**Punto 5 . Comparativa Appium vs Espresso**

Espresso se encuentra dentro de la aplicación y está escrito por Google con el único propósito de probar funcionalmente una interfaz de usuario de Android. En términos de estilo de prueba, la descripción de Espresso menciona "pruebas de caja blanca". Como Espresso está dentro de la aplicación, tiene acceso y sabe cómo usar el código que realmente ejecuta la aplicación para realizar pruebas más exhaustivas de cada elemento.

Appium está diseñado para ser una plataforma de prueba multiplataforma. Como tal, una de las ventajas y desventajas se centra en la "prueba de caja negra" que solo prueba lo que se ha expuesto externamente a la aplicación. Con el uso del marco de Android UIAutomator, Appium puede acceder a todos los elementos de la interfaz de usuario que un usuario verá.

Dos de los beneficios de ser externo a la aplicación son la capacidad de validar los datos que entran y salen de la capa de servicio detrás de escena, y la capacidad de ejecutar simulaciones diciendo a la aplicación que el dispositivo ha girado, o que el botón de inicio ha sido empujado.

Si bien Appium y Espresso pueden satisfacer la necesidad de realizar pruebas de UI para su aplicación Android, realmente se reduce al alcance de sus pruebas. Idealmente, podría aprovechar ambos marcos para agrandar la cantidad de pruebas realizadas a la aplicación, pero como sucede con la mayoría de las cosas, es más tiempo del que la mayoría de las personas tiene para invertir por adelantado.

**Como resumen final**

Si está seleccionando solo un marco, entonces los desarrolladores que construyan una aplicación nativa de Android que tenga su alcance limitado solo a la aplicación y deseen pruebas de interfaz de usuario completas e integradas, Espresso definitivamente satisfará esa necesidad.

Si las pruebas necesitan ser compatibles con múltiples plataformas (por ejemplo, iOS, híbrido y Android), y necesita validar cómo la aplicación reacciona a factores externos como la rotación de la pantalla y / o desea ejecutar la prueba en paralelo usando un servicio, Appium satisfará mejor sus necesidades.

**Tabla comparativa;**

|  |  |  |  |
| --- | --- | --- | --- |
| **Criterio** | **Appium** | **Espresso** | **Comentarios** |
| **Alcance** | Cualquier plataforma, IOS, Android. | Solo Android. | Appium cuenta con mayor número de plataformas para funcionar. |
| **Lenguaje Escritura** | Java, C#, Ruby, Phyton. | Java. | También Appium cuenta con más lenguajes de escritura. |
| **Requisitos de Prueba** | Caja Negra. | Caja Blanca. | Lo definimos anteriormente Appium cuenta con los requisitos de prueba caja negra y espresso caja blanca. |
| **Similares** | Selenium. | Ninguno. | Tecnologías Similares , Appium cuenta con una tecnología similar que es Selenium. |
| **Velocidad** | Cada prueba sobre 1 minuto. | Cada prueba sobre 2 segundos. | Espresso tiene un registro de prueba mejor que Appium. |
| **Compañías que lo usan** | Amazon, Perfecto Mobile, Testdroid. | Trabajar sobre la nube. | En esta casilla vemos que compañías pueden usar Appium y como Espresso se dedica más a plataformas en la nube. |
| **Facilidad de configuración** | Difícil. | Fácil. | Espresso tiene una mayor velocidad de configuración. |
| **Adecuado para** | Equipos que prueben apps en IOS y Android. | Específico para desarrolladores. | Appium es adecuado para equipos testing de apps mientras que Espresso es más específico para desarrolladores |
| **Impulsores** | Open Source. | Google. | En esta casilla vemos sus principales impulsores |
| **Localizadores de Objetos** | Xpath. | Id(From R file). | Los localizadores de objetos que usan estas tecnologías. |