Contenido

[1. Autores del trabajo, planificación y entrega 3](#_Toc511067823)

[1.1 Autores 3](#_Toc511067824)

[1.2 Planificación 3](#_Toc511067825)

[1.3 Entrega 3](#_Toc511067826)

[2. Descripción de las tecnologías 4](#_Toc511067827)

[2.1 Descripción de Appium. 4](#_Toc511067828)

[Appium: 4](#_Toc511067829)

[2.2 Descripción de Espresso 5](#_Toc511067830)

[Espresso: 5](#_Toc511067831)

[3. Criterios de comparación 7](#_Toc511067832)

[3.1 Categoría A: General 7](#_Toc511067833)

[3.1.1 Criterio A.1: Autor de la herramienta 7](#_Toc511067834)

[3.1.2 Criterio A.2: Fecha de creación de la herramienta. 7](#_Toc511067835)

[3.1.3 Criterio A.3: Versión actual de la herramienta. 7](#_Toc511067836)

[3.1.4 Criterio A.4: Requisitos de prueba. 7](#_Toc511067837)

[3.1.5 Criterio A.5: Herramientas similares. 7](#_Toc511067838)

[3.1.6 Criterio A.6: Compañías que lo usan. 7](#_Toc511067839)

[3.1.7 Criterio A.7: Dificultad de configuración. 8](#_Toc511067840)

[3.1.8 Criterio A.8: Impulsores. 8](#_Toc511067841)

[3.2 Categoría B: Rendimiento 8](#_Toc511067842)

[3.2.1 Criterio B.1: Lenguaje Escritura 8](#_Toc511067843)

[3.2.2 Criterio B.2: Sistema operativo 8](#_Toc511067844)

[3.2.3 Criterio B.3: Velocidad de las pruebas. 8](#_Toc511067845)

[3.3 Categoría C: Utilidades 8](#_Toc511067846)

[3.3.1 Criterio C.1: Alcance. 8](#_Toc511067847)

[3.3.2 Criterio C.2: Adecuado para. 8](#_Toc511067848)

[3.3.3 Criterio C.3: Localizadores de objetos. 9](#_Toc511067849)

[3.3.4 Criterio C.4: Testing Factores externos. 9](#_Toc511067850)

[3.3.5 Criterio C.5: Testing Actividades externas. 9](#_Toc511067851)

[3.3.6 Criterio C.6: Mensajes toast. 9](#_Toc511067852)

[3.3.7 Criterio C.7: Medición de test. 9](#_Toc511067853)

[3.3.8 Criterio C.8: Multilenguaje. 9](#_Toc511067854)

[3.3.9 Criterio C.9: Soporte de apps web. 9](#_Toc511067855)

[4. Evaluación de los criterios por tecnología 10](#_Toc511067856)

[4.1 Evaluación de los criterios para Appium 10](#_Toc511067857)

[4.2 Evaluación de los criterios para Espresso 11](#_Toc511067858)

[5. Comparación de las tecnologías 12](#_Toc511067859)

[Tabla comparativa 13](#_Toc511067860)

[6. Recomendaciones 14](#_Toc511067861)

[6.1 Situación 1 14](#_Toc511067862)

[6.1.1 Descripción de la situación 14](#_Toc511067863)

[6.1.2 Recomendación de tecnología a utilizar 14](#_Toc511067864)

[6.2 Situación 2 14](#_Toc511067865)

[6.2.1 Descripción de la situación 14](#_Toc511067866)

[6.2.2 Recomendación de tecnología a utilizar 14](#_Toc511067867)

# 1. Autores del trabajo, planificación y entrega

## 1.1 Autores

Grupo M1

Amir Sholkami Berube (coordinador)

