CESUR

Tu Centro Oficial de FP

PROYECTO FINAL DE CICLO

TÍTULO

Importancia del testing y quality assurance en el desarrollo móvil

CICLO FORMATIVO

TÉCNICO EN DESARROLLO DE APLICACIONES MULTIPLATAFORMA

José Antonio Alacid Pérez

CURSO 2022-2024

CESUR
Tu Centro Oficial de FP

CESUR
Tu Centro Oficial de FP

INFORME EVALUACIÓN TRABAJOS FIN DE CICLO

ALUMNO: José Antonio Alacid Pérez
TITULO DEL PROYECTO: Importancia del testing y quality assurance en el desarrollo móvil
CENTRO DE PRACTICAS Tiki Taka

VALORACIÓN

Originalidad / Innovación_____

Viabilidad/ Realista_____

Forma/Presentación_____

_

Defensa Oral _____

Observaciones y Comentarios

_____
_____
_____
_____

Murcia, días y de de 2024

Tribunal constituido a de de 2024 para la defensa de los trabajos:

Fdo. _____ Fdo. _____

CESUR
Tu Centro Oficial de FP

CESUR
Tu Centro Oficial de FP

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD PROYECTO FINAL DE CICLO

Yo, **José Antonio Alacid Pérez**, con documento de identificación **48753113J** y estudiante del CICLO DE GRADO SUPERIOR DESARROLLO DE APLICACIONES MULTIPLATAFORMA en CESUR MURCIA, declaro que asumo la autoría y originalidad de la memoria de prácticas entregada.

Murcia, a 23 de mayo 2024

Fdo.: _____

CESUR
Tu Centro Oficial de FP

AGRADECIMIENTOS

Gracias a mi familia por apoyarme cuando decidí dejar la carrera para hacer el grado superior.

Gracias a mis profesores por todo lo que me han enseñado estos dos años.

Gracias a mis compañeros de clase por estos dos años maravillosos en los que hemos pasado muchas horas juntos.

Gracias a los miembros del Club de Magic de la Facultad de Informática de la Universidad de Murcia por estar ahí todas las semanas y aguantarme todos los viernes.

Finalmente quiero agradecer a mi abuelo, que lamentablemente falleció un par de días antes de empezar las prácticas, por ser la mejor persona que he conocido en mi vida y un gran ejemplo a seguir, una persona tan grande que no puede ser descrita con palabras y eso se nota cuando toda persona que te conoció te tiene un aprecio increíble. Y aunque no puedas estar hoy aquí conmigo espero que ahí donde estés te sientas orgulloso de tu nieto. Te quiero mucho abuelo, D.E.P

CESUR
Tu Centro Oficial de FP

RESUMEN

En este proyecto se va a hablar sobre el testing y quality assurance orientado al mundo del desarrollo móvil y su importancia, se tratarán temas en profundidad como el contexto de la aplicación móvil y en que dista del contexto de una aplicación independiente en un sobremesa, portátil o servidor, los beneficios tanto para el usuario final como para los clientes de realizar buenas pruebas y tener un quality assurance correcto tanto en el ámbito económico como en el social e incluso el técnico, las mejores prácticas a la hora de realizar las pruebas para mejorar la eficiencia y efectividad de estas, que es la integración continua y entrega continua, el desarrollo guiado por pruebas, el desarrollo guiado por comportamiento, algunas de las herramientas más utilizadas en estos procesos como funcionan y porqué se utilizan, los diferentes tipos de pruebas y porque se realizan, algunas de las dificultades a la hora de realizar las pruebas en dispositivos móviles y las modas actuales y cómo van a influir en el futuro de este campo.

ABSTRACT

This project is about testing and quality assurance oriented to mobile development and its importance, topics covered include in depth topics such as the context of the mobile application and how it differs from the context of a standalone application on a desktop, laptop or server, the benefits for both the end user and for customers of performing good tests and having a good quality assurance process both economically, socially and even on a technical level, best practices in testing to improve the efficiency and effectiveness of tests, what is continuous integration and continuous deployment, test-driven development, behavior-driven development, some of the most commonly used tools in testing and quality assurance and how do they work, the different types of tests and why they are performed, some of the difficulties of testing on mobile devices and current trends and how they will influence the future of this field.

CESUR
Tu Centro Oficial de FP

CESUR
Tu Centro Oficial de FP

ÍNDICE

Introducción

Durante estos dos años que he estado estudiando el ciclo he visto y trabajado con distintas tecnologías, algunas que ya conocía y otras que como mucho me sonaban. Pero de entre todo lo que hemos visto hubo algo que me llamó la atención y fue lo relacionado con las pruebas unitarias en Java con JUnit, algo que no había visto ni conocía, y como lo vimos muy por encima al final del primer año y no lo hemos vuelto a ver o aprendido más cosas sobre este campo durante este último año me he estado informando y formando por mi propia cuenta y cómo estudiante del ciclo de desarrollo de aplicaciones multiplataforma me ha sorprendido que dentro de este mundo del testing y el quality assurance no he visto mucho contenido relacionado con el desarrollo móvil así que he decido investigar su importancia a la hora de desarrollar una aplicación móvil y porque incluso para algunos profesionales del sector este puede ser un tema complicado.

En este proyecto se va a hablar sobre el testing y quality assurance orientado al mundo del desarrollo móvil y su importancia, se tratarán temas en profundidad como el contexto de la aplicación móvil y en que dista del contexto de una aplicación independiente en un sobremesa, portátil o servidor, los beneficios tanto para el usuario final como para los clientes de realizar buenas pruebas y tener un quality assurance correcto tanto en el ámbito económico como en el social e incluso el técnico, las mejores prácticas a la hora de realizar las pruebas para mejorar la eficiencia y efectividad de estas, que es la integración continua y entrega continua, el desarrollo guiado por pruebas, el desarrollo guiado por comportamiento, algunas de las herramientas más utilizadas en estos procesos como funcionan y porqué se utilizan, los diferentes tipos de pruebas y porque se realizan, algunas de las dificultades a la hora de realizar las pruebas en dispositivos móviles y las modas actuales y cómo van a influir en el futuro de este campo.

El fin del proyecto es informar y dar a conocer el mundo del testing y el quality assurance en el desarrollo móvil, las dificultades y beneficios de realizar estas técnicas durante el desarrollo de una aplicación y su importancia en este proceso.

Marco teórico

Para hablar del testing hay que hablar de la historia del desarrollo de software desde los inicios de este campo los profesionales del sector querían que sus programas funcionaran correctamente en cualquier situación, por lo que probaban sus programas antes de sacarlos al público, cuando el desarrollo de software y la tecnología avanzaba también lo hacía el testing.

Durante los primeros años del software alrededor de los años cincuenta no existía una distinción clara entre el testing y el debugging, tampoco existía el concepto de tester, cuando un desarrollador encontraba un error en su código simplemente lo depuraba, por lo que se conoce a esta época como la orientada a la depuración.

En el año 1957 los desarrolladores empezaron a probar sus programas en situaciones reales por lo que ya se creó una distinción clara entre las pruebas y la depuración de código, a partir de esto empiezan a surgir los primeros equipos dedicados exclusivamente al testing, el objetivo principal de estos era asegurarse de que se cumplían los objetivos del software y de que se satisfacía las expectativas de los clientes, por eso esta época es conocida como la época orientada a la demostración, durante estos años surgieron múltiples ideas que a día de hoy se siguen utilizando en el desarrollo de software, como el desarrollo en cascada o las pruebas funcionales.

A finales de 1978 y principio de 1979 surge una nueva metodología a la hora de probar el código esta se centraba en romper el código y descubrir errores que podían haber pasado desapercibidos, esta época se conoce como la orientada a la destrucción pero no fue muy duradera porque los desarrolladores se dieron cuenta de que el software nunca se publicaría de esta forma ya que encontrar un error y arreglarlo podía provocar que surgieran nuevos errores.

En el año 1979 sale The Art of Software Testing de Glenford J. Myers, un libro que rápidamente se volvió un referente en el mundo del testing e introdujo muchos conceptos que a día de hoy siguen siendo relevantes, por lo que es considerado una guía fundamental del testing y su contenido todavía es útil incluso con todos los cambios que ha sufrido la industria tanto de hardware como de software. En el libro Myers comparte sus conocimientos basados en su experiencia trabajando en la empresa norteamericana IBM donde trabajó en proyectos de desarrollo relacionados con el

IBM System/360 y liderando el equipo de desarrollo del sistema denominado “SWARD” (Software Oriented Architecture, arquitectura orientada al software en inglés).

Con el problema planteado por la anterior época en 1983 con la salida del estándar IEEE 829 los desarrolladores se centraron en medir y evaluar la calidad de su código, los testers probaban el software hasta un punto aceptable donde el número de errores detectados era reducido por eso esta época se conoce como la orientada a la evaluación.

Para 1988 el testing ya había llegado a un buen nivel, lo que llevó al desarrollo de nuevas metodologías de trabajo como el desarrollo guiado por pruebas o el desarrollo guiado por comportamiento y la creación de herramientas de testing poderosas como Selenium, esta época es conocida como la orientada a la prevención puesto que el código se separa en dos tipos distintos, el código que se puede probar y el que no se puede donde el que se puede probar termina teniendo menos errores que el que no se puede probar.

En la actualidad existen distintos tipos de pruebas que comprueban diferentes aspectos del software en escenarios distintos, como las pruebas unitarias, las de integración, las de aceptación o las de rendimiento. Además se están utilizando nuevos métodos de trabajo como el uso de inteligencia artificial basada en el comportamiento de los usuarios finales.

