesidades de la aplicación [29].

4.2.2. Flutter

Flutter es un framework de desarrollo de aplicaciones Cross Platform o multiplataforma cuyo origen se remonta al año 2018. Este framework está apoyado por Google y en la actualidad ha logrado gran importancia en el desarrollo de este tipo de aplicaciones debido a que permite que estas puedan emplearse de forma nativa en las diversas plataformas. En la actualidad es posible el despliegue para las siguiente plataformas: Linux, Windows, MacOs, Android, iOS, Android, Web con un solo código base [6], [30].

Entre varias bondades que trae consigo Flutter se encuentran, la rapidez y reducción de tiempo de desarrollo de aplicaciones, la mantenibilidad debido a que la interfaz de usuario y lógica no cambian según la plataforma y el rendimiento es similar a una aplicación nativa. Flutter además se caracteriza ya que provee un motor de renderizado separado llamado SKIA, el cual permite la posibilidad de desplegar a casi a todas las plataformas existentes en la actualidad [11], [31], [32].

Varias son las características que destacan a este framework entre ellas tenemos las siguientes:

- Rápido desarrollo de aplicaciones, debido a que provee la función de hot reload, dicha función permite ver los cambios en tiempo real, es decir Flutter posee un compilador de código en tiempo de ejecución que acelera el proceso de desarrollo [30].
- Diseño flexible y personalizado, el framework provee una amplia gama de widgets que pueden ajustarse o adaptarse a cualquier interfaz de usuario o diseño. Además de permitir crear widgets a partir de los widgets base en función a las necesidades de los desarrolladores [32].
- Alta velocidad y rendimiento, el lenguaje de programación usado por Flutter es Dart, el cual es capaz de compilar a código nativo, esto permite que las aplicaciones tengan mayor velocidad y rendimiento, además de que los widgets provistos por el framework se encuentran optimizados para aportar en el buen rendimiento [11].
- Compatibilidad, gracias a Flutter es posible crear aplicaciones con un solo código fuente para múltiples plataformas en las que se tienen iOS, Android, web, Desktop [11].

A. Casos de éxito y tendencias

- Casos de éxito de aplicaciones desarrolladas con Flutter:

Flutter en la actualidad ha sido empleado por muchas empresas para desarrollar aplicaciones móviles, web, de calidad. Algunos de los casos de éxito más destacados son los siguientes:

- BMW, en el presente caso de éxito, BMW empleó Flutter para el desarrollo de una nueva aplicación la cual fue lanzada en 47 países en julio de 2020. Esta aplicación buscaba ser amigable para el usuario, segura y confiable. Adicionalmente a ello el grupo de desarrollo de BMW estableció una plataforma llamada Mobile 2.0 Platform, que buscaba la automatización del proceso de construcción, pruebas, y despliegue de la aplicación móvil y todas sus variantes [33].
- PUBG Mobile, el presente caso de éxito se trata de un videojuego desarrollado con Flutter, el cual dispone de 1 billón de jugadores, y según detalla el equipo de desarrollo que gracias a Flutter con un equipo de desarrollo reducido pudieron desplegar el juego tanto para sistemas operativos iOS y Android. También se destaca la curva de aprendizaje del framework el cual se caracteriza por la rapidez de aprendizaje [34].
- Google Pay, en el presente caso de estudio el equipo de ingeniería de Google decidió emplear Flutter debido a las ventajas que trae a la hora de agregar más funciones a la solución. También se destaca que el código base se redujo en un 35% en comparación a la solución inicial, permitiendo esto al equipo un mantenimiento más rápido de la solución. El resultado logrado con Flutter fue una solución más eficiente y compatible con Android y iOS [35].
- NuBank, es un banco digital que tiene alrededor de 48 millones de clientes, este banco buscaba un framework que permita a los equipos trabajar de forma autónoma desarrollando la aplicación móvil y entregando valor con una sola arquitectura, lenguajes y convenciones. Fueron varios factores los que analizaron antes de elegir la tecnología, entre ellos estaban la curva de aprendizaje, riesgo no lineal de abstracción, costo incremental de abstracción, entre otros, de este modo decidieron elegir Flutter para el desarrollo de esta aplicación bancaria.

  Los resultados obtenidos al emplear Flutter fueron significativos y de este modo lograron una solución escalable sin sacrificar la calidad, además de reducir los

tiempos de entregas de nuevas funciones y sobre todo el tiempo para el lanzamiento de un nueva función se redujo de un año, a tan solo tres meses [36].

- Rive, es una aplicación que permite el diseño de animaciones y luego publicarlas en cualquier plataforma. El uso de Flutter en el desarrollo de esta plataforma se dio debido al soporte de Flutter de CanvasKit y su soporte multiplataforma. Así se logró una solución con un gran rendimiento [37].

4.2.3. Herramientas para la automatización CI/CD compatibles con Flutter.

Existen varias herramientas en el mercado para ejecutar procesos de automatización CI/CD, no obstante, la mayor parte de estas son únicamente compatibles con ciertos frameworks y lenguajes de programación. Para establecer las herramientas compatibles con el framework Flutter se ha hecho un análisis exploratorio siendo así las herramientas más populares las siguientes:

A. **Jenkins,** es una herramienta de automatización para la integración continua y entrega continua, es open source y generalmente es empleada para automatizar el proceso de construcción, pruebas y distribución de software [20].
B. **GitLab DevOps,** es una plataforma provista por Gitlab la cual incluye la gestión de repositorios, integración continua y entrega continua, también dispone de herramientas de colaboración, monitoreo y gestión de proyectos, esta herramientas es de pago por uso, no obstante dispone de un plan gratuito [24].
C. **Codemagic,** es una plataforma de integración y entrega continua inicialmente enfocada para proyectos de Flutter, actualmente soporta despliegues para otros Frameworks como React Native. Esta plataforma permite automatizar el proceso de construcción, prueba y distribución de aplicaciones, también permite ejecutar el análisis estático de código, diversas integraciones con otras herramientas y otras funciones de seguimiento. Es de pago por uso, aunque dispone de un plan gratuito flexible [21].
D. **Bitrise,** es una plataforma que permite la automatización del proceso de integración y entrega continuas de aplicaciones móviles. Esta herramienta permite la automatización del proceso de construcción, prueba y distribución de aplicaciones, tiene soporte para varios frameworks entre ellos Flutter, posee varios planes de paga y un plan gratuito [38].

E. **Github Actions,** es una plataforma para la automatización del proceso de integración y entrega continúa provista por Github, la cual permite automatizar el proceso de construcción, prueba y entrega de software. Admite variedad de lenguajes de programación y frameworks, entre ellos Flutter, es gratuita para proyectos públicos u open source, en caso de repositorios privados posee ciertas características de paga [39].

F. **Firebase,** se constituye en una plataforma que ofrece servicios para el desarrollo de aplicaciones móviles, entre los cuales se encuentran servicios para autenticación de usuarios, almacenamiento, base de datos, distribución de aplicaciones, monitoreo de aplicaciones, entre otros. Posee un plan gratuito mediante el cual se puede ejecutar varias operaciones [40].

G. **App Store Connect,** es la Plataforma de Apple para desarrolladores de aplicaciones para dispositivos Apple, como iPhone, iMac, iPad, entre otros. Mediante esta plataforma se puede gestionar las aplicaciones y distribuirlas a la App Store o bien para pruebas a TestFlight. Ofrece entre otras funciones, un servicio de análisis de métricas de las aplicaciones. Para hacer uso de esta plataforma es necesario adquirir una licencia anual de desarrollador cuyo valor asciende a 100 dólares americanos [41].

H. **Google Play Console,** es la plataforma de Google para desarrolladores de aplicaciones Android. Mediante esta plataforma los desarrolladores pueden gestionar y distribuir sus aplicaciones para que de este modo se encuentren disponibles en Google Play Store, ofrece otras funciones como métricas de las aplicaciones. Para poder hacer uso de esta plataforma debe adquirirse una licencia de por vida cuyo valor asciende a 25 dólares americanos [42].

I. **YAML,** es un lenguaje de serialización de datos que suele emplearse para definir archivos de configuración. Este lenguaje cuenta con funciones de lenguajes como Perl, C, XML, HTML, entre otros. Se fundamenta en JSON y emplea sangría al estilo del lenguaje de programación Python para incorporar un elemento dentro de otro. Estos archivos generalmente usan la extensión yml o yaml. En el caso de codemagic se emplea este lenguaje para definir los pipelines en forma de script [43], [44].

J. **SonarQube,** se constituye en una plataforma de análisis de calidad de código la cual permite mejorar la calidad y seguridad de código de las aplicaciones, esta

herramienta permite detectar errores, vulnerabilidades y malas prácticas a nivel del código [45].

5. Metodología

Para el desarrollo del TT en el presente inciso se describe el área de estudio, procedimiento y actividades complementarias definidas para lograr cumplir los objetivos propuestos. De este modo a continuación se presentan los aspectos antes indicados.

5.1. Área de estudio

El ámbito de conocimiento que comprende el presente TT aborda las áreas relacionadas a DevOps y el desarrollo de aplicaciones con el Framework Flutter, para ello se ha considerado el análisis referente al ámbito antes descrito considerando trabajos, herramientas, entre otros elementos de actualidad. Se ha considerado para la evaluación de la propuesta que resultará del presente TT el desarrollo de una solución basada en el catastro turístico del Ecuador.

Entre los materiales software empleado en el proceso de ejecución del presente TT se emplearon una Laptop, software como Visual Studio Code, Google Chrome, Flutter/Dart, y la información pública del catastro turístico disponible en la página de este organismo.

A continuación, se detalla el procedimiento para cumplir cada uno de los objetivos propuestos en el presente TT.

5.2. Procedimiento

5.2.1. Objetivo 1:

Para el análisis de herramientas que pueden emplearse para automatizar el proceso CI/CD de aplicaciones desarrolladas con el Framework Flutter se consideró realizar un estudio comparativo para la selección y establecimiento de herramientas para la automatización, de esta forma se empleó un análisis comparativo, el cual consiste en la recolección y análisis de información para comparar dos o más objetos. Para el presente caso se consideró herramientas populares para la automatización del proceso CI/CD, tanto open source como servicios o herramientas en la nube, como requerimiento esencial se tiene la compatibilidad con Flutter. Para ello se establecieron varios criterios para ejecutar la evaluación de las herramientas los cuales son descritos a continuación:

- Planes, en este criterio se ponderó si la herramienta que es objeto de análisis posee plan gratuito, es de libre acceso, o bajo suscripción. La ponderación comprenderá valores entre el 1 y 10, siendo 1 la inexistencia de planes gratuitos y 10 cuando sea de libre acceso sin costo a todas las funciones de la herramienta.
- Máquina virtual, con este criterio se ponderó las características de la máquina virtual provista por la herramienta, de este modo en función a las características se pondera del 1 al 10.
- Procesos en paralelo, este criterio evalúa la capacidad de ejecutar procesos en paralelo de parte de la herramienta, de este modo se pondera entre el 1 al 10.
- Compatibilidad con Framework, en el presente criterio se analizó aspectos relacionados con la compatibilidad del framework que es objeto de estudio, el cual es Flutter. Para ello se pondera entre el 1 al 10, siendo 1 la incompatibilidad y 10 la compatibilidad total de la herramienta con Flutter.
- Nivel de personalización de pipeline, en el presente criterio se analizó la capacidad de la herramienta para poder establecer pipelines con tareas complejas o adicionales. De este modo se pondera entre el 1 y el 10, representando 1 la imposibilidad de generar pipelines personalizados y 10 la flexibilidad de la herramienta en la creación de pipelines personalizados.
- Capacidad de integración con Google Play y App Store, mediante el presente criterio se analizó la complejidad al momento de configurar las integraciones para el despliegue de las aplicaciones en las tiendas de aplicaciones, de este modo acorde al nivel de complejidad para integrar se pondera entre el 1 al 10.
- Facilidad para definir pipeline, en el presente criterio se analizó la facilidad que provee la plataforma para generar un pipeline simple, de esta forma se considera configuraciones adicionales, rapidez y facilidad para la creación. Así 1 comprendería un proceso complejo para crear un pipeline sencillo y 10 un proceso ágil y fácil.
- Integraciones, en el presente criterio se analizó las integraciones que pueden ejecutarse en la herramienta. Para ello se pondera entre el 1 y 10 según la flexibilidad de integración de las herramientas.
- Documentación, en el presente criterio se evaluó la documentación existente para ejecutar la documentación De este modo se considerará la documentación oficial de la herramienta, siendo 1 la inexistencia de documentación y 10 la presencia de documentación completa.

- Capacidad para trabajar en equipo, en el presente apartado se analizó si la herramienta permite la colaboración, es decir es fácil que varios miembros de un equipo puedan acceder a la herramienta para consultar o configurar diversos aspectos.

Para la definición de la herramienta a emplearse se tuvo presente la que obtenga la mayor ponderación. Siendo de este modo guiado el proceso de automatización mediante la herramienta que resultó seleccionada.

Para el caso de herramientas adicionales se consideró el empleo de alternativas populares, es por ello por lo que solo se ejecutó la evaluación para definir la herramienta de automatización y no para definir las herramientas auxiliares, ya que el eje central del presente TT es generar el proceso de automatización.

5.2.2. Objetivo 2:

Para realizar la automatización del despliegue e integración continua CI/CD mediante la aplicación de las herramientas seleccionadas. Con la definición de las herramientas en el objetivo 1 se ejecutan las siguientes fases para lograr la automatización del proceso DevOps. De este modo se tienen las siguientes fases:

A. Configuración de herramientas:

Definidas las herramientas es necesario la configuración de estas para poder ejecutar su integración como parte de la automatización. De este modo es importante ejecutar la configuración de Codemagic, Gitlab, App Store Connect, Google Play Console, Firebase, Sonarcloud el proceso de configuración de estos servicios y herramientas se describe desde el Anexo 2 al Anexo 7. Este proceso es mandatorio ya que al momento de automatizar se requieren de insumos como tokens, llaves, entre otros.

