

Test your front-end against a real API

Proyecto: REQ|RES

Por: Teodoro Calle Lara

Estrategia de Prueba Automatizadas
Training sofka

Historia de revisiones

Versión	Autor(es)	Descripción	Fecha
1.0	Teodoro Calle Lara	Estrategias de pruebas REST API-REQRES	Octubre 2021

Tabla de Contenidos

1. Introducción
2. Alcance
3. Análisis de riesgos
4. Ambiente y Herramientas de Pruebas
 - 4.1 Herramientas de Pruebas
 - 4.2 Arquitectura del framework de automatización
5. Reportes
 - 5.1 Api/unknown

1.Introducción

En esta Estrategia para la realización de pruebas automatizadas se describe el alcance de las pruebas, el ambiente de pruebas, los recursos necesarios, las herramientas a utilizar para la ejecución de las pruebas del sitio Reqres

2. Alcance

Se realizarán pruebas de caja negra (automatizadas) a las funcionalidades asignadas para el reto del curso "Screenplay - Serenity BDD - REST API".

En este caso el alcance son los Recursos son:

- **Recurso /api/unknown**

3. Análisis de riesgos

Criterios de aceptación:

- **CA001:** Get user por id
- **CA002:** Get user por id invalido

Antes de empezar con la ejecución de pruebas se empezara con un testeo de software (smoke test) para asegurarnos de que el ambiente, funcionalidades básicas y críticas del programa funcionan correctamente, este será un testeo rápido y no exhaustivo, por lo tanto debemos enfocarnos en reconocer cuales son las funciones básicas y críticas, para este caso se verificara que se abra de manera correcta el sitio web y que la información desplegada sea legible, entendible, ordenada y coherente con lo que se esperaría encontrar en un sitio web.

Vamos a realizar las pruebas en cierto orden en base a un análisis de riesgos, donde se calculara el riesgo como: **riesgo = probabilidad*impacto**, y el orden será del mayor riesgo al menor.

También se tendrá en cuenta la cobertura que tendremos al realizar estas pruebas en cuanto a los navegadores y dispositivos móviles, sin embargo no se utilizaran todas las opciones del mercado ya que son muchas, nos enfocaremos en Google Chrome en Windows 10.

No	Criterios de aceptación.	Probabilidad (1-4)	Impacto (1-4)	Riesgo
1	CA001	1	3	3
2	CA002	1	3	3

4. Ambiente y Herramientas de Pruebas

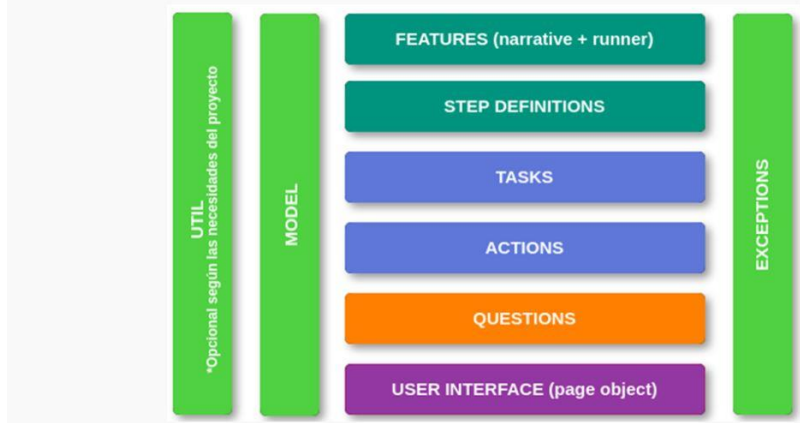
4.1 Herramientas de Pruebas

Herramienta	Función
-------------	---------

Serenity BDD	Ejecución y Reporte de las pruebas
Gradle	Creación de la estructura de proyectos y uso e importación de librerías
Cucumber	Escribir Features y Scenarios con lenguaje Gherkin
Servicios rest	Interfaz para conectar varios sistemas basados en el protocolo HTTP.

4.2 Arquitectura del framework de automatización

Arquitectura de referencia para Screenplay



Usando el patrón Screenplay que tiene un enfoque de desarrollo encaminado por comportamiento Behaviour Driven Development (BDD). **Esta no es una herramienta para testing sino una estrategia de desarrollo que se enfoca en prevenir defectos en lugar de encontrarlos** en un ambiente controlado.

Para automatizar procesos, también se utiliza la herramienta Cucumber para implementar metodologías como BDD, **las cuales permiten ejecutar descripciones funcionales escritas en texto plano como pruebas de software automatizadas.**

Para la implementación en los temas de la automatización, debemos tener en cuenta que se deben manejar ciertos conceptos que abarquen los siguientes elementos: **Runners, User Interface, Features, StepDefinitions, task, Interactions, Questions y Reports.**

Runners: es el ejecutor de los features, tags, glue (donde se encuentra el step definition), URL del driver y la conexión con excel del drive.

User Interface: las UI son el mapeo de la interfaz, donde capturaremos todos los elementos con los cuales podríamos llegar a interactuar durante la automatización. Además, se le puede añadir la URL donde se iniciará la prueba.

Features: los features son las historias de usuario que se llevarán a cabo en las pruebas y proveerá los métodos que utilizaremos más adelante para los StepDefinitions.

