Contenido

[1. Autores del trabajo, planificación y entrega 2](#_Toc448254544)

[1.1 Autores 2](#_Toc448254545)

[1.2 Planificación 2](#_Toc448254546)

[1.3 Entrega 2](#_Toc448254547)

[2. Requisitos del prototipo a implementar 3](#_Toc448254548)

[2.1 Requisitos funcionales 3](#_Toc448254549)

[2.2 Otros requisitos 3](#_Toc448254550)

[3. Criterios de comparación en la implementación 4](#_Toc448254551)

[3.1 Criterio 1: Nombre del criterio 4](#_Toc448254552)

[3.2 Criterio 2: Nombre del criterio 4](#_Toc448254553)

[3.N Criterio N: Nombre del criterio 4](#_Toc448254554)

[4. Proyecto de implementación de un prototipo del sistema utilizando la tecnología A 5](#_Toc448254555)

[4.1 Documentación de diseño 5](#_Toc448254556)

[4.2 Documentación de construcción 5](#_Toc448254557)

[4.3 Documentación de pruebas 5](#_Toc448254558)

[4.4 Documentación de instalación 5](#_Toc448254559)

[4.5 Manual de usuario 5](#_Toc448254560)

[5. Proyecto de implementación de un prototipo del sistema utilizando la tecnología B 6](#_Toc448254561)

[5.1 Documentación de diseño 6](#_Toc448254562)

[5.2 Documentación de construcción 6](#_Toc448254563)

[5.3 Documentación de pruebas 6](#_Toc448254564)

[5.4 Documentación de instalación 6](#_Toc448254565)

[5.5 Manual de usuario 6](#_Toc448254566)

[6. Comparación de las dos implementaciones 7](#_Toc448254567)

[6.1 Evaluación de los criterios en la implementación usando la tecnología A 7](#_Toc448254568)

[6.2 Evaluación de los criterios en la implementación usando la tecnología B 7](#_Toc448254569)

[7. Comparación de la implementación de las tecnologías 8](#_Toc448254570)

[8. Conclusiones 10](#_Toc448254571)

# 1. Autores del trabajo, planificación y entrega

## 1.1 Autores

Grupo M1

Amir Sholkami Berube - Amir95sb(coordinador)

Javier Ballesteros García - JavierBallesteros5

Ismael Sainz Maza Jiménez - Ismaeel19

Stephany Susan Vela Vellachich - stephanyvela

## 1.2 Planificación



Ya que este trabajo cuenta 3 Puntos de la nota final de la asignatura, cada uno ha sido asignado trabajos equivalentes o mayores a 45 horas totales.

https://app.teamweek.com/#pg/1g82G63vww4N07cj0sN0WsaZLnDfTiuz

## 1.3 Entrega

https://github.com/Amir95sb/TG3

# 2. Requisitos del prototipo a implementar

## 2.1 Requisitos funcionales

| **REQ.** | **DESCRIPCIÓN** |
| --- | --- |
| RF01 | Testing de Botones |
| RF02 | Testing de introducción de texto |
| RF03 | Guardar Procedimientos Test |

## 2.2 Otros requisitos

| **REQ.** | **DESCRIPCIÓN** |
| --- | --- |
| R01 | Testing de interfaz de usuario |
| R02 | Velocidad de pruebas >1 minuto |
|  |  |

# 3. Criterios de comparación en la implementación

***3.1 Interfaz de Usuario***

Nombre del criterio: Interfaz de Usuario.

Descripción: En este criterio comparamos la facilidad de la interfaz de usuario con respecto a otras aplicaciones. Más tarde veremos esa comparación de nuestros dos software frente a frente

Tipo de valor: Texto.

***3.2 Tiempo de Aprendizaje***

Nombre del criterio: Tiempo de Aprendizaje.

Descripción: En este criterio valoramos el tiempo que hemos tardado desde que nosotros arrancamos desde el momento que arrancamos nuestro software y podemos empezar a realizar ese proyecto. También introducimos el tiempo que le hemos dedicado a la adquisición de conocimientos de aprendizaje del software.

Tipo de valor: Numérico (horas).

***3.3 Tiempo de Configuración***

Nombre del criterio: Tiempo de Configuración.

Descripción: Incluimos el tiempo que hemos empleado en la configuración de nuestro software para utilizarlo. Aquí debemos incluir la instalación.

Tipo de valor: Numérico (horas).

***3.4 SS.OO Útiles***

Nombre del criterio: SS.OO Útiles.

Descripción: Diremos los SS.OO que hemos usado en la instalación y realización del proyecto. En este punto introducimos también la dificultad que hemos tenido.

Tipo de valor: Texto.

***3.5 Horas Empleadas en el Desarrollo***

Nombre del criterio: Horas Empleadas en el Desarrollo.

Descripción: Decimos el número de horas que se hemos tardado desde que empezamos en su momento el desarrollo hasta tener un proyecto viable al menos mínimamente.

Tipo de valor: Numérico (horas).

***3.6 Herramientas Utilizadas en el Desarrollo***

Nombre del criterio: Herramientas Utilizadas en el Desarrollo.

Descripción: Vemos las herramientas que hemos usado pues nos ayudaban a realizar ese mínimo de nuestro proyecto.

Tipo de valor: Texto.

***3.7 Costes del software***

Nombre del criterio: Costes del software

Descripción: Decimos cuanto a costado el software con todas las herramientas que también usamos para hacerlo funcionar.

Tipo de valor: Valor económico.