B. Creación del pipeline

Con las configuraciones ejecutadas es posible la generación del pipeline, de este modo se describe el proceso y consideraciones propias de Codemagic al momento de la definición de la configuración del pipeline.

Codemagic permite la creación del pipeline mediante su editor gráfico y también mediante un archivo de configuración o script yaml. En el presente caso se ha optado por la definición del pipeline mediante el script yaml debido a la flexibilidad que permite al momento de ejecutar ciertas integraciones. Además de que el editor gráfico no permite realizar todas las integraciones soportadas por Codemagic según lo descrito en la documentación. En la construcción del pipeline se emplearon principios DRY.

Se definió un workflow para cada una de las ramas principales entre las cuales tenemos dev, testing y master. Para cada workflow se estableció una estrategia de triggering, es decir cuando se ejecute una operación de push, el workflow se ejecutará de forma automática según la rama que reciba dicha operación.

Los workflows tienen fases en común como por ejemplo el análisis estático de código, construcción de artefactos, publicación, las cuales difieren en cómo se publican cada uno de los artefactos.

C. Validación de pipeline

Una vez establecido el pipeline es importante comprobar el correcto funcionamiento, de este modo se recomienda ejecutar el pipeline de forma integral con el objetivo de corroborar su funcionamiento. Codemagic provee un dashboard mediante el cual puede visualizarse el estado de ejecución del pipeline, además de notificar de la fallida o exitosa ejecución vía email.

5.2.3. Objetivo 3:

Para evaluar el proceso de CI/CD implementado, se empleó la solución para la consulta de establecimientos del catastro turístico. Se han considerado tener en cuenta diversos escenarios para probar el correcto funcionamiento de las reglas y diversos aspectos establecidos en el pipeline. Entre varias métricas que se pueden analizar para evaluar el proceso CI/CD según [46] se tienen:

- Calidad del código, la cual se puede medir mediante métricas como por ejemplo el número de líneas de código, complejidad ciclomática, índice de mantenibilidad.

- Número de problemas en producción, esta métrica permite medir el número de problemas surgidos luego de cumplirse la implementación mediante el pipeline.
- Porcentaje de pruebas automatizadas: esta métrica se basa en el porcentaje de pruebas automatizadas ejecutadas en el pipeline CI/CD, cuanto mayor sea este porcentaje, menor será el número de errores en producción.
- Tiempo promedio entre implementaciones, esta métrica permite medir el tiempo promedio que transcurre entre dos implementaciones, es importante de medir este valor ya que es importante minimizar el tiempo y garantizar que los cambios se implementen de forma rápida y eficaz.

De este modo mediante el caso de aplicación el cual aborda una aplicación de consulta de establecimientos turísticos en función al catastro turístico del Ministerio de Turismo del Ecuador. Esta información es de carácter público y puede encontrarse en el siguiente sitio web siguiente https://servicios.turismo.gob.ec/portfolio/catastro-turistico-nacional y se destaca por disponer información de aproximadamente 21000 establecimientos a nivel nacional, de diversos tipos como por ejemplo hoteles, restaurantes, bares, entre otros [47].

Para el desarrollo del caso de aplicación se consideró la aplicación de Scrum para lograr la definición de esta aplicación. Se optó por esta alternativa debido a la necesidad de buscar un proceso de mejora continua que permita ir logrando en el proceso versiones funcionales de la solución propuesta.

El factor que se ha determinó para evaluar la eficiencia del pipeline es el tiempo que lleva lograr ejecutar todo el proceso teniendo presentes a todas las fases necesarias para publicar una aplicación en Google Play Console y App Store Connect. Para ello se llevó a cabo la ejecución del proceso manual y mediante el pipeline con el objetivo de determinar en que se destaca un proceso del otro.

Se tuvieron presentes los escenarios que se describen a continuación para la evaluación del pipeline y proceso manual:

- Generación y distribución de artefactos a Google Play Console y App Store Connect, cuando se trata de una nueva versión de aplicación.
- Generación y distribución de artefactos a Google Play Console y App Store Connect, cuando se trata de una corrección.

Para la ejecución de estos escenarios se consideró ejecutar el proceso en un equipo con las siguientes características: procesador M2 PRO, 16 Gb de RAM, debido a que para el despliegue de aplicaciones para iOS es mandatorio disponer de un equipo con sistema operativo macOS. También se consideró buscar ejecutar este proceso de benchmarck en entornos similares al considerar que Codemagic trabaja bajo un entorno macOS.

Para ejecutar el proceso de evaluación manual se consideró abordar las fases del pipeline contenidas en el Flujo de trabajo de producción. Este flujo se constituye en el más crítico ya que distribuye las versiones testeadas y aprobadas para su publicación en las tiendas de aplicaciones, abordando de este modo las siguientes fases:

- Obtención de código fuente.
- Configuración de entornos.
- Medición de calidad de código.
- Validación de que se cumple los Quality Gates.
- Construcción de aab para Android.
- Construcción de ipa para iOS.
- Publicación de aab en Google Play Console.
- Publicación de ipa en App Store Connect.

Así se obtuvieron los tiempos de ejecución importantes para definir la optimización del pipeline, además de surgir consideraciones adicionales que fueron detectadas en el proceso seguido para ejecutar cada una de las fases.

6. Resultados

6.1. Objetivo 1:

Para la definición de las herramientas que se evaluaron se han considerado aquellas herramientas que tienen soporte con el framework Flutter, de este modo en el apartado 4.2.3 se han establecido las herramientas que son compatibles y fueron objeto de análisis y evaluación.

A continuación, en la Tabla I se muestran los resultados obtenidos luego de ejecutar la evaluación de las herramientas con las métricas descritas en el procedimiento propuesto para el presente objetivo.

TABLA I
ANÁLISIS COMPARATIVO DE HERRAMIENTAS PARA LA AUTOMATIZACIÓN CI/CD COMPATIBLES CON FLUTTER

Métrica	**Jenkins**	**Gitlab Devops**	**Codemagic**	**Bitrise**	**Github Actions**
Precio y Planes	10- Open Source	9	9	7	8
Máquina Virtual VM	8- Depende de VM	8	9	8	8
Procesos en paralelo	9	8	7	8	8
Compatibilidad con Flutter	5- Requiere config extra	7	10	8	7
Nivel de personalización	10	9	9	9	8
Capacidad e integración con Google Play y App Store	8	9	10	9	7
Facilidad para definir pipelines	7	8	9	8	7
Integraciones	8	9	8	9	8
Documentación	7	9	9	9	8
Capacidad trabajo en equipo	8	9	9	9	9
Total	**80**	**85**	**89**	**84**	**78**

En la tabla I presentada previamente se muestra la ponderación para cada herramienta conforme los criterios establecidos para ejecutar la evaluación. De este modo se ha ponderado del 1 al 10 cada criterio en función a un análisis de información relacionada con cada herramienta en función a cada aspecto relacionado con cada una de las métricas obteniendo de este modo la ponderación. De esta forma se definió a Codemagic como herramienta para ejecutar el proceso de automatización DevOps de aplicaciones Flutter.

También se definieron herramientas que apoyarían en la ejecución del proceso de automatización. Así, como plataforma para gestionar el repositorio de código fuente se seleccionó a Gitlab debido a sus múltiples características entre ellas la más relevante su compatibilidad e integración con Codemagic. También se decidió emplear a Sonarqube como herramienta de análisis estático de código, la cual en los últimos años se ha convertido en una herramienta popular por sus características y soporte con la mayor parte de lenguajes de programación.

6.2. Objetivo 2:

Definidas y configuradas las herramientas seleccionadas para el pipeline o canalización, se desarrolló el archivo de configuración. También se establecieron los distintos flujos de trabajo que permitirán ejecutar la automatización para cada ambiente de este modo tenemos:

A. **Development Workflow,** El presente workflow se generó con el objetivo de que mediante el análisis estático de código se logre validar si se cumple con los criterios de calidad de código. También mediante este workflow se genera una apk y se notifica al desarrollador de si su código cumplió los criterios de calidad. En la Fig. 1 se puede visualizar el pipeline.

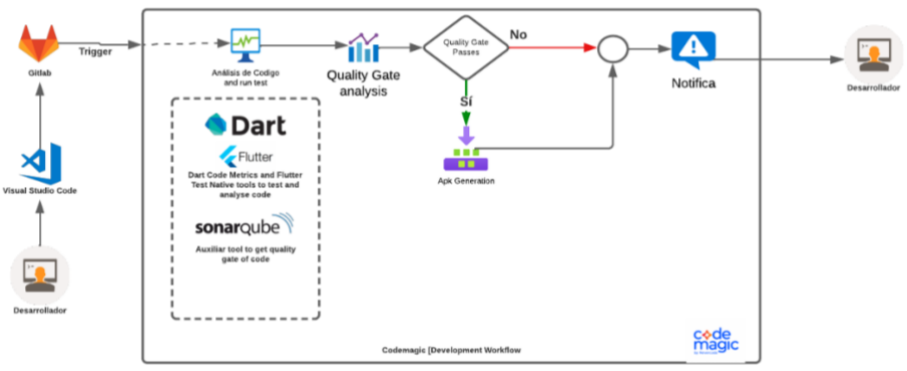


Fig. 1. Flujo de trabajo para desarrollo

En la Fig. 2, presentada a continuación se puede observar además el código resultante para el workflow previamente presentado.

```yaml
develop-workflow:
  name: Develop Workflow
  triggering:
    events:
      - push
    branch_patterns:
      - pattern: develop
        include: true
        source: true
  environment:
    flutter: stable
    groups:
      - sonarcloud_credentials
      - notifications
  cache:
    cache_paths:
      - ~/.pub-cache
      - ~/.sonar/cache
  scripts:
    - *code-quality
    - name: Check Quality Gate status
      script: |
        quality_gate_status=$(curl -s -u $SONAR_TOKEN: https://sonarcloud.io/api/qualitygates/project_status?projectKey=$SONAR_PROJECT_KEY&branch=develop | jq -r '.projectStatus.status
        echo "Quality Gate status: $quality_gate_status"
        quality_line=$(echo "$quality_gate_status" | grep -o '"status": *"[^"]*"')
        quality_gate=$(echo "$quality_line" | grep -o '"[^"]*"$' | sed 's/"//g')
        echo "Quality Gate status: $quality_gate"
        if [ "$quality_gate" == "ok" ]; then
          exit 1
        else
          echo "Quality gate ok"
        fi
    - name: Build APK
      script: |
        flutter build apk --debug
  artifacts:        You, 2 weeks ago • sonar check …
    - app/build/outputs/apk/debug/app-debug.apk
  publishing:
    email:
      recipients:
        - $DEV_EMAIL
        - jean.mosquera@unl.edu.ec
      notify:
        success: true
        failure: true
```

Fig. 2. Código yaml Development workflow

B. Testing Workflow, este workflow ejecuta el mismo proceso de análisis estático de código, no obstante, difiere en la fase de construcción de artefactos ya que en la presente fase una vez cumplido de forma exitosa el workflow de desarrollo en este caso se distribuye la aplicación a los testers. Para esto se ha considerado la distribución mediante Firebase App Distribution para dispositivos Android, y mediante Testflight para dispositivos iOS. De este modo el diagrama correspondiente a este workflow se encuentra en la Fig. 3.

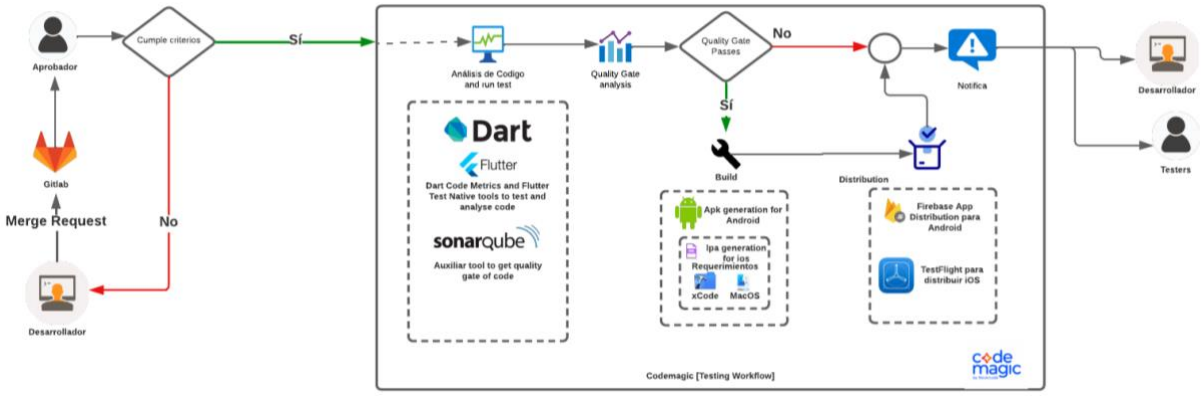


Fig. 3. Flujo de trabajo para pruebas

En la Fig. 4 se puede observar los pasos adicionales en los cuales se diferencia este workflow del anterior.

