

Plan de pruebas

Tienda de comercio electrónico SwagLabs

Versión 1.0

Diciembre - 2022

Hoja de control

Sistema	Sistema de comercio electrónico SwagLabs		
Entregable	Plan de pruebas detallado		
Autor	Emanuel Martínez Pinzon - Ceiba Software		
Versión	1.0	Fecha de versión	27/12/2022
Aprobado por		Fecha de aprobación	
		Páginas	21

Registro de cambios

Versión	Causa del cambio	Responsable	Fecha
1.0	Versión inicial	Emanuel Martínez Pinzón	27/12/2022

Tabla de contenido

1. Introducción	4
2. Objetivo del plan de pruebas	4
3. Definiciones, acrónimos y abreviaturas	4
4. Alcance de las pruebas	7
5. Estrategia de las pruebas	9
5.1. Planificación	9
5.2. Diseño	12
5.2.1. Tipo de pruebas	12
5.2.2. Tecnologías a usar	12
5.2.3. Escenarios en Gherkin a validar	15
5.3. Reporte	17
6. Entregables de pruebas	18
7. Matriz de riesgos	19

PLAN DE PRUEBAS

1. Introducción

SwagLabs es una plataforma de comercio electrónico centrada en la venta de implementos y artículos deportivos, su intuitiva interfaz permite al usuario navegar por un listado de productos, ver su información detallada y realizar el proceso completo de compra.

Para garantizar la buena experiencia de usuario y la calidad de la plataforma se decide realizar un conjunto de pruebas fundamentadas en altos estándares que no solo permiten el seguimiento, sino también las correcciones oportunas del producto.

El propósito de este documento es ser el plan de pruebas que guíe al equipo de calidad para garantizar la calidad, operatividad y funcionalidad del sistema de comercio electrónico.

2. Objetivo del plan de pruebas

El objetivo de este plan de pruebas es entregar las pautas y definir el alcance y la estrategia que permita la certificación en calidad de las funcionalidades requeridas para SwagLabs.

3. Definiciones, acrónimos y abreviaturas

Lenguaje de programación: Es un lenguaje formal que le proporciona al programador la capacidad de escribir una serie de instrucciones o algoritmos con el fin de controlar el comportamiento físico o lógico de un sistema informático.

Java: Es un lenguaje de programación ampliamente usado para codificar aplicaciones para amplios dispositivos. Pueden desarrollarse desde aplicaciones móviles, plataformas web hasta robustos sistemas empresariales con alta exigencia.

Groovy: Es un lenguaje compatible con la sintaxis de Java y que corre sobre la JVM. Puede usarse tanto como lenguaje de programación como de lenguaje de secuencias de comandos.

JVM: La máquina virtual de Java (Java Virtual Machine en Inglés) es la que permite la ejecución de programas Java en múltiples plataformas,

sin embargo también permite la ejecución de programas escritos en otros lenguajes como Groovy o Scala.

Gherkin: Es un lenguaje de programación de sintaxis simple que permite a los equipos de negocio escribir sus propias pruebas de software. Usa el enfoque BDD permitiendo que los desarrolladores realicen sus implementaciones basados en los escenarios escritos por equipos no técnicos.

IDE: Entorno de desarrollo integrado (Integrated development environment en Inglés). Corresponde a una aplicación robusta cargada de herramientas y opciones que permiten la escritura y ejecución del código de software.

Librería: Es un conjunto de implementaciones funcionales que buscan facilitar el desarrollo incorporando elementos comunes que ya fueron resueltos por otros desarrolladores.

Framework: Son un conjunto estandarizado de conceptos, prácticas y herramientas que buscan resolver problemas específicos.

Gradle: Es un sistema de automatización y construcción de software. Gradle soporta múltiples lenguajes de programación pero usa el lenguaje Groovy como lenguaje para su escritura.

Git: Es un sistema de control de versiones distribuido.

