Hibot Mobile Farm

Julian Andres Quiroz Taborda

Informe de práctica para optar al título de Ingeniero de Sistemas

Asesores

Roberto Flórez Rueda, Profesional en Ingeniería Civil

Universidad de Antioquia

Facultad de Ingeniería

Ingeniería de Sistemas

Medellín, Antioquia, Colombia

2022

TABLA DE CONTENIDO

LISTA DE TABLAS

LISTA DE FIGURAS

SIGLAS, ACRÓNIMOS Y ABREVIATURAS

API	Application Programming Interface
AWS	Amazon Web Services
IMAT	Interactive Mobile App Testing
Jdk	Java Development Kit
Mb	Megabytes
QA	Quality Assurance
SSL	Secure Sockets Layer
UdeA	Universidad de Antioquia
USD	United States Dollar

RESUMEN

Hibot es un producto desarrollado por la compañía colombiana Sofka Technologies, el cual se ofrece como un servicio de chat automático empresarial. Durante su ciclo de desarrollo, se evidenció la necesidad de escalar la ejecución de sus pruebas automatizadas a alternativas en nube, buscando acoplarse a metodologías como DevOps, y a su vez, incrementar el grado de confiabilidad de su aplicación móvil. Tras investigaciones internas del equipo, que fueron abandonadas con anterioridad, no se obtuvieron resultados satisfactorios para solventar esta necesidad. En consecuencia, se plantea el presente proyecto, el cual tuvo como finalidad, investigar herramientas en nube, que permitan crear granjas de dispositivos móviles, para así lograr la integración de las pruebas automatizadas que se tienen para el producto mencionado, pasando por configuraciones internas al código fuente, de infraestructura y de los dispositivos remotos que sirven para ejecutar las pruebas. Esto, con el propósito de ayudar en el mejoramiento de los procesos de certificación del equipo, brindando una herramienta flexible, escalable y que genere valor.

Palabras clave — **Automatización de pruebas, AWS Device Farm, calidad de software, DevOps, granja de dispositivos, Hibot.**

ABSTRACT

Hibot is a software product developed by colombian company Sofka Technologies, which is offered as a business automatic chat service. During its development cycle, Hibot team evidenced the need to execute their automated tests by using cloud services, and looking to be aligned with current methodologies such as DevOps, also, trying to increase application's coverage degree. After some internal investigations, which were discarded, they had not successfully results for their need. According to the above, the current project was proposed, and its purpose was to investigate cloud tools able to create mobile device farms, seeking to achieve integration with Hibot's automated tests, considering different things like source code configuration, infrastructure and remote device's settings. This, looking for improve the team's certificacion process, and giving a flexible tool, scalable and value generator.

Keywords — **AWS Device Farm, DevOps, Hibot, Mobile Farm, Software Quality Assurance, Test automation.**

I. INTRODUCCIÓN

Sofka es una empresa colombiana del sector TI, cuyo nombre deriva de la unión de las palabras software y kaizen, del japonés mejora continua, filosofía interna de la compañía. Esta, a lo largo de su trayectoria, se ha desempeñado ofreciendo servicios del área, como el desarrollo de software y el aseguramiento de la calidad. Además de estos servicios, Sofka cuenta con un producto interno llamado Hibot, cuyo propósito es el de servir como asistente automático de chats organizacionales. Dicho producto, cuenta con un equipo conformado por un grupo de desarrolladores, analistas de calidad, arquitectos y equipo comercial.

Durante el ciclo de desarrollo del software, y a medida que avanzaba el proceso de certificación, surgió dentro del equipo, la necesidad de ejecutar sus pruebas automatizadas a través de servicios en nube, pues hasta el momento, todo el proceso de certificación de su módulo para dispositivos móviles, se realizaba mediante ejecuciones en ambiente local, y utilizando los smartphones que son propiedad de cada miembro del equipo.