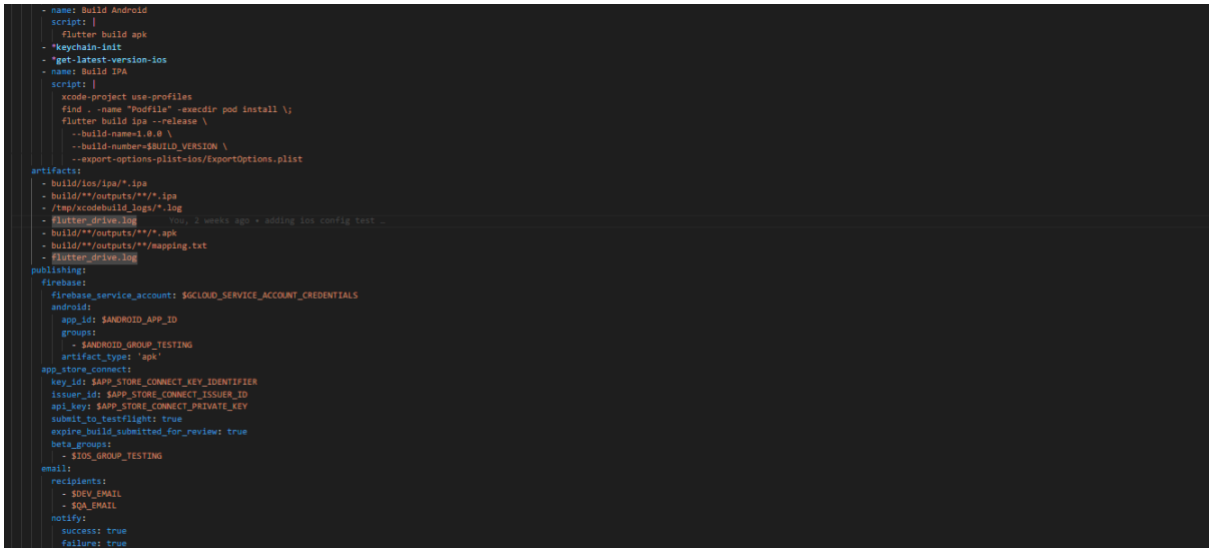


Fig. 4. Fase de construcción y distribución Testing Workflow

C. **Production Workflow,** el objetivo de este workflow es una vez dada la aprobación del equipo de ejecutar el lanzamiento de una nueva versión de la aplicación se ejecute el proceso de entrega o distribución de forma automática. Es decir, el resultado de ejecutar este flujo permitirá distribuir la aplicación a las tiendas de aplicaciones. Este proceso difiere en la fase de publicación del flujo de trabajo para el ambiente de testing, debido a que en este caso los artefactos se distribuirán para Google Play Console y App Store Connect para ejecutar su publicación y distribución. Este workflow se puede observar en la Fig. 5.

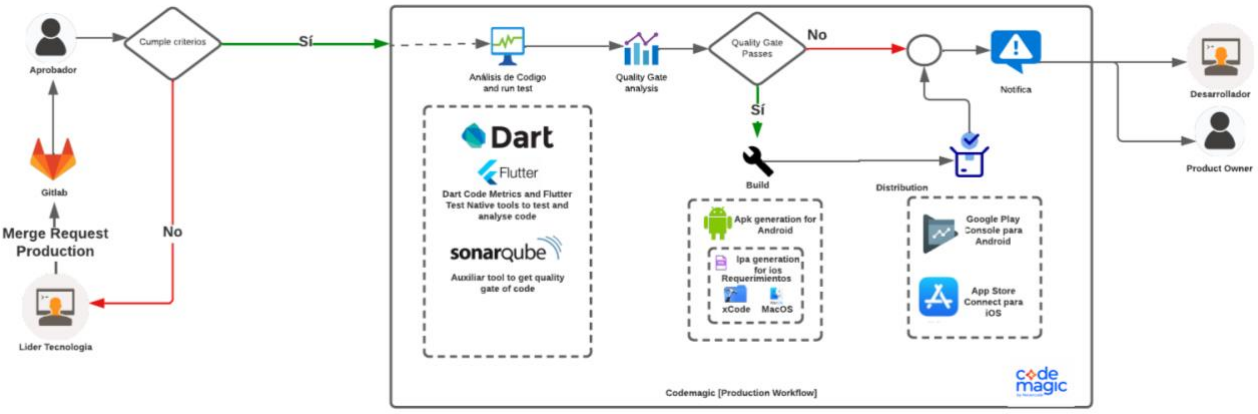


Fig. 5. Flujo de trabajo de producción

En la Fig. 6 presentada a continuación, se muestra la fase de publicación, dicha fase es la que difiere del flujo de trabajo de testing. Debido a que el presente flujo se encarga de distribuir como borrador los artefactos a las plataformas de distribución de aplicaciones para dispositivos Android y iOS.

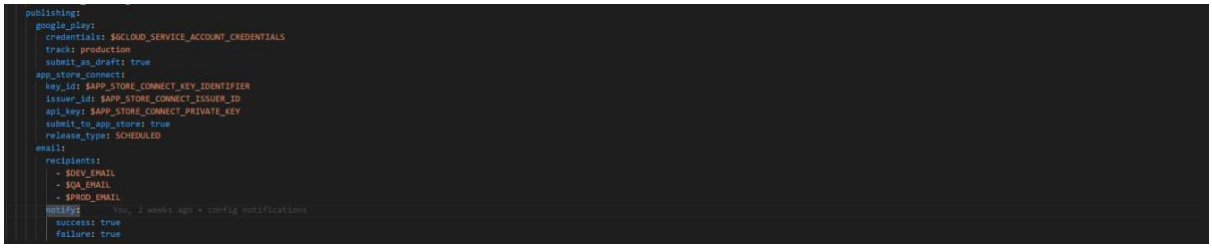


Fig. 6. Fase de publicación Production Workflow

Una vez establecido el pipeline se realizaron pruebas para validar las integraciones y el correcto funcionamiento. De este modo en la Fig. 7 se puede observarse como se puede visualizar el resultado de ejecución de un proceso en el dashboard de Codemagic.

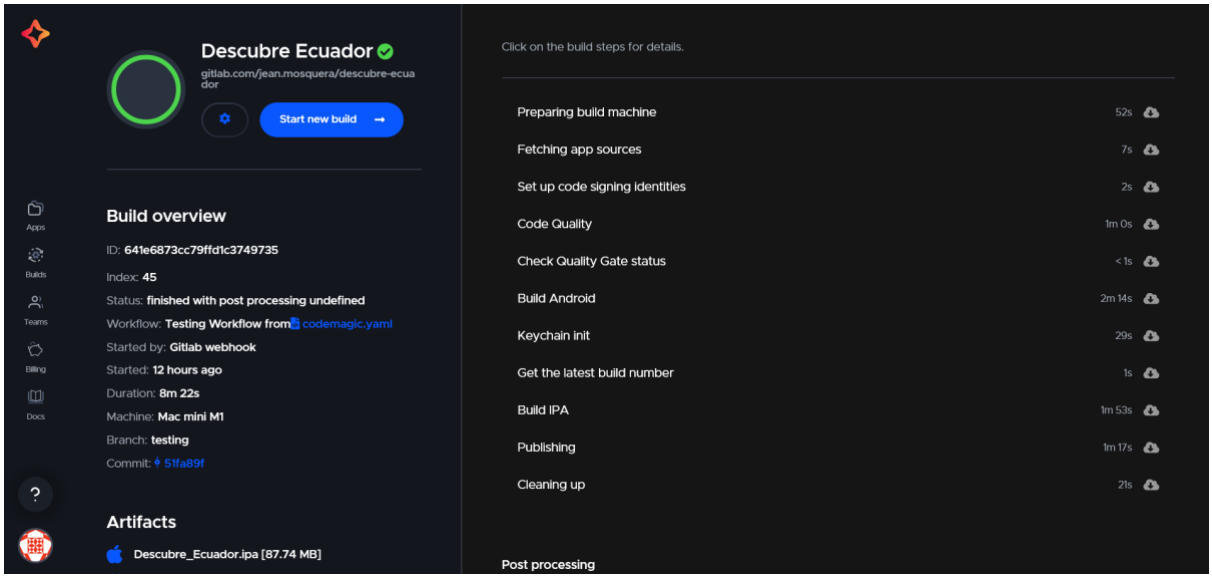


Fig. 7. Vista de resumen de pipeline ejecutado.

6.3. Objetivo 3:

Previo a ejecutar el proceso de evaluación de la propuesta se procedió al desarrollo del caso de aplicación. Así, el caso de aplicación busca dar valor a la información compartida públicamente por el Ministerio de Turismo la cual aborda los establecimientos turísticos del Ecuador que está contenida en el catastro turístico. Dentro del catastro turístico se encuentra un total de 21.912 establecimientos turísticos a lo largo del Ecuador.

Entre otros aspectos que se han tomado en cuenta se encuentran que la aplicación será de uso general, es decir disponible para el público general. El principal objetivo de la aplicación es facilitar el acceso a la información de los establecimientos turísticos dando uso y generando valor de esta información disponible públicamente. Para ello se

definieron historias de usuario con diversas funcionalidades que se han considerado útiles para el usuario final.

- **Desarrollo de caso de aplicación**

Entre las historias de usuario que se han establecido para la aplicación se encuentran:

- HU-1: Consultar establecimientos del catastro turístico.
- HU-2: Filtrar por provincia, cantón tipo.
- HU-3: Visualizar información de cada establecimiento turístico.
- HU-4: Localizar el establecimiento en el mapa.
- HU-5: Llamar al número registrado como contacto del establecimiento.
- HU-6: Enviar un correo al email registrado como mail de contacto del establecimiento.
- HU-7: Buscar información en la web sobre el establecimiento.
- HU-8: Compartir en diversos medios la información del establecimiento.
- HU-9: Copiar información referente al detalle de un establecimiento.

Con las historias de usuario establecidas se procedió a ejecutar el proceso de descripción de estas. A continuación, en la Figura 8 se presenta un ejemplo de una historia de usuario definida. En los anexos se encontrarán todas las historias de usuario descritas previamente.

Número: 1	**Usuario:** Público general
Nombre historia: Consultar establecimientos del catastro turístico	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 3	**Iteración asignada:** 1
Programador responsable: Jean Paul Mosquera	
Descripción: Como usuario, quiero poder consultar los establecimientos del catastro turístico para obtener información detallada sobre ellos. Quiero poder hacer esta consulta de forma sencilla y rápida, para poder tomar decisiones informadas sobre los establecimientos turísticos que me interesen.	
Observaciones:	

Fig. 8. Historia de usuario función filtrar por provincia, cantón y tipo

Con la definición de las historias de usuario se procedió a ejecutar la priorización de las historias de usuario por sprint, la cual puede observarse en la Tabla II. Se ha considerado una duración de sprint de 7 días, es decir sprints semanales para la ejecución del caso de aplicación previamente descrito.

TABLA II
PRIORIZACIÓN DE LAS HISTORIAS DE USUARIO

	Sprint 1	**Sprint 2**	**Sprint 3**
Historias de Usuario	HU-1 HU-2	HU-3 HU-4 HU-5 HU-6	HU-7 HU-8 HU-9

En lo relacionado con la arquitectura de la aplicación se ha definido el uso del patrón de diseño Modelo, Vista y Controlador (MVC). Este patrón de diseño se caracteriza por separar en capas los datos y la lógica de negocio, el controlador y la vista con el objetivo de lograr la separación de responsabilidad y facilitar la mantenibilidad del software[48], a continuación en la Fig. 9 se ilustra el patrón de diseño.

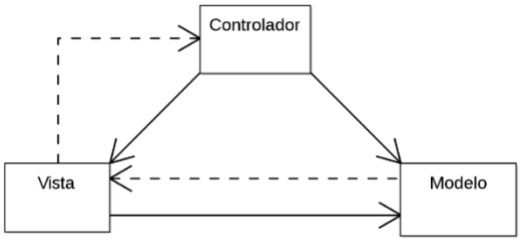


Fig. 9. Patrón de diseño MVC

Con la arquitectura de la solución definida se implementó el caso de aplicación del cual a continuación, en la Fig. 10 se presentan algunas capturas del proceso.

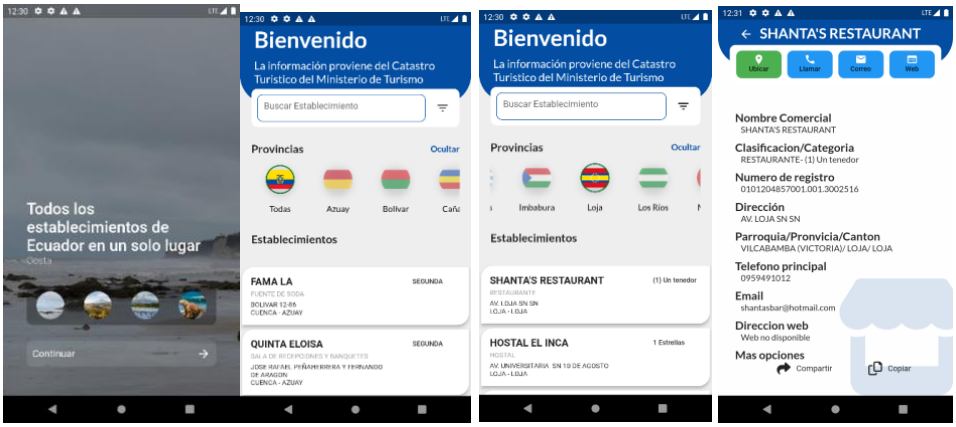


Fig. 10. Capturas de la aplicación desarrollada como caso de aplicación

También se definió el flujo funcional de la aplicación, es decir cómo será la navegación entre vistas y las diversas funcionalidades de este modo a continuación, en la Fig. 11 se presenta el flujo del caso de aplicación mediante el cual se describe el funcionamiento de la aplicación propuesta como solución al caso de aplicación propuesto.