GitHub: Es un servicio basado en la nube que aloja proyectos de código usando Git.

BDD: Desarrollo guiado por comportamiento (Behavior Driven Development en inglés). Es un proceso de desarrollo que busca la colaboración y entendimiento entre desarrolladores, gestores del proyecto y el equipo de negocio.

Serenity BDD: Es un framework que facilita la implementación de BDD en los proyectos de pruebas de software. Permite escribir pruebas automatizadas con mayor calidad y de forma eficiente.

Cucumber: Es una herramienta de calidad de software que permite escribir pruebas siguiendo el estilo BDD, usando Gherkin permite escribir funcionalidades y escenarios con lenguaje simple de entender y al tener soporte en múltiples lenguajes permite su implementación en diferentes proyectos.

Patrón de diseño: Son técnicas comunes para resolver problemas en el desarrollo de software y otros ámbitos referentes al análisis o diseño de sistemas.

ScreenPlay: Es un patrón de diseño con un enfoque BDD que busca una aproximación centrada en el usuario. Ayuda a escribir y leer pruebas de una manera más sencilla mostrando los escenarios como si fuesen guiones.

Pirámide de Cohn: También conocida como pirámide de pruebas es la estrategia más aceptada para realizar pruebas de software en metodologías ágiles. Se divide en tres niveles siendo la base las pruebas unitarias, el centro las pruebas de servicios y las pruebas funcionales.

Pruebas unitarias: Son las pruebas de más bajo nivel, buscan comprobar el correcto funcionamiento de las unidades de menor tamaño como los métodos.

Pruebas mutantes: Son un tipo de pruebas que realizan pequeñas modificaciones al código para validar la calidad de las pruebas unitarias. Consisten en validar que las pruebas unitarias están validando solo el escenario que les corresponde y si este cambia ellas deberían fallar.

Pruebas de servicios: Corresponden al nivel medio en la pirámide de Cohn y hacen referencia a esas pruebas que intentan validar el correcto funcionamiento de varios bloques de código, ejemplo son las pruebas de integración.

Pruebas de integración: Estas pruebas están pensadas para verificar que los distintos módulos o servicios funcionan bien en conjunto.

Pruebas funcionales: Estas pruebas son similares a las pruebas de integración, pero se centran en los requisitos empresariales de una aplicación. Verifican el resultado de una funcionalidad y no sus estados intermedios.

Pruebas de extremo a extremo: También conocidas como pruebas integrales replican el comportamiento de un usuario con el software.

Pruebas de rendimiento: Evalúan el comportamiento del sistema bajo diferentes escenarios de carga.

