PARCIAL 1 PRUEBAS AUTOMÁTICAS 201820

JESUS GABRIEL ANGEL 201413561

REPO: https://github.com/jgangel/parcial1

# Estrategia de pruebas

## Descripción de la app bajo pruebas (AUT)

En este trabajo se desarrollarán y ejecutarán pruebas automatizadas sobre las aplicaciones Los Estudiantes y AntennaPod.

Los Estudiantes es una aplicación web, desplegada en la url [www.losestudiantes.co](http://www.losestudiantes.co), que tiene como objetivo que los estudiantes de las distintas universidades puedan calificar a los profesores de las materias para futuras referencias y los profesores tengan retroalimentación sobre sus clases.

AntennaPod por su parte es una aplicación que permite la gestión y reproducción de podcasts, tanto gratuitos como pagos. Debido a los recursos provistos para el presupuesto del proceso de pruebas, estas sólo estarán enfocadas en la gestión y reproducción de podcasts gratuitos.

## Contexto de ejecución proceso de pruebas

Para las pruebas sobre las aplicaciones seleccionadas como AUT, se tendrá disponible un presupuesto de 10 horas persona, 10 horas máquina y ningún recurso monetario adicional. Las pruebas se ejecutarán en una computadora portátil, con procesador Intel Core i7 5500U, velocidad de procesador 2.4GHz, memoria RAM de 8GB. Para pruebas en Android, se utilizó un Motorola Moto G4 con versión de Android 7.0, con memoria RAM de 2GB.

## Objetivos del proceso de pruebas

Se desea probar la parte funcional de ambas aplicaciones, tanto la web como la móvil.

## Tipos de prueba

Se realizarán pruebas tipo E2E testing, headless testing, random testing y behaviour-driven testing.

# Generación y ejecución de estrategia de pruebas

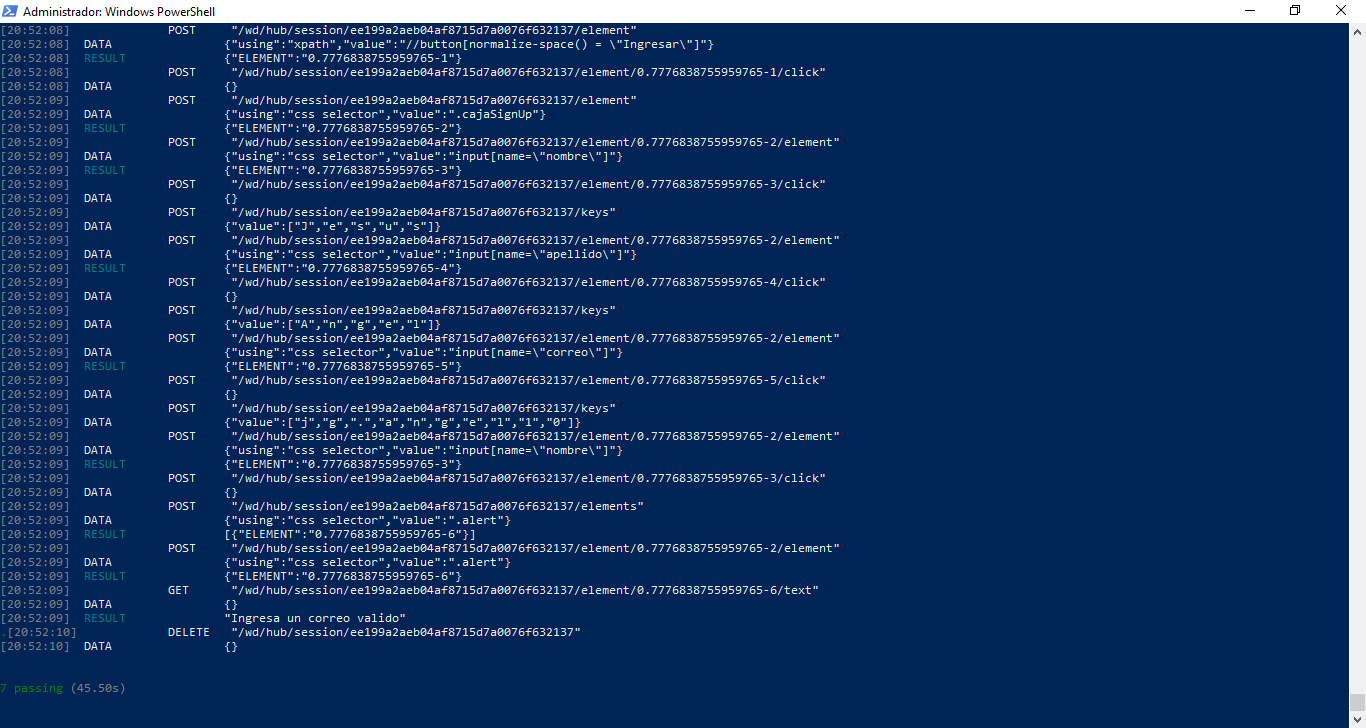
Como se menciona en la sección de definición de la estrategia de pruebas, las pruebas que se tendrán en cuenta en este trabajo son automatizadas, en este caso la ejecución y la validación mediante los oráculos son realizados automáticamente, aunque la generación es generada manualmente.