***3.8 Requisitos en el ordenador***

Nombre del criterio: Requisitos en el ordenador

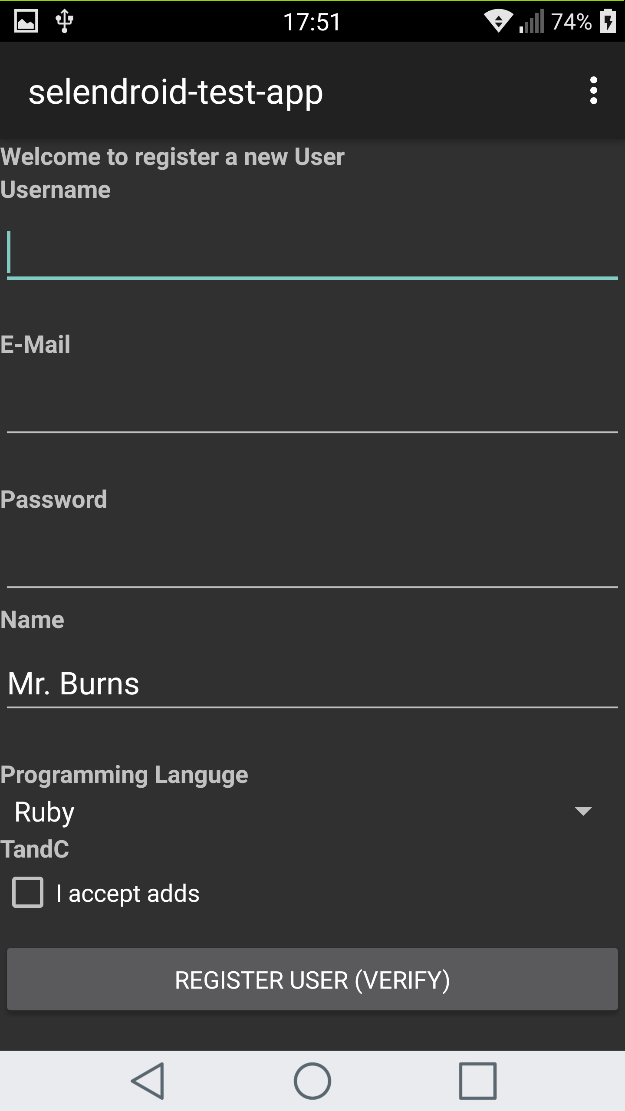
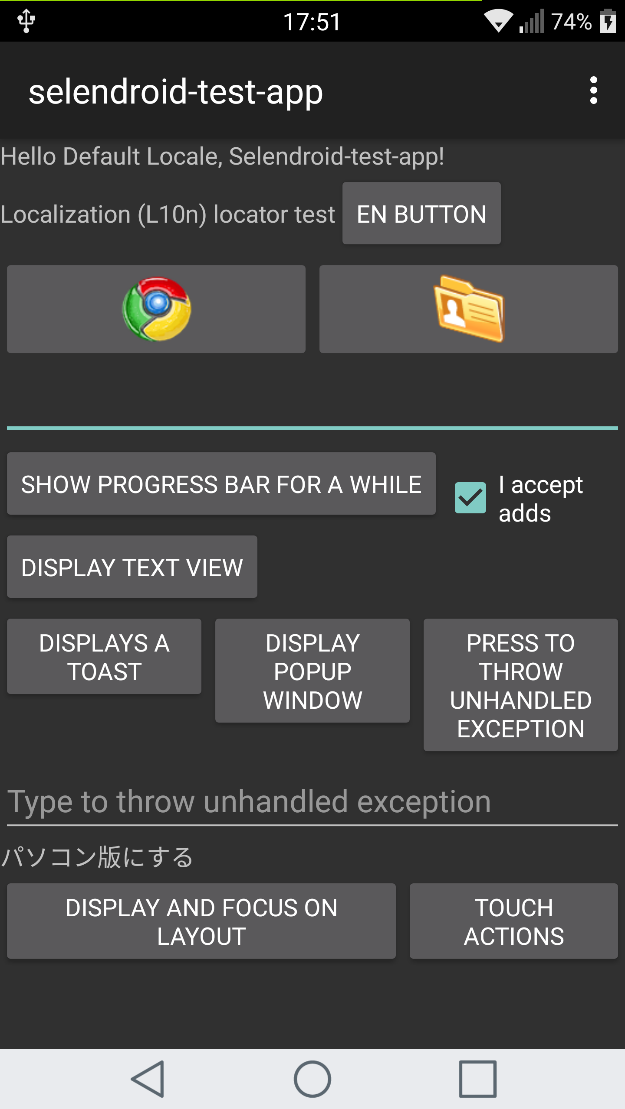
Descripción: Decimos cuales son los requisitos para nuestro ordenador o dispositivo para que nuestro software pueda funcionar a pleno rendimiento.

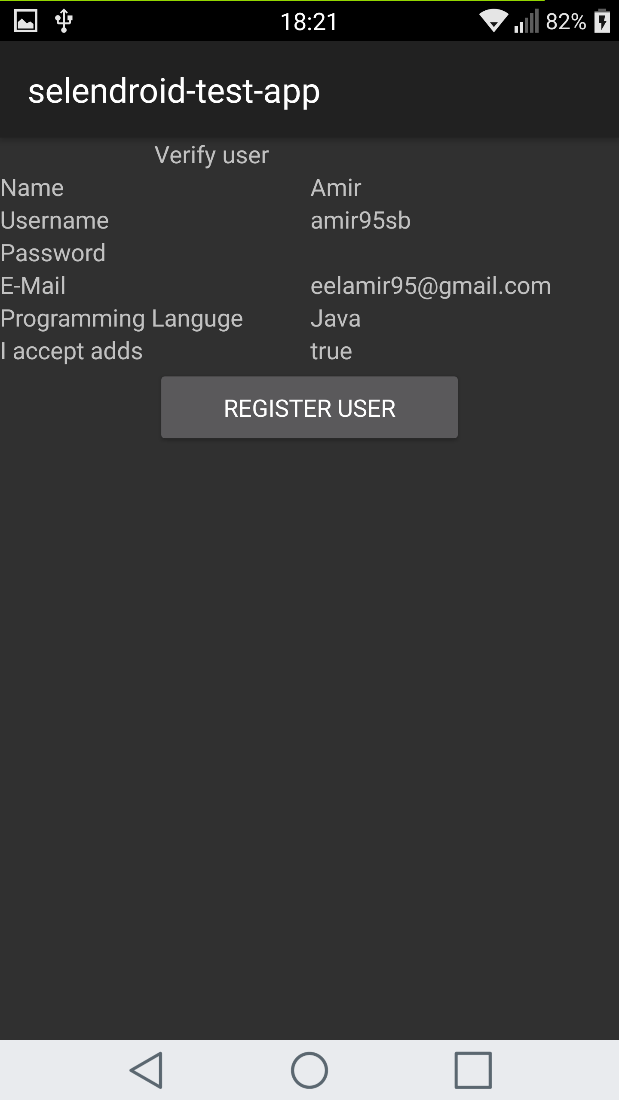
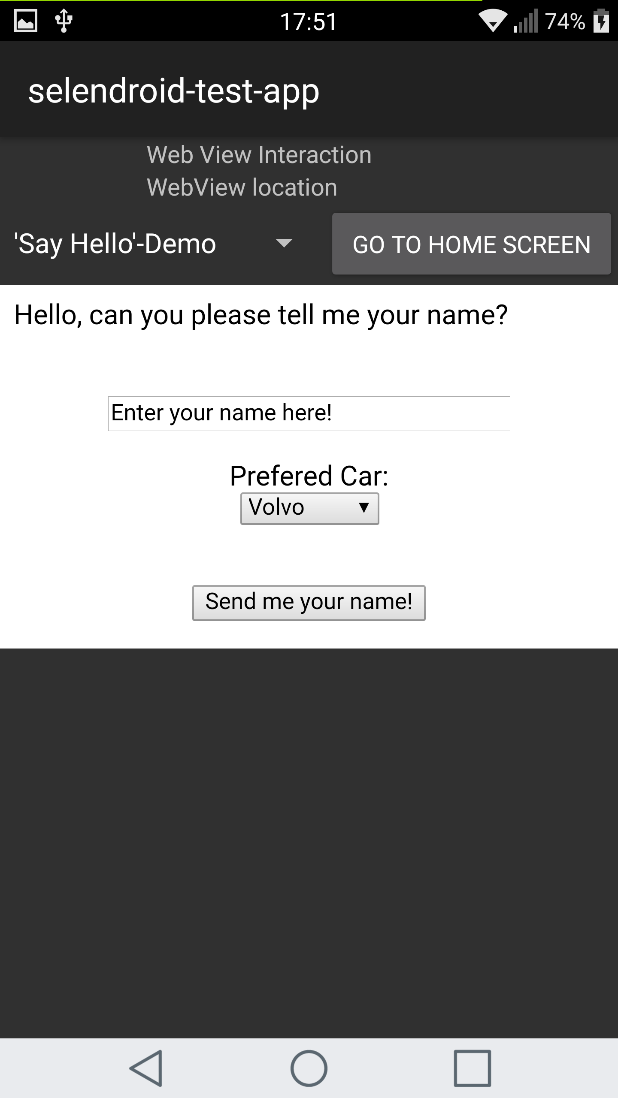
Tipo de valor: Texto.