4. Alcance de las pruebas

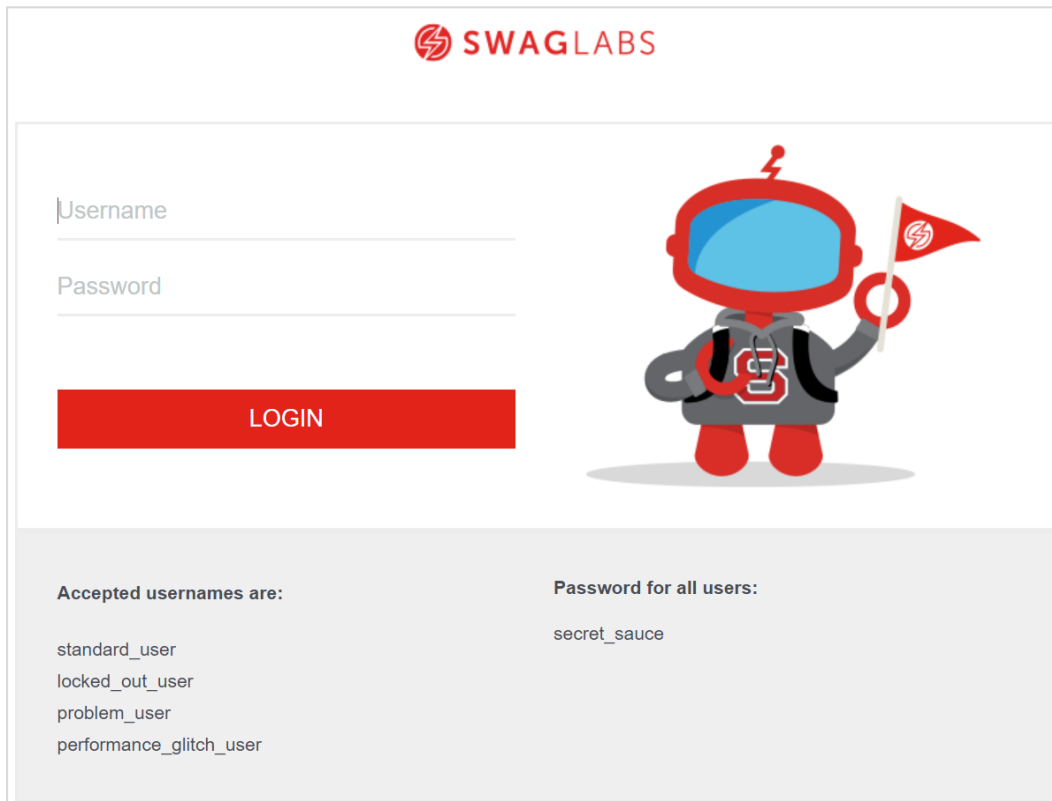
Para garantizar la alta calidad de la aplicación se desarrollarán pruebas automatizadas sobre tres funcionalidades principales definidas por SwagLabs como críticas: autenticación de usuario, carrito de compra y confirmación de pedido.

De la siguiente manera se describe cada funcionalidad así como los escenarios que se tendrán en cuenta:

Autenticación de usuario: Se realizarán pruebas de inicio de sesión sobre los siguientes usuarios registrados en el sistema:

- standard_user
- locked_out_user
- problem_user
- performance_glitch_user

Se probará ingresando datos de usuario válidas e inválidos sobre el formulario



The screenshot shows the SwagLabs login interface. At the top is the SwagLabs logo. Below it is a login form with two input fields: 'Username' and 'Password'. To the right of the form is a cartoon robot character holding a flag. Below the form is a red 'LOGIN' button. At the bottom, there is a section with two columns of text: 'Accepted usernames are:' followed by a list of usernames, and 'Password for all users:' followed by the password 'secret_sauce'.

Accepted usernames are:

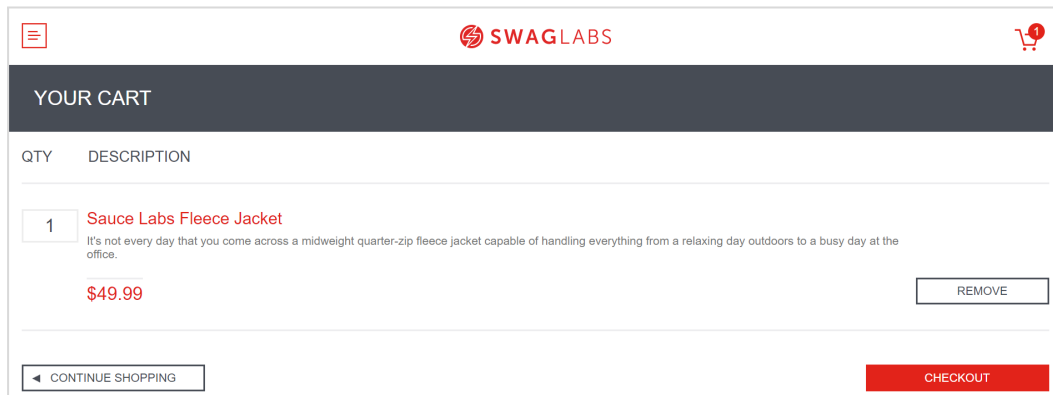
- standard_user
- locked_out_user
- problem_user
- performance_glitch_user

