

Reto: Reto de automatización - Serenity BDD - Servicios Rest

Estrategia de Automatización de Pruebas

Por:

Hansee Jiménez Pérez

**Historia de revisiones**

|  |  |  |  |
| --- | --- | --- | --- |
| Versión | Autor(es) | Descripción | Fecha |
| 1.0 | Hansee Jimenez | Creación del documento | 27 de Marzo 2022 |

Tabla de contenido

[1.Introducción 4](#_Toc99294630)

[2. Alcance 5](#_Toc99294631)

[3. Roles y Responsabilidades 5](#_Toc99294632)

[4. Análisis Riesgos 6](#_Toc99294633)

[4.1 Riesgos del Servicio PokeApi 6](#_Toc99294634)

[4.2 Riesgos del Servicio {JSON} Placeholder 6](#_Toc99294635)

[4.3 Riesgos del Servicio REQ | RES 7](#_Toc99294636)

[5. Ambiente y Herramientas de Pruebas 7](#_Toc99294637)

[5.1 Herramientas de Pruebas 7](#_Toc99294638)

[6. Criterios de Entrada y Salida 8](#_Toc99294639)

[6.1 Criterios de Entrada 8](#_Toc99294640)

[6.2 Criterios de Salida 8](#_Toc99294641)

[7. Planificación de ejecución de las pruebas 8](#_Toc99294642)

[8. Reporte de Pruebas 9](#_Toc99294643)

# 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, los riesgos y la planificación de ejecución de las pruebas para los servicios REST que ofrecen las siguientes páginas:

* **Sitio Web** **PokeApi**  (<https://pokeapi.co>).
* **Sitio Web** **{JSON} Placeholder** (<https://jsonplaceholder.typicode.com/>).
* **Sitio Web** **REQ | RES**  (<https://reqres.in/>).

# 2. Alcance

Se realizarán cuatro pruebas de caja negra (automatizadas) utilizando tres servicios web, empleando en cada una de las pruebas diferentes métodos Http, que ofrecen dichos servicios verificando que las peticiones solicitadas contengan el estado y la información esperada.

Las funcionalidades a ser automatizadas serán seleccionadas con base a los servicios y los siguiented endpoint:

* **Servicio PokeApi**
  + GET – Petición de información sobre una habilidad – Endpoint=”/api/v2/ability/{Id o Nombre}”.
* **Servicio {JSON} Placeholder**
  + PUT – Modificación del comentario de un post – Enpoint="/comments/{id}".
  + POST – Creación de un nuevo comentario en un post – Endpoint=”/comments”.
* **Sitio Web REQ | RES**
  + DELETE – Eliminación de un usuario – Endpoint= "/api/users/{id}".

# 3. Roles y Responsabilidades

|  |  |
| --- | --- |
| Roles | Responsabilidades |
| Coach de QA | Monitoreo de las pruebas automatizadas  Reporte de progreso de las pruebas |
| Estudiante QA de Automatización | Diseño e implementación de las pruebas.  Ejecución de las pruebas automatizadas.  Reporte de resultados de las pruebas. |

# 4. Análisis Riesgos

## 4.1 Riesgos del Servicio PokeApi

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Riesgos** | **Probabilidad de Ocurrencia**  **(1-5)** | **Impacto**  **(1-5)** | **Criticidad**  **(Prob\*Impacto)** |
| 1 | La petición de información se realiza exitosamente | 2 | 5 | 10 |
| 2 | La petición contiene información errónea | 1 | 3 | 3 |
| 3 | La respuesta de la solicitud no es la correcta | 1 | 3 | 3 |
| 4 | El servicio no se encuentra disponible | 1 | 5 | 5 |

## 4.2 Riesgos del Servicio {JSON} Placeholder

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Riesgos** | **Probabilidad de Ocurrencia**  **(1-5)** | **Impacto**  **(1-5)** | **Criticidad**  **(Prob\*Impacto)** |
| 1 | La petición PUT se realiza exitosamente | 2 | 5 | 10 |
| 2 | La petición POST se realiza exitosamente | 2 | 5 | 10 |
| 3 | El estado de respuesta no es el adecuado | 2 | 4 | 8 |
| 4 | El cuerpo de la solicitud contiene la información correcta | 2 | 5 | 10 |
| 5 | El servicio no se encuentra disponible | 1 | 5 | 5 |

## 4.3 Riesgos del Servicio REQ | RES

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Riesgos** | **Probabilidad de Ocurrencia**  **(1-5)** | **Impacto**  **(1-5)** | **Criticidad**  **(Prob\*Impacto)** |
| 1 | Solicitud de eliminación se realiza correctamente | 2 | 5 | 10 |
| 2 | El estado de respuesta de la solicitud no es el adecuado | 1 | 3 | 3 |
| 3 | El servicio no se encuentra disponible | 1 | 5 | 5 |

# 5. Ambiente y Herramientas de Pruebas

## 5.1 Herramientas de Pruebas

|  |  |
| --- | --- |
| Herramienta | Función |
| JUnit Framework | Ejecución de pruebas |
| Gradle | Creación de la estructura de proyectos y uso e importación de librerías |
| Cucumber Framework | Implementar BDD y facilitar la automatización de pruebas |
| SerenityBDD Framework | Generación de reportes y documentación viva  Aplicar el patrón de Screenplay en el diseño de pruebas y conexión con API |
| JavaFaker | Librería para la creación de información de usuarios de forma realista |

# 6. Criterios de Entrada y Salida

## 6.1 Criterios de Entrada

Las página web de PokeApi, {JSON} Placeholder y REQ | RES deben estar completamente funcional y se debe poder tener acceso a ella.

Tener todas las herramientas (ambiente) de automatización instaladas y configuradas correctamente.

Tener información previa del comportamiento de las páginas y sus endpoint con el tipo de información que transfiere.

## 6.2 Criterios de Salida

Ejecución de al menos cuatro escenarios de prueba en su totalidad.

Reporte de resultados de las pruebas realizadas.

# 7. Planificación de ejecución de las pruebas

Lista de funcionalidades a ser automatizadas por operación Http

|  |  |  |
| --- | --- | --- |
| Feature | Http | Funcionalidad / Escenario |
| Habilidades de Pokemon | GET | Pokemon que posee una habilidad |
| Comentarios de usuario en los post | POST | Crear un nuevo comentario en un post |
| PUT | Modificar nombre y cuerpo de un comentario |
| Administración de usuarios | DELETE | Eliminar usuario |

Es necesario que las funcionalidades a automatizar se desarrollen, implementen y prueben manualmente con anterioridad para tener conocimiento de las respuestas e información que se entrega.

# 8. Reporte de Pruebas

El Reporte automático de pruebas se obtendrá a través de SerenityBDD. Este Reporte informará sobre los resultados de la ejecución de cada caso de prueba. Incluirá las pruebas que pasaron y las que fallaron, los errores encontrados, la tasa de éxito y el tiempo transcurrido.

A continuación se evidencia una captura del reporte generado por SerenityBDD, sin embargo, para poder explorar todos los detalles del reporte se recomienda acceder a GitHub y clonar el repositorio, seguido acceder a la ruta **\target\site\serenity** y abrir en el navegador el archivo **index.html**.