El entorno del desarrollo móvil también ha evolucionado mucho en los últimos años con la implementación de nuevas tecnologías como el 5G o distintos tipos de pantallas, factores que se tienen que tener en cuenta a la hora de crear una aplicación móvil para el correcto funcionamiento de esta.

Objetivos

La meta del proyecto es informar sobre la importancia del testing y el quality assurance en el ámbito del desarrollo móvil, para cumplir esta meta se cumplen los siguientes objetivos:

- Explicar el contexto de la aplicación móvil y sus diferencias con la aplicación de sobremesa.

- Describir los beneficios del testing en el desarrollo móvil, tanto económicos como sociales y técnicos.

- Manifestar las mejores prácticas a la hora de realizar el proceso de testing en una aplicación móvil.

- Explicar los diferentes tipos de pruebas que existen y su importancia en los procesos de testing y quality assurance.

- Enseñar algunos de los programas más utilizados en el testing de aplicaciones móviles, como funcionan y para que se utilizan.

- Esclarecer las dificultades que puede tener el testing en el desarrollo de aplicaciones móviles.

- Comentar las modas actuales del testing para aplicaciones móviles y cómo van a influir a futuro.

- Explicar los conceptos de integración continua y entrega continua y porque se utilizan en el desarrollo de aplicaciones móviles.

- Mostrar que es el desarrollo guiado por pruebas y el desarrollo guiado por comportamiento.

Metodología

Participantes

José Antonio Alacid Pérez, funciones:

1. Búsqueda de información
2. Redacción del proyecto

Descripción del proceso

Para cumplir los objetivos expuestos en el punto anterior se van a realizar los siguientes procesos:

Búsqueda de información y opiniones de otros desarrolladores sobre el testing en el desarrollo móvil en libros, artículos, páginas web.

Aprender a utilizar algunos de los programas más utilizados en el testing de aplicaciones móviles para poder explicar cómo funciona el programa y para qué se utiliza.

Cronología

26 de Marzo a 7 de Abril – Búsqueda de información

8 de Abril a 20 de Abril – Introducción y marco teórico

22 de Abril a 4 de Mayo – Objetivos y metodología

6 de Mayo a 20 de Mayo – Desarrollo, conclusiones y bibliografía

Desarrollo

El testing y el quality assurance de las aplicaciones móviles están diseñados para garantizar que una aplicación cumpla sus especificaciones, funcione como se espera y ofrezca una experiencia de usuario satisfactoria en una amplia gama de dispositivos, sistemas operativos y condiciones de red.

El desarrollo móvil es una herramienta muy poderosa para que las empresas y organizaciones capten clientes, agilicen sus operaciones y desbloqueen nuevas fuentes de ingresos.

Por eso, es fundamental garantizar la calidad y rendimiento de las aplicaciones móviles que se desarrollan. Las aplicaciones de mala calidad tienen mucha más probabilidad de ser rechazadas por los usuarios y pueden dañar la reputación e ingresos de una empresa.

En este proyecto se explora la importancia de las pruebas y el control de calidad en el ciclo de vida del desarrollo de aplicaciones móviles.

El entorno de desarrollo móvil

Los dispositivos móviles como teléfonos o PDAs están diseñados para usar tecnologías de transmisión de datos inalámbricas como Bluetooth, 3G, 4G, 5G, etc. Por eso los desarrolladores tienen que tener en cuenta que estas tecnologías a veces pueden ser poco fiables, puede que no tengamos señal o la conexión no sea muy buena. Por lo que a la hora de desarrollar sus aplicaciones deben decidir si su aplicación está pensada para utilizar estas tecnologías y si lo están, desarrollar pruebas que tengan en cuenta estas situaciones en las que la conexión no es buena para comprobar el funcionamiento de la aplicación y poder solucionar los posibles problemas que esto puede acarrear.

La conectividad de estos dispositivos no es el único aspecto a tener en cuenta a la hora de trabajar con estos, otros factores que influyen en el desarrollo de una aplicación móvil son la diversidad de dispositivos y sus limitaciones, estas limitaciones induyen distintos sistemas operativos, no es lo mismo desarrollar una aplicación para Android que para iOS, o los distintos tamaños de pantalla

una aplicación puede estar pensada para un tamaño de pantalla determinado y en dispositivos con tamaños distintos puede no mostrarse correctamente, entre otros muchos aspectos.

Por último, otro factor a tener en cuenta es como instalar y mantener la aplicación. Algunos proveedores como Apple o Google, tienen sus propias plataformas para la instalación o compra de aplicaciones, pero solo después de que estos proveedores certifiquen tu aplicación para su plataforma. Este sistema de distribución simplifica en gran medida la instalación y mantenimiento de la aplicación al usuario final además de mostrar confianza ya que uno de estos proveedores ha certificado la calidad de tu aplicación. Por lo que es extremadamente importante realizar pruebas y controles de calidad adecuados para cumplir las exigencias de los proveedores.

Beneficios del testing y el quality assurance

Garantizar la satisfacción del usuario

En un mercado tan competitivo como el de las aplicaciones móviles, la satisfacción del usuario es primordial. Los usuarios esperan que las aplicaciones sean intuitivas y rápidas.

Incluso problemas menores, como una interfaz de usuario mala o un tiempo de respuesta lento, pueden provocar frustración y, en última instancia, abandono. Problemas más graves, como fallos o pérdida de datos, pueden dar lugar a críticas negativas y dañar irreparablemente la reputación de la aplicación.

Un buen proceso de testing y control de calidad ayuda a garantizar una experiencia de usuario de calidad al identificar y resolver los problemas antes de que lleguen al usuario final. Al probar a fondo una aplicación, los desarrolladores pueden asegurarse de que funciona correctamente y ofrece una experiencia de usuario fluida y agradable.

Mejorar el rendimiento de la aplicación

Los usuarios esperan que las aplicaciones móviles sean rápidas y fiables. Los problemas relacionados con el rendimiento, como tiempos de carga lentos o lentitud de respuesta, pueden provocar rápidamente una pérdida de usuarios.

Las pruebas de rendimiento son cruciales para garantizar que una aplicación pueda soportar picos de carga, responder rápidamente a las interacciones de los usuarios y funcionar sin problemas en distintas condiciones de red.

Las pruebas de rendimiento pueden ayudar a identificar cuellos de botella, optimizar el uso de recursos y mejorar el rendimiento general de la aplicación.
También puede simular diferentes condiciones y cargas de red para garantizar que la aplicación funcione satisfactoriamente incluso en condiciones extremas.

Garantizar la seguridad

A medida que aumentan las filtraciones de datos y los ciberataques, garantizar la seguridad de las aplicaciones móviles es más importante que nunca.

Las aplicaciones móviles suelen manejar datos sensibles, como información personal, datos de pago y datos críticos de la empresa. Los cibercriminales pueden aprovecharse de cualquier vulnerabilidad y provocar filtraciones de datos, robos de identidad y otros problemas graves.

Las pruebas de seguridad son una parte esencial del testing y el control de calidad de las aplicaciones móviles. Consiste en identificar posibles vulnerabilidades, comprobar la eficacia de las medidas de seguridad y garantizar que la aplicación cumple las normas y reglamentos de seguridad pertinentes.

También ayuda a generar confianza en el usuario, proteger datos confidenciales y evitar problemas legales, multas y daños a la reputación de la aplicación asociados a fallos de seguridad.

Preparar la aplicación para el mercado móvil

El mercado de los dispositivos móviles es muy diverso, con muchos dispositivos, sistemas operativos, tamaños de pantalla y condiciones de red. Una aplicación que funciona perfectamente en un dispositivo puede fallar en otro debido a diferencias de hardware, software o conectividad de red.

Las pruebas de compatibilidad son cruciales para garantizar que una aplicación funciona correctamente en un entorno tan diverso. Consiste en probar la aplicación en varios dispositivos, sistemas operativos y condiciones de red para garantizar una buena experiencia de usuario.

Las pruebas de compatibilidad ayudan a garantizar que la aplicación está lista para el mercado y puede llegar al público más amplio posible.

Rentabilidad

Realizar pruebas en las primeras fases del ciclo de desarrollo puede ahorrar mucho tiempo, esfuerzo y dinero.

Identificar y solucionar problemas después de lanzar la aplicación puede ser costoso y consume mucho tiempo. También puede dañar la reputación de la aplicación y la satisfacción de los usuarios.

Las pruebas tempranas permiten detectar y resolver los problemas cuando son relativamente fáciles y baratos de solucionar. También ayuda a evitar modificaciones, retrasos en la entrega y posibles pérdidas de ingresos.

Buenas prácticas de testing y quality assurance en el ámbito del desarrollo móvil

La aplicación de buenas prácticas en las pruebas y en el control de calidad de las aplicaciones móviles puede mejorar significativamente la eficiencia y la eficacia de estas. Las siguientes prácticas son algunas de las recomendadas:

Empezar a probar pronto y a menudo

Cuanto antes se incorporen las pruebas al proceso de desarrollo, mejor. Empezar a probar en las fases iniciales del proceso de desarrollo puede ayudar a identificar y abordar los problemas antes de que se arraiguen profundamente en el código, lo que facilita y abarata su solución.

Además, las pruebas no deben ser una actividad puntual, sino un proceso continuo. Las pruebas periódicas a lo largo del ciclo de desarrollo pueden ayudar a detectar nuevos problemas a medida que surgen y garantizar que las correcciones no han introducido otros nuevos.

Pruebas en dispositivos reales

Aunque los simuladores y emuladores pueden ser útiles en el proceso de testing, no pueden reproducir completamente el rendimiento y comportamiento de los dispositivos reales. Los dispositivos reales tienen características específicas de hardware, software y red que pueden afectar significativamente al rendimiento de una aplicación y a la experiencia del usuario.