Fig. 11. Flujo del caso de aplicación

En la figura anterior se describe el flujo de la aplicación, además se muestran las funciones establecidas en las historias de usuario. El presente flujo describe de forma secuencial las diversas funciones que posee el caso de aplicación.

- **Proceso de Evaluación**

Para el proceso de evaluación se ha considerado tomar como referencia el tiempo de ejecución de los procesos o tareas para poder construir un artefacto y posterior publicación de este en App Store Connect y Google Play Console.

De este modo al ejecutar el proceso manual y ejecución del pipeline se obtuvieron los resultados mostrados en la Tabla III:

TABLA III
RESULTADOS TIEMPO DE EJECUCIÓN PROCESO MANUAL VS PIPELINE

Fase o proceso	**Proceso Manual**	**Pipeline**
Obtención de código fuente	600s	10s
Configuración de entornos	3600s	300s
Medición de calidad de código	60s	60s
Validación Quality Gates	60s	1s
Construcción aab Android	600s	134s
Construcción ipa iOS	180s	114s
Publicación aab Google Play Console	305s	135s
Publicación ipa App Store Connect	495s	135s
Total, en segundos	**6000s**	**889s**

La tabla presentada previamente demostró una reducción considerable de tiempo entre el proceso automatizado y el proceso manual. Esta reducción significa un 85.18% del tiempo que lleva ejecutar todas las fases cuando se emplea el pipeline.

En el caso del escenario de subir una corrección se consideró que ya se dispone el código fuente en el equipo en el que se ejecuta el proceso manual y también que el entorno necesario para ejecutar el proceso ya está configurado. Así, a continuación, en la Tabla IV, se presentan los resultados obtenidos en este escenario.

TABLA IV
TIEMPO DE EJECUCIÓN PROCESO MANUAL VS PIPELINE, GESTIÓN DE CAMBIOS

Fase o proceso	**Proceso Manual**	**Pipeline**
Obtención de código fuente (pull)	10s	10s
Configuración de entornos	200s	300s
Medición de calidad de código	75s	55s
Validación Quality Gates	60s	1s
Construcción aab Android	580s	145s
Construcción ipa iOS	200s	127s

Publicación aab Google Play Console	305s	145s
Publicación ipa App Store Connect	300s	145s
Total, en segundos	**1925s**	**928s**

De este modo se comprueba que efectivamente existe una notoria reducción del tiempo que conlleva la ejecución del proceso de construcción y distribución de la aplicación en App Store Connect y Google Play Console. Se puede evidenciar la ejecución del proceso manual en el anexo 3, las variaciones de tiempo se deben a que la ejecución de la primera iteración considerada para los resultados conllevaba la obtención de insumos como la llave para firma de aplicaciones Android y la configuración de los certificados para la distribución de la aplicación para la plataforma iOS, entre otras configuraciones necesarias.

Se definieron ciertos requerimientos obligatorios que han surgido al momento de ejecutar algunas de las fases cuando el proceso se realiza de forma manual, las cuales se describen a continuación :

A. Configuración de entornos:

- En caso de que no se disponga del entorno de Sonarqube, es necesario ejecutar el proceso de instalación y configuración de esta herramienta necesaria para el análisis de código estático.
- En caso de no disponer del framework Flutter en el equipo, es necesario ejecutar el proceso de instalación y configuración., se requieren de herramientas auxiliares como Android Studio necesario para la construcción del artefacto para Android, lo cual conlleva tiempo extra.
- En caso de no disponer de xCode en el equipo y otras herramientas como Transporter es necesario su instalación y configuración, lo cual conlleva un tiempo extra.

B. Medición de la calidad de código:

- Es necesario y mandatorio disponer en el entorno de Sonarqube que se ha definido como herramienta de análisis estático de código fuente, es importante destacar que Flutter provee de herramientas para este fin que podrían emplearse.

C. Construcción de artefacto para Android:

- Es mandatorio disponer la llave jks y credenciales para ejecutar el proceso de firma para su publicación. Esto significa que es necesario proveer de esta información a la persona que requiera ejecutar este proceso provocando una mala práctica al filtrar credenciales críticas.

D. Construcción de artefacto para iOS

- En el presente caso es necesario disponer de un entorno macOS debido a que se requiere xCode y Transporter, aplicaciones necesarias para construir una aplicación iOS para Flutter las cuales solo están disponibles para este sistema operativo. De este modo en caso de no disponer de un entorno con este sistema operativo se imposibilita la creación de este artefacto.

E. Publicación en Google Play Console y App Store Connect

- Es necesario generar un perfil con permisos para subida del artefacto para habilitar la posibilidad de que este pueda subir la nueva versión de la aplicación, es decir se deben tener las cuentas de desarrollador para cada servicio.

7. Discusión

La automatización del proceso DevOps de aplicaciones desarrolladas en Flutter reduce de forma significativa el tiempo de construcción y entrega, además de permitir lograr que estas tareas se ejecuten de forma ordenada y controlada. En la actualidad existen variedad de servicios que permiten la automatización del proceso DevOps, no obstante, en el caso particular del framework Flutter si se requiere la distribución de la aplicación para dispositivos iOS es necesario disponer de una máquina virtual o bien un dispositivo con el sistema operativo macOS.

El presente TT busca mediante el empleo de Codemagic la automatización del proceso DevOps de aplicaciones desarrolladas con Flutter, así se han logrado definir diversos hallazgos que distinguen al proceso manual y el proceso automatizado. En la tabla V se muestra un cuadro comparativo con los hallazgos detectados.

TABLA V
DIFERENCIAS ENTRE EL PROCESOS MANUAL Y PROCESO PROPUESTO

Característica	Proceso Manual	Proceso automatizado
Control	Se pueden obviar fases importantes por buscar generar los artefactos rápidos.	Todo el proceso está definido de forma tal que no se pueden omitir fases.
Tiempo de ejecución	Puede variar dependiendo los conocimientos de la persona que ejecuta, equipo y diversas variables.	El tiempo de ejecución no depende de variables externas y no se ven variaciones.
Seguridad	Es necesario brindar acceso a insumos para la firma y distribución de la aplicación como por ejemplo credenciales.	No es necesario brindar accesos a otros miembros del equipo ya que estos insumos forman parte de la configuración.
Trazabilidad	El proceso no se podría controlar, ni visualizar los resultados obtenidos por parte del equipo.	Se puede observar todo el proceso, las operaciones y eventos ejecutados. Se reciben notificaciones al finalizar de forma exitosa o fallida el proceso
Requerimientos	En el presente caso se requeriría de un equipo y una persona que conozca el proceso de construcción y distribución	Los requerimientos al inicio de un experto para configurar. Los procesos posteriores se ejecutarán sin requisitos más que el pago por uso de la herramienta en caso de que se supere la cuota gratuita.

Para la automatización DevOps de aplicaciones Flutter se deben tener presente varios aspectos para garantizar el éxito del proceso. A continuación, se destacan los más relevantes:

A. **Configuración del entorno de desarrollo:** Es importante tener un entorno adecuado para el desarrollo y para la automatización de DevOps. Esto incluye tener una instalación adecuada de Flutter, un entorno de integración y despliegue continuos (CI/CD) y herramientas de automatización de prueba, entre otras.
B. **Implementación de pruebas automatizadas:** Las pruebas automatizadas son esenciales para la automatización de DevOps de aplicaciones Flutter. Es importante definir y escribir pruebas automatizadas para cada componente de la aplicación.
C. **Automatización de la integración y el despliegue continuos:** La automatización de la integración y el despliegue continuos son esenciales para la automatización de DevOps. Es importante definir y configurar pipelines de CI/CD que permitan la integración y el despliegue continuo de la aplicación.
D. **Gestión de versiones y control de código fuente:** Es importante tener un control de versiones adecuado y un control de código fuente para el proceso de automatización de DevOps. Se deben definir y seguir las mejores prácticas de gestión de versiones y control de código fuente.
E. **Monitoreo y recopilación de incidentes de la aplicación:** Es importante implementar un sistema de monitoreo y de recopilación de incidentes de la aplicación para garantizar el buen funcionamiento de la aplicación y detectar posibles problemas en tiempo real y así evitar que se ponga en riesgo la disponibilidad de la aplicación.

8. Conclusiones

La automatización del proceso DevOps resulta ser muy efectiva y reduce de forma sustancial el tiempo de entrega de los artefactos para su distribución al equipo de testers y posterior publicación para que se encuentren disponibles para los usuarios finales.

Codemagic permite diseñar pipelines ya sea de forma asistida y también mediante la generación de un script de configuración del pipeline yaml, esta última alternativa permite mayor flexibilidad al momento de buscar integraciones con diversos servicios como Sonarqube para el análisis estático de código, entre otros servicios compatibles con codemagic empleados para el análisis de código, testing, comunicación. Al emplear el lenguaje yaml para ejecutar la configuración del proceso, se logra mejor legibilidad y comprensión de las instrucciones, portabilidad y sobre todo flexibilidad al permitir incluir desde instrucciones simples hasta procesos más complejos.

Codemagic se constituye en una herramienta ideal para la automatización del proceso DevOps para aplicaciones Flutter, ya que permite acceder a máquinas virtuales con el sistema operativo macOS siendo esto importante para la generación de artefactos para dispositivos iOS y así como también gestionar su distribución mediante App Store Connect y TestFlight, esta herramienta además provee una cantidad de minutos de ejecución importante que se renuevan mensualmente.

Al automatizar el proceso DevOps se evita fugas de información como proveer archivos y credenciales para gestionar la firma de aplicaciones y distribución de los artefactos, además de agilizar este proceso y estandarizar los pasos que se deben cumplir para publicar un artefacto, también se pueden incluir procesos adicionales como análisis estático de código para garantizar la mantenibilidad, validación de pruebas unitarias y otros atributos de calidad, los cuales al ejecutarse de forma manual podrían omitirse.

Además, los pipelines reducen el tiempo requerido para establecer el entorno de trabajo, es decir los pasos relacionados con la instalación y configuración de herramientas prácticamente se omitirían lo cual agilizaría el proceso de construcción y entrega.

9. Recomendaciones

Es importante generar el proceso de distribución de artefactos de forma manual para que el proceso de automatización en ejecuciones posteriores se ejecute sin bloqueos. De este modo se recomienda generar la distribución de la versión inicial de la aplicación tanto en Google Play Console y App Store Connect. Esto se pudo visibilizar al ejecutar experimentos para probar la efectividad del pipeline generado.

También es recomendable ejecutar el proceso de análisis de herramientas para de este modo seleccionar aquellas que sean compatibles con el stack tecnológico de la solución a la cual se desea configurar la automatización. Otro aspecto al momento de definir la herramienta es analizar si la herramienta permite ejecutar diversas integraciones, y entre otros aspectos el costo y el entorno bajo el cual se ejecutará el pipeline. Este último punto es sumamente importante ya que en el caso de que se requiera la construcción y distribución para dispositivos iOS es necesario disponer de una máquina virtual con sistema operativo macOS.

Se recomienda revisar la documentación que dispone cada herramienta para de este definir un proceso adecuado de instalación y configuración, ya que esto posteriormente puede generar bloqueos o interrupciones en la ejecución del proceso y sobre todo gastos de recursos al ejecutar un pipeline que no es eficiente.

10. Bibliografía

[1] J. Verona, *Practical DevOps*. 2016.

[2] Red Hat, "What is DevSecOps?," 2023. https://www.redhat.com/en/topics/devops/what-is-devsecops.

[3] S. Vadapalli, *DevOps: Continuous Delivery, Integration, and Deployment with DevOps*. 2018.

[4] Instituto Nacional de Estadisticas y Censo INEC, "Tecnologías de la Información y Comunicación-TIC," 2020. .

[5] Simon Kemp, "Digital 2023: Ecuador," 2023. https://datareportal.com/reports/digital-2023-ecuador.

[6] D. Meiller, *Modern App Development with Dart and Flutter 2*. 2021.

[7] C. Ebert, G. Gallardo, J. Hernantes, and N. Serrano, "DevOps," *IEEE Softw.*, vol. 33, no. 3, pp. 94–100, 2016, doi: 10.1109/MS.2016.68.

[8] A. Hemon, B. Lyonnet, F. Rowe, and B. Fitzgerald, "From Agile to DevOps: Smart Skills and Collaborations," *Inf. Syst. Front.*, vol. 22, no. 4, pp. 927–945, 2020, doi: 10.1007/s10796-019-09905-1.

[9] M. Hajian, "Applying DevOps in Flutter mobile development," 2020. https://www.wearedevelopers.com/en/videos/applying-devops-in-flutter-mobile-development.

[10] S. Moqrab Khan, A. ul Nabi, and T. Bhanbhro, "Comparative Analysis between Flutter and React Native," *Int. J. Artif. Intell. Math. Sci.*, vol. 1, 2022, doi: https://doi.org/10.58921/ijaims.v1i1.19.

[11] F. Martin, *Flutter and Dart the complete guide*. 2021.

[12] A. Maulana and H. Kabetta, "Implementation of DevSecOps by Integrating Static and Dynamic Security Testing in CI/CD Pipelines," 2022, doi: 10.1109/ICOSNIKOM56551.2022.10034883.

[13] A. E. Fentaw, *Cross platform mobile application development: a comparison study of React Native Vs Flutter*. 2020.

[14] S. Anwar, "Comparison and evaluation of cross-platform framework and development of a digital health platform using selected framework," no. December, 2021, [Online]. Available: http://www.teknat.uu.se/student.

[15] E. Ramírez, *Canalización CI/CD para la entrega continua de aplicaciones móviles en la nube de Amazon Web Services*. 2020.

[16] S. Fleming, "The DevOps Engineer's career guide," 2019.