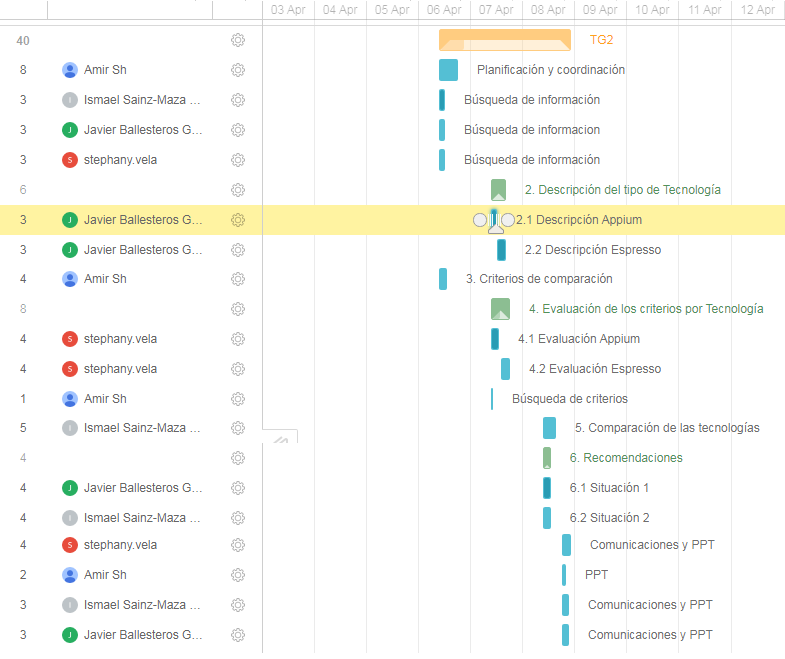
Javier Ballesteros García

Ismael Sainz Maza Jiménez

Stephany Susan Vela Vellachich

## 1.2 Planificación

Debido a la reducción de funcionalidades gratuitas de la herramienta propuesta (Gantt), se han hecho capturas de pantalla sobre la planificación realizada, como así mismo incluyendo al profesor como participante del proyecto.



## 1.3 Entrega

La herramienta utilizada en este proyecto ha sido GitHub, siendo el siguiente enlace la ubicación del proyecto:

<https://github.com/Amir95sb/TG2>

# 2. Descripción de las tecnologías

## 2.1 Descripción de Appium.

### Appium:

[](http://appium.io/)

Appium es una herramienta de código abierto para automatizar aplicaciones nativas móviles en plataformas de escritorio con iOS, Android y Windows. Las aplicaciones nativas son aquellas escritas usando iOS, Android o SDK de Windows.

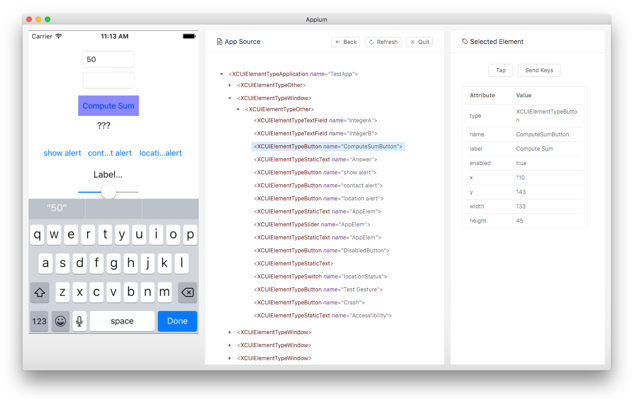
Es importante destacar que Appium es "multiplataforma": le permite escribir pruebas en múltiples plataformas (iOS, Android, Windows), utilizando la misma API. Esto permite la reutilización de códigos entre las salas de pruebas de iOS, Android y Windows.

También cabe resaltar que permite escribir los scripts en casi cualquier lenguaje de programación, por ejemplo, Java, Objective-C, , JavaScript, PHP, Ruby, Python or C#, etc)

Filosofía de Appium:

Appium fue diseñado para satisfacer las necesidades de automatización móvil de acuerdo con una filosofía descrita por los siguientes cuatro principios:

* No debería tener que volver a compilar su aplicación o modificarla de ninguna manera para automatizarla.
* No debe estar encerrado en un idioma o marco específico para escribir y ejecutar sus pruebas.
* Un marco de automatización móvil no debe reinventar la rueda.
* Un marco de automatización móvil debe ser de código abierto, tanto en espíritu como en práctica.



2.2 Descripción de Espresso.

### Espresso:

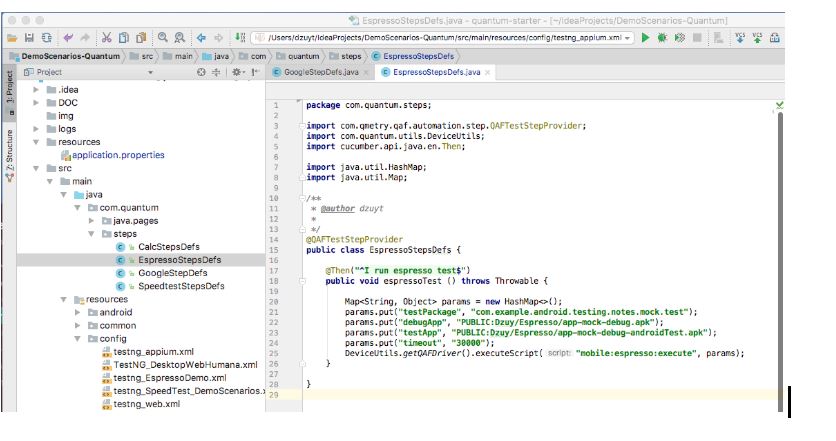
[](https://google.github.io/android-testing-support-library/docs/espresso/)

Espresso es una herramienta para empresas. Como herramienta de código abierto es muy fácil de usar y puede extenderse dentro del entorno de trabajo con facilidad.