Por lo tanto, es esencial probar la aplicación en una muestra representativa de los dispositivos que utiliza el público objetivo. Esto proporcionará una evaluación más precisa del rendimiento, la funcionalidad y la experiencia de usuario de la aplicación.

Automatizar cuando sea posible

Las pruebas manuales pueden llevar mucho tiempo y ser propensas a errores, especialmente en el caso de aplicaciones grandes y complejas. La automatización puede acelerar considerablemente el proceso de pruebas, aumentar la cobertura y reducir el riesgo de error humano. Las pruebas automatizadas pueden ejecutarse repetida y sistemáticamente, lo que las hace especialmente útiles para las pruebas de regresión.

Sin embargo, la automatización no lo es todo. Algunos aspectos, como las pruebas de usabilidad, requieren el juicio humano y son difíciles de automatizar con eficacia. Por eso, un enfoque equilibrado que combine pruebas automatizadas y manuales suele ser la mejor estrategia.

Priorizar tests en función de los escenarios de usuario

Dada la variedad de dispositivos, sistemas operativos y escenarios de usuario, es imposible probar todas las combinaciones posibles. Por lo tanto, es esencial priorizar las pruebas en función de los escenarios de usuario.

Las funcionalidades más críticas y las más utilizadas por el público objetivo deben probarse primero y con mayor detenimiento. Del mismo modo, los dispositivos y sistemas operativos más utilizados por el público objetivo deben recibir la máxima prioridad en las pruebas.

Tipos de pruebas

El proceso de testing de las aplicaciones móviles no se limita a comprobar si la aplicación funciona como se espera. Se realizan distintos tipos de pruebas, cada una centrada en un aspecto diferente de la aplicación.

Pruebas funcionales

Las pruebas funcionales son un tipo de pruebas de software que evalúan la funcionalidad de una aplicación comparándola con los requisitos especificados. El objetivo de las pruebas funcionales es

garantizar que el software funciona según las especificaciones deseadas y realiza correctamente las funciones previstas.

Las pruebas funcionales son importantes porque ayudan a verificar si la aplicación funciona según lo previsto o no. Aunque una aplicación supere las pruebas no funcionales y funcione bien, tiene que ofrecer los resultados esperados para ser considerada funcional por los usuarios finales.

En el caso de estos tests es recomendable automatizar el proceso. Identificar todos los fallos de manera manual es caro, consume mucho tiempo de desarrollo y es casi imposible. Esta automatización se suele conseguir a través de herramientas de procesamiento de lenguaje natural.

Tipos de pruebas funcionales:

Pruebas unitarias. Las pruebas unitarias son un tipo vital de pruebas funcionales que implican probar la unidad de código funcional y comprobable más pequeña. Para garantizar la máxima calidad del software, las pruebas unitarias son una parte esencial del proceso de desarrollo de software. Gracias a las pruebas unitarias, los desarrolladores pueden identificar y corregir los defectos en una fase temprana del proceso de desarrollo, lo que en última instancia puede ayudar a ahorrar tiempo y dinero, garantizando al mismo tiempo que el software sea de la máxima calidad.

Pruebas de integración. Durante las pruebas de integración, el objetivo es validar la interacción entre dos o más componentes a los que se les han realizado una prueba unitaria. Existen dos tipos de pruebas de integración: Escalonadas y Big-Bang. Las pruebas de integración escalonadas consisten en combinar uno o varios componentes por etapas y probarlos hasta que todo el sistema se haya probado con éxito. Por otro lado, el enfoque Big-Bang consiste en integrar y probar todos los componentes como un sistema completo.

Pruebas de interfaz. Las pruebas de interfaz y las pruebas de integración son dos pruebas de software que se realizan para garantizar el perfecto funcionamiento de las aplicaciones informáticas. Las pruebas de interfaz se centran principalmente en comprobar la comunicación

entre distintas interfaces, mientras que las pruebas de integración se centran en comprobar el grupo integrado de módulos como una sola unidad. Durante las pruebas de interfaz, se comprueba la corrección del intercambio de datos, la transferencia de datos, los mensajes, las llamadas y los comandos entre dos componentes integrados. La comunicación entre bases de datos, servicios web, API o cualquier componente externo y la aplicación también se comprueba durante las pruebas de interfaz. Durante esta comunicación de datos o comandos no debe producirse ningún error ni desajuste de formato. Si se encuentra algún problema de este tipo, hay que corregirlo. En resumen, tanto las pruebas de interfaz como las de integración son esenciales para garantizar que las aplicaciones de software funcionen a la perfección y sin problemas.

Pruebas de sistema. Las pruebas de sistema son una fase crucial del ciclo de vida de las pruebas de software que garantiza que todos los componentes del sistema están perfectamente integrados y funcionan juntos según las especificaciones de los requisitos. Esta técnica de testing es un método de prueba Black-Box (caja negra) que valida el sistema integrado frente a los requisitos predefinidos. Las pruebas de sistema se realizan en un entorno casi real y de acuerdo con el uso en la vida real. Es esencial realizar las pruebas de sistema antes de las pruebas de aceptación para garantizar que el sistema es totalmente funcional y cumple los requisitos de la empresa.

Pruebas de regresión. Después de que los desarrolladores introduzcan mejoras o corrijan el código, es importante ejecutar un conjunto de pruebas de regresión para garantizar que los cambios no han afectado a las funcionalidades existentes y no han introducido nuevos defectos.

Pruebas de humo. Después del desarrollo de una aplicación, es esencial asegurarse de que todas las funcionalidades principales funcionan sin problemas. Las pruebas de humo se realizan en la nueva versión para garantizar que la aplicación está lista para el siguiente nivel de pruebas. Las pruebas de humo se realizan normalmente para las builds creadas durante la fase inicial de desarrollo, que todavía no son estables. El objetivo de las pruebas de humo es verificar que las funcionalidades críticas de la aplicación funcionan correctamente, y que la compilación es lo suficientemente estable como para pasar al siguiente nivel de pruebas. Si alguna funcionalidad importante no funciona como se esperaba durante las pruebas, se rechaza esa compilación concreta y los desarrolladores deben corregir los errores y crear una nueva compilación para

seguir probando. Una vez que las pruebas de es exitoso, la aplicación está lista para el siguiente nivel de pruebas. En resumen, las pruebas de humo son una prueba esencial que ayuda a garantizar que la nueva versión cumple con los estándares de calidad requeridos.

Pruebas de cordura. Las pruebas de cordura son un tipo de prueba esencial que cubre las principales funcionalidades de una aplicación. Normalmente se realiza en una nueva compilación creada por los desarrolladores para una aplicación relativamente estable. El objetivo principal de estas pruebas es verificar si la aplicación funciona como se espera o no. Una vez que la aplicación supera las pruebas de sanidad, se considera que está lista para el siguiente nivel de pruebas.

Pruebas de aceptación. Las pruebas de aceptación son un paso crucial en el ciclo de vida del desarrollo de software, cuyo objetivo es garantizar que el sistema desarrollado cumple todos los requisitos acordados durante la creación de los requisitos empresariales. Este tipo de pruebas se realizan justo después de las pruebas del sistema y antes del lanzamiento final de la aplicación. Las pruebas de aceptación se convierten en un criterio para que el usuario acepte o rechace el sistema, lo que las convierte en un paso crucial en el proceso de desarrollo de software.

Los distintos tipos de pruebas de aceptación son:

Pruebas de aceptación de usuario: Este tipo de pruebas las realizan usuarios reales para asegurarse de que la aplicación satisface sus necesidades y expectativas. También se conoce como prueba beta.

Pruebas de aceptación de negocio: Este tipo de prueba se realiza para garantizar que la aplicación cumple los requisitos y objetivos de la empresa. Se realiza para garantizar que la aplicación está lista para su despliegue y ayudará a la empresa a alcanzar sus objetivos.

Se realiza para garantizar que la aplicación está lista para su despliegue y ayudará a la empresa a alcanzar sus objetivos.

Pruebas de aceptación de la normativa: Este tipo de prueba se realiza para garantizar que la aplicación desarrollada no infringe ninguna normativa legal establecida por los gobiernos. Se

realiza para garantizar que la aplicación cumple los requisitos legales y puede implantarse sin problemas.

Pruebas de usabilidad

Las pruebas de usabilidad son un tipo de investigación de usuarios que evalúa su experiencia al interactuar con una aplicación. Ayuda a los desarrolladores y equipos de diseño a evaluar lo intuitivos y fáciles de usar que son las aplicaciones.

El proceso de pruebas de usabilidad identifica problemas de la aplicación que de otro modo podría haber pasado por alto, pidiendo a usuarios reales (en lugar de desarrolladores o diseñadores) que completen una serie de tareas de usabilidad en el producto. A continuación, se analizan los resultados, el índice de éxito y el camino seguido para completar las tareas con el fin de identificar posibles problemas y áreas de mejora.

El objetivo final de las pruebas de usabilidad es crear una aplicación que resuelva los problemas del usuario y le ayude a alcanzar sus objetivos con una experiencia positiva.

Pruebas de compatibilidad

Las pruebas de compatibilidad comprueban si el software puede ejecutarse en hardware, sistemas operativos, aplicaciones, entornos de red o dispositivos móviles diferentes.

Es una parte vital de una estrategia de control de calidad, y es una parte crucial en el desarrollo de cualquier aplicación.

Dado que hay muchos dispositivos/navegadores, es recomendable tener un plan estratégico para asegurase de que las pruebas de compatibilidad se tienen en cuenta en la estrategia de pruebas. Las pruebas de compatibilidad son una forma de no discriminar los dispositivos/navegadores que puede utilizar un usuario concreto.