El propósito con el cual se ha desarrollado el presente proyecto, es el de suplir dicha necesidad, integrando su proyecto de automatización de pruebas móviles con herramientas en nube. Esto, con el fin de brindar el mayor valor posible al equipo, y buscando que sirva como pilar para mejorar sus procesos de certificación. De igual manera, procurando que sea una solución escalable en el tiempo, considerando un eventual crecimiento del producto. Y al mismo tiempo, documentando todo el proceso de integración, de tal forma que sirva como guía para otros equipos de la compañía, en caso de llegar a tener necesidades similares en el futuro.

II. PLANTEAMIENTO DEL PROBLEMA

A lo largo del desarrollo de Hibot, se han construido diferentes pruebas automatizadas para los módulos que componen la totalidad del producto, entre los cuales se encuentran el aplicativo móvil y el web. Este último, cuenta con pruebas automatizadas que se ejecutan en máquinas virtuales a través de pipelines montados en la plataforma de Azure. Sin embargo, en el caso de las pruebas elaboradas para el aplicativo móvil, su ejecución se realizó localmente desde un principio, y contando únicamente con los dispositivos físicos que son propiedad de los miembros del equipo. Los analistas de calidad del proyecto reconocieron que allí existía un posible sesgo, al contar con una cantidad minúscula de dispositivos físicos. Adicionalmente, se identificó que dicha situación generaba un desalineamiento del proyecto con un principio básico de las pruebas de software, la repetibilidad, y otro de automatización, siendo necesario realizar configuraciones adicionales para cada equipo en el cual se deseaba ejecutar dicho autómata.

A. Antecedentes

A raíz del problema planteado, el equipo se vio en la necesidad de investigar la forma de ejecutar sus pruebas en una granja de dispositivos en la nube, que les permitiese aumentar el rango de cobertura de su producto, y acoplarse a los lineamientos básicos de pruebas y automatización. Para ello, dentro de sus actividades de QA, se inició con un spike, buscando la solución a esta necesidad, donde se exploraron herramientas bastante posicionadas en el mercado, como lo son las de Azure y BrowserStack. No obstante, debido a la alta complejidad y el nivel de trabajo acumulado en el equipo, la actividad debió de ser abandonada antes de que rindiera frutos. Por tal motivo, el proyecto fue ingresado al backlog de trabajos de grado disponibles dentro del equipo de QA de Sofka Technologies.

En cuanto a arquitectura se refiere, la aplicación móvil de Hibot ha sido construida para el sistema operativo de Android, no se consideran móviles que usen iOS. Igualmente, el proyecto de automatización se encuentra construido bajo las herramientas de Gradle, Java, un framework de automatización interno de la compañía, basado en Selenium y Appium, y finalmente Cucumber.

III. OBJETIVOS

A. Objetivo general

Implementar una solución, basada en servidores en nube, para la ejecución de pruebas automatizadas móviles, con la cual se puedan mejorar los procesos de DevOps del equipo interno de Hibot, y de otros equipos que lo requieran en un futuro.

B. Objetivos específicos

- Investigar soluciones basadas en nube, que permitan crear granjas de dispositivos, con los cuales ejecutar pruebas automatizadas de aplicaciones móviles.
- Con base en los resultados obtenidos durante la fase de investigación, construir la solución, adaptada a las necesidades y herramientas utilizadas por el equipo de Hibot.
- Generar una solución escalable, que pueda ser aplicada en diferentes equipos de automatización de pruebas móviles.
- Construir arquitectura base de notificación, que informe al equipo de pruebas acerca del resultado general de las regresiones.

IV. MARCO TEÓRICO

A. Calidad de Software

Pressman, en su libro Ingeniería del Software, define el concepto de calidad de software como el grado de concordancia que tiene un sistema con los requisitos funcionales y de rendimiento, que se definen durante la fase de análisis y diseño, y que buscan cumplir las expectativas y necesidades del cliente [1]. Adicional a esto, y antes de ahondar en el tema de las pruebas de software, es importante recordar las fases principales por las cuales debería de pasar un producto de software, las cuales nos presentan E. Khan y F. Khan en su artículo Importance of software Testing in software development life cycle [2]:

1. Análisis y recolección de requerimientos.
2. Especificación de requerimientos.
3. Estudios de factibilidad.
4. Diseño del software.
5. Construcción del software.
6. Certificación.
7. Depuración.
8. Despliegue y mantenimiento.