Para poder reproducir las pruebas con los comandos que están descritos en cada uno de los tipos de prueba, se debe primero descargar el proyecto y ejecutar .

### E2E Testing

Se generaron pruebas de E2E testing para nuestra aplicación web [www.losestudiantes.co](http://www.losestudiantes.co), usando el framework de automatización webdriver.io, usando un lenguaje de descripción de pruebas Jasmine. Dado que ya se habían realizado pruebas E2E sobre la aplicación web en los talleres (taller 2, link repositorio: <https://github.com/jgangel/taller2>), se utilizaron éstas como base en esta iteración de pruebas para nuestra aplicación web, y se añadieron dos funcionalidades adicionales, al igual que se aumentaron considerablemente los escenarios de prueba que se ejecutaban de un escenario a dos o tres por funcionalidad.

Las pruebas E2E están definidas en el proyecto, dentro del subdirectorio , y desde npm se permite ejecutar las pruebas como scripts con el comando o o directamente con . Los resultados de las pruebas definidas se ejecutan satisfactoriamente, como se muestra a continación

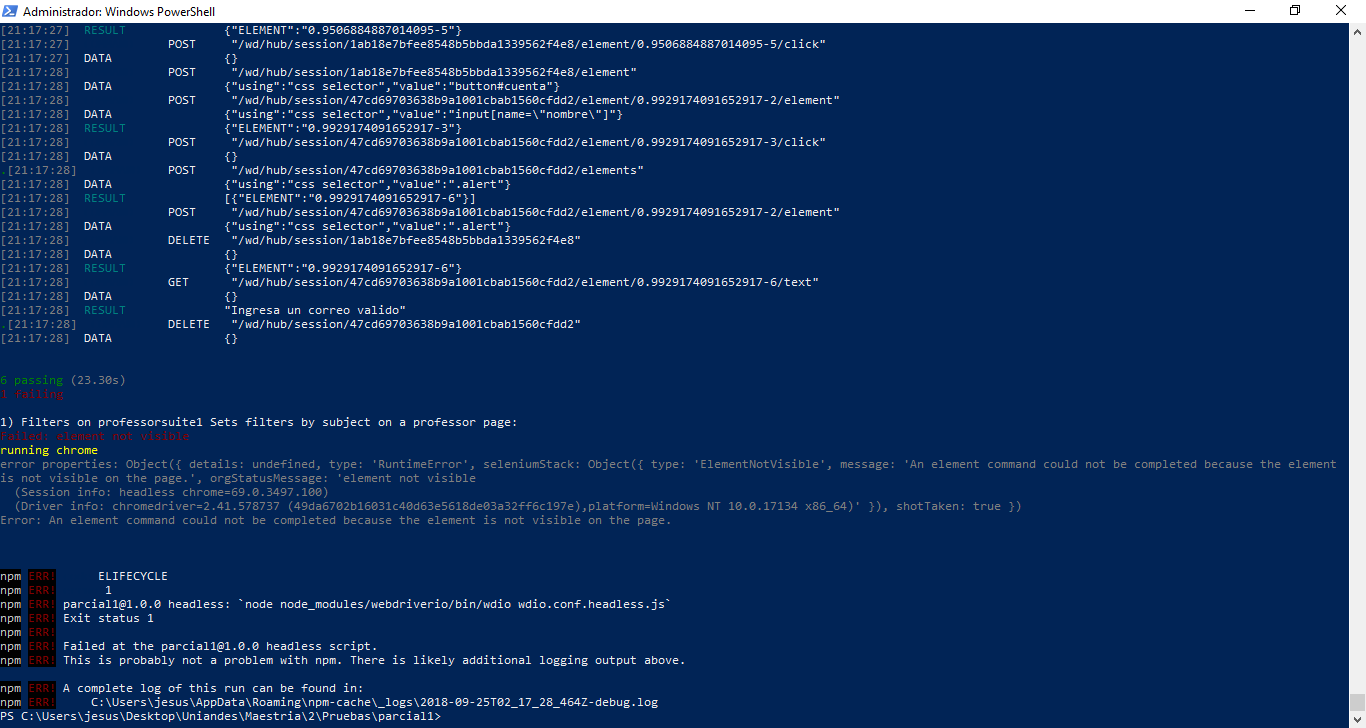


Para la aplicación Android, AntennaPod, se estudió Espresso en Android Studio, pero dado que se necesita el código fuente para la ejecución de pruebas bajo este framework, se decidió no realizar pruebas tipo E2E en aplicaciones Android.