Pruebas de rendimiento

Las pruebas de rendimiento consisten en evaluar el rendimiento de un sistema en términos de capacidad de respuesta y estabilidad bajo una carga de trabajo determinada. Las pruebas de rendimiento suelen realizarse para examinar la velocidad, la robustez, la fiabilidad y el tamaño de la aplicación.

Las pruebas de rendimiento ayudan a garantizar que el software cumple los niveles de servicio esperados y proporciona una buena experiencia de usuario. Las aplicaciones lanzadas al público sin haber sido sometidas a pruebas de rendimiento podrían ver dañada su reputación de marca, en algunos casos de forma irrevocable. Los resultados de las pruebas indican las mejoras que se deben introducir en relación con la velocidad, la estabilidad y la escalabilidad antes de que la aplicación entre en producción.

La adopción, el éxito y la productividad de las aplicaciones dependen directamente de la correcta ejecución de las pruebas de rendimiento. Aunque resolver los problemas de rendimiento de producción puede resultar extremadamente caro, el uso de pruebas de rendimiento es clave para el éxito de una aplicación.

Pruebas de seguridad

Las pruebas de seguridad son un tipo de pruebas de software que identifican posibles riesgos y vulnerabilidades de seguridad en aplicaciones, sistemas y redes.

Las pruebas de seguridad son una parte esencial del ciclo de vida del desarrollo de software. El objetivo de las pruebas de seguridad es descubrir cualquier punto débil que pueda ser aprovechado por los atacantes para acceder a datos confidenciales o interrumpir el funcionamiento del sistema.

Si bien las pruebas de seguridad de las aplicaciones son uno de los primeros pasos hacia el objetivo de mejorar la seguridad, estos tests también aportan otras ventajas.

Las pruebas de seguridad pueden aumentar el tiempo de actividad y la productividad neta. Mejorar la seguridad después de un fallo de seguridad es siempre más laborioso que prevenirlo en primer lugar. Cuántas empresas de todo el mundo se han enfrentado a demandas colectivas por violaciones de datos. Los altos costes de esas demandas se podrían haber prevenido con el coste de incorporar un hacker ético a su equipo.

Integrar una cultura de evaluación de riesgos de seguridad y dedicar tiempo a pensar como un ciberdelincuente no sólo mejora la seguridad del software, sino también la calidad del código. Dedicar un tiempo adicional a revisar el código en busca de vulnerabilidades ofrece la oportunidad de detectar también otros errores.

Integración continua y entrega continua

La integración continua y entrega continua es un enfoque de desarrollo de software cuyo objetivo es mejorar la velocidad, la eficiencia y la fiabilidad de la entrega de software. Este enfoque implica la integración frecuente del código, pruebas automatizadas y el despliegue continuo de los cambios de software en la producción.

Antes de la adopción de la integración continua y entrega continua en la industria del desarrollo de software, el enfoque común era un modelo tradicional de desarrollo de software en cascada.
En este enfoque, los desarrolladores trabajaban con sistemas complejos aislados, con cada etapa del ciclo de vida de desarrollo de software completada en secuencia. El proceso solía incluir la recopilación de requisitos, el diseño del software, la codificación, las pruebas y el despliegue.

Este enfoque tiene múltiples desventajas, como costes elevados asociados a los tests, la depuración y corrección de errores que se realizaba de forma manual, agilidad limitada debido a que este enfoque es lineal y no es posible realizar cambios con rapidez ni responder a las necesidades del cliente en tiempo real, o la limitada colaboración entre los testers y los desarrolladores lo que dificultaba la corrección de errores.

La integración continua y la entrega continua surgieron como solución a estos inconvenientes, al introducir un enfoque más ágil y colaborativo del desarrollo de software. La integración continua

y la entrega continua permite a los equipos trabajar juntos, integrando sus cambios de código con frecuencia y automatizando el proceso de pruebas y despliegue.

La integración continua y entrega continua es un proceso automatizado que implica la integración frecuente del código, las pruebas automatizadas y la entrega continua de los cambios de software en producción.

Integración del código

El primer paso en el proceso de integración continua y entrega continua es la integración del código. En este paso, los desarrolladores envían sus cambios de código a un repositorio remoto (como GitHub o GitLab), donde el código se integra con el código base principal.

El objetivo de este paso es garantizar que los cambios en el código sean compatibles con el resto de la base de código y no interrumpan la compilación.

Pruebas automatizadas

Una vez integrado el código, el siguiente paso son las pruebas automatizadas. Las pruebas automatizadas consisten en ejecutar un conjunto de pruebas para garantizar que los cambios en el código son funcionales, cumplen las normas de calidad previstas y no presentan defectos.

Este paso ayuda a detectar problemas en una fase temprana del proceso de desarrollo, lo que permite a los desarrolladores solucionarlos con rapidez y eficacia.

Entrega continua

Una vez que los cambios de código superan la fase de pruebas automatizadas, el siguiente paso es la entrega continua. En este paso, los cambios de código se despliegan automáticamente en un entorno de ensayo para su posterior comprobación.

Este paso tiene como objetivo garantizar que el software se actualiza continuamente con los últimos cambios de código, ofreciendo nuevas características y funcionalidades a los usuarios de forma rápida y eficiente.

Despliegue en producción

El último paso del proceso de integración continua y entrega continua es el despliegue en producción. En este paso, los cambios en el software se ponen a disposición de los usuarios finales. Este paso implica supervisar el entorno de producción, asegurarse de que el software funciona correctamente e identificar y solucionar cualquier problema que surja.

Los cuatro pasos del enfoque de integración continua y entrega continua trabajan juntos para garantizar que los cambios en el software se prueben, integren y desplieguen en producción automáticamente. Esta automatización ayuda a reducir errores, aumentar la eficacia y mejorar la calidad general del software.

Al adoptar un enfoque de integración continua y entrega continua, los equipos de desarrollo pueden conseguir ciclos de lanzamiento más rápidos, reducir el riesgo de defectos en el software y mejorar la experiencia del usuario.

Hay que tener en cuenta que el proceso puede variar dependiendo del proyecto o la empresa. Es decir, algunos equipos pueden o no utilizar pruebas automatizadas, algunos equipos pueden o no tener un entorno dividido en fases, etc.

Las partes clave que componen la integración continua y entrega continua son la integración y el despliegue. Esto significa que el código debe integrarse continuamente en un repositorio remoto, y que este código debe desplegarse continuamente en un entorno determinado después de cada integración.

Beneficios clave

Las principales ventajas de la integración continua y entrega continua son:

Ciclos de lanzamiento más rápidos: Al automatizar el proceso de pruebas y despliegue, la integración continua y entrega continua permite a los equipos lanzar software con mayor frecuencia, respondiendo rápidamente a las necesidades de los clientes.

Mejora de la calidad: Las pruebas automatizadas garantizan que los cambios en el software no introduzcan nuevos errores o problemas, mejorando la calidad general del software.

Mayor colaboración: La integración frecuente del código y las pruebas requieren que los desarrolladores trabajen en estrecha colaboración, lo que conduce a una mejor colaboración y comunicación.

Reducción del riesgo: La implantación continua permite a los desarrolladores identificar y solucionar problemas rápidamente, reduciendo el riesgo de fallos importantes y tiempos de inactividad.

Rentabilidad: La integración continua y entrega continua reduce la cantidad de trabajo manual necesario para desplegar cambios de software, ahorrando tiempo y reduciendo costes.

En resumen, la integración continua y entrega continua surgió como solución a las limitaciones del enfoque tradicional y lineal del desarrollo de software. Al introducir un enfoque más ágil y colaborativo en el desarrollo de software, la integración continua y entrega continua permite a los equipos trabajar juntos, publicar software con más frecuencia y responder rápidamente a las necesidades de los clientes.

Desarrollo guiado por pruebas

El desarrollo y las pruebas van de la mano: uno no puede estar completo sin el otro. Tradicionalmente, siempre se ha hecho hincapié en el desarrollo. Los desarrolladores primero programan, luego prueban, corrigen los errores que encuentran, envían el código y el ciclo se repite. El desarrollo basado en pruebas adopta un enfoque ligeramente opuesto: probar primero y escribir el código después. Los resultados de las pruebas guían las actividades de desarrollo.

El desarrollo basado en pruebas consiste en escribir y ejecutar pruebas automatizadas antes de escribir el código. Los resultados de las pruebas ayudan al desarrollador a mejorar el código.

Ciclo del desarrollo guiado por pruebas

Escribir un test específico. Este paso empieza escribiendo un caso de prueba para una característica específica. No hay absolutamente ningún código escrito para esta característica todavía. Esencialmente, estamos escribiendo la prueba para una característica que ni siquiera se ha creado.

Ejecutar la prueba. Debería ser bastante fácil ver que esta prueba fallará definitivamente. Por contraintuitivo que parezca, aquí es donde reside el ingenio del desarrollo guiado por pruebas. Hay cuatro razones principales por las que la prueba tiene que fallar:

Ejecutar las pruebas en las primeras etapas del desarrollo asegura la correcta configuración y funcionamiento del entorno de pruebas. Cualquier problema con el entorno de pruebas puede ser abordado desde el principio.

Una prueba fallida no es intrínsecamente algo malo. El código no está escrito todavía, y la prueba falla, esto es una buena señal de que la prueba es fiable.

Esta primera prueba, junto con las futuras, es una guía de tu desarrollo. Se convierten en objetivos a seguir. Desglosa el proyecto, aparentemente abrumador, en trozos más pequeños que pueden abordarse uno a uno. Al final del día, el desarrollo guiado por pruebas da a los desarrolladores metas y al project manager una visión más clara de las características que se están desarrollando.