[17] G. Kim, J. Humble, P. Debois, and J. Willis, *The DevOps Handbook*. 2016.

[18] D. Harrison and K. Lively, *Achieving DevOps*. 2019.

[19] HashiCorp, "Terraform," 2023. https://www.terraform.io/.

[20] Jenkins, "Jenkins," 2023. https://www.jenkins.io/.

[21] Nevercode, "Codemagic," 2023. https://codemagic.io/.

[22] Prometheus, "Prometheus," 2023. https://prometheus.io/.

[23] Sentry, "Sentry," 2023. .

[24] C. Singh, N. S. Gaba, M. Kaur, and B. Kaur, "Comparison of different CI/CD Tools integrated with cloud platform," *Proc. 9th Int. Conf. Cloud Comput. Data Sci. Eng. Conflu. 2019*, pp. 7–12, 2019, doi: 10.1109/CONFLUENCE.2019.8776985.

[25] C. Null, "10 companies killing it at DevOps," 2023.

[26] A. Ocampo, "Principales tendencias y alcances futuros de DevOps," 2020. https://andresfelipeocampo.medium.com/principales-tendencias-y-alcances-futuros-de-devops-c32c6d1c9a1a.

[27] B. Anand, "Top 7 DevOps Trends of 2023," 2023. https://www.knowledgehut.com/blog/devops/devops-trends.

[28] Weaveworks, “GitOps: GitOps is a way to do Continuous Delivery, it works by using Git as a single source of truth for declarative infrastructure and applications.,” 2023. https://www.weave.works/technologies/gitops/.

[29] IBM, “Understanding serverless architectures,” 2023. https://www.ibm.com/cloud/learn/serverless-architecture.

[30] P. Tyagi, *Pragmatic Flutter*. 2022.

[31] Y. Cheon and C. Chavez, “Converting Android Native Apps to Flutter Cross-Platform Apps,” in *2021 International Conference on Computational Science and Computational Intelligence (CSCI)*, 2021, pp. 1898–1904, doi: 10.1109/CSCI54926.2021.00355.

[32] E. Freitas, *Flutter UI*. 2021.

[33] Flutter, “BMW,” 2023. https://flutter.dev/showcase/bmw.

[34] Flutter, “PUBG Mobile,” 2023. https://flutter.dev/showcase/pubg-mobile.

[35] Flutter, “Google Pay,” 2023. https://flutter.dev/showcase/google-pay.

[36] Flutter, “NuBank,” 2023. https://flutter.dev/showcase/nubank.

[37] Flutter, “Rive,” 2023. https://flutter.dev/showcase/rive.

[38] Bitrise, “Bitrise,” 2023. https://bitrise.io.

[39] Github, “Github Actions,” 2023. https://github.com/features/actions.

[40] Google Inc., “Firebase for Android,” 2018. https://firebase.google.com/docs/reference/android/packages?hl=es-419.

[41] Apple Inc., “App Store Connect,” 2023. https://appstoreconnect.apple.com/login.

[42] P. Madan, “Review : Graph Databases,” vol. 4, no. 5, pp. 195–200, 2014.

[43] Red Hat, “Yaml,” 2023. https://www.redhat.com/es/topics/automation/what-is-yaml.

[44] Yaml Org, “Yaml,” 2023. https://yaml.org/.

[45] Sonar, “Sonarqube,” 2023. https://www.sonarsource.com/products/sonarqube/.

[46] Jetbrains, “Measuring & Monitoring CI/CD Performance,” 2023. https://www.jetbrains.com/teamcity/ci-cd-guide/devops-ci-cd-metrics/.

[47] Ministerio de Turismo, “Catastro nacional de establecimientos turisticos,” 2022. https://servicios.turismo.gob.ec/portfolio/catastro-turistico-nacional.

[48] M. Katz, K. Moore, V. Ngo, and V. Guzzi, “Flutter Apprentice,” 2021.

11. Anexos

Anexo 1. Historias de usuario

Historia de Usuario	
Número: 3	**Usuario:** Público general
Nombre historia: Visualizar información de cada establecimiento turístico	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 3	**Iteración asignada: 2**
Programador responsable: Jean Paul Mosquera	
Descripción: Como usuario, quiero poder visualizar información detallada sobre cada establecimiento turístico que consulte en el catastro. Esta información debe incluir datos como la ubicación, la categoría, las instalaciones, las opiniones de otros turistas, entre otros aspectos relevantes. De esta forma, podré tener una visión completa del establecimiento antes de tomar una decisión sobre si visitarlo o no.	
Observaciones:	

Historia de Usuario	
Número: 4	**Usuario:** Público general
Nombre historia: Localizar el establecimiento en el mapa	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 3	**Iteración asignada: 2**
Programador responsable: Jean Paul Mosquera	
Descripción: Como usuario, quiero poder localizar el establecimiento turístico en un mapa para tener una idea de su ubicación exacta y cómo llegar hasta allí. Esto me permitirá planificar mi viaje de manera más efectiva y asegurarse de que llegue al lugar que deseo visitar.	
Observaciones:	

Historia de Usuario

Número: 5	**Usuario:** Público general
Nombre historia: Llamar al número registrado como contacto del establecimiento	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 3	**Iteración asignada: 2**
Programador responsable: Jean Paul Mosquera	
Descripción: Como usuario, quiero poder llamar al número de contacto registrado para cada establecimiento turístico, para poder hacer preguntas o reservaciones directamente con ellos. Esta función me permitirá tener una comunicación más efectiva y obtener respuestas rápidas a cualquier duda o inquietud que tenga sobre el establecimiento.	
Observaciones:	

Historia de Usuario	
Número: 6	**Usuario:** Público general
Nombre historia: Enviar un correo al email registrado como mail de contacto del establecimiento	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 3	**Iteración asignada: 2**
Programador responsable: Jean Paul Mosquera	
Descripción: Como usuario, quiero poder enviar un correo electrónico al correo de contacto registrado para cada establecimiento turístico, para poder hacer preguntas o reservaciones con ellos a través de esta vía. Esta función me permitirá tener una forma alternativa de comunicación y me asegurará de que mi mensaje sea recibido por el establecimiento de manera oportuna.	
Observaciones:	

Historia de Usuario	
Número: 7	**Usuario:** Público general

Nombre historia: Buscar información en la web sobre el establecimiento	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 3	**Iteración asignada: 3**
Programador responsable: Jean Paul Mosquera	
Descripción: Como usuario, quiero poder buscar información adicional sobre el establecimiento turístico en la web, para poder tener una visión más completa y detallada de sus características, servicios y opiniones de otros turistas. Esta función me permitirá complementar la información que ya tengo sobre el establecimiento y tomar una decisión más informada sobre si visitarlo o no.	
Observaciones:	

	Historia de Usuario
Número: 8	**Usuario:** Público general
Nombre historia: Compartir en diversos medios la información del establecimiento	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 3	**Iteración asignada: 3**
Programador responsable: Jean Paul Mosquera	
Descripción: Como usuario, quiero poder compartir la información del establecimiento turístico en diversos medios, como redes sociales o mensajería instantánea, para poder recomendarlo a mis amigos y familiares o simplemente para tener un registro personal. Esta función me permitirá expandir mi conocimiento y experiencia del establecimiento y compartirla con otros que puedan estar interesados en visitarlo.	
Observaciones:	

	Historia de Usuario
Número: 9	**Usuario:** Público general

Nombre historia: Copiar información referente al detalle de un establecimiento	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 3	**Iteración asignada: 3**
Programador responsable: Jean Paul Mosquera	
Descripción: Como usuario, quiero poder copiar información relevante sobre el detalle de un establecimiento turístico, como su nombre, dirección o número de teléfono, para poder tener una copia en mi dispositivo o para poder compartirla con otras personas. Esta función me permitirá tener un registro fácil y accesible de la información sobre el establecimiento y me ahorrará tiempo en buscarla de nuevo.	
Observaciones:	

Anexo 2. Proceso de configuración de Git

Se ejecutó la creación de un nuevo proyecto en Gitlab y posterior asociación del proyecto local con el proyecto generado en Gitlab empleando Git. Para acceder al proyecto puede hacerse mediante el siguiente enlace https://gitlab.com/jean.mosquera/descubre-ecuador, en la Fig. 12 se puede observarse el repositorio.

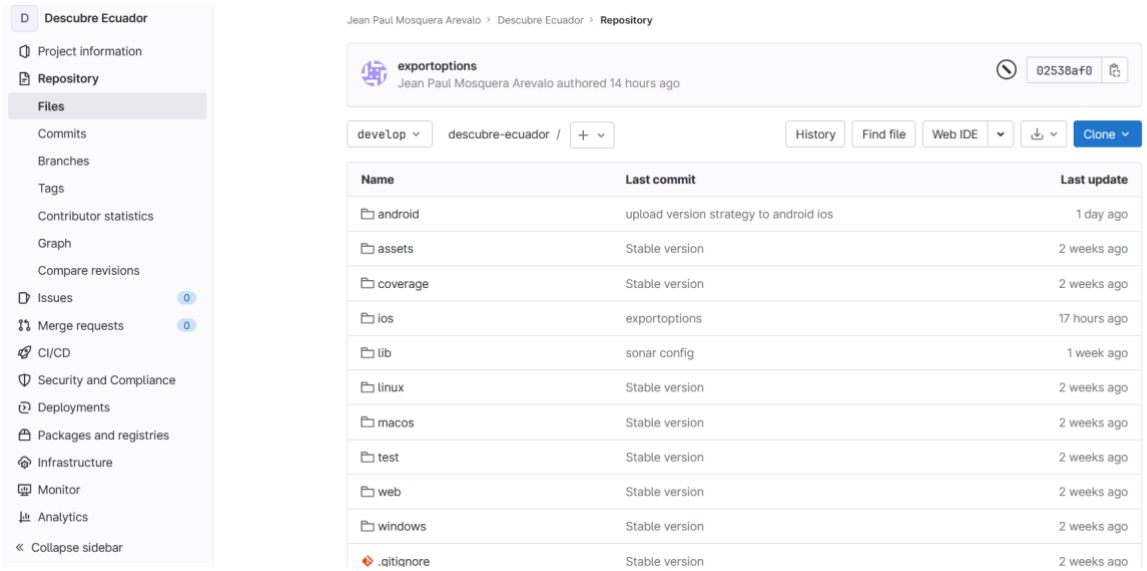


Fig. 12. Repositorio de caso de aplicación que servirá para configurar y evaluar pipeline

Anexo 3. Proceso de configuración de Codemagic

Se procedió a conectar el repositorio antes configurado en Codemagic, de esta forma debemos seleccionar el proveedor o gestor de repositorios de este modo Codemagic soporta Bitbucket, Github, Gitlab, además de permitir asociar mediante una URL, como puede observarse en la Fig. 13.

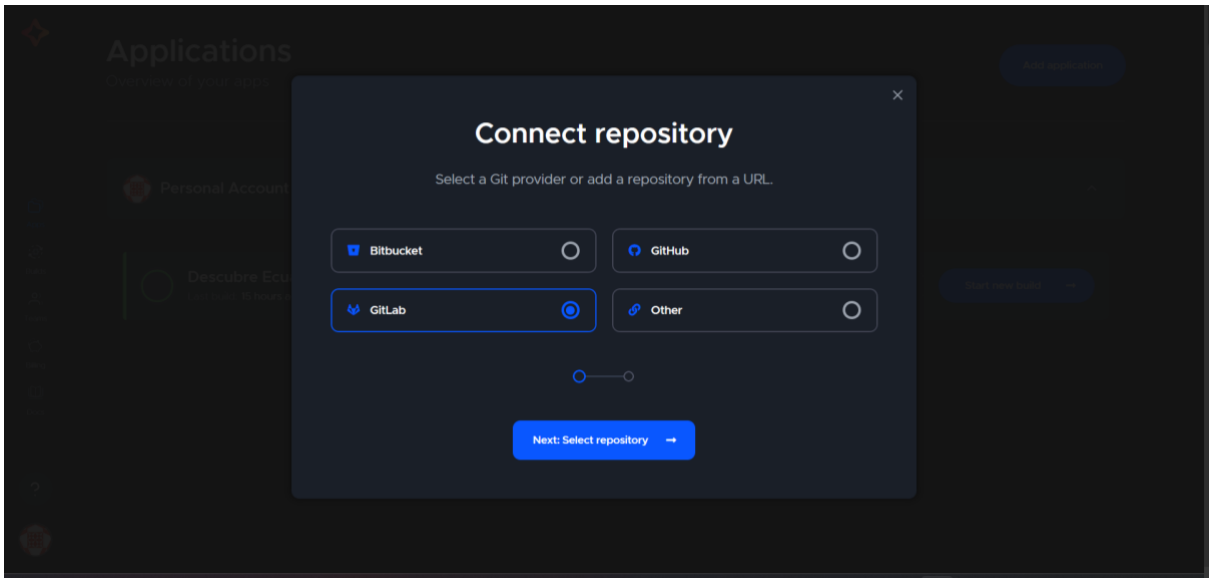


Fig. 13. Conexión de repositorio en codemagic

Se realizó la selección del repositorio y elección del tipo de proyecto, entre las opciones disponibles se encuentran Aplicación Android o iOS, Aplicación Flutter, React Native, Cordova, Ionic, Unity. Es decir, Codemagic tiene soporte para aplicaciones nativas y desarrolladas con varios frameworks, en la Fig. 14 se puede ver este proceso.

