Framework multiplataforma de pruebas automatizadas para Ticketmaster

Emanuel Gaviria Echeverri

Informe de práctica presentado para optar al título de Ingeniero de Sistemas

Asesor

Luz Viviana Cobaleda Estepa, Doctora en Ingeniería Electrónica

Universidad de Antioquia

Facultad de Ingeniería

Ingeniería de Sistemas

Medellín, Antioquia, Colombia

2024

Cita	Gaviria Echeverri [1]
Referencia Estilo IEEE (2020)	[1] E. Gaviria Echeverri, “Framework multiplataforma de pruebas automatizadas para Ticketmaster”, Informe de práctica profesional, Ingeniería de Sistemas, Universidad de Antioquia, Medellín, Antioquia, Colombia, 2024.

Project Manager Live Nation Team, Globant: Zulim Abigail Hernandez Nava

Agradecimientos

Agradezco a Globant por permitirme utilizar mi experiencia dentro del proyecto al que pertenezco para dar cumplimiento a mi proceso de práctica académica. Agradezco a Abigail Hernandez, Project Manager en Live Nation, por su disposición y acompañamiento en esta tarea, y a la profesora Luz Viviana Cobaleda por su asesoría a lo largo del proceso

TABLA DE CONTENIDO

LISTA DE FIGURAS

SIGLAS, ACRÓNIMOS Y ABREVIATURAS

NA	Norte América
MX	México
UK	Reino Unido
IE	Irlanda
AU	Australia
NZ	Nueva Zelanda
CI	Integración continua
UI	Interfaz de usuario
MR	Merge Request

RESUMEN

Tomando en cuenta la necesidad de Ticketmaster, cliente de Globant, de aprovechar los beneficios de la automatización de pruebas para mejorar su proceso de despliegue, este documento expone la forma en que se ha abordado el diseño de un framework que permita ejecutar casos de prueba previamente analizados y priorizados, en plataformas móviles (iOS y Android). La aplicación de Ticketmaster está disponible en diferentes regiones que pueden o no compartir casos de prueba entre sí, y en un principio se espera que el nivel de cobertura sea similar en todas ellas. No obstante, debido a la diferencia en la cantidad de casos de prueba por región, se encuentra que no es posible alcanzar un alto nivel de cobertura general, sin antes nutrir el conjunto de casos de prueba para las regiones que no han sido prioritarias en los esfuerzos de testing hasta el momento. También se discute la forma en que se aprovecha la integración continua para una detección temprana de defectos durante el proceso de desarrollo.

Palabras clave — **Pruebas automatizadas, casos de prueba, plataformas móviles, integración continua**

ABSTRACT

One of Globant's clients, Ticketmaster, had the necessity of leveraging the benefits of automated testing in order to improve its release process. With that as a baseline, this document presents the strategy behind the design of a test automation framework which allows the execution of previously analyzed and prioritized test cases on mobile platforms (iOS and Android). The Ticketmaster app is available in different regions that may or may not share test cases among them, and it is initially expected to reach the same level of coverage across all regions. However, since the number of test cases per region isn't consistent, reaching a high level of coverage across-the-board will be an unlikely goal until test suites for neglected regions are improved upon. This paper also delves into the advantages of continuous integration for early bug detection during the development cycle.

Keywords — **Automated testing, test cases, mobile platforms, continuous integration.**

I. INTRODUCCIÓN

La automatización de pruebas es un proceso cada vez más prominente en el mundo del desarrollo de software, ya que garantiza niveles altos de calidad en cada entrega incremental de un producto, a través de procesos de integración continua que contribuyen a la detección temprana de defectos y agilizan el proceso de testing.

Como cliente de Globant, la compañía norteamericana Ticketmaster, que se dedica a la venta y gestión de entradas para eventos de todo tipo, a través de plataformas web y mobile, pretende hacer uso de la automatización de pruebas en diferentes proyectos, para aprovechar sus ventajas en cada fase de testing. Concretamente, el proyecto que se discute en este documento abarca una porción de las aplicaciones móviles para Android y iOS. Estas apps están disponibles en seis regiones diferentes (NA, MX, UK, IE, AU y NZ), que no necesariamente comparten las mismas funcionalidades.