Su principal ventaja es que nos permite la automatización automática de aplicaciones sin escribir ni una sola línea de código. Solo puede trabajar en el OS Android, pero lo hace con mucha más velocidad y eficiencia que otras tecnologías.

Espresso permite realizar pruebas tanto en dispositivos físicos como virtuales ([emuladores](https://es.wikipedia.org/wiki/Emuladores)).La principal desventaja de ejecutar las pruebas en dispositivos físicos (reales) se debe a su alto costo y tiempo que requiere por la gran variedad de dispositivos existentes en la actualidad. Una posible solución es ejecutar estas pruebas en un emulador (ambiente controlado que permite soportar múltiples versiones de [SO](https://es.wikipedia.org/wiki/SO), tamaños de pantalla y limitantes de memoria).

A modo de ejemplo, si comparamos la velocidad de ejecución de las pruebas en un Nexus 4 contra un emulador, este último requiere un 65% menos de tiempo para llevarla a cabo. De acuerdo con el equipo de trabajo detrás de Espresso, su solución logró detectar el 99% de los bugs en las aplicaciones Android, dejando unos pocos para las pruebas tradicionales (reduciendo el esfuerzo requerido para las mismas).



# 3. Criterios de comparación

Las categorías por las que hemos dividido los criterios son 3;

-A. General.

-B. Rendimiento.

-C. Utilidades.

## 3.1 Categoría A: General

### 3.1.1 Criterio A.1: Autor de la herramienta

Nombre del criterio: Autor.

Descripción: Nombre de la persona, institución o empresa creadora de la herramienta.

Tipo de valor: Texto libre.

### 3.1.2 Criterio A.2: Fecha de creación de la herramienta.

Nombre del criterio: Fecha

Descripción: La fecha en la cual se publicó por primera vez la herramienta.

Tipo de valor: Fecha

### 3.1.3 Criterio A.3: Versión actual de la herramienta.

Nombre del criterio: Versión.

Descripción: La versión de la última actualización de la herramienta.

Tipo de valor: Cadena alfanumérica.

### 3.1.4 Criterio A.4: Requisitos de prueba.

Nombre del criterio: Requisitos de prueba.

Descripción: Que tipo de casos de prueba utiliza.

Tipo de valor: Texto libre.

### 3.1.5 Criterio A.5: Herramientas similares.

Nombre del criterio: Similares.

Descripción: Herramientas similares a la herramienta en cuestión,

Tipo de valor: Texto libre.

### 3.1.6 Criterio A.6: Compañías que lo usan.

Nombre del criterio: Compañías.

Descripción: Por qué compañías es usada la herramienta

Tipo de valor: Texto libre.

### 3.1.7 Criterio A.7: Dificultad de configuración.

Nombre del criterio: Dificultad.

Descripción: Grado de dificultad de configuración de la herramienta.

Tipo de valor: Texto libre

### 3.1.8 Criterio A.8: Impulsores.

Nombre del criterio: Impulsores.

Descripción: Que impulsores tiene cada herramienta

Tipo de valor: Texto libre

## 3.2 Categoría B: Rendimiento

### 3.2.1 Criterio B.1: Lenguaje Escritura

Nombre del criterio: Lenguaje.

Descripción: Lenguaje de programación en el que está programada la herramienta.

Tipo de valor: Texto libre

### 3.2.2 Criterio B.2: Sistema operativo

Nombre del criterio: Sistema operativo.

Descripción: Sistema operativo en el cual la herramienta puede ser usada.

Tipo de valor: Texto libre.

### 3.2.3 Criterio B.3: Velocidad de las pruebas.

Nombre del criterio: Velocidad.

Descripción: Tiempo aproximado de duración de la prueba.

Tipo de valor: Numérico (segundos)

## 3.3 Categoría C: Utilidades

### 3.3.1 Criterio C.1: Alcance.

Nombre del criterio: Alcance

Descripción: El alcance de la herramienta, a que plataformas está orientada, Android, IOS, Windows…

Tipo del valor: Texto libre.