# 4. Proyecto de implementación de un prototipo del sistema utilizando la tecnología Appium

## 4.1 Documentación de diseño

El prototipo usado en este documento es una aplicación sencilla pero completa, llamada Selendroid test app, la cual ha sido diseñada para ejecutar y ver los resultados a pruebas desde cualquier herramienta de testing.





La interfaz del usuario es muy básica, ya que solo queremos probar la funcionalidad de la app, y las animaciones han sido desactivadas para evitar errores.

Las miniaturas usadas se pueden encontrar en la carpeta del código fuente llamada:

\selendroid-master\selendroid-test-app\res

## 4.2 Documentación de construcción

La construcción de la app es básica, como mostramos en el siguiente código fuente:

1. **public** **class** UserRegistrationTest **extends** BaseAndroidTest {
2. @Test
3. **public** void assertUserAccountCanRegistered() throws Exception {
4. UserDO user =
5. **new** UserDO("u**$erNAme**", "me@myserver.com", "mySecret", "Selendroid User",
6. PreferedProgrammingLanguage.JAVA\_SCRIPT);
7. registerUser(user);
8. verifyUser(user);
9. }
11. **private** void registerUser(UserDO user) throws Exception {
12. WebElement button = driver().findElement(By.id("startUserRegistration"));
14. button.click();
16. WebElement username = driver().findElement(By.id("inputUsername"));
18. username.sendKeys(user.getUsername());
19. WebElement nameInput = driver().findElement(By.id("inputName"));
20. [Assert](http://www.php.net/assert).assertEquals(nameInput.[getText](http://www.php.net/gettext)(), "Mr. Burns");
21. nameInput.clear();
22. nameInput.sendKeys(user.getName());
24. driver().findElement(By.id("inputEmail")).sendKeys(user.getEmail());
25. driver().findElement(By.id("inputPassword")).sendKeys(user.getPassword()); //se introduce la contraseña del usuario
26. try {
27. nameInput.submit();
28. [Assert](http://www.php.net/assert).fail("submit is not supported by SelendroidNativeDriver");
29. } catch (WebDriverException e) {
30. // expected behavior
31. }

Este codigo es el del formulario, en el cual se rellenan los datos del usuario como el ID, email, o contraseña.

1. <resources>
3. <string name="hello">Hello Default Locale, Selendroid-test-app!</string>
4. <string name="button">EN Button</string>
5. <string name="app\_name">Selendroid-test-app</string>
6. <string name="search\_hint">Search for users</string>
7. <string name="welcomeRegisterUser">Welcome to register a **new** User</string>
8. <string name="verifyNewUser">Register User (verify)</string>
9. <string name="label\_password">Password</string>
10. <string name="label\_username">Username</string>
11. <string name="label\_email">E-Mail</string>
12. <string name="label\_name">Name</string>
13. <string name="label\_preferedProgrammingLanguage">Programming Languge</string>
14. <string name="prompt\_preferedProgrammingLanguage">Programming Languge</string>
15. <string name="prompt\_webdriverHtmlTestFile">Webdriver Test File</string>
16. <string name="label\_acceptAdds">I accept adds</string>
17. <string name="label\_conditions">TandC</string>
19. <string-[array](http://www.php.net/array) name="arrays\_preferedProgrammingLanguage">
20. <item>Ruby</item>
21. <item>PHP</item>
22. <item>Scala</item>
23. <item>Python</item>
24. <item>Javascript</item>
25. <item>Java</item>
26. <item>C++</item>
27. <item>C*#</item>*
28. </string-array>
30. </resources>

En este código están incluidos todos los “strings” usados por la app.

Estaran subidos tanto el código fuente del proyecto como la .APK en el repositorio github.

## 4.3 Documentación de pruebas

Casos de prueba establecidos y resultados de las pruebas y acciones de corrección. No es creíble que no hayan aparecido errores en los caso de prueba.

## 4.4 Documentación de instalación

Para que Appium funcione correctamente en Windows debemos tener instalado NodeJs, Java Development Kit (JDK), Paquetes de Android SDK y ADB, además de los drivers de los dispositivos que usaremos, ya que Appium los necesita para conectarse y probar la aplicación. En caso de no poseer un dispositivo se requiere un emulador.

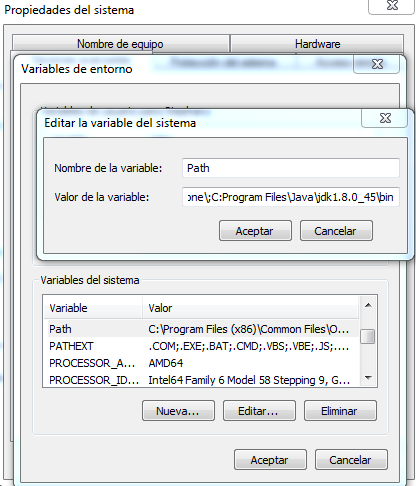
Para saber la versión de node.js (se recomienda tener al menos la v4.4.7):

node -v

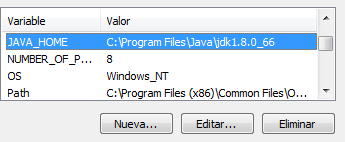
Para la versión de java (se recomienda tener a partir de la v1.8.0):

java –version

En caso de que desde la cmd no se permita ver la versión de java, deberemos indicar el camino. Para ello iremos a Panel de Control > Sistema y Seguridad > Sistema y entraremos a la pestaña de Configuración avanzada del sistema.

Desde opciones avanzadas daremos click a Variables de entorno y añadiremos a Path el camino al bin de nuestro software Java, pero sin quitar los demás ya que son necesarios.

Además añadiremos la variable JAVA\_HOME:



## 4.5 Manual de usuario

Descripción suficiente para que una persona que no ha participado en el proyecto pueda utilizar toda la funcionalidad que ofrece el prototipo. Que debe coincidir con los requisitos funcionales incluidos en el apartado 2.

# 5. Proyecto de implementación de un prototipo del sistema utilizando la tecnología Espresso

## 5.1 Documentación de diseño

El prototipo usado es exactamente el mismo que ha sido usado en la tecnología A (Appium)



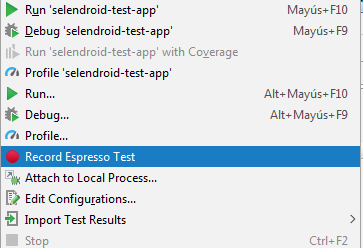
## 5.2 Documentación de construcción

El código fuente, ya está explicado en el apartado 4.2, pero cabe decir que ha sido modificado con Android Studio para su correcto funcionamiento, ya que algunos módulos estaban desactualizados, y al actualizarlos, salieron a la superficie problemas debido a funciones no admitidas en los módulos actualizados nuevos.