Dentro de dicho ciclo, existe un rol fundamental, encargado de velar por la concordancia entre todos los niveles mencionados anteriormente. Tal rol, es el de analista de calidad, y su propósito es el de encontrar y eliminar errores y defectos en fases tempranas, de tal modo, que estos no se conviertan en problemas más graves en el futuro, como si se tratase de un efecto bola de nieve [3]. Las pruebas de software, son entonces, una herramienta transversal que busca eliminar disparidades entre las diferentes fases del desarrollo, y que durante este ciclo pasan por tres estados:

1. Análisis y comprensión del proyecto.
2. Diseño de las pruebas.
3. Ejecución de las pruebas.

B. Automatización de pruebas

A su vez, es importante saber que estas pruebas se pueden realizar de manera manual o automatizada. Sin embargo, para efectos del presente proyecto, se hará énfasis en las bondades de la automatización de pruebas. Entre las principales ventajas, destacadas por Fewster y Graham en su libro Software Test Automation [4], se destacan:

- Si bien, en un principio su construcción suele ser más costosa, realizando la comparación contra la forma manual, una correcta construcción de estas, ahorra tiempo y recursos valiosos en fases más maduras del ciclo de desarrollo.
- Los autómatas son mucho más rápidos que un humano, tanto en la interacción con el aplicativo, como en la detección y documentación de errores.
- Las regresiones pueden ser ejecutadas por cualquier miembro del equipo, cuantas veces sea necesario, y sin necesidad de una contextualización previa acerca del funcionamiento del aplicativo.
- Hace más fácil la reproducción de errores.
- Los robots de pruebas son más precisos, y disminuyen el riesgo de pasar algo por alto.
- Las pruebas automatizadas hacen parte fundamental del ciclo de DevOps, el cual busca maximizar la eficiencia de los equipos de desarrollo.

Adicional, como nos presentan Riveiro y Reigosa en el foro de BBVA Next Technologies, Mike Cohn realizó el diseño de una pirámide de pruebas, y plasmó en ella, una propuesta sobre cómo deberían de estar conformados los diferentes niveles de automatización. En el nivel más bajo de la pirámide, se encuentra el más fundamental de cualquier producto de software, que es el nivel funcional y de clases. Según Cohn, es importante tener una base sólida de pruebas unitarias, que minimicen el riesgo de encontrar defectos en niveles superiores, además de que son más fáciles y menos costosas de realizar. Luego, se escala al nivel de integración de componentes, reduciendo notoriamente el número de pruebas en este nivel, y por último, se llega al nivel de la interfaz, donde se reduce aún más la cantidad de pruebas automatizadas, pues su costo computacional y de construcción es mucho más elevado [5].

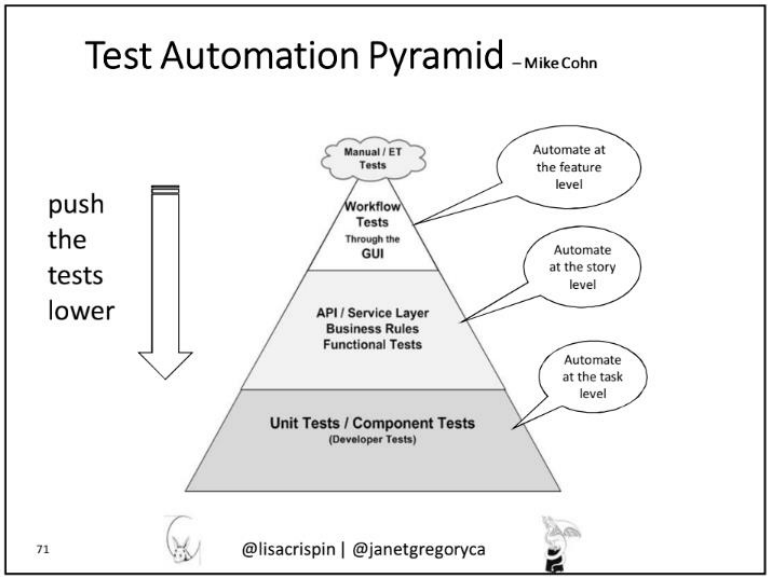

Fig. 1. Pirámide de Cohn

Nota: Fuente [5]