Password for all users:

secret_sauce

Formulario de inicio de sesión

Carrito de compra: Se realizará una prueba sobre el carrito de compra agregando el artículo “Sauce Labs Fleece Jacket”

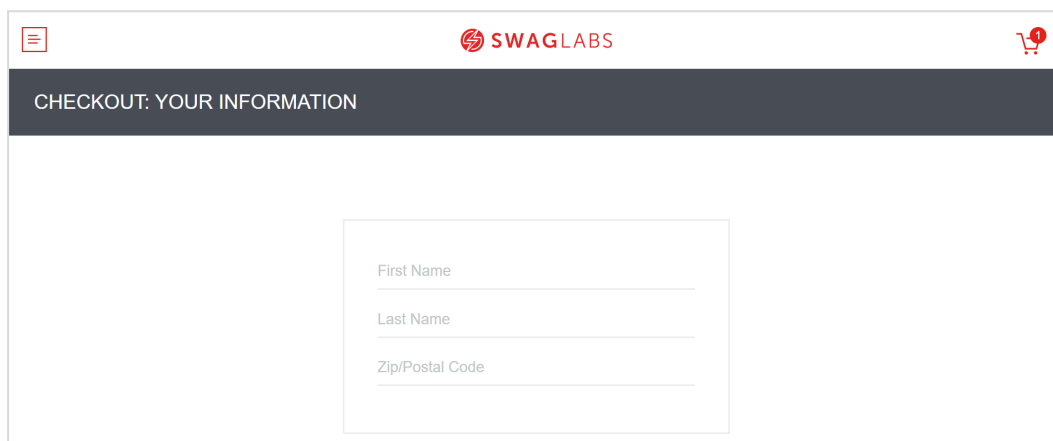


The screenshot shows the SWAGLABS website's shopping cart. At the top, there's a navigation bar with the SWAGLABS logo and a shopping cart icon. Below this is a dark header with the text "YOUR CART". The main area has a table with two columns: "QTY" and "DESCRIPTION". There is one item in the cart: "Sauce Labs Fleece Jacket" with a quantity of 1. The description reads: "It's not every day that you come across a midweight quarter-zip fleece jacket capable of handling everything from a relaxing day outdoors to a busy day at the office." The price is listed as "\$49.99". To the right of the item is a "REMOVE" button. At the bottom of the cart, there are two buttons: "CONTINUE SHOPPING" and "CHECKOUT".

Visualización del carrito de compra

Confirmación de pedido: Se realizará una prueba para confirmar el pedido del artículo “Sauce Labs Fleece Jacket” con los siguientes escenarios:

- Ingresando todos los datos de envío en el formulario de confirmación
- No ingresando o ingresando parcialmente los datos de envío en el formulario de confirmación



The screenshot shows the SWAGLABS website's checkout page. At the top, there's a navigation bar with the SWAGLABS logo and a shopping cart icon. Below this is a dark header with the text "CHECKOUT: YOUR INFORMATION". The main area contains a form with three input fields: "First Name", "Last Name", and "Zip/Postal Code".

Formulario de datos de envío

SwagLabs cuenta con otros equipos que se harán cargo de otros tipos de pruebas como lo son: pruebas unitarias, de integración, de seguridad, mutantes y otras. Estas pruebas no hacen parte de este plan de pruebas.

5. Estrategia de las pruebas

5.1. Planificación

Para realizar las pruebas automatizadas se tomará cada funcionalidad y en cada funcionalidad se validarán los escenarios descritos en la sección de [alcance de las pruebas](#), los siguientes son los diagramas de flujo de cada funcionalidad a probar con cada uno de los escenarios contemplados.