### 3.3.2 Criterio C.2: Adecuado para.

Nombre del criterio: Adecuado

Descripción: Hacia qué tipo de perfiles están adecuadas las funcionalidades de las utilidades

Tipo del valor: Texto libre.

### 3.3.3 Criterio C.3: Localizadores de objetos.

Nombre del criterio: Localizadores

Descripción: El método que usa la herramienta para localizar objetos

Tipo del valor: Texto libre.

### 3.3.4 Criterio C.4: Testing Factores externos.

Nombre del criterio: Factores Externos.

Descripción: La capacidad de la herramienta a testear valores externos de la aplicación como la rotación del dispositivo.

Tipo del valor: Booleano (si/no).

### 3.3.5 Criterio C.5: Testing Actividades externas.

Nombre del criterio: Actividades Externas.

Descripción: La capacidad de la herramienta a testear actividades externas como cámara, navegador, y llamadas.

Tipo del valor: Booleano (si/no).

### 3.3.6 Criterio C.6: Mensajes toast.

Nombre del criterio: Mensajes toast.

Descripción: La capacidad de la herramienta a testear mensajes toast, auto-completado y diálogos fuera de la app

Tipo del valor: Booleano (si/no).

### 3.3.7 Criterio C.7: Medición de test.

Nombre del criterio: Medición.

Descripción: Capacidad de analizar los test realizados y visualizar métricas sobre ellos.

Tipo del valor: Booleano (si/no).

### 3.3.8 Criterio C.8: Multilenguaje.

Nombre del criterio: Multilenguaje.

Descripción: Capacidad de la herramienta a testear aplicaciones de distintas plataformas; IOS, Android, Windows....

Tipo del valor: Booleano (si/no).

### 3.3.9 Criterio C.9: Soporte de apps web.

Nombre del criterio: Apps web.

Descripción: Soporte de la herramienta a aplicaciones que usan una página web, y la envuelven en un entorno adaptado al dispositivo.

Tipo del valor: Booleano (si/no).

# 4. Evaluación de los criterios por tecnología

## 4.1 Evaluación de los criterios para Appium

|  |  |
| --- | --- |
| CRITERIOS | EVALUACIÓN |
| Criterio A.1: Autor de la Herramienta | Dan Cuellar |
| Criterio A.2: Fecha de creación de la herramienta | 2012 |
| Criterio A.3: Versión actual de la herramienta | 1.5.0 |
| Criterio A.4: Requisitos de prueba | Prueba de caja negra\* |
| Criterio A.5: Herramientas similares | Calabash, BrowserStack, Ranorex |
| Criterio A.6: Compañías que lo usan | Apple, propiedades.com, sparks42 GmbH, N26 |
| Criterio A.7: Dificultad de configuración | Sencillo |
| Criterio A.8: Impulsores | JS Foundation |
| Criterio B.1: Lenguaje Escritura | Inglés |
| Criterio B.2: Sistema operativo | iOS, Windows, Linux, Android |
| Criterio B.3: Velocidad de las pruebas | Depende de las pruebas\*\* |
| Criterio C.1: Alcance | iOS\*\*\* |
| Criterio C.2: Adecuado | Appium es adecuado ya que es fácil de programar y tiene varios lenguajes de programación para ello, por lo que no requiere ser experto en un lenguaje concreto. Además, tiene una amplia comunidad en la que buscar información. |
| Criterio C.3: Localizadores de objetos | La sintaxis usada para encontrar objetos es Selenium, y se puede buscar por varios elementos (ID, nombre, clase, etc). El código es driver.findElement |
| Criterio C.4: Testing Factores externos | Sí |
| Criterio C.5: Testing Actividades externas | Sí |
| Criterio C.6: Mensajes toast | Sí |
| Criterio C.7: Medición | No |
| Criterio C.8: Multilenguaje | Sí |
| Criterio C.9: Soporte de apps web | Sí |

\*Caja negra: se refiere a que solo se puede probar lo expuesto externamente en la aplicación.

\*\*Appium es más lento en casos más largos y tiene algunas dificultades localizando ciertos elementos.

\*\*\* Aunque Appium ofrece una versión de escritorio para Windows y Linux está mucho más orientado a la prueba de iOS.