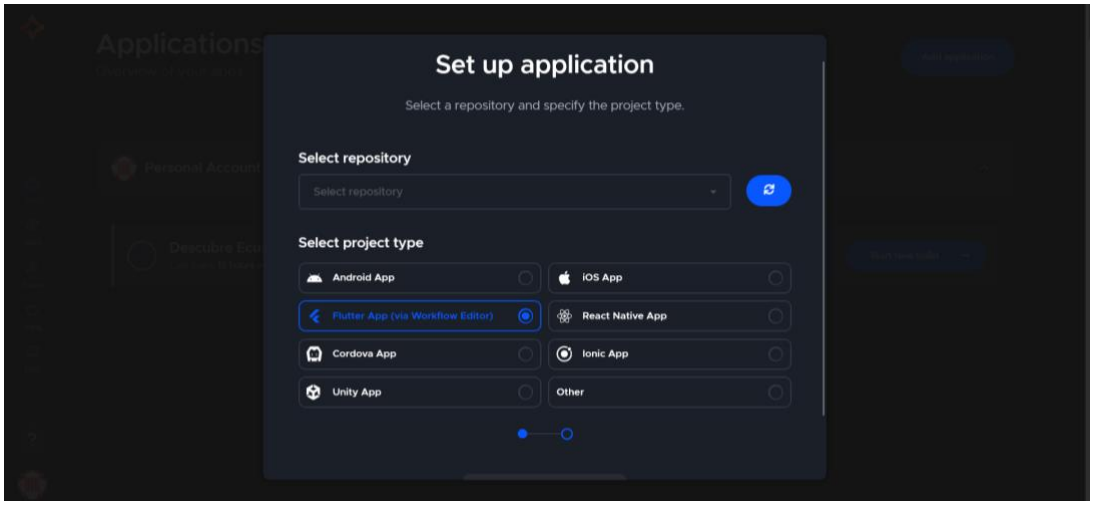


Fig. 14. Selección de repositorio

Se definió el uso de codemagic.yaml es decir la generación vía scripting. Se debe destacar que Codemagic posee un editor intuitivo para configurar el pipeline, no obstante, se optó por la alternativa de hacer uso de script debido a que permite ejecutar varias integraciones entre ellas integración con SonarQube para análisis de la calidad de código, entre otras, en la Fig. 15 se muestra la pantalla de configuración.

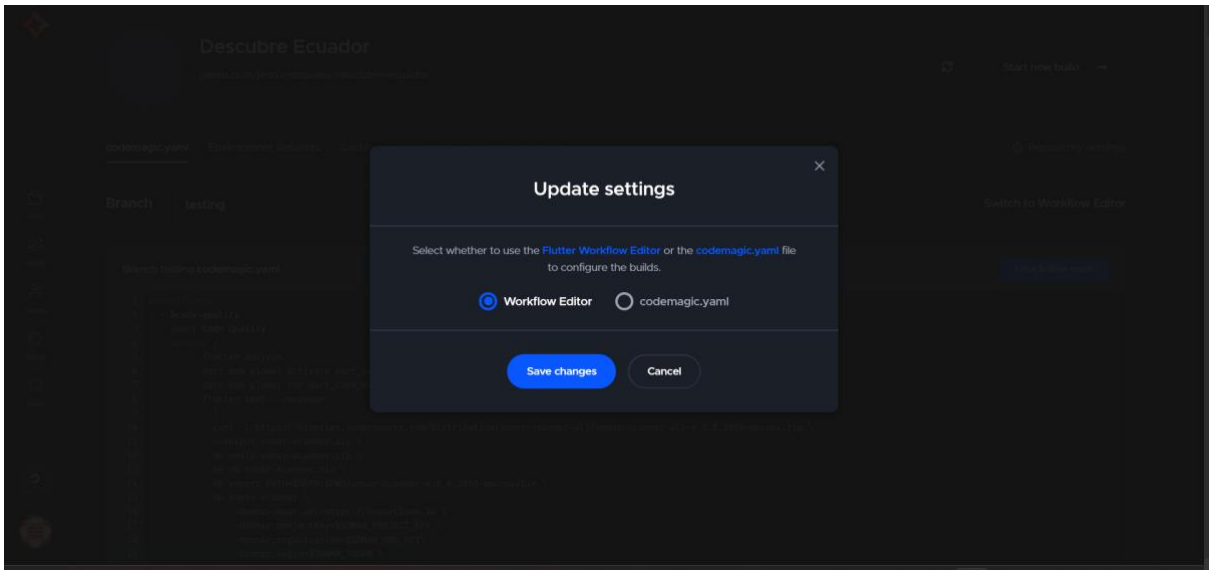


Fig. 15. Selección de tipo de configuración para generación de pipeline

Anexo 4. Proceso de configuración de Firebase

Para configurar Firebase, inicialmente se realizó la creación de un nuevo proyecto, para lo cual mediante la opción crear agregar proyecto establecemos un nombre para nuestro proyecto y continuamos con el proceso guiado de creación, en la Fig. 16 se muestra el inicio de proceso de creación.

Fig. 16. Creación de proyecto en firebase.

a. Para configurar Firebase en la aplicación Flutter, ejecutamos la configuración de mediante la librería FlutterFire, la cual es un asistente que nos permite configurar en Flutter Firebase de forma ágil y rápida. Para ejecutar el proceso de instalación se puede seguir la siguiente guía https://firebase.flutter.dev/docs/overview.
b. Una vez instalado mediante el comando "flutterfire configure", podremos asociar nuestros proyectos de firebase en nuestra aplicación, tal como se describe en la Fig. 17.

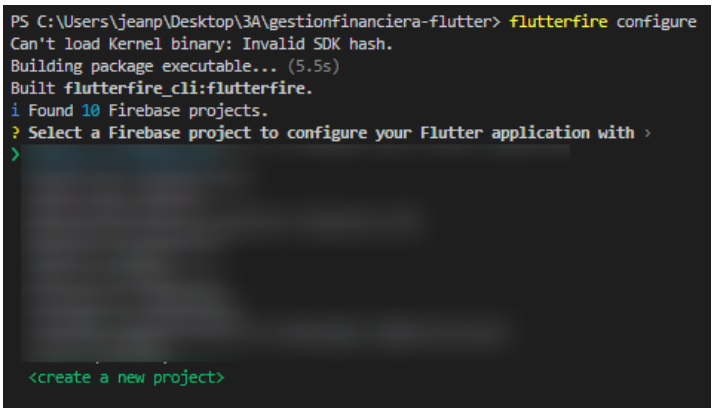


Fig. 17. Configuración de firebase en proyecto mediante flutterfire.

c. Para validar la correcta configuración podemos acceder a la configuración del proyecto generado en firebase y visualizar nuestras aplicaciones en el proyecto listadas como puede verse en la Fig. 18.

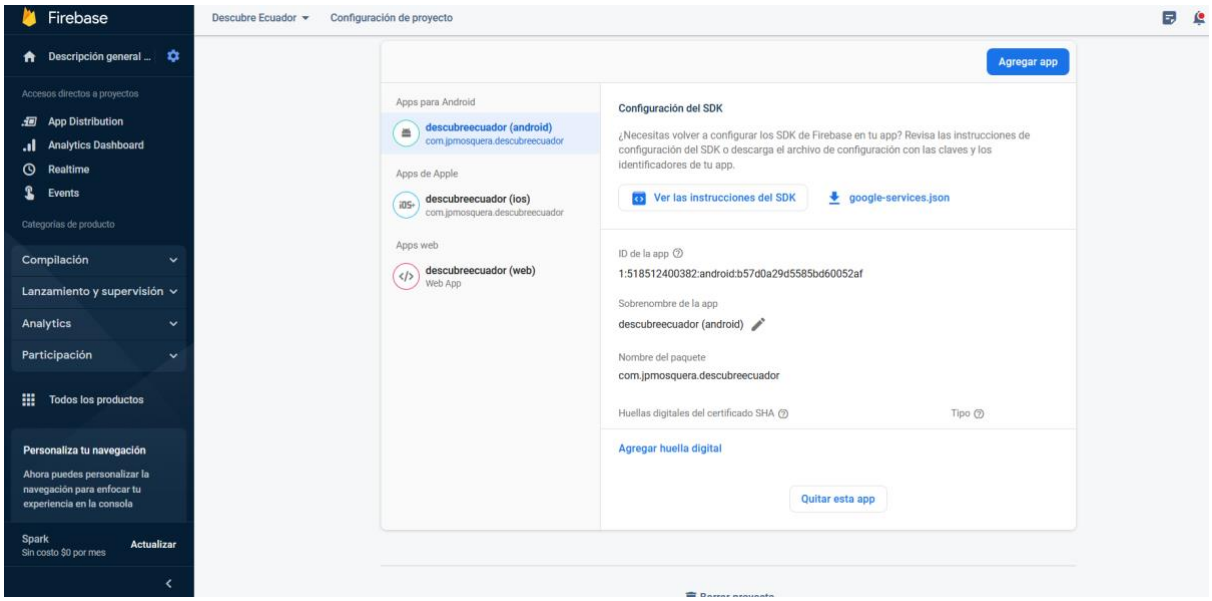


Fig. 18. Aplicaciones creadas en firebase.

Anexo 5. Proceso de configuración de App Store Connect

Para configurar App Store Connect es necesario obtener un Api Key, para lo cual se debe acceder al apartado de Usuarios y accesos, en la figura siguiente se puede observar la sección donde se administran el api keys, una vez generada es importante guardar el api key debido a que solo puede obtenerse una vez, tal como se puede observar en la Fig.19.

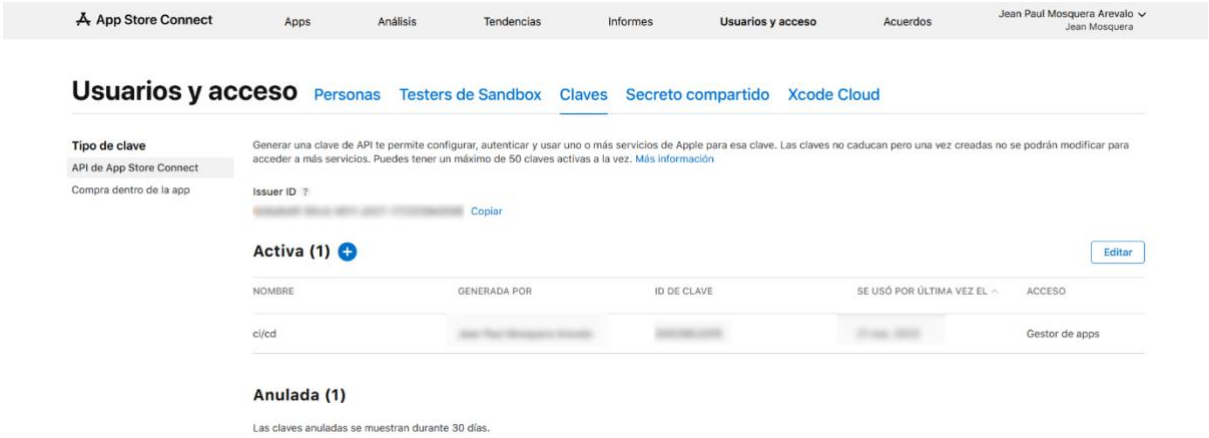


Fig. 19. Sección de administración de api keys en App Store Connect

En App Store Connect es necesario contar con una app para ello desde el apartado de Apps procedemos a generar la aplicación, en la figura siguiente se puede visualizar el formulario de registro de una app. Es necesario cumplir con este proceso ya que necesitamos el id de la aplicación para configuraciones posteriores, en la Fig. 20 se muestra el formulario de creación.

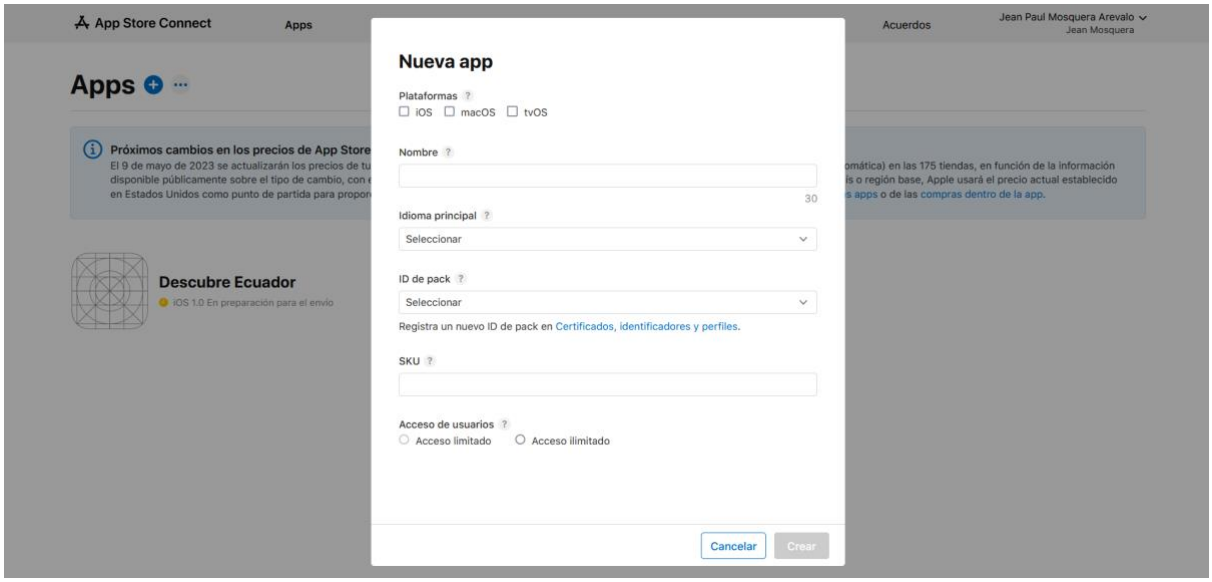


Fig. 20. Sección para crear una nueva app en App Store Connect

Posteriormente a la obtención de estos insumos es necesario ejecutar la configuración de las variables de entorno tanto para la llave privada como el id de emisor que puede

obtenerse desde el mismo apartado anterior. Codemagic permite definir grupos de variables, para ello debemos acceder a la configuración de la aplicación en el Dashboard de Codemagic, tal como se muestra en la Fig. 21.