C. DevOps

Anteriormente, se mencionó a DevOps como una de las principales ventajas de la automatización de pruebas, pero, ¿Qué es DevOps? Este término nace de la combinación de las palabras en inglés development (desarrollo) y operations (operaciones), y su finalidad es lograr la unión entre personas, procesos y tecnología, para brindar un valor adicional a los clientes. Este, permite que roles anteriormente aislados, se coordinen y contribuyan en conjunto a producir mejores productos y más confiables [6].

Algunas de las prácticas principales que promueve DevOps son:

- *Continuous integration:* Consiste en compilar y ejecutar pruebas en lapsos cortos de tiempo, sobre la versión más actualizada de un proyecto de software. Lo que permite tener un monitoreo continuo sobre las métricas de calidad del proyecto, además de la detección temprana de errores.
- *Continuous delivery:* Esta es una práctica en la cual, el software desarrollado iterativamente, puede ser entregado a producción en cualquier momento. Esto no

implica necesariamente un despliegue a producción con cada cambio, dependerá de lo que sea definido por los encargados del proyecto.

- *Continuous management:* Hace referencia a todo el conjunto de elementos que se deben configurar para que el proyecto sea exitoso. La gestión de la configuración se compone principalmente de un repositorio del código fuente, repositorio de artefactos y una base de datos de la gestión de la configuración.

Al seguir estas prácticas y herramientas que nos brinda DevOps, los equipos se ven en la capacidad de afrontar de una mejor forma los requerimientos del cliente, aumentar la productividad y confianza en sus productos, y alcanzar los objetivos en un tiempo menor.

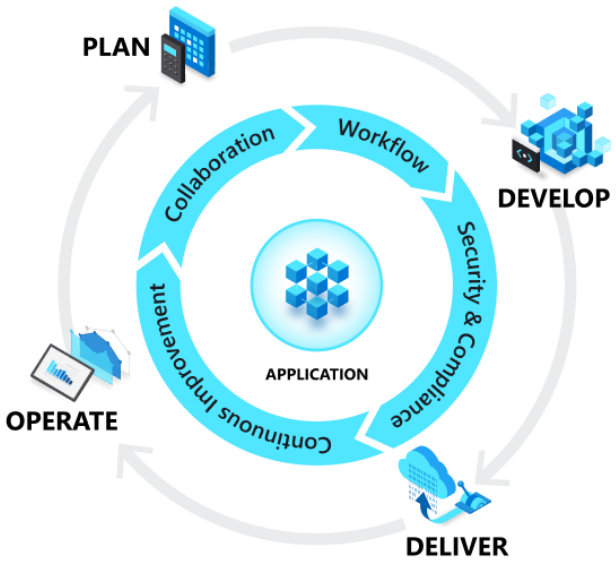

Fig. 2. Proceso de DevOps

Nota: Fuente [6]

Además de aprovechar las bondades que ofrece DevOps, también es pertinente aprovechar aquellas que ofrece una herramienta como la computación en nube, aquella que permite alojar servicios, código fuente, infraestructura, entre otros, en servidores públicos o privados, y que los hace accesibles desde casi cualquier parte del mundo. Las ventajas principales que tiene el ejecutar pruebas en la nube son las siguientes:

- La ejecución de las pruebas no depende de la arquitectura local del equipo, las máquinas virtuales pueden ser configuradas de tal forma que se garantice el correcto funcionamiento bajo condiciones específicas.
- Las pruebas son accesibles para todo el equipo de desarrollo, con lo cual pueden ejecutar regresiones, o consultar los resultados de pruebas anteriormente realizadas.
- Se previenen situaciones que puedan afectar el flujo de trabajo del equipo, como la pérdida de la máquina en la cual se realizan las pruebas.
- Permite ejecuciones paralelas en n dispositivos, que en un ordenador físico se ven limitadas por sus recursos.