El desarrollo guiado por pruebas ofrece información inmediata sobre la calidad del código.

En otras palabras, una prueba fallida es una buena prueba.

Escribir el código. Ahora es el momento de programar. Pero lo bueno es que no hace falta crear una aplicación totalmente funcional y optimizada. Solo hace falta programar lo justo y necesario para que el programa pase los tests.

Cuando se programa para pasar una prueba a la vez, no hace falta preocuparse por el impacto que un segmento de código tiene sobre otro. Están diseñados para funcionar de forma independiente. Mientras funcionen por sí solos, todo va bien.

Una vez programado solo hay que ejecutar las pruebas hasta que ninguna falle. Un test superado indica que el código cumple los requisitos especificados en el caso de prueba.

Fase de refactorización. Refactorizar es reestructurar para mejorar el código (por supuesto sin cambiar ningún comportamiento externo). El ciclo se repite. Se escribe otra prueba. Se ejecuta y se observa si falla. Se escribe código para esa prueba hasta que pase. Con el tiempo, se obtiene cada vez más información de las pruebas, se refactoriza continuamente el código y se mejora su diseño, legibilidad y mantenimiento.

Desarrollo guiado por comportamiento

El desarrollo guiado por comportamiento es un enfoque en el desarrollo de software que hace hincapié en el comportamiento de una aplicación para las necesidades empresariales. Se concibió para resolver los problemas derivados de unas especificaciones de requisitos mal definidas y para alinear a los profesionales de la empresa y del quality assurance.

Los tres principios del desarrollo guiado por comportamiento son:

Centrarse en el comportamiento o los resultados deseados.

Colaboración entre desarrolladores, testers y partes interesadas del negocio.

Uso del lenguaje natural para la comunicación y el entendimiento.

El desarrollo guiado por comportamiento es una evolución de la metodología de desarrollo guiado por pruebas, pero cambia el enfoque de una alta cobertura de pruebas a la definición del comportamiento de la aplicación.

El desarrollo guiado por comportamiento es una poderosa herramienta en el desarrollo de software, que sirve de puente entre las partes interesadas técnicas y no técnicas y que permite a todos entenderse con claridad. El desarrollo guiado por comportamiento garantiza que los proyectos de desarrollo sigan centrándose en las necesidades de la empresa, al tiempo que satisfacen los requisitos del usuario.

Y más importante, el desarrollo guiado por comportamiento mejora la calidad del código, lo que ayuda a reducir gastos variables como los de mantenimiento y minimiza el riesgo del proyecto, lo que garantiza que se mantenga firme durante años.

En esencia, las prácticas de desarrollo guiado por comportamiento son una poderosa herramienta para ayudar a los equipos a desarrollar mejor software mediante ejemplos en lenguaje natural.

Gherkin

En el mundo dinámico del desarrollo de software, la comunicación eficaz es primordial. A menudo, traducir requisitos técnicos complejos en instrucciones comprensibles puede parecer complicado. Afortunadamente, Gherkin, un lenguaje adaptado al desarrollo guiado por comportamiento, ofrece un formato estructurado y legible por humanos para definir características, escenarios y pasos del software con notable claridad y precisión.

Gherkin se desarrolló originalmente como parte del framework de testing Cucumber, que fomenta la colaboración entre desarrolladores de software y partes interesadas no técnicas al permitirles escribir especificaciones ejecutables en lenguaje natural.

Gherkin sigue una sintaxis sencilla con palabras clave estructuradas como Feature, Scenario, Given, When, Then, y Scenario Outline. Esta estructura coherente mejora la legibilidad y facilita a los equipos la comprensión y redacción de escenarios.

Una de las principales ventajas de Gherkin es su capacidad para facilitar la colaboración entre equipos. Al utilizar un lenguaje común para definir los requisitos, desarrolladores, testers y otras partes interesadas pueden comunicarse eficazmente y alinear su comprensión de las características del software.

Los escenarios de Gherkin pueden ejecutarse como pruebas automatizadas, lo que ayuda a garantizar que el software se comporta según lo previsto. Esta naturaleza ejecutable permite a los equipos validar sus requisitos continuamente a lo largo del proceso de desarrollo, lo que reduce el riesgo de malentendidos o interpretaciones erróneas.

Gherkin se integra perfectamente con diversos frameworks y herramientas de pruebas, como Cucumber, SpecFlow y Behat. Esta integración permite a los equipos automatizar la ejecución de los escenarios de Gherkin y generar informes de prueba detallados, mejorando la eficiencia y la eficacia de su testing.

Gherkin fomenta una mentalidad de mejora continua mediante la promoción de prácticas de desarrollo iterativo. Los equipos pueden utilizar los resultados obtenidos de la ejecución de escenarios Gherkin para identificar áreas de mejora en su software y refinar sus requisitos en consecuencia.

La comunidad Gherkin es activa, con una gran cantidad de recursos disponibles para apoyar a los usuarios en todos los niveles de habilidad. Desde tutoriales y documentación hasta foros y grupos de usuarios, hay muchas oportunidades para aprender y compartir las mejores prácticas relacionadas con Gherkin y desarrollo guiado por comportamiento.

Al adoptar las normas y la estructura de Gherkin, los equipos pueden aclarar dudas en sus requisitos. Esto acelera el desarrollo y aumenta la confianza en el producto final. Gherkin ayuda tanto a la gente que forma parte del equipo de desarrollo como a la que no lo es a trabajar mejor juntos, trayendo una nueva era de trabajo en equipo en la que todo el mundo entiende lo que hacer.

Gherkin ayuda a los equipos a expresar muy claramente las necesidades de su software. Esto facilita el trabajo en equipo y la aportación de nuevas ideas. Cuando los equipos entienden bien Gherkin, pueden manejar situaciones complicadas con confianza, asegurándose de que su software hace lo que se supone que debe hacer y satisface a sus dientes en el cambiante mercado actual.

Herramientas útiles

Elegir las herramientas adecuadas puede marcar una diferencia significativa en la eficacia y eficiencia del testing y el quality assurance de las aplicaciones móviles. Estas son algunas de las herramientas más populares:

Appium

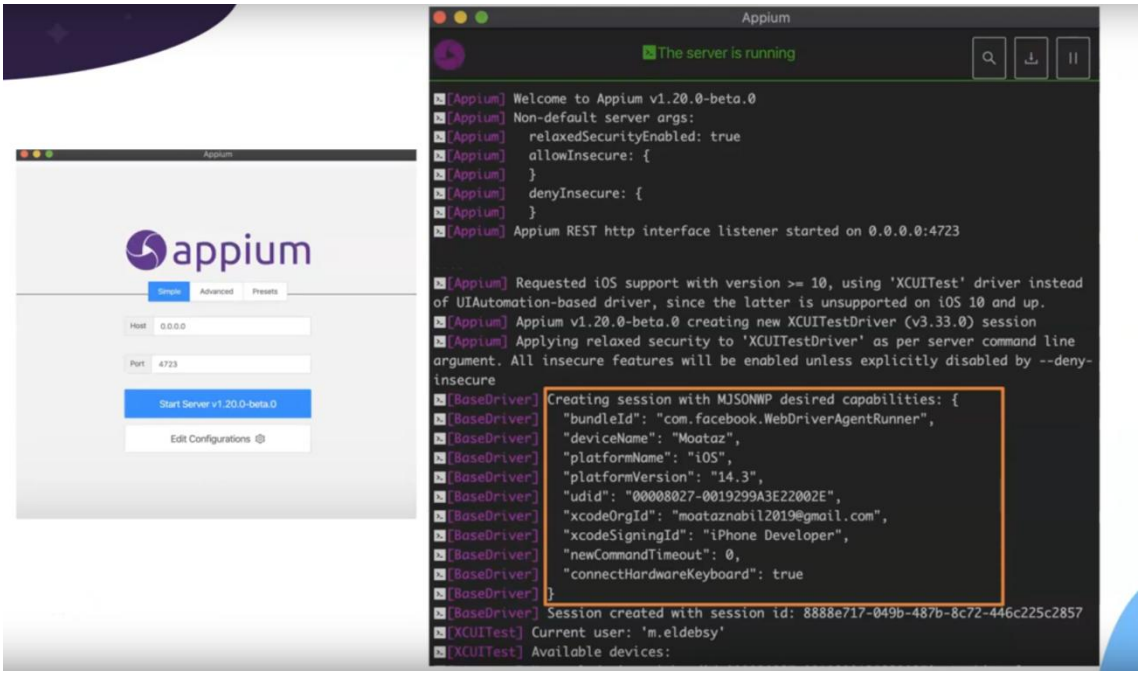

Ilustración 1 Ejemplo de uso de Appium Fuente: Moataz Nabil

Appium es una herramienta de código abierto para automatizar las pruebas de aplicaciones móviles. Es compatible con aplicaciones nativas, web móviles e híbridas en plataformas iOS y Android. Appium permite escribir pruebas en varios lenguajes de programación, lo que la convierte en una opción flexible para los desarrolladores y testers.

La API de Appium está basada en la API de Selenium, WebDriver. Selenium trabajó con varios proveedores de navegadores web y con el grupo de estándares W3C para convertir su API en un estándar oficial para navegadores web, denominado especificación WebDriver. Todos los navegadores principales ahora implementan capacidades de automatización en línea con la especificación WebDriver, sin que el equipo de Selenium tenga que mantener ningún software que realice la automatización real.

Appium tiene los siguientes objetivos:

Hacer que las capacidades de automatización específicas de cada plataforma estén disponibles en una API estándar multiplataforma.

Facilitar el acceso a esta API desde cualquier lenguaje de programación.

Proporcionar herramientas que permitan a la comunidad desarrollar cómodamente extensiones de Appium.