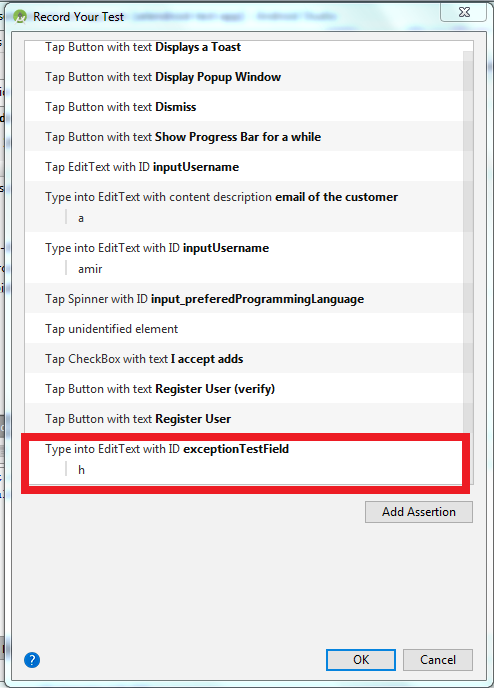
## 5.3 Documentación de pruebas

Se han realizado dos casos de prueba, un caso de prueba fallido:

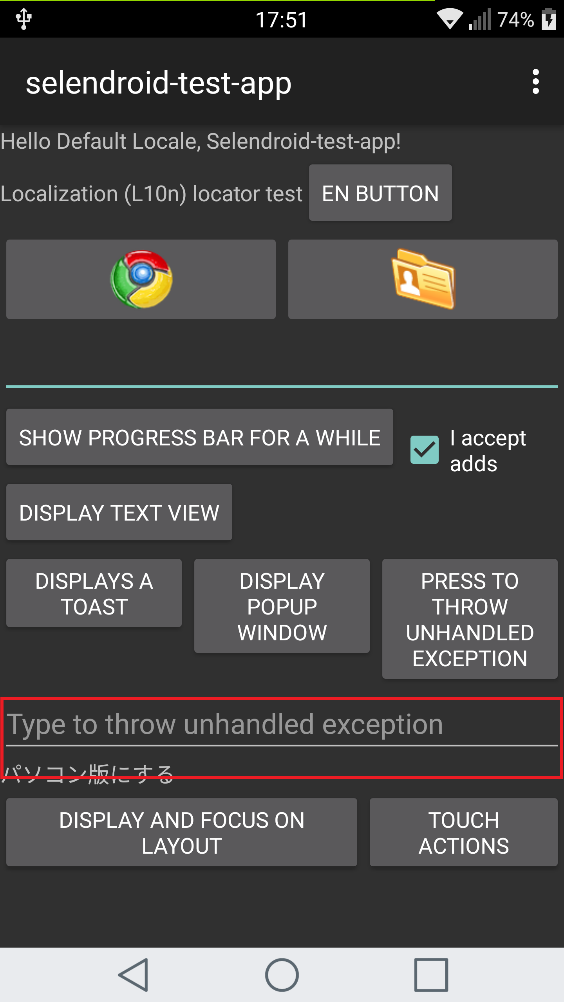
Para demostrar el correcto funcionamiento de Espresso, hemos realizado una prueba fallida, no porque la aplicación este mal programada, sino porque se incluye una función que lanza un error, para demostrar que es lo que sucede ante estos casos.

Para crear la prueba, se ha utilizado una herramienta llamada incluida en Android Studio: 

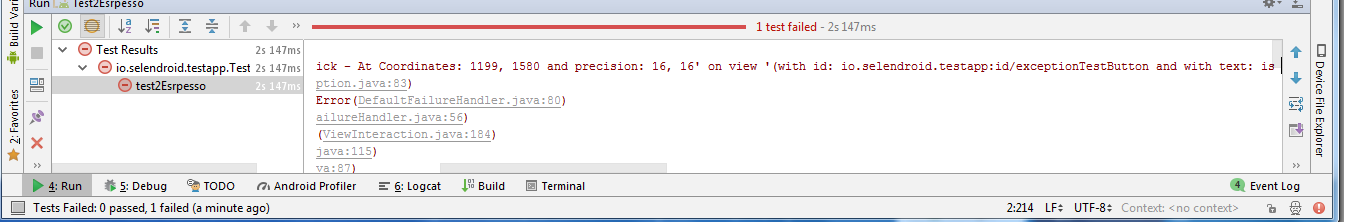
Con la cual la aplicación se inicia en el dispositivo seleccionado, y graba la serie de acciones que nosotros realizamos en la aplicación desde el dispositivo, transformándolo directamente en código de test:



Como se aprecia en la imagen, la ultima acción grabada es en el campo llamado “exceptionTestField” el cual lanza una excepción al ser escrito cualquier carácter en el campo.



Al realizar la prueba, el resultado es fallido, indicando que el fallo esta en la línea de código la cual lanza la excepción. Los resultados están incluidos en el zip llamado PrototipoTecnologiaB\_final.rar



Para realizar una prueba Valida, es decir, sin fallos, hemos realizado los mismos pasos, excepto sin apretar el botón llamado “Press to throw unhandled exception” ni escribir en el campo “Type to throw unhandled exception” dando lugar a esta secuencia de acciones:

-(R.id.***showToastButton***), *withText*(**"Displays a Toast"**)

*-allOf*(*withId*(R.id.***topLevelElementTest***), *withText*(**"Display and focus on layout"**)

*-allOf*(*withId*(R.id.***visibleButtonTest***), *withText*(**"Display text view"**)

*-allOf*(*withId*(R.id.***waitingButtonTest***), *withText*(**"Show Progress Bar for a while"**)

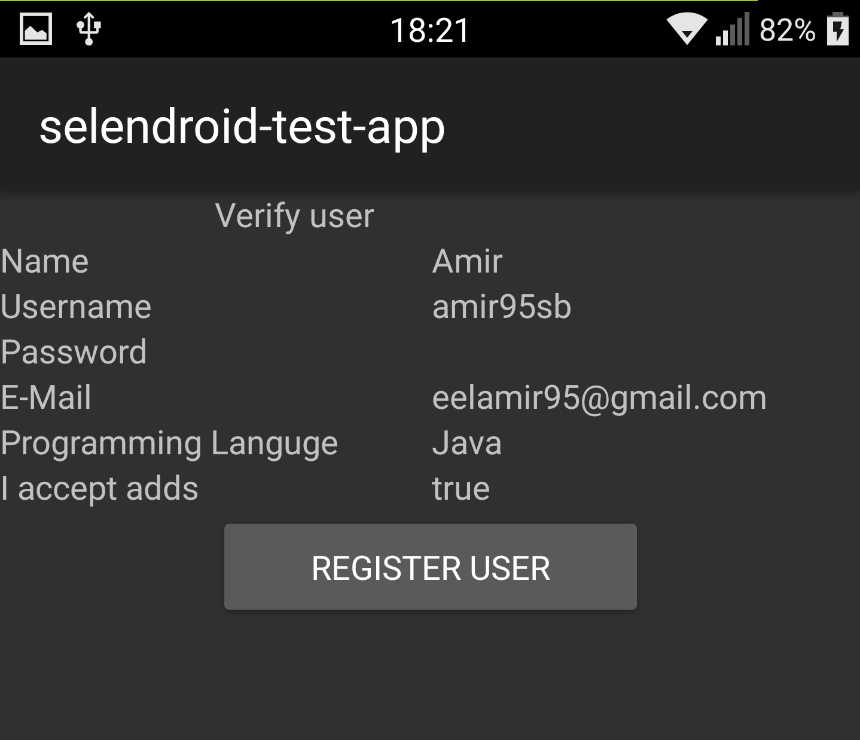
-editText.perform(*scrollTo*(), *replaceText*(**"amir"**)