V. METODOLOGÍA

La metodología de trabajo implementada durante la ejecución del proyecto, ha sido la de Scrum, una metodología ágil con la cual se cuenta con una amplia experiencia dentro de los servicios que ofrece la compañía. Para ello, se definieron sprints con duración de dos semanas, para un total de trece durante la totalidad de la ejecución del proyecto.

De igual manera, para el desarrollo del proyecto de Hibot Mobile Farm, se definieron cuatro fases principales, que comprenden:

- *Fase I – Preparación e inicio:* Incluye todos los ejercicios iniciales de puesta a punto para iniciar la ejecución del proyecto, como solicitud de permisos y contextualizaciones.
- *Fase II – Investigación:* En esta fase se encuentran los procesos de investigación de herramientas, análisis de costos, análisis de compatibilidad, y actividades similares.
- *Fase III – Solución:* Durante esta fase se realizó todo el proceso de integración del proyecto de automatización, incluyendo el estudio de la documentación oficial, preparación del proyecto de automatización, preparación del pipeline, desarrollo de notificación, y demás actividades afines.
- *Fase IV – Cierre:* Finalmente, se encuentra la fase de cierre, donde se encuentran agrupadas todas las tareas alusivas a la finalización del proyecto, como lo es la socialización de los resultados con el equipo de Hibot, documentación del proceso, recopilación de informes de evaluación y seguimiento, y la redacción del presente documento.

Adicional, es importante mencionar que las actividades y ceremonias correspondientes a la metodología ágil de Scrum, como lo son la planeación, revisión, retrospectiva, entre otras, se realizaron de manera transversal a las fases anteriormente mencionadas.

VI. RESULTADOS

Durante la primera fase, específicamente en el planteamiento de la propuesta, se definieron cuatro objetivos específicos para desarrollar durante el presente proyecto. Estos, comprenden las tareas de investigación, integración, escalabilidad y notificación. A continuación, se detalla paso a paso, los resultados obtenidos para cada numeral.

A. Investigación

Durante el trabajo de investigación, se tomaron en cuenta las tres herramientas mejor posicionadas en el mercado, que permitieran realizar la creación de granjas de dispositivos móviles. Estas fueron AWS Device Farm, BrowserStack Interactive Mobile App Testing y Azure DevOps. Posteriormente, se hizo un análisis de las diferentes características que ofrecía cada plataforma, donde se consideraron, entre otras cosas, los costos en su utilización, dispositivos disponibles para prueba, y compatibilidad del proyecto de automatización con la infraestructura de la herramienta. Allí, se obtuvieron los siguientes resultados:

TABLA I
CARACTERÍSTICAS DE LAS HERRAMIENTAS

Plataforma	Cantidad de dispositivos físicos disponibles	Compatible con proyecto de automatización
AWS Device Farm	100 +	Sí
Azure DevOps	0 (Máquinas virtuales)	Sí
BrowserStack IMAT	2000 +	Es necesario migrar a Maven

TABLA II
PLANES DISPONIBLES POR HERRAMIENTA (USD)

Plataforma	Plan mensual	Otros planes	Plan bienvenida
AWS Device Farm	$ 250	$ 0,17 por minuto	1000 minutos de prueba
Azure DevOps	$ 52	-	30 días calendario
BrowserStack IMAT	$ 199 $ 125 $39	-	100 minutos de prueba

Adicional a esto, es importante resaltar que, a diferencia de las herramientas de Azure y BrowserStack, la herramienta de Amazon Web Services permite la ejecución de pruebas paralelas a partir de su plan básico, abriendo la posibilidad de tener ejecuciones más rápidas, sin algún costo adicional.

De acuerdo con la información presentada anteriormente, y después de realizar un análisis de costo-beneficio, la herramienta seleccionada para realizar la solución fue AWS Device Farm, debido a su compatibilidad con la arquitectura de automatización, plan bienvenida de mil primeros minutos de servicio gratuito, ejecución paralela de pruebas, y plan complementario de cobro por demanda, que puede resultar provechoso para el equipo de Hibot, al no ser un proyecto demasiado grande.