El propósito general consiste en construir un framework de automatización multiplataforma que permita ejecutar pruebas en todas las regiones, tanto para Android como para iOS. A continuación, se discuten los objetivos en detalle, la forma en que se ha venido realizando el trabajo (tomando en cuenta que el proyecto ha estado en desarrollo desde antes del inicio del proceso de práctica), los resultados obtenidos, y las conclusiones derivadas de los seis meses de trabajo.

II. OBJETIVOS

A. Objetivo general

Diseñar un framework multiplataforma para la ejecución de pruebas automatizadas en dispositivos móviles Android y iOS para el SDK mobile de Ticketmaster, con el fin de facilitar y agilizar el proceso de aseguramiento de la calidad a lo largo del ciclo de desarrollo de la aplicación, mediante ejecuciones automáticas para procesos de regresión y smoke testing.

B. Objetivos específicos

1. Analizar casos de prueba existentes, identificar candidatos a automatizar y definir prioridades.
2. Mantener y actualizar el framework, de acuerdo con los cambios en la aplicación.
3. Aumentar la cobertura de automatización para las regiones internacionales, de modo que se acerque a la cobertura de Norteamérica.
4. Crear y mantener el pipeline de integración continua en Gitlab para ejecución de pruebas automatizadas durante el ciclo de desarrollo.
5. Capacitar al equipo de QA manual en conceptos de automatización y estructura del framework, con el fin de integrarlos al proceso de automatización.

III. MARCO TEÓRICO

Como lo menciona Ham Vocke en [1] la importancia de la automatización de pruebas radica en la habilidad de ejecutar tareas repetitivas sin supervisión e identificar posibles problemas en la aplicación de forma temprana, cada vez que hay cambios en el código. Sin embargo, para aprovechar estas ventajas, es indispensable contar con un proceso definido de Integración continua [2], que compile la aplicación y ejecute los tests automáticamente. Gitlab CI es la herramienta usada para este propósito.

Para este proyecto de automatización de UI multiplataforma para aplicaciones móviles, inicialmente se consideraron las herramientas de automatización nativas de cada sistema operativo, XCTest [3] para iOS y Espresso [4] para Android, junto con Appium [5], herramienta de automatización móvil compatible con ambos sistemas operativos.

Al ser herramientas desarrolladas y mantenidas directamente por Apple y Google, tanto XCTest como Espresso gozan de una alta velocidad de ejecución y no sufren problemas de compatibilidad. Sin embargo, requieren estar integrados directamente dentro del proyecto de desarrollo, lo que implica el mantenimiento de dos repositorios diferentes para el único ingeniero de automatización, requiere automatizar cada caso de prueba dos veces (una por proyecto), e impide la reutilización de código.

Por otro lado, Appium suele tener velocidades de ejecución mas bajas en iOS y puede presentar problemas de incompatibilidad cuando se lanzan nuevas versiones de los sistemas operativos, pero es completamente independiente a los proyectos de desarrollo, ya que solo necesita acceso a la aplicación de prueba para ejecutar tests (bien sea el archivo o una aplicación previamente instalada en el dispositivo de pruebas).

Tomando en cuenta que las aplicaciones de prueba en cada sistema operativo son muy similares y los casos de prueba se diseñan con ambas apps en mente, tener un único proyecto de automatización independiente que permita la reutilización de código y la ejecución de tests en

paralelo en ambos sistemas operativos, se constituyó como la opción más viable, y llevo a elegir Appium como el núcleo para la creación del framework.

WebdriverIO [6], un framework enfocado en testing de UI se utiliza como base para el proyecto, gracias a su compatibilidad con Appium y su fácil integración con otras herramientas como generadores de reportes y sistemas de integración continua.

IV. METODOLOGÍA

Tomando en cuenta que el proyecto ha estado en desarrollo desde antes del inicio del proceso de prácticas, se tiene un framework funcional de entrada, y el trabajo a seguir se enfoca en el mejoramiento constante y la consecución de los objetivos planteados.

El proyecto sigue scrum como metodología base, enfocando el trabajo a realizar por releases, en lugar de sprints, debido a la naturaleza del negocio. Los releases inicialmente ocurrían con una periodicidad mensual que, durante el transcurso de los seis meses de trabajo, pasó a ser cada mes y medio. La planeación de las actividades de automatización se lleva a cabo mediante reuniones del equipo de QA tras cada release. El objetivo principal es priorizar los casos de prueba a automatizar, pero también se consideran actividades de mantenimiento y desarrollo de nuevas funcionalidades para el framework. De igual forma, cada mes se llevan a cabo reuniones de socialización de avances de automatización junto con los líderes del equipo, con el fin de presentar el trabajo realizado con respecto a la planeación inicial.