-editText2.perform(*scrollTo*(), *replaceText*(**"amir95sb@gmail.com"**)

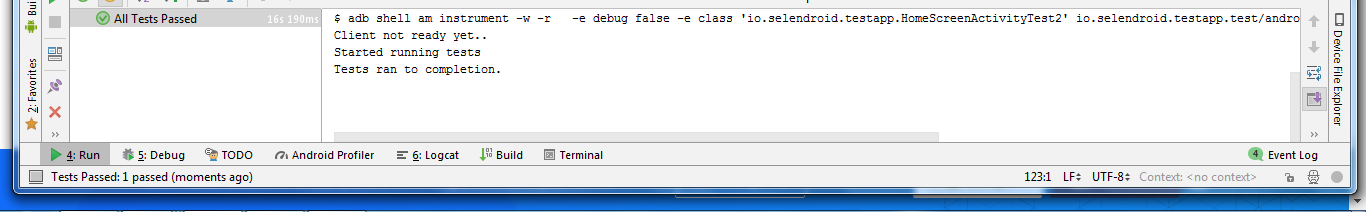
-*allOf*(*withId*(R.id.***btnRegisterUser***), *withText*(**"Register User (verify)"**)

-*allOf*(*withId*(R.id.***buttonRegisterUser***), *withText*(**"Register User"**)

Dando lugar a la siguiente pantalla:



A la hora de realizar el test, el resultado es positivo:



## 5.4 Documentación de instalación

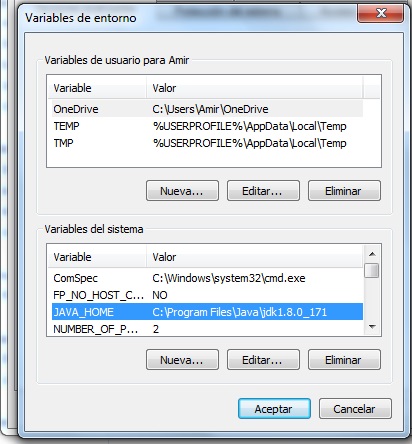
La instalación de Espresso es relativamente sencilla, ya que esta integrada con la aplicación Android Studio, y el proceso es totalmente automatizado.

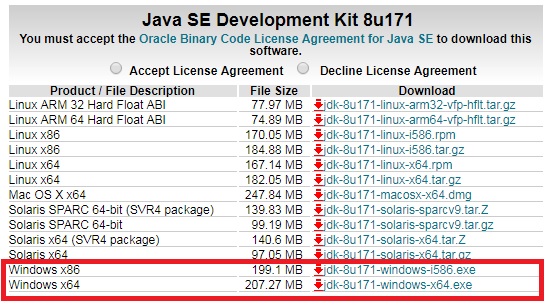
Las herramientas a descargar son las siguientes:

-Android Studio: <https://developer.android.com/studio/?hl=es-419>

-Java JDK: <http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>

Primero sería recomendable descargar las herramientas Java JDK, ya que luego Android Studio las localizara automáticamente. De lo contrario habrá que incluir en las variables del entorno una variable llamada JAVA\_HOME con la dirección de la carpeta de instalación:

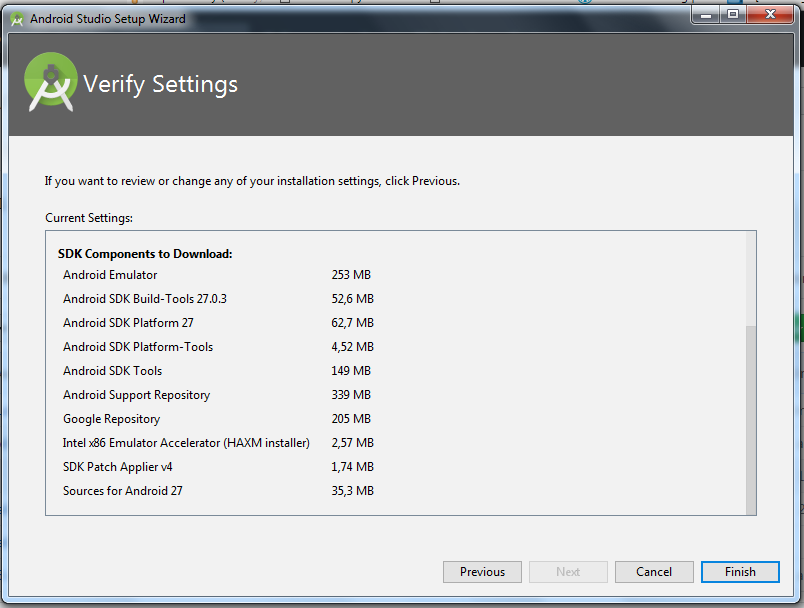




En nuestro caso seleccionamos las herramientas para Windows x64, porque es el sistema que usamos.

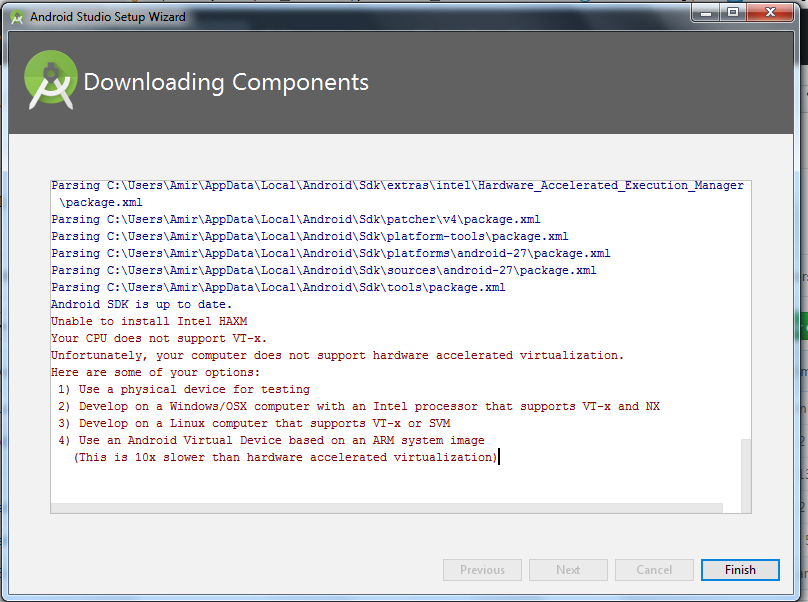
Se instala siguiendo las opciones por defecto que viene el instalador, sin tener que elegir ninguna opción.

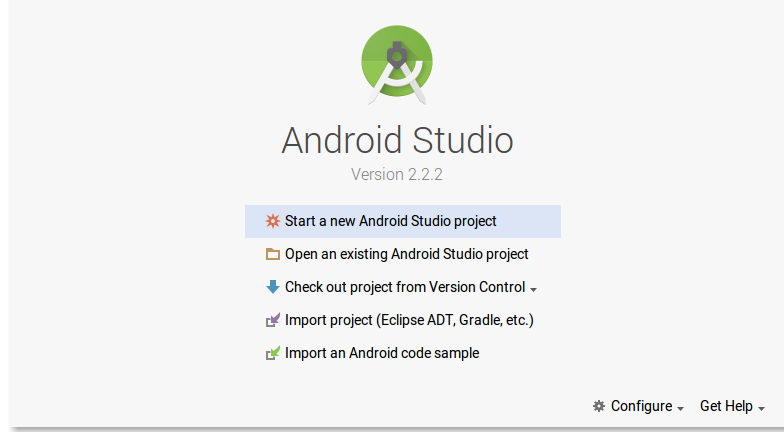
Seguimos con Android Studio, el cual también tenemos que seguir las opciones por defecto, incluidas las Android SDK.