B. Integración

Para la tarea de integración, el equipo de calidad de Hibot realizó la entrega de tres escenarios automatizados completamente estables, que cuentan con una complejidad media, y donde se puede verificar el correcto funcionamiento de los diferentes componentes que pueden contener los demás escenarios de prueba, por ejemplo, la interacción con elementos gráficos, consumo de servicios web, o la conexión de AWS con las bases de datos.

Una vez seleccionada la herramienta, y teniendo disponibles los escenarios base a ejecutar, comienza el estudio de las configuraciones necesarias para poder ejecutar allí el proyecto de automatización.

Otro de los criterios que determinó la selección de AWS Device Farm como herramienta de solución, fue la amplia documentación que se encuentra disponible en la red, tanto en su portal oficial, como en comunidades y foros afines a TI.

"AWS Device Farm es un servicio de prueba de aplicaciones que le permite mejorar la calidad de sus aplicaciones web y móviles al probarlas en un amplio rango de navegadores de escritorio y dispositivos móviles reales, sin tener que aprovisionar y administrar ninguna

infraestructura de pruebas. El servicio permite ejecutar sus pruebas de forma concurrente en varios navegadores de escritorio y dispositivos reales para agilizar la ejecución de su suite de pruebas. Además, genera videos y registros que lo ayudarán a identificar rápidamente los problemas con su aplicación." [7]

Esta plataforma permite la ejecución de código fuente escrito con los lenguajes de Java, Python, Ruby y JavaScript, este último utilizando Node.js. Para ello, es necesario realizar la carga de por lo menos dos archivos:

- El apk de la aplicación que se desea probar.
- Un archivo zip que contenga el proyecto de automatización a ejecutar, con sus respectivas dependencias como archivos jar.

Posterior a la carga de ambos archivos, es necesario configurar un archivo yaml dentro de AWS Device Farm, donde se define el paso a paso a seguir durante la ejecución de la prueba, y que comprende tres fases:

- *pre-test:* En esta fase, se prepara el dispositivo para realizar la ejecución. Aquí se realiza la descarga del archivo zip del proyecto, posteriormente se descomprime, y finalmente, se levanta la instancia de Appium.
- *test:* En esta fase se realizan las operaciones relacionadas a las pruebas, como lo es la definición de variables de entorno utilizadas durante la ejecución, además de la propia ejecución de la automatización.
- *post-test:* Durante esta fase se realiza la recolección de evidencias, que quedarán guardadas en el job de ejecución de AWS.

El paso siguiente es configurar el pool de dispositivos en los cuales se desea probar la aplicación. La plataforma cuenta actualmente con un aproximado de cien (100) dispositivos disponibles, que varían entre los fabricantes de Google Pixel, Motorola, OnePlus, Samsung y Sony. Además, se especifica su versión de Android y el estado de disponibilidad del equipo:

- *Highly Available:* Completamente disponible.
- *Available:* Ocupado o en segundo plano, pero próximo a estar disponible.
- *Busy:* Ocupado.

A continuación, se presentan las opciones de configuración del dispositivo móvil, donde se configurarán opciones de red, bluetooth, idioma, entre otros. Finalmente, se confirma la ejecución, la cual se realizará de forma paralela en caso de contar con la disponibilidad de los equipos.

AWS Device Farm ofrece un amplio rango de evidencias de ejecución. Entre ellas, destaca la opción de video, que por defecto, se encuentra activa, y estos podrán ser consultados en el histórico de ejecuciones. Además de ello, cuenta con otras opciones de captura de evidencias, como lo son la salida de consola, logs de conectividad, capturas de pantalla y performance. Todo ello, representa una herramienta muy completa, intuitiva, y que permite identificar rápidamente los errores, en caso de que se presenten.

Durante esta tarea de integración se presentaron algunos obstáculos, tanto de arquitectura, como de código fuente, y se describen a continuación.