Norteamérica es la región prioritaria y como tal, tiene una mayor cobertura que las demás regiones y se utiliza como base para definir actividades. A futuro, se espera que los casos automatizados para Norteamérica que sean aplicables a nivel internacional, funcionen correctamente en todas las regiones.

Un caso de prueba se considera automatizado una vez que funcione en Android y iOS para cada una de las cinco regiones en las que la aplicación está disponible. Los casos de prueba

automatizados se añaden al set de regresión que se ejecuta antes de cada reléase, a través del pipeline, para cada plataforma y región.

En un principio, se realizaban sesiones prácticas de automatización con los testers manuales cada semana, buscando que fueran capaces de contribuir al proceso de automatización a futuro. Sin embargo, durante los últimos meses de trabajo, se sumaron dos nuevos miembros al equipo de automatización y el entrenamiento de los testers manueles dejó de ser una prioridad.

V. RESULTADOS

Durante el transcurso del proyecto, se han agregado nuevos casos de prueba al framework de acuerdo con las prioridades definidas. También se han actualizado y deprecado casos de prueba previamente automatizados, a raíz de actualizaciones en la aplicación. Las sesiones de planning llevadas a cabo antes de cada reléase, han permitido identificar flujos específicos para los cuales no se tiene una cobertura de casos de prueba adecuada en regiones internacionales (fuera de NA), información que ha sido vital para la estrategia general de calidad del año 2024.

Debido a que le cliente sostiene una postura reservada con respecto a sus métricas y cifras, no ha sido posible presentar datos concretos que soporten el incremento mencionado en la cobertura de los casos de prueba automatizados.

En cuanto a los procesos de CI, se han implementado Trigger Jobs en los pipelines de desarrollo de Backend y iOS, que permiten activar Jobs específicos en el pipeline de automatización cada vez que se genera un MR en el repositorio de desarrollo.

Fig. 1 ilustra el funcionamiento de los Trigger Jobs: Una vez creada la MR en el repositorio de desarrollo, el Trigger Job activa el pipeline de automatización que se encarga de ejecutar smoke tests compuestos por un subconjunto de casos de prueba previamente definidos. El pipeline cuenta con un Job de smoke test por cada región disponible. Al estar asociados a un pipeline externo al de desarrollo, los smoke tests se ejecutan en paralelo con los Jobs del pipeline de desarrollo. Una vez

finalizada la ejecución, el pipeline de automatización comparte los resultados con el pipeline de desarrollo, y el reporte generado es accesible desde cualquiera de los dos pipelines.

La capacitación de los miembros del equipo de QA manual en temas de automatización se concibió inicialmente como un entrenamiento consistente de dos sesiones semanales, donde se abordarían temas generales de la automatización de pruebas, empezando desde los conceptos básicos. Dos ingenieros de calidad formaron parte del entrenamiento, y la metodología consistía en una sesión de teoría y explicación de conceptos y una sesión práctica, donde se esperaba que uno de los testers compartiera su pantalla con el fin de resolver en vivo ejercicios relacionados con el tema abordado durante la sesión teórica. Los temas a abordar incluían: configuración local del proyecto de automatización y sus dependencias, estructura de un caso de prueba automatizado, ejecución de tests en emuladores y simuladores para Android y iOS respectivamente, conceptos básicos de programación orientada a objetos, el patrón de diseño Page Object Model, tipos de localizadores para interactuar con elementos en una aplicación, uso de la herramienta Appium Inspector para identificar localizadores; y posteriormente se esperaba poder abordar temas más avanzados, como: uso de hooks para establecer precondiciones y postcondiciones, configuración de suites o subconjuntos de casos de prueba para flujos específicos, automatización de gestos en el teléfono, ejecuciones paralelas e introducción a la integración continua. Sin embargo, debido a la necesidad del cliente de incrementar la cobertura rápidamente, se sumaron dos nuevos ingenieros de automatización al equipo y las capacitaciones perdieron relevancia paulatinamente, hasta que dejaron de hacerse. Si bien uno de los ingenieros de calidad adquirió el nivel suficiente para automatizar casos de prueba sencillos, sus contribuciones son esporádicas debido a que su enfoque principal sigue siendo el testing manual.