Calabash

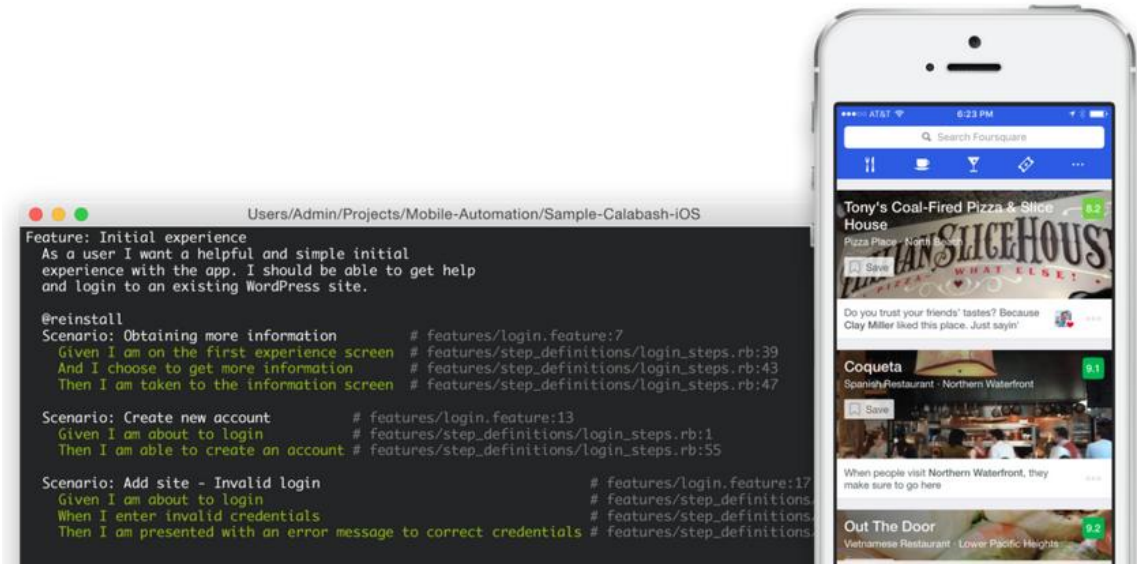

Ilustración 2 Ejemplo de uso de Calabash Fuente: Jean-Baptiste Crouigneau

Calabash es una herramienta de código abierto para pruebas de aceptación automatizadas de aplicaciones Android e iOS. Permite escribir y ejecutar pruebas en Ruby usando Cucumber (una herramienta de desarrollo guiado por comportamiento) por lo que se pueden escribir un formato de lenguaje natural, lo que facilita la comprensión y la contribución de los interesados sin conocimientos técnicos.

Funciona permitiendo interacciones de interfaz de usuario automáticas dentro de una aplicación móvil, como pulsar botones, introducir texto, validar respuestas, etc. Puede configurarse para ejecutarse en diferentes dispositivos Android e iOS, lo que proporciona respuestas y validaciones en tiempo real.

Espresso

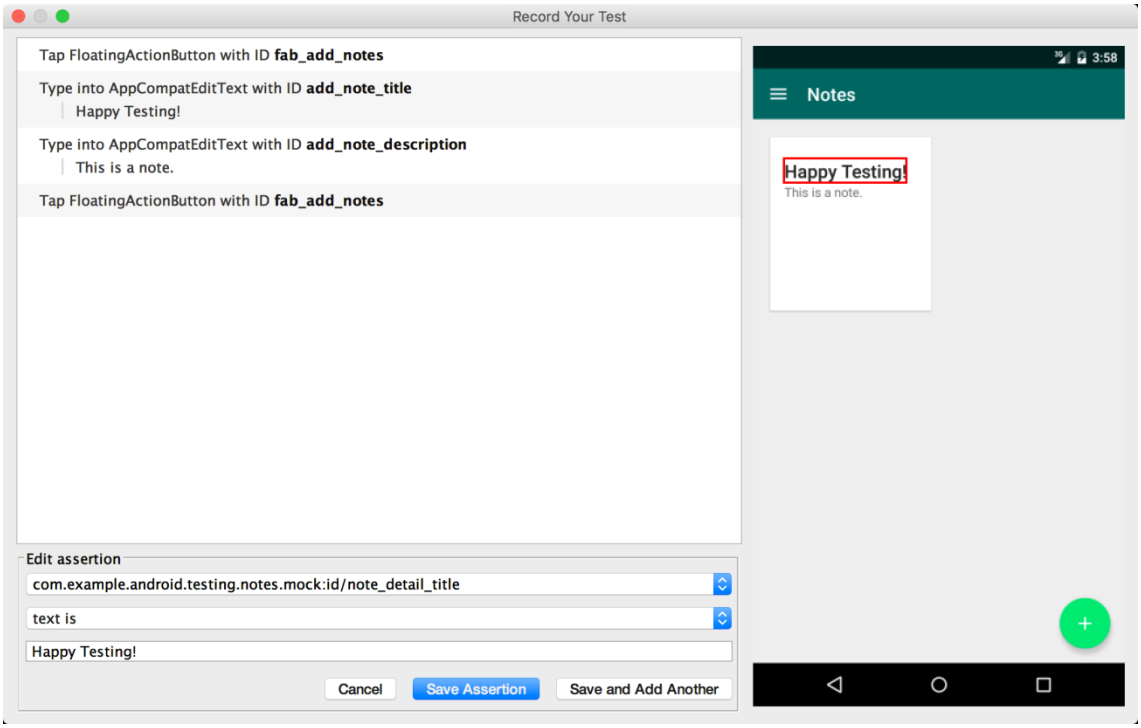

Ilustración 3 Ejemplo de uso de Espresso Fuente: Android Developers

Espresso es un framework de testing proporcionado por Google para aplicaciones Android. Es muy utilizado por sus funciones avanzadas de sincronización, que gestionan automáticamente la velocidad de ejecución de las pruebas para garantizar resultados fiables.

La API principal es pequeña, predecible y fácil de aprender, pero permanece abierta para la personalización. Las pruebas con Espresso exponen claramente las expectativas, las interacciones y las aserciones sin la distracción del contenido estándar, la infraestructura personalizada o los detalles de implementación desordenados que se interponen.

XCTest

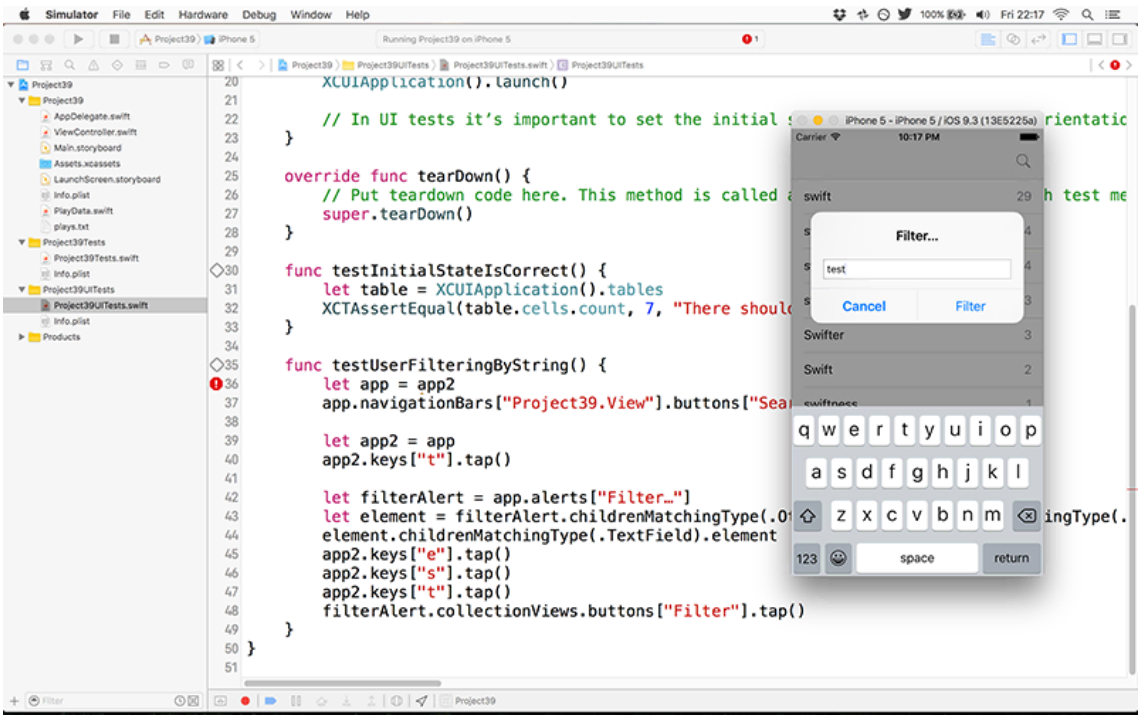

Ilustración 4 Ejemplo de uso de XCTest Fuente: Paul Hudson

XCTest es un framework proporcionado por Apple para aplicaciones iOS. Se integra perfectamente con Xcode, el IDE de Apple, lo que facilita la escritura y ejecución de pruebas. XCTest admite tanto pruebas unitarias como pruebas de interfaz de usuario.

XCTest se basa en el protocolo WebDriver. El protocolo WebDriver es un estándar del sector para la automatización de navegadores. Apple ha creado un marco de pruebas potente y fácil de usar, perfecto para cualquiera que desee automatizar las pruebas de sus aplicaciones.

XCTest se anunció por primera vez en la WWDC de 2016, junto con el lanzamiento de iOS 10. Antes de XCTest, las pruebas automatizadas en iOS solo eran posibles utilizando el framework UI Automation. UI Automation tenía algunas limitaciones, como la necesidad de un Mac y conocimiento de JavaScript.