Las dificultades iniciales, se presentaron al tratar de ejecutar los test utilizando las opciones de ejecución de código fuente Java, que pueden ser ejecutadas utilizando JUnit o TestNG. El problema principal radica en que estas opciones fueron diseñadas para ser ejecutadas utilizando Maven, que difiere de las tecnologías implementadas en la automatización de Hibot, siendo Gradle su gestor de dependencias. Por este motivo, fue necesario investigar alternativas que fuesen compatibles. Dicha investigación, condujo a la opción que permite ejecutar JavaScript, ya que, según la documentación expuesta por el equipo de Amazon, esta opción permite ejecutar casi cualquier comando desde su cmd. Lo que habilitó la ejecución del gradle wrapper dentro de la máquina de AWS.

Posterior a la ejecución del proyecto de automatización en el Device Farm, se evidenciaron los primeros errores a nivel de código fuente, que se relacionan con las configuraciones y propiedades que Appium exige para levantar una instancia, como lo son la url y el identificador

único del dispositivo. Para ello, se realizó una refactorización sobre el código, que permitiese consultar estas propiedades mediante las variables de entorno expuestas por el Device Farm.

Finalmente, el obstáculo más grande que se presentó para la integración de la automatización de Hibot con AWS Device Farm, fue realizar la conexión con la base de datos del aplicativo, que se encuentra en la plataforma de MongoDB. Tras numerosos intentos, se logró identificar que el error de conexión, se presentaba debido a la versión del jdk expuesta en la máquina de AWS Device Farm, la cual carece del certificado SSL en su archivo cacerts. Para la solución de este obstáculo, fue necesario cargar un jdk más actualizado junto con el proyecto de automatización, y forzando su ejecución con la nueva versión. Esto, debido a que la plataforma de Amazon no permite realizar la actualización del software.

Una vez solucionados estos inconvenientes, se logró la ejecución exitosa de los escenarios de prueba que fueron provistos en un principio por el equipo de QA de Hibot.

C. Escalabilidad

Buscando la descentralización y escalabilidad interna del conocimiento, se realizó la documentación de la configuración necesaria para ejecutar un proyecto de automatización, de características similares a Hibot, dentro de la herramienta de AWS Device Farm. En dicho documento, se detallan las configuraciones que se deben realizar dentro del proyecto, tanto a nivel de código como de infraestructura, configuraciones que se deben realizar en los pipelines de AWS Device Farm, además de una guía de errores frecuentes que se encontraron durante la elaboración del presente proyecto, y su respectiva solución. Por último, al final del documento mencionado, se adjunta enlace a un repositorio privado, alojado en el Gitlab de la compañía, donde se encuentra el proyecto de Hibot, completamente configurado y listo para ser ejecutado en la herramienta de Amazon.

D. Notificación

AWS Device Farm no cuenta con una herramienta propia capaz de realizar la notificación por correo, que permita a los interesados, enterarse acerca de la finalización de los test, razón por la cual, el desarrollo de este objetivo se debió de implementar al interior del código fuente del proyecto. Para esto, se utilizó la API que Google ofrece para realizar la conexión con Gmail, utilizando el lenguaje de programación Java. Gracias a ello, es posible la lectura y envío de correos electrónicos, luego de configurar una cuenta de Gmail.

La notificación se realiza cuando finaliza la prueba en cada dispositivo, es decir, en caso de tener tres dispositivos en el pool de AWS, se realizan tres notificaciones. En dicho mensaje, se especifica el identificador único del móvil que ejecutó la prueba, y se adjuntan en un archivo zip, los logs y capturas de pantalla generados durante la prueba.