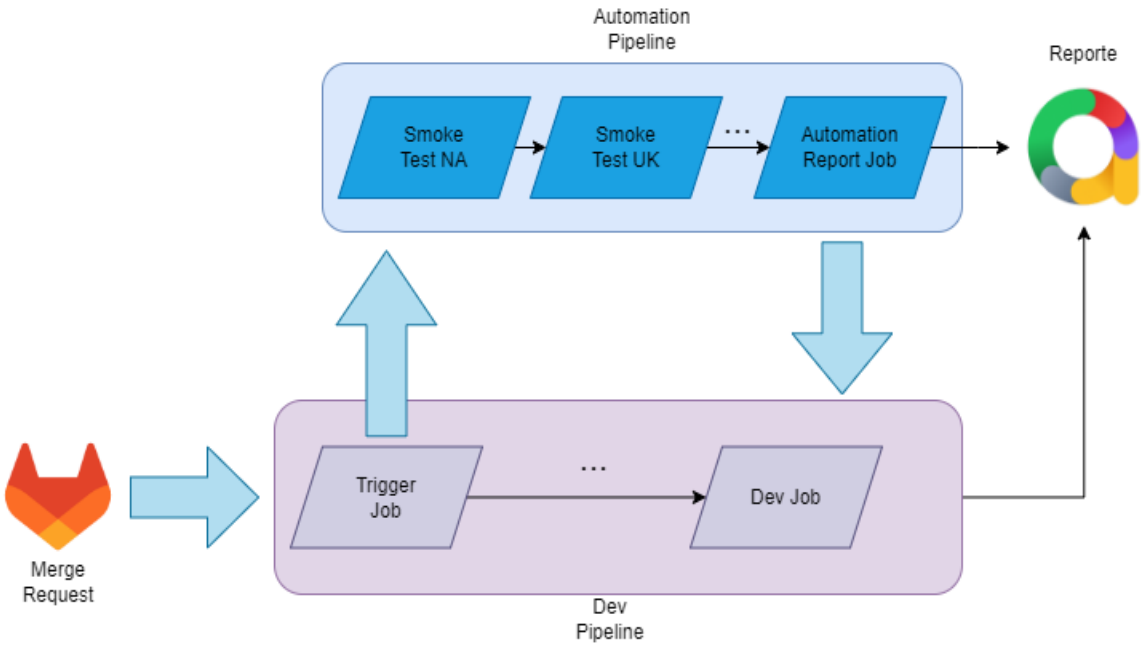

Fig. 1 Diagrama de pipeline de automatización activado desde el pipeline de desarrollo

VI. ANÁLISIS

A. Incremento de la cobertura a corto y largo plazo

El esfuerzo de automatización depende de los casos de prueba existentes como insumo principal para obtener el avance esperado. Debido a la alta prioridad que históricamente ha tenido la región de NA para el cliente, lo esfuerzos en el ámbito de pruebas se han enfocado en esa región, generando un desajuste con respecto a la cantidad de casos de prueba existentes en las demás regiones. Esto implica inviabilidad en el incremento general de la cobertura de pruebas automatizadas, ya que, si bien una gran cantidad de casos presentes en NA puede ser aplicable a otros mercados, resulta imprudente tomar esa suposición como realidad sin antes someterla a las verificaciones propias de los procesos de aseguramiento de la calidad.

Como se ha descrito previamente, el aumento en la cobertura tomando una región como eje (NA) puede ser viable a corto plazo, mas no sostenible a futuro, como consecuencia del alcance a considerar para todas las regiones existentes, así como para otras regiones que puedan sumarse en algún momento.

Gracias a las prioridades ya establecidas entre los casos de prueba existentes, se tiene conocimiento de los flujos clave en la aplicación que podrán usarse como insumo para el diseño de los casos de prueba faltantes, lo que permitirá alcanzar un nivel de paridad entre NA y las demás regiones. Mantener ese nivel de paridad sería un factor determinante para el aumento sostenido de la cobertura de las pruebas automatizadas a largo plazo.

B. El impacto de la integración continua para el proceso de testing

El propósito principal del uso de CI en testing es obtener resultados en las etapas iniciales del ciclo de desarrollo, sin necesidad de que haya intervención humana. Si un cambio reciente en desarrollo se comprueba una vez enviado al repositorio, y desemboca en fallos en casos de prueba que no tenían problemas previamente, será posible evitar que los mismos errores ocurran en etapas cruciales, como el despliegue.