### Headless Testing

Para correr nuestras pruebas en modo headless, sólo tuvo que hacerse algunas modificaciones al archivo de configuración definido en E2E testing para el caso web. Para AntennaPod no se ejecutaron pruebas en modo headless debido a que no se generó ninguna prueba E2E.

De esta manera, las pruebas definidas para headless testing son las mismas que se definieron para E2E testing, sólo que la ejecución se realiza sin la GUI de la aplicación y paralelizando ejecuciones gracias al navegador Chrome que permite este modo de operación. Al ejecutar, desde npm con el comando o haciendo el llamado a wdio con el archivo de configuración correspondiente, se obtiene un resultado que no es el esperado, como se muestra a continuación

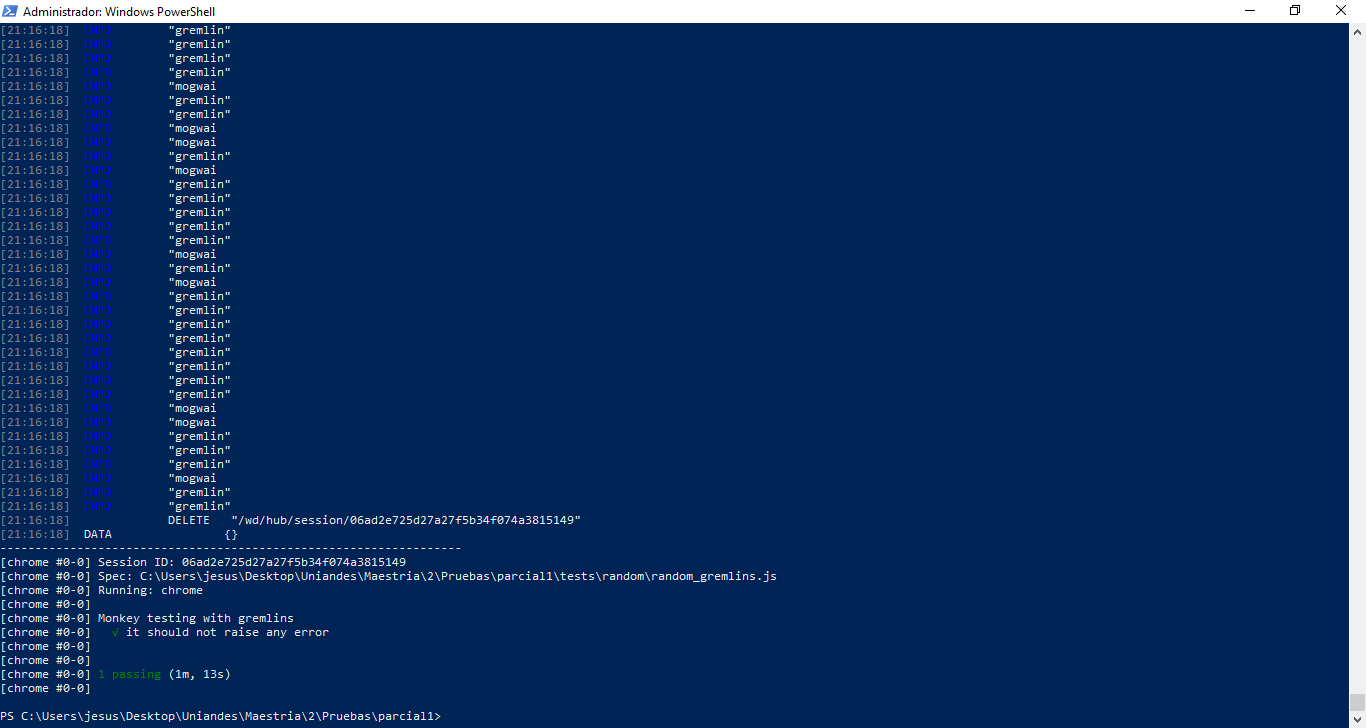


Como se observa, hay una prueba (de aplicar filtros a profesores) que falla en modo headless. Se tuvo en cuenta que podría deberse a una prueba flaky, sin embargo al ejecutar repetidas veces (cerca de 15 veces) en modo headless para ver si la prueba pasaba, el resultado siempre fue el mismo. Se ejecutó análogamente con el modo normal y no hubo variación en los resultados.