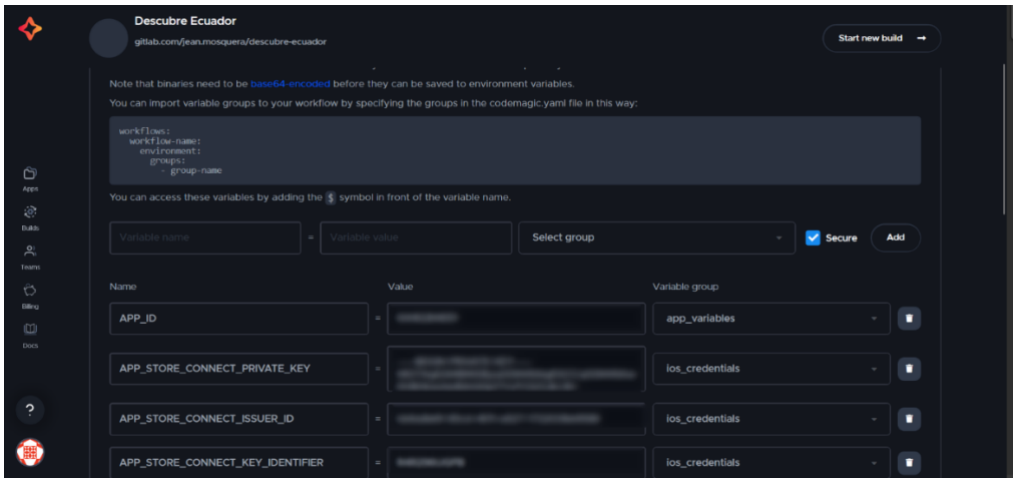


Fig. 21. Configuración de variables en codemagic

Anexo 6. Proceso de configuración de Google Play Console y Firebase App Distribution

Inicialmente se debe crear un proyecto para la aplicación que se desea desplegar, puesto posteriormente se requerirá de este.

Seguidamente en el apartado Acceso de api se debe generar el api key, en caso de no disponer de un proyecto de Google Cloud se generará de forma automática, para luego poder acceder al mismo mediante la opción ver en Google Cloud Platform como se puede ver la Fig. 22.

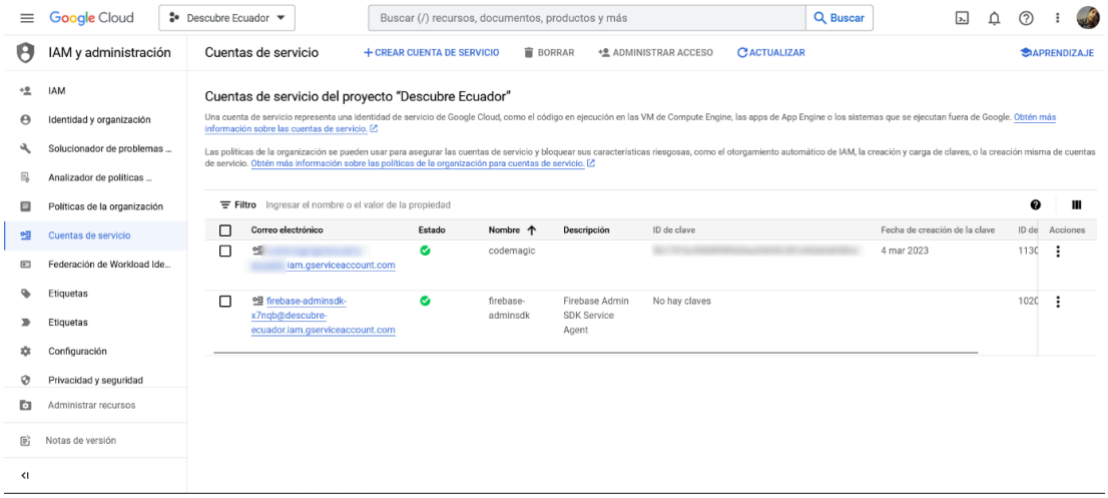


Fig. 22. Creación de cuenta de servicio.

Una vez ejecutada esta acción se redirecciona a Google Cloud apartado cuentas de servicio, en donde debemos generar una nueva cuenta de servicio, una vez creada podemos gestionar las llaves.

Una vez generada la cuenta de servicio debemos asignar la aplicación a la que tendrá acceso, la previamente generada, en la Fig. 23 se puede mostrar la sección donde se ejecuta este proceso.

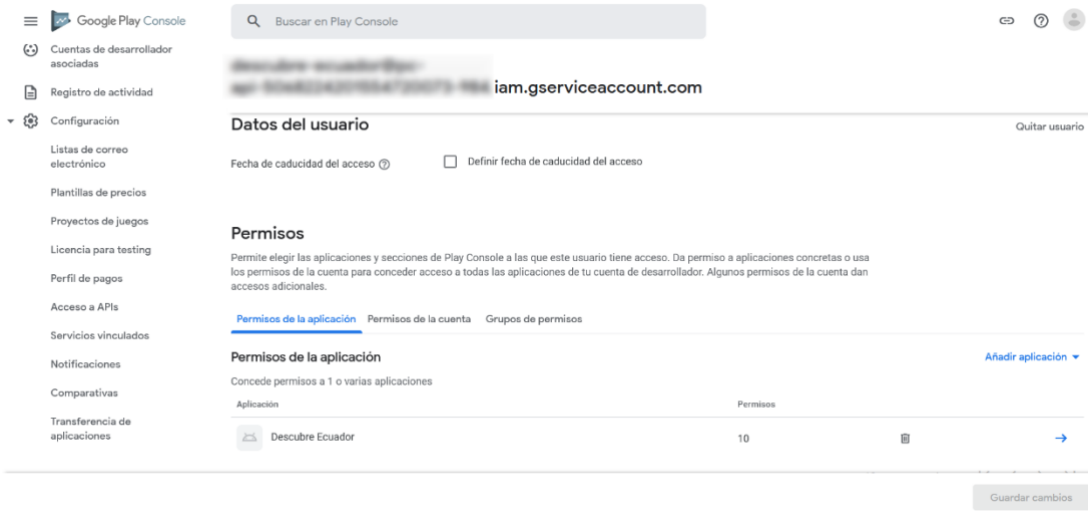


Fig. 23. Asociación de api con aplicación de Google Play Console

Es necesario configurar como variables de entorno los insumos obtenidos previamente, para ello se definió el grupo de variables denominado Firebase credentials, el cual contiene variables relacionadas con Google Cloud y Firebase, como puede verse en la Fig. 24.

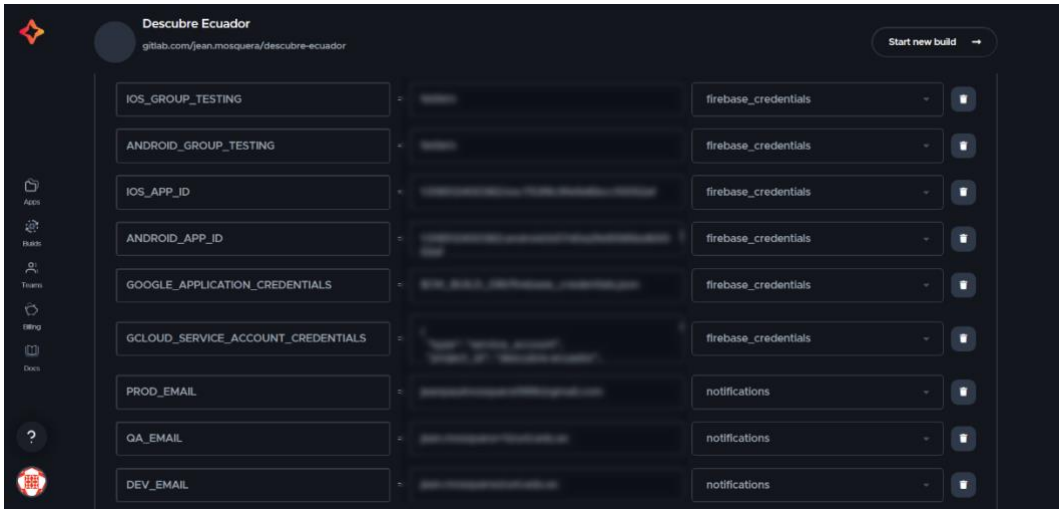


Fig. 24. Configuración de variables para Google y firebase.

Anexo 7. Proceso de configuración de Sonarcloud

Para integrar Sonarcloud en el pipeline, es importante registrarse y generar una cuenta previamente en SonarCloud. Este proceso puede realizarse mediante el siguiente enlace: https://www.sonarsource.com/products/sonarcloud/signup/.

Una vez registrados se procedió a crear una organización, mediante el siguiente enlace: https://sonarcloud.io/create-organization.

Con la organización creada se procedió a obtener el token, mediante el apartado mi cuenta y luego la pestaña de seguridad, como se observa en la Fig. 25.

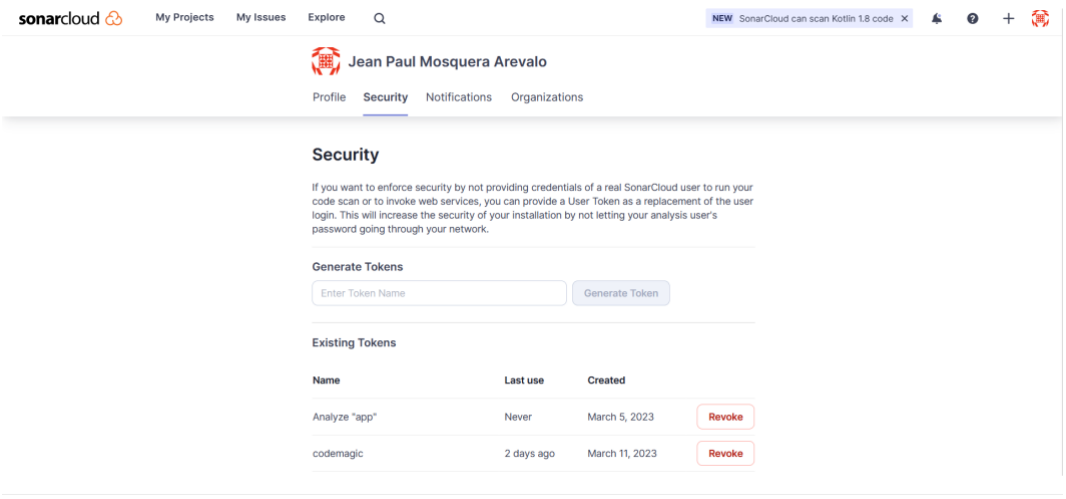


Fig. 25. Generación de token en sonarcloud.

Es importante almacenar el token generado como variable en Codemagic tal como puede verse en la Fig. 26.