XCTest se creó para hacer frente a estas limitaciones permitiendo ejecutar pruebas directamente en un iPhone o iPad, sin necesidad de un Mac. Con XCTest, Apple quería desarrollar un framework que fuera fácil de usar y no requiriera conocimientos previos de programación. También querían asegurarse de que XCTest se basara en protocolos estándar del sector, como el protocolo WebDriver. De este modo, confirmaron que XCTest dispondría de una amplia gama de recursos y asistencia.

Beneficios de usar XCTest:

En primer lugar, XCTest es muy fácil de usar y sólo requiere experiencia en Swift, que es un lenguaje de desarrollo estándar. Esto hace que sea la herramienta perfecta para cualquier persona que quiera empezar a programar pruebas automatizadas, pero no sabe por dónde empezar.

En segundo lugar, XCTest es muy intuitivo y fácil de entender. La sintaxis es similar a la del inglés, por lo que resulta sencillo leer y escribir pruebas.
Por último, XCTest es un framework completo que se puede utilizar para algo más que pruebas de interfaz de usuario. Con XCTest, se puede escribir pruebas unitarias, pruebas de rendimiento e incluso probar la compatibilidad de tu aplicación con otras aplicaciones y servicios.

JIRA

JIRA es una herramienta popular de gestión de proyectos para el seguimiento de errores, problemas y tareas a lo largo del ciclo de vida del desarrollo. Ofrece funciones completas de planificación, seguimiento y elaboración de informes, lo que facilita la gestión y coordinación de las pruebas.

Desarrollado por la empresa australiana Atlassian en 2002, JIRA recibe actualizaciones periódicas. Originalmente, JIRA se creó para ayudar a los equipos de software en el seguimiento de errores. Ahora, Jira es conocido en todo el mundo y se ha convertido en una de las soluciones de software de gestión de proyectos más utilizadas.

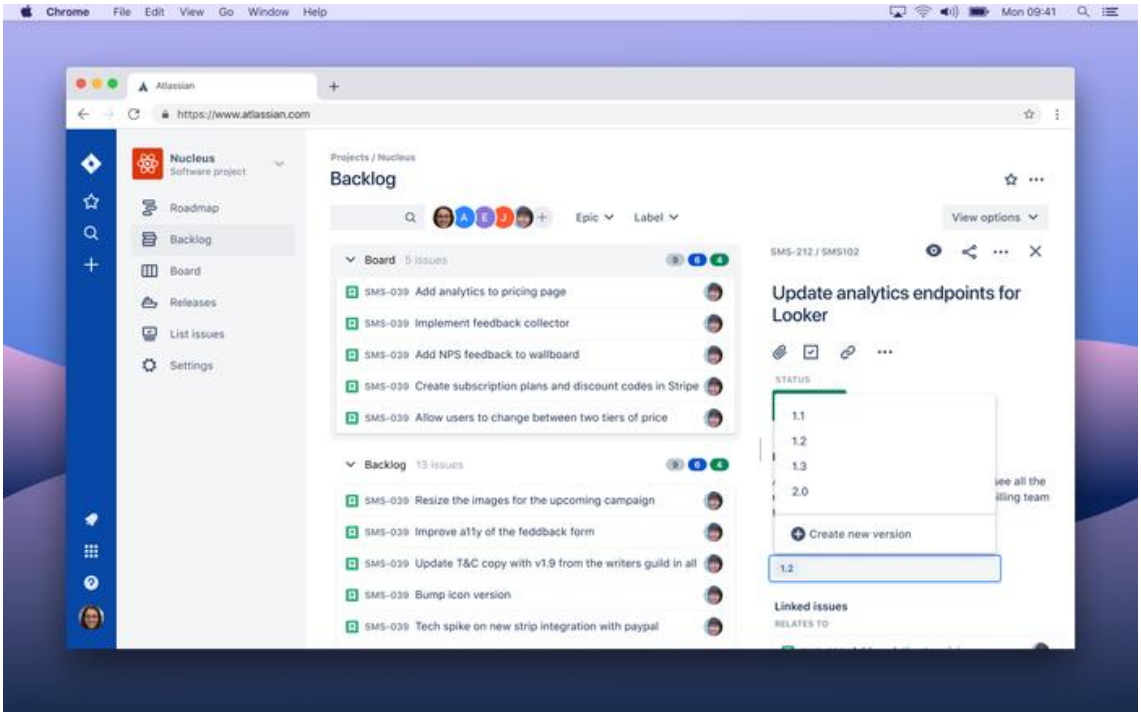

Ilustración 5 Interfaz de usuario de JIRA Fuente: Atlassian

TestRail

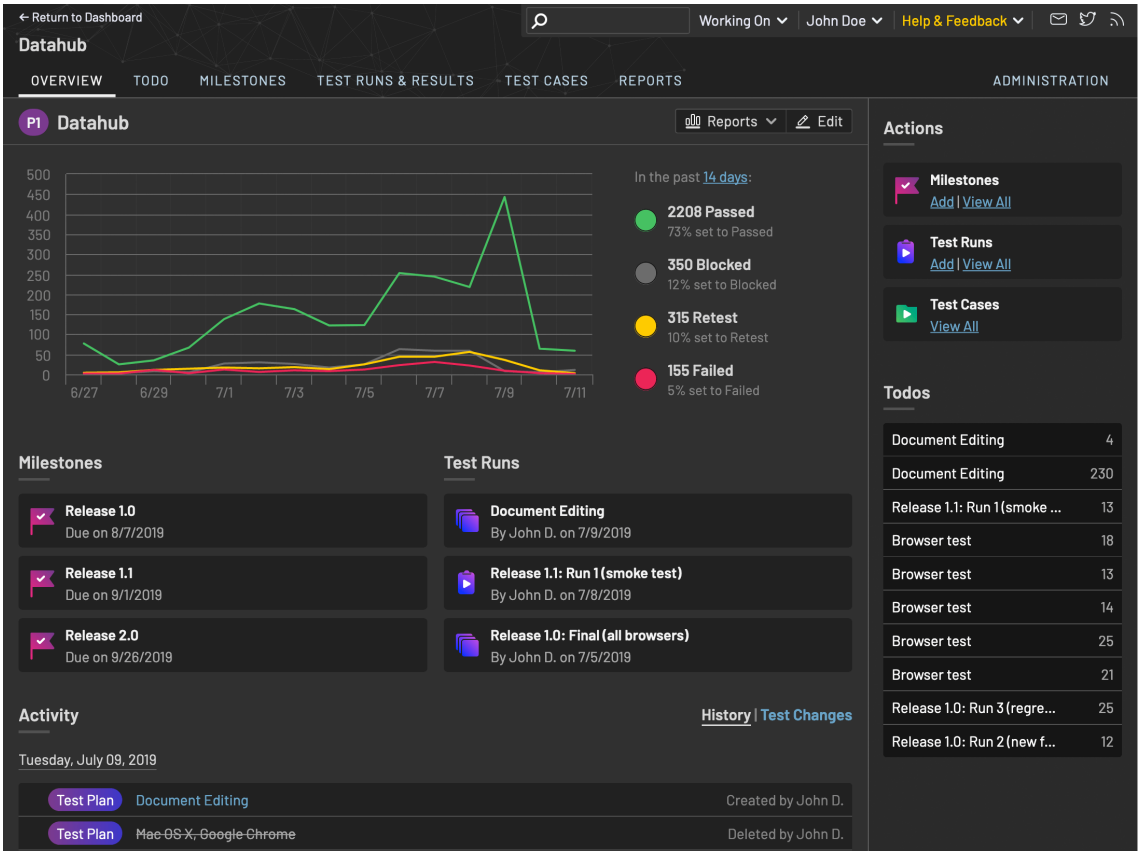

Ilustración 6 Interfaz de usuario de TestRail Fuente: TestRail

TestRail es una herramienta de gestión de casos de prueba que ayuda a gestionar y realizar un seguimiento de los tests de software. Ofrece potentes funciones para crear casos de prueba, gestionar las ejecuciones de las pruebas e informar de los resultados.

La forma tradicional de gestionar y documentar los casos de prueba en papel, en wikis, herramientas de seguimiento de errores o sistemas generales de gestión de documentos alcanza rápidamente sus límites. TestRail permite crear, gestionar y organizar casos y suites de prueba dentro de una interfaz de usuario y una estructura de aplicación optimizadas.

Las listas de tareas, los filtros y las notificaciones por correo electrónico de TestRail ayudan a coordinar las pruebas y a aumentar la productividad y la responsabilidad de los testers. El objetivo es que cada miembro del equipo conozca sus tareas en todo momento y que los jefes de equipo puedan asignar nuevas tareas a los probadores en función de su carga de trabajo.

Jenkins

Ilustración 7 Entorno de trabajo de Jenkins Fuente: Dotan Horovits

Jenkins es una herramienta de código abierto utilizada para la integración continua y entrega continua. Automatiza el proceso de creación, prueba y despliegue de aplicaciones, lo que facilita las pruebas continuas y la entrega de aplicaciones de alta calidad.

Jenkins es uno de los primeros servidores de integración continua de código abierto y sigue siendo la opción más utilizada en la actualidad. Originalmente formaba parte del proyecto Hudson, pero la comunidad y el código base se separaron a raíz de los conflictos de marca con Orade tras la

adquisición de Sun Microsystems, los desarrolladores originales. Hudson se lanzó originalmente en 2005, mientras que la primera versión como Jenkins se realizó en 2011.