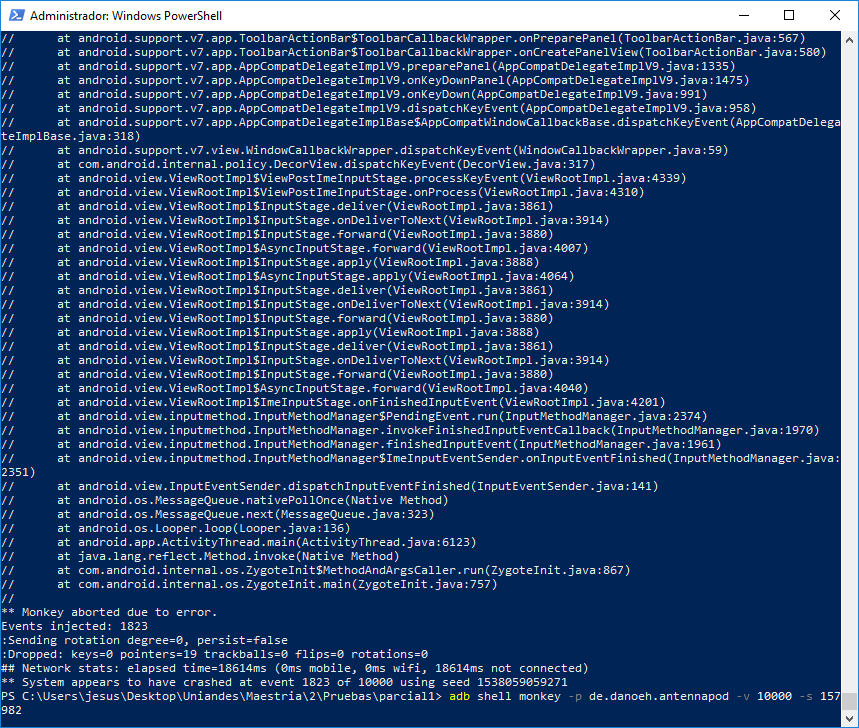
## Random testing

Para Random Testing, sí se ejecutaron pruebas sobre ambas aplicaciones, [www.losestudiantes.co](http://www.losestudiantes.co) de tipo Web y AntennaPod para dispositivos Android.

Para random testing sobre aplicaciones web, se utilizó la herramienta de gremlins.js, basada en webdriver.io, que facilita considerablemente la creación de tests tipo random sobre aplicaciones web utilizando “gremlins” que ejecutan eventos aleatorios sobe la GUI de la app. A continuación se muestra una ejecución satisfactoria de random testing, con semilla 189645463 y tiempo de ejecución de 1 minuto. Observando a detalle las acciones que hace sobre la GUI,



Para random testing en Android, se utilizó un monkey desde la shell con la que nos permite comunicar adb. Se ejecutó el comando desde un dispositivo Android real, tal como se especifica en la definición de nuestra estrategia de pruebas.



El resultado de un escenario en el cual se crashea la app se muestra en el siguiente video (dar doble click sobre el enlace). A pesar de que se usó la semilla que se muestra en la imagen 157982, no es suficiente para reproducir los eventos, ya que los eventos dependen adicionalmente del estado en que se encuentra la aplicación y el estado del dispositivo.



## Behaviour Driven Testing

Finalmente, se ejecutan pruebas de BDT. Sólo se ejecutan en Cucumber para [www.losestudiantes.co](http://www.losestudiantes.co), debido a que para ejecutar Calabash en Android se requiere del apk de la aplicación en cuestión, el cual no se tiene, sólo la aplicación publicada en Google Play.

Para ello, se tiene como base los escenarios creados en Cucumber para talleres anteriores, al ejecutar el script con , se obtienen resultados satisfactorios, tanto para los antiguos como para los features nuevos, como se observa a continuación

