Preguntas teoricas- Prueba tecnica QA automation RappiPay

RappiPay

1. ¿Cuáles son las principales diferencias entre automatizar en Android vs iOS con Appium?

R//: Las principales diferencias están en el motor de automatización que usa Appium para comunicarse con cada plataforma, que son UiAutomator2 y XCUITest. Esto hace que para cada plataforma cambie la forma de localizar elementos, la forma de interactuar con ciertos elementos e incluso la estabilidad de algunas acciones. Por ejemplo, para Android es más común usar resource-id y content-desc, y para iOS se suele usar accessibility id o name. También toca tener en cuenta que los tiempos de ejecución cambian para cada plataforma, la gestión de permisos, las configuraciones del entorno y, para iOS, tenemos que tener presente la existencia de certificados y firmas, ya que suele ser un entorno más cerrado y controlado.

2. ¿Cómo diseñaría un framework de automatización mobile multiplataforma que sea mantenible y escalable?

R//: Lo primero en lo que me centraría es elegir el patrón con el cual lo trabajaría, y eso lo haría dependiendo de la necesidad del negocio, ya sea Page/Screen Object o Screenplay, teniendo claras las diferencias entre ambos: uno es más directo y práctico, y el otro es más robusto, modular y escalable, pero puede ser más difícil de mantener si el equipo no tiene buena adopción y disciplina.

Estos patrones ayudan a la separación de la lógica de pruebas de las interacciones con la interfaz, lo cual se puede manejar muy bien con Screen Object o, en Screenplay, con capas como Interactions y Tasks.

De igual forma, agregaría una capa de manejo de drivers, manejo de esperas y utilidades comunes. La de drivers para tener estructurada la toma de decisiones por plataforma y no tener código quemado, lo cual nos permite manejar multiplataforma. Las esperas explícitas serían importantes para el control del comportamiento de la app y las dejaría en una capa de utils si son recurrentes.

También complementaría con una configuración mediante archivos de propiedades o perfiles por plataforma, ya sea Android o iOS, permitiéndome no tener código quemado, ya que esta parte de configuración estaría desacoplada.

Para las pruebas tendría una capa de servicios o pasos recurrentes (Tasks/Utils en Screenplay) o flujos reutilizables. Además, agregaría logging, reportes y manejo de excepciones, lo cual en Screenplay también se puede estructurar por capas.

3. ¿Qué criterios usa para seleccionar qué casos se deben automatizar y cuáles no?

Para seleccionar casos de prueba me baso en escenarios repetitivos, críticos para el negocio y que sean poco cambiantes (estables). También tengo en cuenta los escenarios que se ejecutan en regresión o que son flujos críticos del usuario, es decir, los principales.

Preguntas teoricas- Prueba tecnica QA automation RappiPay

RappiPay

Cuando decido no automatizar escenarios, es porque tengo en cuenta escenarios que son muy cambiantes o difíciles de automatizar y poco prácticos, como pruebas muy visuales o escenarios que más bien corresponden a pruebas exploratorias, las cuales cambian según el criterio del tester o del entorno, es decir, factores externos.

4. ¿Qué estrategias aplicaría para integrar pruebas automáticas en un pipeline CI/CD?

Agregaría las pruebas automatizadas al pipeline de acuerdo al modelo de negocio y las necesidades del cliente. Lo más común es agregarlas como stages después de un stage de despliegue a ambientes de pruebas, es decir, se ejecutarían después del despliegue (para mobile, creación del APK/IPA apuntando a ambientes de prueba).

A nivel de pipeline, si tengo varios tipos de pruebas, les aplicaría paralelismo de CI/CD para ejecutarlas al tiempo, por ejemplo, pruebas de performance, seguridad y E2E automatizadas en paralelo. Y a nivel de las E2E también tendría en cuenta que, si las pruebas son muy robustas o extensas, aplicar paralelismo de ejecución, siempre que el negocio tenga la capacidad de dispositivos o emuladores para soportarlo.

De igual forma, agregaría reportes automáticos para una fácil interpretación de resultados y usaría etiquetas o suites de pruebas para permitir ejecutar por plataformas, por contexto o por necesidad, de acuerdo a la naturaleza del despliegue (regresión, smoke, etc.).

5. Ventajas y desventajas de dispositivos físicos vs granja de dispositivos

Los dispositivos físicos son la manera más efectiva de ver el comportamiento real de una aplicación de cara al usuario, por ejemplo en rendimiento, red y hardware. Sin embargo, es más costoso mantenerlos y es difícil escalar, especialmente para paralelismo.

Las granjas de dispositivos son útiles para ejecutar pruebas en paralelo, siguiendo la recomendación de 1 hilo por dispositivo. Esto permite cubrir muchos casos de prueba y tener acceso a diferentes sistemas operativos. Pero hay que tener en cuenta que pueden existir limitaciones de acceso, latencias y configuraciones compartidas.

6. ¿Qué patrones de diseño ha implementado?

He trabajado con Screenplay usando Serenity y también con Page/Screen Object. Para que el framework sea más mantenible, los complemento con DriverFactory para crear drivers según la plataforma, y con clases base o capas base para centralizar configuraciones y utilidades recurrentes.