A lo largo de los años, Jenkins ha evolucionado hasta convertirse en un sistema potente y flexible de automatización de tareas relacionadas con el software. Jenkins en sí sirve principalmente como un framework de automatización con gran parte de la lógica importante implementada a través de una biblioteca de plugins. Todo, desde escuchar web hooks o ver repositorios hasta construir entornos y soporte de lenguajes, es manejado por plugins. Si bien esto proporciona una gran flexibilidad, su proceso de integración continua puede llegar a depender de numerosos plugins de terceros, que pueden ser frágiles.

Sauce Labs

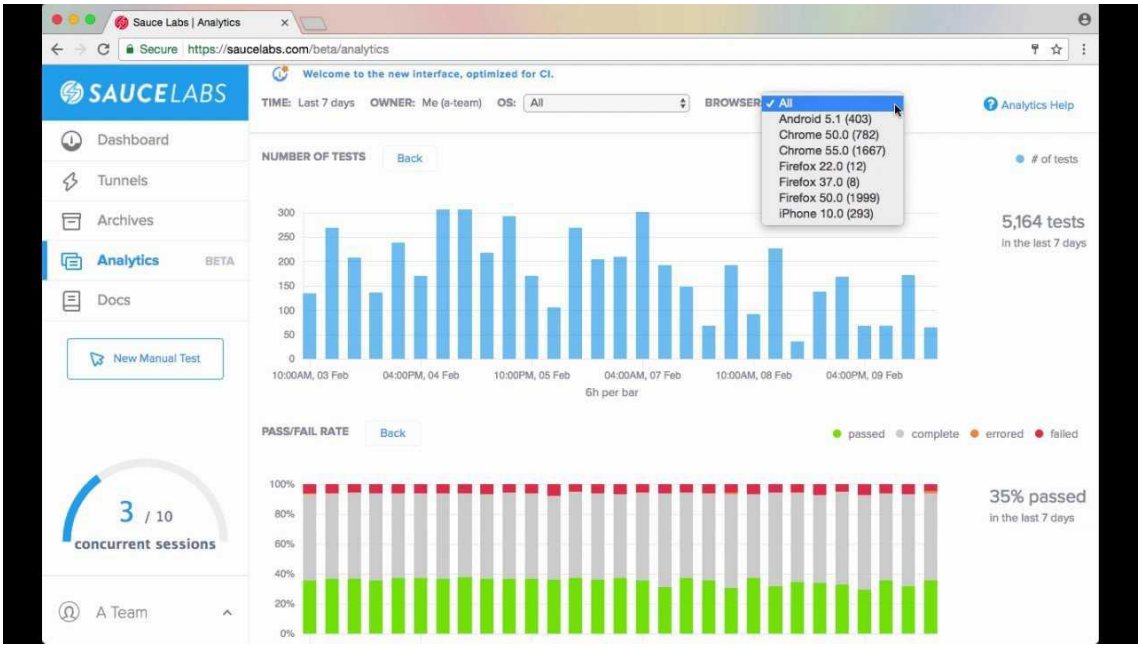

Ilustración 8 Entorno de trabajo de Sauce Labs Fuente: Apps & Reports

Sauce Labs es una plataforma en la nube que sirve para probar aplicaciones web y móviles. Proporciona acceso a muchos dispositivos y sistemas operativos virtuales y reales, lo que permite realizar pruebas de compatibilidad exhaustivas sin necesidad de un laboratorio físico de dispositivos.

Modas actuales

El sector de las aplicaciones móviles está en constante evolución, impulsado por los avances tecnológicos y los cambios en las expectativas de los usuarios. A medida que cambia el panorama, también lo hacen las metodologías y prioridades de las pruebas de aplicaciones móviles. He aquí un vistazo a algunas tendencias emergentes que están configuradas para dar forma al futuro de este dominio crucial.

Cambio hacia la integración continua

En la era de la agilidad y DevOps, el modelo tradicional de pruebas, en el que se tratan como una fase separada después del desarrollo, se está quedando obsoleto. La integración continua integra las pruebas en todas las fases del ciclo de vida del desarrollo. Este enfoque garantiza que cualquier problema se identifique y aborde en tiempo real, facilitando lanzamientos más rápidos y frecuentes. A medida que las empresas traten de lanzar actualizaciones y nuevas funciones con rapidez, la integración continua se convertirá en la norma, garantizando que la calidad no quede relegada a un segundo plano.

El auge de la IA y el Machine Learning en el testing

La Inteligencia Artificial (IA) y el Machine Learning están haciendo importantes incursiones en el testing de aplicaciones móviles. Estas tecnologías pueden analizar grandes cantidades de datos de los resultados de las pruebas más rápido que cualquier ser humano. Los análisis predictivos pueden prever posibles áreas problemáticas, mientras que los algoritmos de aprendizaje automático pueden optimizar los conjuntos de pruebas identificando qué pruebas son más relevantes en función de los cambios de código. Además, la IA puede simular la interacción de miles de usuarios con la aplicación, lo que ofrece información sobre escenarios de uso reales y posibles problemas que puede tener la aplicación.

Mayor importancia de las pruebas de seguridad ante los crecientes ciberataques

A medida que los ciberataques se vuelven más sofisticados, también deben serlo las defensas contra ellos. Las aplicaciones móviles, que a menudo almacenan o tienen acceso a datos sensibles de los usuarios, son objetivos prioritarios de los cibercriminales. En el futuro se hará aún más hincapié en las pruebas de seguridad. Esto no sólo implica la comprobación de vulnerabilidades, sino también la simulación de ciberataques avanzados para evaluar la resistencia de una aplicación. A medida que las normativas sobre privacidad de datos se hacen más estrictas en todo el mundo, garantizar la seguridad de una aplicación no sólo consistirá en proteger a los usuarios, sino también en cumplir la normativa y evitar posibles repercusiones legales.

En conclusión, el futuro del testing de aplicaciones móviles está preparado para ser dinámico e impulsado tanto por la innovación tecnológica como por los retos en constante evolución del mundo digital. Si los desarrolladores y empresas se mantienen al tanto de estas tendencias y se adaptan en consecuencia, pueden garantizar que sus aplicaciones sigan siendo sólidas, relevantes y se hagan eco entre los usuarios en los próximos años.

Conclusiones

Dada la feroz competencia en el mercado de las aplicaciones móviles, ya no basta con que el rendimiento sea aceptable. Una aplicación debe ofrecer una experiencia de usuario fluida y sin errores para satisfacer a los usuarios y hacer que vuelvan.

El testing de aplicaciones móviles y los procesos de quality assurance requieren tiempo y recursos, pero la recompensa en términos de longevidad y éxito de una aplicación en el mercado es incalculable.

Una aplicación bien probada, validada mediante rigurosos procesos de control de calidad, garantiza un producto de alta calidad que se gana la confianza de los clientes y mejora la reputación de la marca.

Integrar estrategias sólidas de pruebas y control de calidad en el proceso de desarrollo puede sentar las bases para ofrecer aplicaciones de alta calidad y éxito que calen en los usuarios y resistan el paso del tiempo.

Bibliografía

Myers, G. J., Badgett, T., & Sandler, C. (2012). *The Art of Software Testing*. En *Wiley eBooks*. https://doi.org/10.1002/9781119202486

Pecorelli, F., Catolino, G., Ferrucci, F., De Lucia, A., & Palomba, F. (2021). *Software testing and Android applications: a large-scale empirical study*. *Empirical Software Engineering*, *27*(2). https://doi.org/10.1007/s10664-021-10059-5

Humble, J., & Farley, D. (2010). *Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation*. https://openlibrary.org/books/OL25154801M/Continuous_Delivery

Fewster, M., & Graham, D. (2000). *Software Test Automation: Effective Use of Test Execution Tools*. http://ci.nii.ac.jp/ncid/BB0177634X

Tan, K. H. (2012). How Google tests software by James A. Whittaker, Jason Arbon and Jeff Carollo. *Software Engineering Notes*, *37*(5), 44-45. https://doi.org/10.1145/2347696.2347723

North, Dan (March 2006). "Introducing BDD". Dan North. https://dannorth.net/introducing-bdd/

"Gherkin" – Cucumber documentation https://cucumber.io/docs/gherkin/reference/

GeeksforGeeks. (2022, noviembre 17). History of Software Testing. GeeksforGeeks. https://www.geeksforgeeks.org/history-of-software-testing/

Appium in a Nutshell - Appium Documentation. (s. f.). https://appium.io/docs/en/latest/intro/

Sauce Labs Demo, Overview, Reviews, Features and Pricing - 2024. (s. f.). AppsandReports.com. https://www.appsandreports.com/software/Sauce-Labs

Horovits, D. (2022, 31 mayo). Fighting Slow and Flaky CI/CD Pipelines Starts with Observability. Logz.io. https://logz.io/learn/cicd-observability-jenkins/#cicd_observability

Knight, S. (2023, 8 septiembre). Announcing TestRail 6.0 with UI Enhancements and Docker Support - TestRail. TestRail | The Quality OS for QA Teams. https://www.testrail.com/blog/testrail-6-0-release/

Calabash - Software Testing Tools Guide. (2018, 19 febrero). Software Testing Tools Guide. https://www.testingtoolsguide.net/tools/calabash/

Create UI tests with Espresso Test Recorder. (s. f.). Android Developers. https://developer.android.com/studio/test/other-testing-tools/espresso-test-recorder

Whammy. (2019, 27 noviembre). New features for next-gen projects in Jira Software Cloud: Releases and Versions. Atlassian Community. https://community.atlassian.com/t5/Jira-articles/New-features-for-next-gen-projects-in-Jira-Software-Cloud/ba-p/1232149

User interface testing with XCTest – Hacking with Swift https://www.hackingwithswift.com/read/39/8/user-interface-testing-with-xctest