La detección temprana de fallos gracias a la introducción de tests automatizados en el pipeline de iOS ha permitido obtener versiones más estables de la aplicación de para efectos de pruebas mas exhaustivas, como la regresión.

Si bien los mismos beneficios no están siendo aprovechados en Android, el éxito conseguido en iOS justifica la priorización de la implementación de tests automatizados integrados en el pipeline de desarrollo de Android.

C. Viabilidad de capacitación en un entorno que requiere resultados inmediatos

Para perfiles de ingenieros de calidad que se han dedicado exclusivamente al testing manual, entrar en el mundo del testing automatizado implica tener contacto directo con

conceptos y herramientas de desarrollo. Este esfuerzo requiere motivación por parte del individuo, así como tiempo para estudiar, practicar e interiorizar conceptos antes de poder usarlos activamente en un proyecto de automatización.

La expectativa del cliente respecto al plan de capacitación de testers manuales implicaba generar más contribuciones al proyecto de automatización en un corto periodo de tiempo, manteniendo el mismo nivel de desempeño en las tareas de testing manual.

Pese a las recomendaciones y aclaraciones presentadas para el establecimiento del plan de capacitación, la necesidad de obtener resultados a corto plazo sumado a una falta de conocimiento especializado en temas de calidad de software por parte del cliente, impidieron que se tuviera en cuanta las compensaciones necesarias para lograr una capacitación exitosa (el tiempo invertido en entrenamiento implica una disminución en el tiempo invertido en tareas de testing manual).

Futuros esfuerzos similares con el cliente requerirán un mejor planteamiento del plan, describiendo claramente los pros, contras e implicaciones, para asegurar que los tiempos y esfuerzos sean claros desde el inicio.

VII. CONCLUSIONES

- Establecer prioridades claras, fue clave para generar valor al proceso de testing, de acuerdo con las necesidades especificas o funcionalidades clave a tener en cuenta en cada release.
- El análisis periódico de la suite de casos de prueba, en conjunto con los demás miembros del equipo de calidad, permitió identificar tanto los casos de prueba automatizados que requerían actualizaciones, como las falencias en los casos manuales existentes.
- A raíz de la baja cobertura para algunos flujos en los casos de prueba internacionales, no fue posible aumentar la cobertura de la automatización en regiones

internacionales al nivel que se esperaba. Sin embargo, fue posible hacer que los casos previamente automatizados y compatibles a nivel internacional, funcionen en todas las regiones.

- La implementación de los smoke tests automatizados como parte del pipeline de desarrollo ha contribuido a la detección temprana de defectos en iOS, agilizando el proceso de calidad del equipo. La implementación en Android se tiene contemplada para los primeros meses del 2024.
- La capacitación del equipo de testers manuales perdió importancia debido a un cambio de prioridades del cliente, que prefirió sumar nuevos ingenieros de automatización para agilizar el avance.

REFERENCIAS

[1] H. Vocke, «The Practical Test Pyramid,» martinfowler.com, 26 Febrero 2018. [En línea]. Disponible en: https://martinfowler.com/articles/practical-testpyramid.html#TheImportanceOftestAutomation. [Último acceso: 22 enero de 2024].

[2] M. Rehkopf, «¿En qué consiste la integración continua?,» Atlassian, [En línea]. Disponible en: https://www.atlassian.com/es/continuous-delivery/continuous-integration. [Último acceso: 22 enero de 2024].

[3] «XCTest,» Apple Developer Documentation, [En línea]. Disponible en: https://developer.apple.com/documentation/xctest [Ultimo acceso: 24 enero de 2024].

[4] «Espresso,» Android Developers, [En línea]. Disponible en: https://developer.android.com/training/testing/espresso?hl=es-419 [Ultimo acceso: 24 enero de 2024].

[5] «Appium Documentation,» Appium, [En línea]. Disponible en: http://appium.io/docs/en/2.1/. [Último acceso: 22 enero de 2024].

[6] «WebdriverIO · Next-gen browser and mobile automation test framework for Node.js,» WebdriverIO, [En línea]. Disponible en: https://webdriver.io/. [Último acceso: 22 enero de 2024].