## 4.2 Evaluación de los criterios para Espresso

|  |  |
| --- | --- |
| CRITERIOS | EVALUACIÓN |
| Criterio A.1: Autor de la Herramienta | Google inc |
| Criterio A.2: Fecha de creación de la herramienta | 2013 |
| Criterio A.3: Versión actual de la herramienta | 3.0.2 |
| Criterio A.4: Requisitos de prueba | Prueba blanca |
| Criterio A.5: Herramientas similares | UIAutomator, Robotium |
| Criterio A.6: Compañías que lo usan | eBay, Google, Android |
| Criterio A.7: Dificultad de configuración | Sencillo |
| Criterio A.8: Impulsores | Google |
| Criterio B.1: Lenguaje Escritura | Inglés |
| Criterio B.2: Sistema operativo | Android, iOS |
| Criterio B.3: Velocidad de las pruebas | Depende de las pruebas\*\* |
| Criterio C.1: Alcance | Android\*\*\* |
| Criterio C.2: Adecuado | Espresso es adecuado para los perfiles que trabajen especialmente con Android. Además, es fácilmente manejable por lo que se puede aprender a manejar fácilmente desde una base mínima. |
| Criterio C.3: Localizadores de objetos | Espresso encuentra la vista deseada y la reproduce en pantalla. El usuario interactúa con ella haciendo click y verificando el contenido. Se usa ViewMatchers, ViewActions, etc. También se puede usar RecyclerView, que permite ver contenido fuera de la pantalla. |
| Criterio C.4: Testing Factores externos | No |
| Criterio C.5: Testing Actividades externas | Sí |
| Criterio C.6: Mensajes toast | Sí |
| Criterio C.7: Medición de test | Sí\*\*\*\* |
| Criterio C.8: Multilenguaje | Sí |
| Criterio C.9: Soporte de apps web | Sí |

\*Prueba blanca: se refiere a que Espresso puede acceder al código interno de cada elemento.

\*\*En pruebas largas, Espresso es más rápido.

\*\*\* Aunque Espresso soporta bien tanto pruebas en Android como iOS, está más orientado a pruebas Android.

\*\*\*\* Con Test Suite.

# 5. Comparación de las tecnologías

**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. |

# 6. Recomendaciones

## 6.1 Situación 1

### 6.1.1 Descripción de la situación

Una posible situación podría ser el de una empresa que realiza automatización de pruebas móviles para todo tipo de dispositivos y apps. Ya que estos dispositivos pueden tener distintos OS y estar escritos en diferentes lenguajes, la empresa busca una herramienta que le permita esta versatilidad a la hora de trabajar.

### 6.1.2 Recomendación de tecnología a utilizar

Appium permite realizar pruebas de aplicaciones en iOS, Android y Windows además de poder programar en un gran número de lenguajes, los más utilizados. Esto permitiría a la empresa automatizar pruebas en la gran mayoría de dispositivos por la parte del OS y también por los distintos lenguajes utilizados en esas aplicaciones al soportar varios lenguajes.

En el caso de Espresso estas dos ventajas no existen ya que solo permite aplicaciones para OS Android y no permite tantos lenguajes de programación

La tecnología usada en este caso sería **Appium.**

|  |  |  |
| --- | --- | --- |
| Criterios relevantes para la decisión | Ventajas Appium | Ventajas Espresso |
| Multiplataforma | Si | No |
| Variedad de lenguajes de programación | Si | No |

## 6.2 Situación 2

### 6.2.1 Descripción de la situación

Una empresa quiere empezar un proyecto de automatización de pruebas de aplicaciones móviles en sistemas operativos Android y para ello confía a un jefe de proyecto la búsqueda de una herramienta que se adapte a sus necesidades.

Necesita una herramienta fácil de aprender, que se adapte al entorno de trabajo fácilmente para poder empezar el proyecto lo antes posible y que además cumpla con todos los requisitos necesarios para probar las aplicaciones a fondo.

### 6.2.2 Recomendación de tecnología a utilizar

La tecnología utilizada en este caso sería **Espresso.**

Appium puede probar de forma correcta las aplicaciones Android, pero es más complicada de utilizar y no se adapta de forma tan fácil al entorno de trabajo.

Espresso en el caso contrario, además de ser la herramienta más rápida y eficiente en cuanto a las pruebas en OS Android, aunque solo pueda probar sobre este OS, se adapta rápidamente al área de trabajo con su API.

|  |  |  |
| --- | --- | --- |
| Criterios relevantes para la decisión | Ventajas Appium | Ventajas Espresso |
| Adaptación al entorno de trabajo | No | Si |
| Requisitos para las pruebas | Si | Si |
| Facilidad de uso | No | Si |