

Plan de Pruebas

02-2017

Universidad de Los Andes

MISO 4208 - Pruebas Automáticas

Felipe Martinez (f.martinez2@uniandes.edu.co)

Juan Hernández (js.hernandez15@uniandes.edu.co)

Estrategia de pruebas

Alcance

Aplicaciones

La ejecución de pruebas será realizada sobre las siguientes aplicaciones:

- [Habitica](#): es una aplicación web/juego diseñada como un juego con la intención de ayudar a los jugadores a mejorar hábitos de la vida real. Permite convertir todas las tareas de los jugadores en recompensas para un personaje tales como objetos y experiencia.
- [Habitica Mobil](#): aplicación Android para Habitica
- [GNU Cash](#): es una aplicación Android para control de gastos financieros. Sirve como complemento a la versión de escritorio con la que puede sincronizarse. Con ella se puede realizar control de gastos, trazabilidad de cuentas

Testers

- Felipe Martínez
- Juan Hernández

Tareas

Entre las tareas de los testers se encuentran:

- Investigación sobre las herramientas a usar para las pruebas automáticas.
- Pruebas de concepto de los frameworks y herramientas a utilizar.
- Configuración del entorno para cada una de las aplicaciones.

Presupuesto

Los testers asignados para la ejecución de pruebas se encuentran con una dedicación de

- Tiempo de Tester: 4 horas semanales
- Tiempo ejecución: 2 horas semanales

Niveles y Objetivos

Pruebas Unitarias

- Ejecución de pruebas unitarias de desarrollo incluidas dentro del código fuente de las aplicaciones

- Ejecución de pruebas automáticas de tipo Monkey Testing

Entregables

Los documentos entregables están enmarcados en los reportes generados por las herramientas usadas a través del curso de pruebas automáticas de software. Adicionalmente se relacionarán las evidencias de la ejecución de pruebas tales como capturas de pantallas o videos de acuerdo a las características de los frameworks que se utilicen para realizar las pruebas. Los artefactos están almacenados en el [repositorio del equipo](#).

Plan de pruebas - Sprint 1

Objetivo del sprint

- Contextualizar al equipo de pruebas acerca de las características de las aplicaciones a probar
- Usar las herramientas vistas en clase para entender las capacidades de la aplicación en relación con desempeño, PWA y accesibilidad.
- Aplicar herramientas de Monkey Testing para ejecutar rutinas sobre las aplicaciones y reportar los resultados

Test Backlog

<https://trello.com/b/9mr9sE/backlog>

Testing - App1: Habitica (Web)

Tipos de Pruebas

Unitarias

- Ejecución de pruebas unitarias: ejecutar la pruebas unitarias incluidas en el código fuente de la aplicación. Determinar cobertura de las mismas y estado de la compilación (satisfactorio/fallido)
- Reporte lighthouse: ejecutar el reporte de lighthouse sobre la aplicación web para determinar el estado de la aplicación respecto a las aspectos evaluados por esta herramienta.
- Monkey Testing - Gremlins
 1. Ejecución completa: configurar gremlins.js para lanzar todos lo tipos de agentes contra la aplicación web alojada

```
.gremlin(gremlins.species.formFiller())  
.gremlin(gremlins.species.clicker())  
.gremlin(gremlins.species.scroller())  
.gremlin(gremlins.species.typer())
```
 2. Ejecución sin mover el scroll: durante la ejecución de la pruebas de concepto se identificó que el agente que controlaba el scroll del explorador generaba que las acciones de click o teclear no se capturarán adecuadamente por lo que se desactivo para esta ejecución.

```
.gremlin(gremlins.species.formFiller())
.gremlin(gremlins.species.clicker())
.gremlin(gremlins.species.typer())
```

3. Ejecución para completar forms del sitio: utilizar el agente **"formFiller"**
4. Ejecución personalizada: implementar un agente personalizado de acuerdo las características del sitio web para inducir posibles errores

```
.gremlin(gremlins.species.formFiller())
.gremlin(function () {
    $("button[type='submit']").click();
    console.log("custom gremlin click submit");
})
.strategy(gremlins.strategies.distribution()
    .delay(50) // wait 50 ms between each action
    .distribution([0.5, 0.5])
)
```

Testing - App2: Habitica (Android)

Tipos de Pruebas

Unitarias


- Ejecución de pruebas unitarias: Ejecutamos las pruebas unitarias usando el archivo gradlew, este reporte generó 32 pruebas exitosas y 12 con fallas. Adicionalmente instalamos y configuramos el plugin de Jacoco en el script de Gradle para configurar el reporte de cobertura de líneas y ramas de código, este plugin no reportó la cobertura debido a las pruebas fallidas.
- Monkey Testing - Android Monkey: ejecucion de 3 rondas de monkey testing usando las herramientas provistas por el Android SDK

Testing - App3: GNU Cash (Android)

Tipos de Pruebas

Unitarias

- Ejecución de pruebas unitarias: Ejecutamos las pruebas unitarias usando el archivo gradlew, este reporte generó 134 pruebas exitosas y 0 fallidas. Adicionalmente instalamos y configuramos el plugin de Jacoco en el script de Gradle para configurar



el reporte de cobertura de líneas y ramas de código, este plugin reportó que a pesar de la cantidad de pruebas sólo teníamos un 2% de cobertura.

- Monkey Testing - Android Monkey: ejecución de 2 rondas de monkey testing usando las herramientas provistas por el Android SDK.