Estas son las opciones por defecto que se deberían instalar.

En el ordenador en concreto que hemos usado, nuestra CPU no soporta dispositivos emulados, aunque no importa porque en nuestro caso hemos usado dispositivos reales.



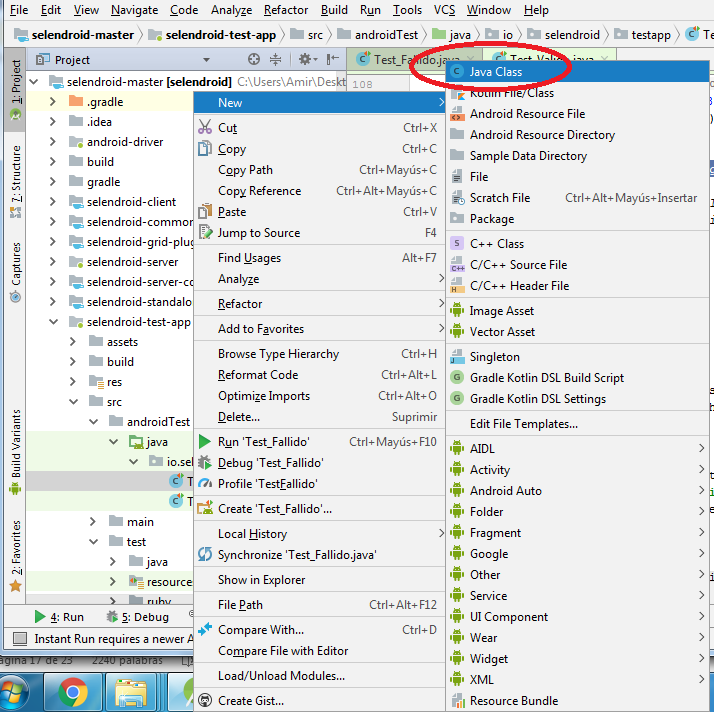


Y ya estaría todo preparado para crear tu propia aplicación, o empezar a testear tu aplicación sobre el código fuente. Se pueden usar archivos APK para crear tus test en Espresso, pero no se podrán ejecutar a no ser que tengas el código fuente.

## 5.5 Manual de usuario

Para comenzar necesitas abrir el proyecto en Android Studio en el cual este el código fuente de la aplicación en cuestión:

Una vez añadido, después de haber arreglado los errores correspondientes, podemos empezar a crear una clase Java, en la cual se escribirá el procedimiento para el test:



Para poder ejecutar pruebas en espresso, es necesario añadir las siguientes líneas al código en el archivo gradle.build:

dependencies {

compile fileTree(dir: 'libs', include: ['\*.jar'])

testCompile 'junit:junit:4.12'

*// Android runner and rules support*

androidTestCompile 'com.android.support.test:runner:0.5'

androidTestCompile 'com.android.support.test:rules:0.5'

*// Espresso support*

androidTestCompile('com.android.support.test.espresso:espresso-core:2.2.2', {

exclude group: 'com.android.support', module: 'support-annotations'

})

*// add this for intent mocking support*

androidTestCompile 'com.android.support.test.espresso:espresso-intents:2.2.2'

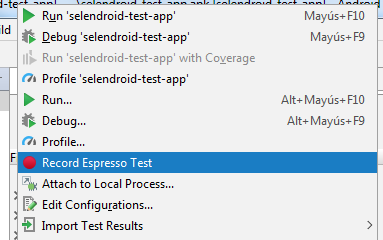
*// add this for webview testing support*

androidTestCompile 'com.android.support.test.espresso:espresso-web:2.2.2'

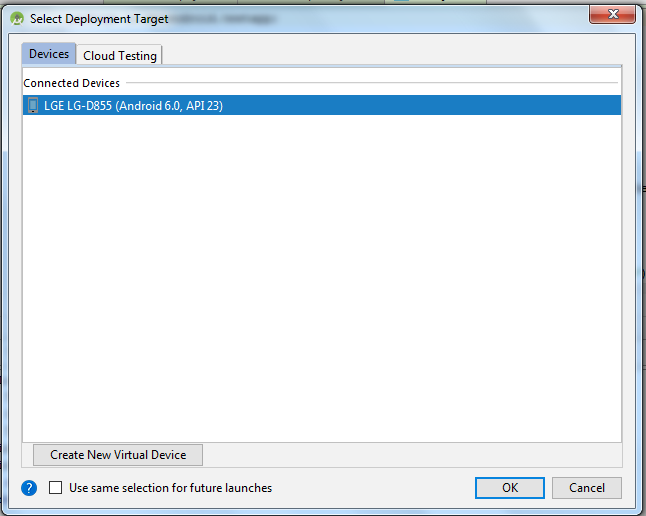
}

O usaremos la herramienta especial para grabar un test sobre el dispositivo y crear el código automáticamente:

En la pestaña RUN seleccionamos: “record Espresso test”:



Luego seleccionamos el modulo de la aplicación a grabar, en caso de haber algún fallo o incoherencia, te pedirá que lo arregles. En caso satisfactorio, se abirá un cuadro, en el cual se activara el ADB “android debug bridge” y seleccionaras el dispositivo en el cual se desean hacer las grabaciones de las pruebas:



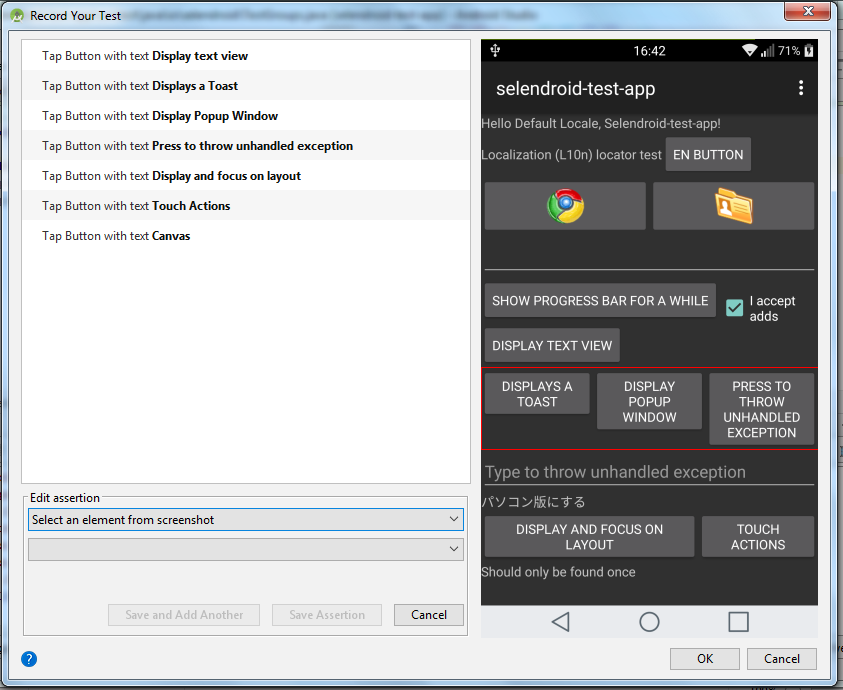
En nuestro caso usaremos un LG G3 d855, con la versión de Android 6.0.