VII. CONCLUSIONES

Una vez finalizado el presente proyecto, y teniendo en cuenta los procesos y herramientas utilizados durante su ejecución, se pueden definir las siguientes conclusiones:

- El aseguramiento de la calidad es una parte fundamental del ciclo de desarrollo de software, que brinda valor al equipo identificando errores de manera temprana.
- Como parte del ciclo de DevOps, es recomendable que se cuente con pruebas automatizadas, tomando en cuenta los beneficios que estas ofrecen, y el valor que brindan en fases críticas del ciclo de desarrollo.
- AWS Device Farm es la herramienta que mejor se adapta en la actualidad a las necesidades del equipo de Hibot, siendo una herramienta bastante provechosa en cuanto a costo-beneficio y versatilidad.
- El equipo de Hibot cuenta ahora con una herramienta que le permite ejecutar sus pruebas automatizadas en nube, y ampliar el rango de cobertura de pruebas de su aplicación móvil.
- Si bien se ha alcanzado una solución útil durante la ejecución del presente proyecto, existe una serie de posibles mejoras que pueden ser implementadas por el equipo, en búsqueda de optimizar los tiempos de certificación y documentación de evidencias, los cuales, se han descrito en el numeral de Recomendaciones.
- Los resultados obtenidos durante el presente proyecto, resultan ser de gran valor para la compañía, pues se abre la puerta a la implementación de dichos resultados dentro del equipo de calidad de software, ampliando el catálogo de servicios de la compañía, y teniendo un efecto positivo a nivel económico.

VIII. RECOMENDACIONES

El objetivo planteado para el presente trabajo fue el de lograr la integración de las pruebas automatizadas de Hibot con una herramienta en nube, que sirviese como punto inicial para optimizar los procesos de certificación del módulo móvil del proyecto. Objetivo que se ha conseguido satisfactoriamente. No obstante, existen algunas oportunidades de mejora que se podrían considerar para un trabajo futuro dentro del equipo, y se describen a continuación:

- En primer lugar, se propone investigar acerca de una posible integración de AWS Device Farm con el repositorio de Azure donde se encuentra alojado el proyecto. Esto, con el fin de mejorar los tiempos de carga del proyecto en la herramienta de AWS, pues el zip generado tiene actualmente un tamaño aproximado de 800Mb.
- Además, se propone implementar a nivel de código fuente, un componente que permita transformar los archivos de evidencia generados por cucumber a una extensión .txt, dado que se generan como archivos .js, y estos generan el rebote del correo enviado. Razón por la cual, no se incluyeron dentro del zip de evidencias generado al finalizar la ejecución.
- Investigar alternativas que permitan realizar la conexión con MongoDB, diferentes a la carga del jdk, pues esto afecta en gran medida el tamaño del archivo zip, y por consiguiente, los tiempos de carga.
- Finalmente, investigar alternativas que permitan la conexión de la herramienta de Device Farm con repositorios privados de artefactos. Además, realizar un comparativo, para validar que el peso perdido en el archivo zip, sí represente una ganancia frente a los tiempos de descarga de dependencias durante la ejecución.

REFERENCIAS

[1] R. Pressman, “Ingeniería Del Software I”, Ingeniería Del Software I, 2010.

[2] M. E. Khan & F. Khan, “Importance of Software Testing in Software Development Life Cycle”, International Journal of Computer Science, T. 11, 2014.

[3] G, Lomprey & S. Hernandez. “La importancia de la calidad en el desarrollo de productos de software”. México: Universidad de Montemorelos. 2008.

[4] Fewster, M. & Graham, D., 1999. Software test automation. New York: ACM Press.

[5] C. Riveiro & C. Reigosa. “BBVA Next Technologies: El gran dilema del QA: ¿Automatización o testing manual?” 2019. Disponible en: https://bit.ly/3aoWxJL

[6] Microsoft Azure. “¿Qué es DevOps? Explicación de DevOps” [En línea]. Disponible en: https://bit.ly/3PfSaiH

[7] Amazon Web Services. “AWS Device Farm” [En línea]. Disponible en: https://go.aws/3ItfA22

[8] Microsoft Azure. “Azure DevOps Services” [En línea]. Disponible en: https://bit.ly/3PhRUzP

[9] BrowserStack. "App Live: Interactive Mobile App Testing” [En línea]. Disponible en: https://bit.ly/3PlMsMt

[10] Amazon Web Services. “Documentación de AWS Device Farm” [En línea]. Disponible en: https://go.aws/3RkD8Ku