StepDefinitions: los Step Definitions son la traducción de los features a código. Los métodos que se utilizaran son los features (historias de usuario), por lo tanto, iremos a la clase “RunnerTags”, le daremos clic derecho sobre ésta, y en la opción “Run as” escogemos “JUnit Test”.

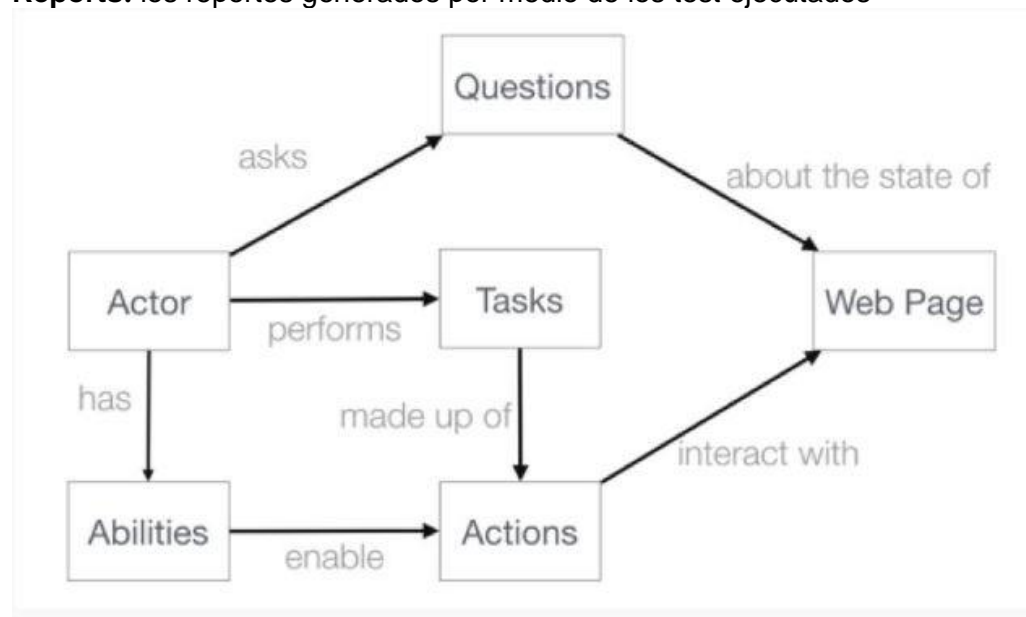
Task: son las interacciones que se llevarán a cabo para cumplir con las historias de usuarios planteadas. **Las tasks se pueden caracterizar porque no se habla en términos de clic, set data o select.** Son verbos más amplios como loguearse formulario, cerrar sesión y buscar.

Tal y como explica Víctor Soto, experto en automatización de pruebas de Pragma, hay que pensar en términos de tareas. Por ejemplo, para ejecutar la tarea “Login”, se requiere de algunas Interacciones y saber interpretar cuáles son esas acciones a llevar a cabo y verificar que sí cumplan con el objetivo principal.

Interactions: indicar acciones como dar clic, select, enviar datos, scroll, entre otras cosas.

Questions: son lo assert a llevar a cabo para asegurar el cumplimiento de ciertos parámetros.

Reports: los reportes generados por medio de los test ejecutados



5. Reportes

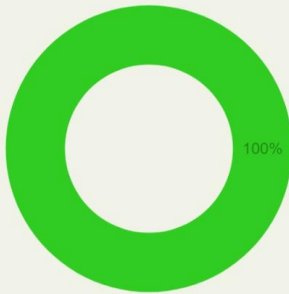
5.1 Api/unknown








Feature: Buscar Un Usuario Desconocido

2 test scenarios | 

 Summary

 Test Results



Scenario Results (including rows of test data)		Automated	
 Passing	2	100%	
 Pending	0		
 Ignored	0		
 Skipped	0		
✖ Unsuccessful			
 Failed	0		
 Broken	0		
 Compromised	0		
Total	2		

Execution Time	Clock Time	Fastest Test	Slowest Test	Average Execution Time
5s	5s	553ms	5s	2s

Functional Coverage Overview

Features

Feature	Scenarios	% Pass	Result	Coverage
Buscar un usuario desconocido	2	100%		<div style="width: 100%; height: 10px; background-color: green;"></div>

Tags

Capability

 Unknown 2

Feature: Users

Placeholder (capability)

Como un usuario que visita una base de datos de usuarios
y necesita filtrar los datos de los usuarios
para obtener su informacion basica

[Specifications](#)[Test Results](#)

Scenario Results (including rows of test data)		Automated	
✓ Passing	3	100%	
⏸ Pending	0		
⌛ Ignored	0		
⚠ Skipped	0		
✗ Unsuccessful			
✗ Failed	0		
⚠ Broken	0		
✗ Compromised	0		
Total	3		

Execution Time	Clock Time	Fastest Test	Slowest Test	Average Execution Time
8s	8s	424ms	7s	2s

Automated Tests

Show 10 entries

Filter

Scenario	Steps	Start Time	Duration	Result
get users completed	3	20:58:44	7s 439ms	✓
filter user by username	3	20:58:52	772ms	✓
filter user wrongly	3	20:58:53	424ms	✓

Showing 1 to 3 of 3 entries

Previous

1

Next

Manual Tests

No manual tests were recorded