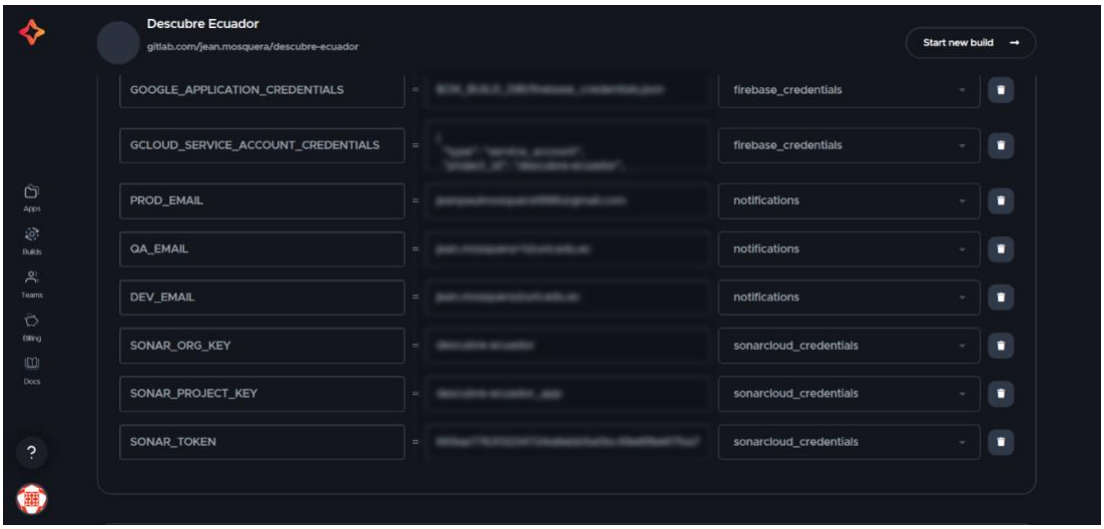


Fig. 26. Configuración de variables para sonarcloud

Anexo 8. Código configuración Pipeline .yaml

```
definitions:
  - &code-quality
    name: Code Quality
    script: |
          flutter analyze
          dart pub global activate dart_code_metrics
          dart pub global run dart_code_metrics:metrics lib --reporter=console-verbose
          flutter test --coverage
          - |
            curl -L https://binaries.sonarsource.com/Distribution/sonar-scanner-cli/sonar-scanner-cli-
4.8.0.2856-macosx.zip \
            --output sonar-scanner.zip \
            && unzip sonar-scanner.zip \
            && rm sonar-scanner.zip \
            && export PATH=$PATH:$PWD/sonar-scanner-4.8.0.2856-macosx/bin \
            && sonar-scanner \
               -Dsonar.host.url=https://sonarcloud.io \
               -Dsonar.projectKey=$SONAR_PROJECT_KEY \
               -Dsonar.organization=$SONAR_ORG_KEY\
               -Dsonar.login=$SONAR_TOKEN \
               -Dsonar.sources=lib \
               -Dsonar.tests=lib \
               -Dsonar.test.inclusions=**/*.test.dart \
               -
Dsonar.test.exclusions=**/*.freezed.dart,**/*.gr.dart,**/*.inject.summary.json,**/*.iconfig.json \
               -Dsonar.dart.coverage.reportPaths=coverage/lcov.info \
               -
Dsonar.exclusions=**/*.g.dart,**/*.freezed.dart,**/*.gr.dart,**/*.inject.summary.json,**/*.iconfig.json \
               -Dsonar.projectConfigFile=sonar-project.properties
  - &keychain-init
    name: Keychain init
    script: |
      find . -name "Podfile" -execdir pod install \;
      keychain initialize
      app-store-connect fetch-signing-files "com.jpmosquera.descubreecuador" \
      --type IOS_APP_DEVELOPMENT \
      --create
      app-store-connect fetch-signing-files "com.jpmosquera.descubreecuador" \
      --type IOS_APP_STORE \
      --create
      keychain add-certificates
  - &get-latest-version-ios
      name: Get the latest build number
      script: |
        LATEST_BUILD_NUMBER=$BUILD_VERSION
        cd ./ios # avgtool must run in the folder where xcodeproj file is located
        agvtool new-version -all $(($LATEST_BUILD_NUMBER + 1))
workflows:
  develop-workflow:
    name: Develop Workflow
    triggering:
      events:
        - push
      branch_patterns:
        - pattern: develop
          include: true
          source: true
    environment:
```

```
    flutter: stable
    groups:
      - sonarcloud_credentials
      - notifications
  cache:
    cache_paths:
      - ~/.pub-cache
      - ~/.sonar/cache
  scripts:
    - *code-quality
    - name: Check Quality Gate status
      script: |
        quality_gate_status=$(curl -s -u $SONAR_TOKEN:
https://sonarcloud.io/api/qualitygates/project_status?projectKey=$SONAR_PROJECT_KEY&branch=de
velop | jq -r '.projectStatus.status')
        echo "Quality Gate status: $quality_gate_status"
        quality_line=$(echo "$quality_gate_status" | grep -o '"status": *"[^"]*"')
        quality_gate=$(echo "$quality_line" | grep -o '"[^"]*"$' | sed 's/"//g')
        echo "Quality Gate status: $quality_gate"
        if [ "$quality_gate" == "ok" ]; then
          exit 1
        else
          echo "Quality gate ok"
        fi
    - name: Build APK
      script: |
        flutter build apk --debug
  artifacts:
    - app/build/outputs/apk/debug/app-debug.apk
  publishing:
    email:
      recipients:
        - $DEV_EMAIL
        - jean.mosquera@unl.edu.ec
      notify:
        success: true
        failure: true
testing-workflow:
  name: Testing Workflow
  triggering:
    events:
      - push
    branch_patterns:
      - pattern: testing
        include: true
        source: true
  environment:
    ios_signing:
      distribution_type: app_store
      bundle_identifier: com.jpmosquera.descubreecuador
    android_signing:
      - descubre-ecuador
    groups:
      - sonarcloud_credentials
      - notifications
      - firebase_credentials
      - ios_credentials
      - app_variables
  cache:
    cache_paths:
```

```
      - ~/.pub-cache
      - ~/.sonar/cache
    scripts:
      - *code-quality
      - name: Check Quality Gate status
        script: |
          quality_gate_status=$(curl -s -u $SONAR_TOKEN:
https://sonarcloud.io/api/qualitygates/project_status?projectKey=$SONAR_PROJECT_KEY&branch=te
sting | jq -r '.projectStatus.status')
          echo "Quality Gate status: $quality_gate_status"
          quality_line=$(echo "$quality_gate_status" | grep -o '"status": *"[^"]*"')
          quality_gate=$(echo "$quality_line" | grep -o '"[^"]*"$' | sed 's/"//g')
          echo "Quality Gate status: $quality_gate"
          if [ "$quality_gate" == "ok" ]; then
            exit 1
          else
            echo "Quality gate ok"
          fi
      - name: Build Android
        script: |
          flutter build apk
      - *keychain-init
      - *get-latest-version-ios
      - name: Build IPA
        script: |
          xcode-project use-profiles
          find . -name "Podfile" -execdir pod install \;
          flutter build ipa --release \
            --build-name=1.0.0 \
            --build-number=$BUILD_VERSION \
            --export-options-plist=ios/ExportOptions.plist
    artifacts:
      - build/ios/ipa/*.ipa
      - build/**/outputs/**/*.ipa
      - /tmp/xcodebuild_logs/*.log
      - flutter_drive.log
      - build/**/outputs/**/*.apk
      - build/**/outputs/**/mapping.txt
      - flutter_drive.log
    publishing:
      firebase:
        firebase_service_account: $GCLOUD_SERVICE_ACCOUNT_CREDENTIALS
        android:
          app_id: $ANDROID_APP_ID
          groups:
            - $ANDROID_GROUP_TESTING
          artifact_type: 'apk'
      app_store_connect:
        key_id: $APP_STORE_CONNECT_KEY_IDENTIFIER
        issuer_id: $APP_STORE_CONNECT_ISSUER_ID
        api_key: $APP_STORE_CONNECT_PRIVATE_KEY
        submit_to_testflight: true
        expire_build_submitted_for_review: true
        beta_groups:
          - $IOS_GROUP_TESTING
      email:
        recipients:
          - $DEV_EMAIL
          - $QA_EMAIL
        notify:
```

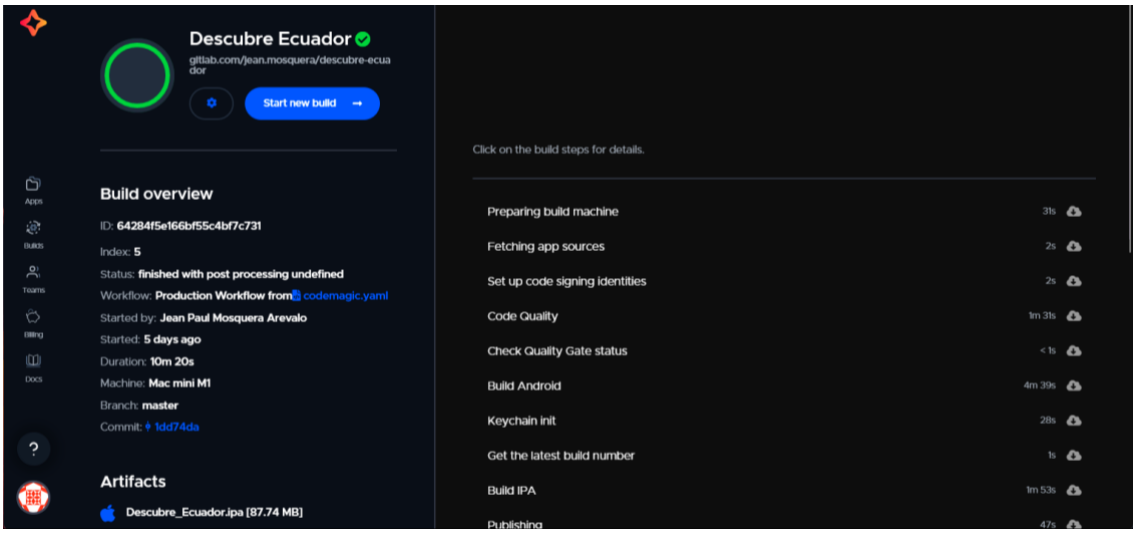
```
        success: true
        failure: true
  production-workflow:
    name:  Production Workflow
    triggering:
      events:
        - push
      branch_patterns:
        - pattern: master
          include: true
          source: true
    environment:
      ios_signing:
        distribution_type: app_store
        bundle_identifier: com.jpmosquera.descubreecuador
      android_signing:
        - descubre-ecuador
      groups:
        - sonarcloud_credentials
        - notifications
        - firebase_credentials
        - ios_credentials
        - app_variables
    cache:
      cache_paths:
        - ~/.pub-cache
        - ~/.sonar/cache
    scripts:
      - *code-quality
      - name: Check Quality Gate status
        script: |
          quality_gate_status=$(curl -s -u $SONAR_TOKEN:
https://sonarcloud.io/api/qualitygates/project_status?projectKey=$SONAR_PROJECT_KEY&branch=m
aster  | jq -r '.projectStatus.status')
          echo "Quality Gate status: $quality_gate_status"
          quality_line=$(echo "$quality_gate_status" | grep -o '"status": *"[^"]*"')
          quality_gate=$(echo "$quality_line" | grep -o '"[^"]*"$' | sed 's/"//g')
          echo "Quality Gate status: $quality_gate"
          if [ "$quality_gate" == "ok" ]; then
            exit 1
          else
            echo "Quality gate ok"
          fi
      - name: Build Android
        script: |
          flutter build appbundle --release
      - *keychain-init
      - *get-latest-version-ios
      - name: Build IPA
        script: |
          xcode-project use-profiles
          find . -name "Podfile" -execdir pod install \;
          flutter build ipa --release \
            --build-name=1.0.0 \
            --build-number=$BUILD_VERSION \
            --export-options-plist=ios/ExportOptions.plist
    artifacts:
      - build/ios/ipa/*.ipa
      - build/**/outputs/**/*.ipa
      - /tmp/xcodebuild_logs/*.log
```

```
    - flutter_drive.log
    - build/**/outputs/**/*.aab
    - build/**/outputs/**/mapping.txt
    - flutter_drive.log
  publishing:
    google_play:
      credentials: $GCLOUD_SERVICE_ACCOUNT_CREDENTIALS
      track: production
      submit_as_draft: true
    app_store_connect:
      key_id: $APP_STORE_CONNECT_KEY_IDENTIFIER
      issuer_id: $APP_STORE_CONNECT_ISSUER_ID
      api_key: $APP_STORE_CONNECT_PRIVATE_KEY
      submit_to_app_store: true
      release_type: SCHEDULED
    email:
      recipients:
        - $DEV_EMAIL
        - $QA_EMAIL
        - $PROD_EMAIL
      notify:
        success: true
        failure: true
```

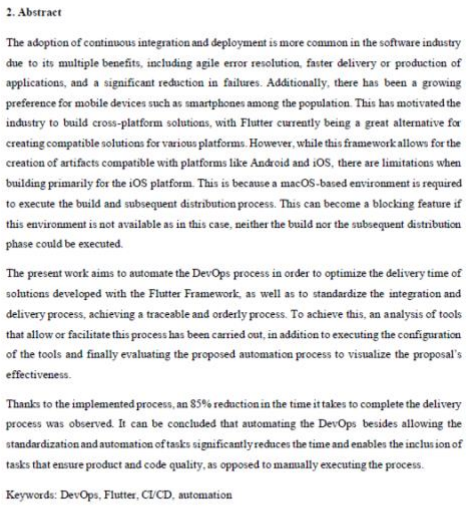
Anexo 9. Video de ejecución de proceso manual para evaluación

https://www.youtube.com/watch?v=6s_k9bwMceI

Anexo 10. Detalle de ejecución de pipeline en dashboard de codemagic



Anexo 11. Certificado de traducción de resumen



2. Abstract

The adoption of continuous integration and deployment is more common in the software industry due to its multiple benefits, including agile error resolution, faster delivery or production of applications, and a significant reduction in failures. Additionally, there has been a growing preference for mobile devices such as smartphones among the population. This has motivated the industry to build cross-platform solutions, with Flutter currently being a great alternative for creating compatible solutions for various platforms. However, while this framework allows for the creation of artifacts compatible with platforms like Android and iOS, there are limitations when building primarily for the iOS platform. This is because a macOS-based environment is required to execute the build and subsequent distribution process. This can become a blocking feature if this environment is not available as in this case, neither the build nor the subsequent distribution phase could be executed.

The present work aims to automate the DevOps process in order to optimize the delivery time of solutions developed with the Flutter Framework, as well as to standardize the integration and delivery process, achieving a traceable and orderly process. To achieve this, an analysis of tools that allow or facilitate this process has been carried out, in addition to executing the configuration of the tools and finally evaluating the proposed automation process to visualize the proposal's effectiveness.

Thanks to the implemented process, an 85% reduction in the time it takes to complete the delivery process was observed. It can be concluded that automating the DevOps besides allowing the standardization and automation of tasks significantly reduces the time and enables the inclusion of tasks that ensure product and code quality, as opposed to manually executing the process.

Keywords: DevOps, Flutter, CI/CD, automation

Certificate of Accurate Translation

I VICTOR ANDRES CARVAJAL PEÑAFIEL, identification number: 0604540369, owner of the SETEC certification ID: MDT-3104-CCL-255680: attest that I have translated the attached document, and that to the best of my knowledge, ability, and belief this a true, accurate, and complete translation with code: VC-0120-04-23-TR of an original document in Spanish with title: 2. Resumen, to English with the title: 2. Abstract, which was provided to me.

TRADUCCIÓN CERTIFICADA

MDT-3104-CCL-255680

Victor Andres Carvajal Peñafiel
Acreditación MDT-3104-CCL-255680
TRADUCCIÓN E INTERPRETACIÓN DE IDIOMAS

Enlace de acceso : https://drive.google.com/file/d/1ld1Krc9QFFS9aaDALtoBZ8-HGuS5tgY1/view?usp=share_link