Antes de este paso, deberas activar la Depuracion usb en tu dispositivo Android, que se encuentra en la sección de herramientas de desarrollador, en los ajustes del móvil. En caso de no tenerlos activados, se deberá hacer “click” en la sección del numero de construcción “build number” repetidas veces hasta que se active.

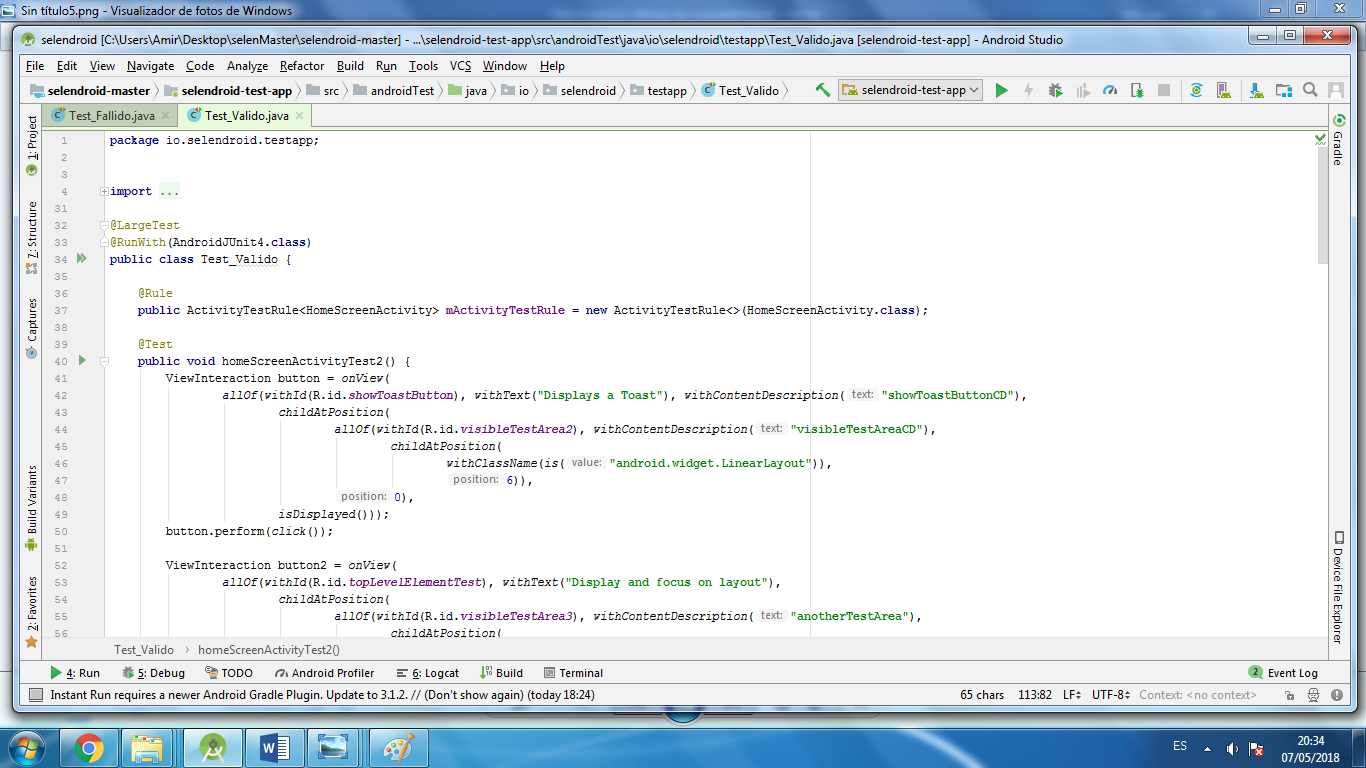
Tambien deberas instalar los drivers de tu dispositivo Android en el ordenador.

Android studio compilara automáticamente la aplicación, generará un archivo .apk y lo instalara automáticamente en tu dispositivo. En cuanto este instalado se abrirá directamente la aplicación, y esperara acciones para ser grabadas:

Tambien se pueden grabar acciones desde el ordenador, es decir, al hacer click en el botón “add assertions” se mostrará por pantalla una captura de pantalla de la aplicación con todos los botones disponibles, y se grabara cualquier acción realizada, o seleccionada en el menú:



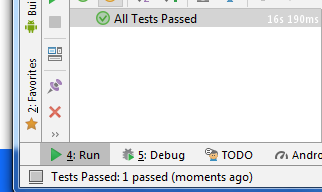
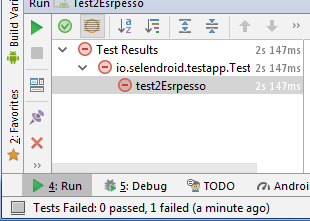
Una vez satisfecho con las acciones, se finaliza con el botón “ok”, se introduce el nombre del archivo test a crear, y se generara automáticamente el código en un archivo Java.



Android Studio añadirá los import necesarios automáticamente, y estará listo para pasar a la prueba.

Para empezar la prueba, se hace click con el botón derecho sobre el archivo de prueba generado, y sobre “Run”. Se pedirá otra vez el dispositivo al que se desea hacer la prueba, y se repetirá el proceso de compilar e instalar la app en el dispositivo móvil.

En este momento, se ejecutara la aplicación y Android studio realizara todas las acciones a ejecutar, y finalizará con un resultado positivo, o fallido:



En el log se indicaran los fallos ocurridos, y el tiempo de ejecución, además de las líneas de código en los cuales han sucedido fallos.

# 6. Comparación de las dos implementaciones

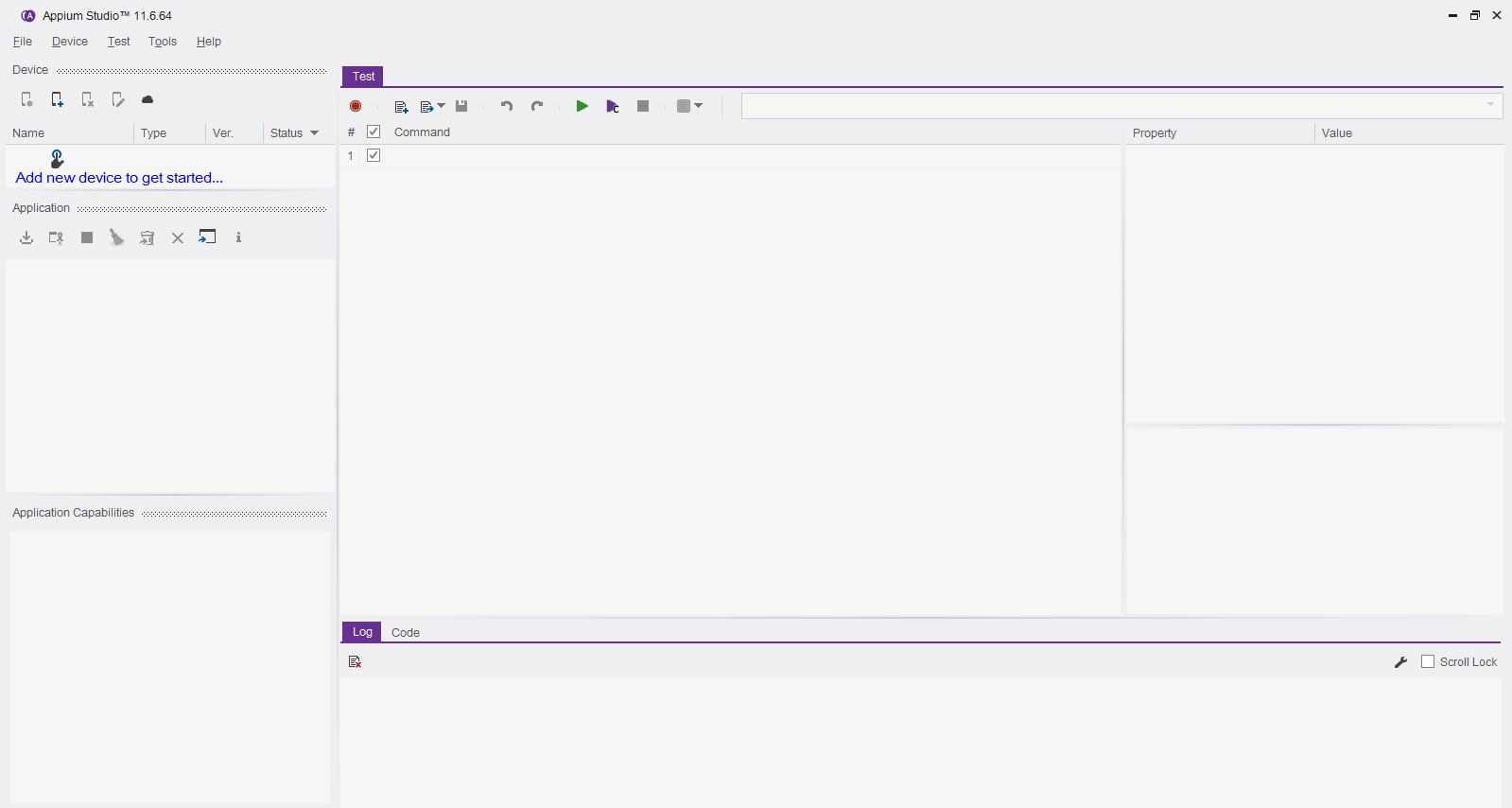
Se trata de dar valores a los criterios de comparación definidos en el apartado 3 sobre la implementación de cada uno de los prototipos.

## 6.1 Evaluación de los criterios en la implementación usando la tecnología A

Debe incluir al menos una tabla con la siguiente estructura.

| **CRITERIO** | **EVALUACIÓN** |
| --- | --- |
| Interfaz de Usuario | Al abrir la aplicación se nos presenta una interfaz sencilla y funcional. Podemos añadir nuevos dispositivos, abrir proyectos, etc. La interfaz es clara.\* |
| Tiempo de aprendizaje | 3 horas (apox) |
| Tiempo de configuración | 5 horas (aprox)\*\* |
| Sistema Operativos útiles | Se ha usado desde Windows para probar una apk de Android. |
| Horas empleadas en el desarrollo |  |
| Herramientas utilizadas en el desarrollo |  |
| Costes de software |  |
| Requisitos en el ordenador | Se requieren diversos software complementarios y drivers, además de un dispositivo o emulador con el sistema operativo deseado.\*\* |

\*Pantalla de la interfaz de usuario:

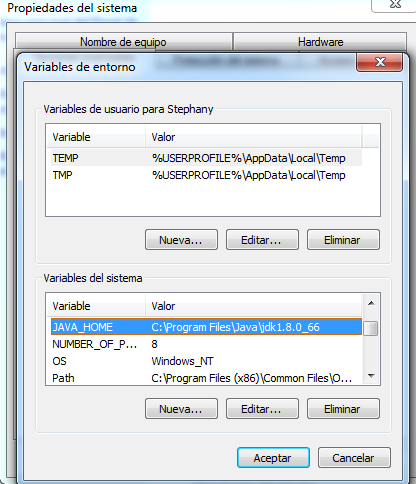


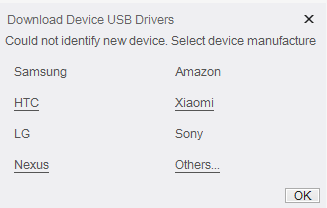
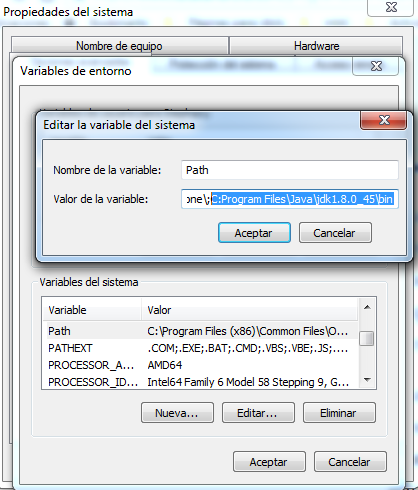
\*\* Al instalar Appium y querer hacer una prueba sobre Android debemos instalar varios complementos, como el Java Development Kit (JDK), Android SDK/ADB y los paquetes deseados, NodeJs, Microsoft .net Framework antes de ponernos a trabajar. Estas instalaciones pueden llevar su tiempo, ya que son bastantes. Por ello, la habilidad a la hora de configurarlo es elevada, ya que además se requiere la preparación de los elementos como se muestra en las próximas capturas de pantalla.

Además, para poder usar el software se nos pide que instalemos unos drivers, es decir, necesitamos tener un dispositivo o un emulador para poder probar el código.

Si no se tiene alguno de los complementos o drivers, es seguro o muy probable que no se pueda realizar la prueba.

Se han tenido diversos problemas con la instalación de Appium por lo que su configuración ha durado más de lo esperado.





## 6.2Evaluación de los criterios en la implementación usando la tecnología B

# 7. Comparación de la implementación de las tecnologías

Debe incluir al menos una tabla resumen, en sección de página horizontal, cruzando los criterios y los valores de cada tecnología. Con una columna de comentarios sobre la comparación

| **CRITERIOS** | **TECNOLOGÍA A** | **TECNOLOGÍA B** | **COMENTARIOS** |
| --- | --- | --- | --- |
| 1 |  |  |  |
| 2 |  |  |  |
| … |  |  |  |
| N |  |  |  |

# 8. Conclusiones

A partir de la información incluida en el apartado 7 y de la experiencia al realizar el trabajo, el grupo debe estar en condiciones de manifestar su opinión sobre la implementación del sistema utilizando ambas tecnologías, y debe plasmarla en este apartado, indicando las ventajas e inconvenientes más relevantes de utilizar una u otra tecnología para implementar el sistema.

---------------------------

(Hay que cumplir la estructura básica indicada de secciones. Pero si se desea se pueden añadir otras secciones como anexos. Por ejemplo, alguna encuesta de opinión realizada sobre las tecnologías, etc.)