Autenticación de usuario

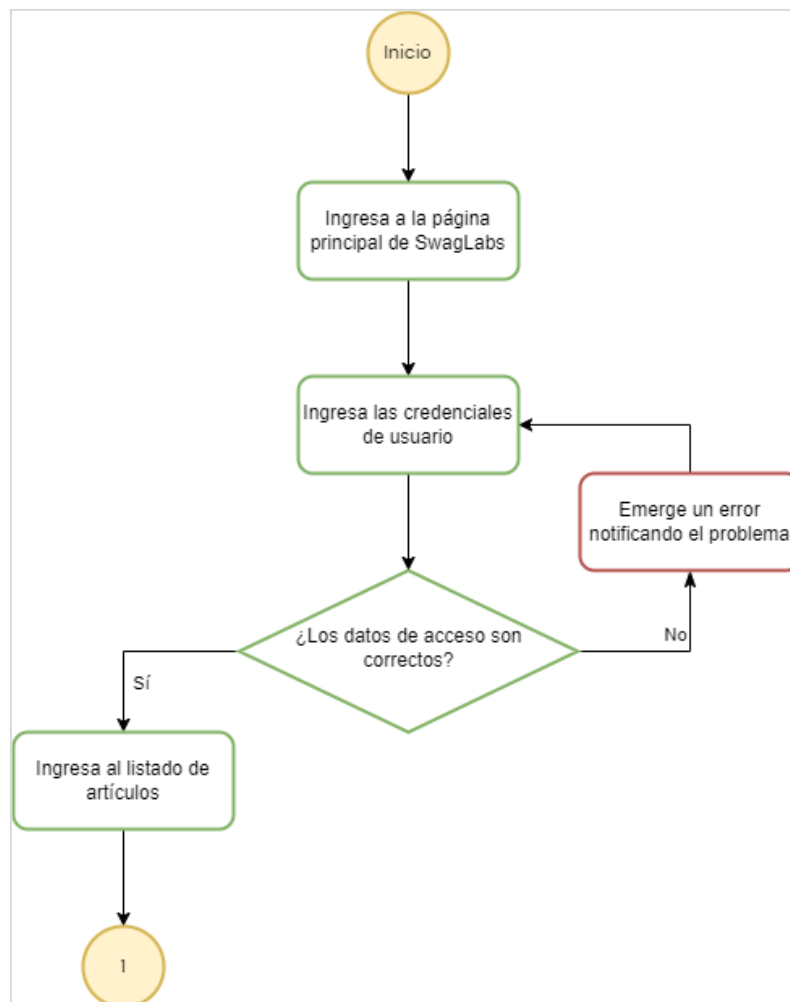
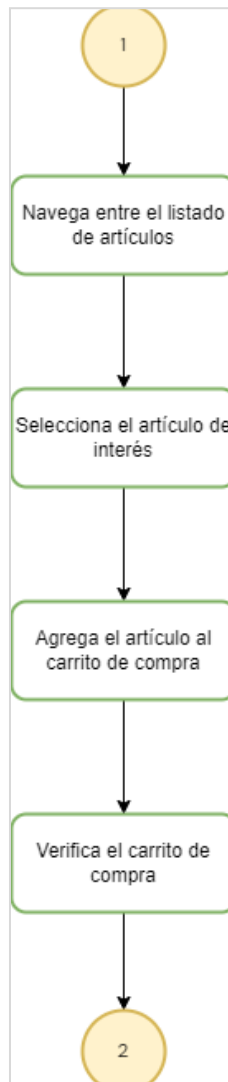


Diagrama de flujo para autenticación de usuario

Los dos escenarios descritos para esta funcionalidad que deben probarse son:

- Inicio de sesión con credenciales válidas: Debe permitir el ingreso al listado de artículos de la tienda.
- Inicio de sesión con credenciales inválidas: Debe emitir un mensaje de error informando al usuario qué ha sucedido.

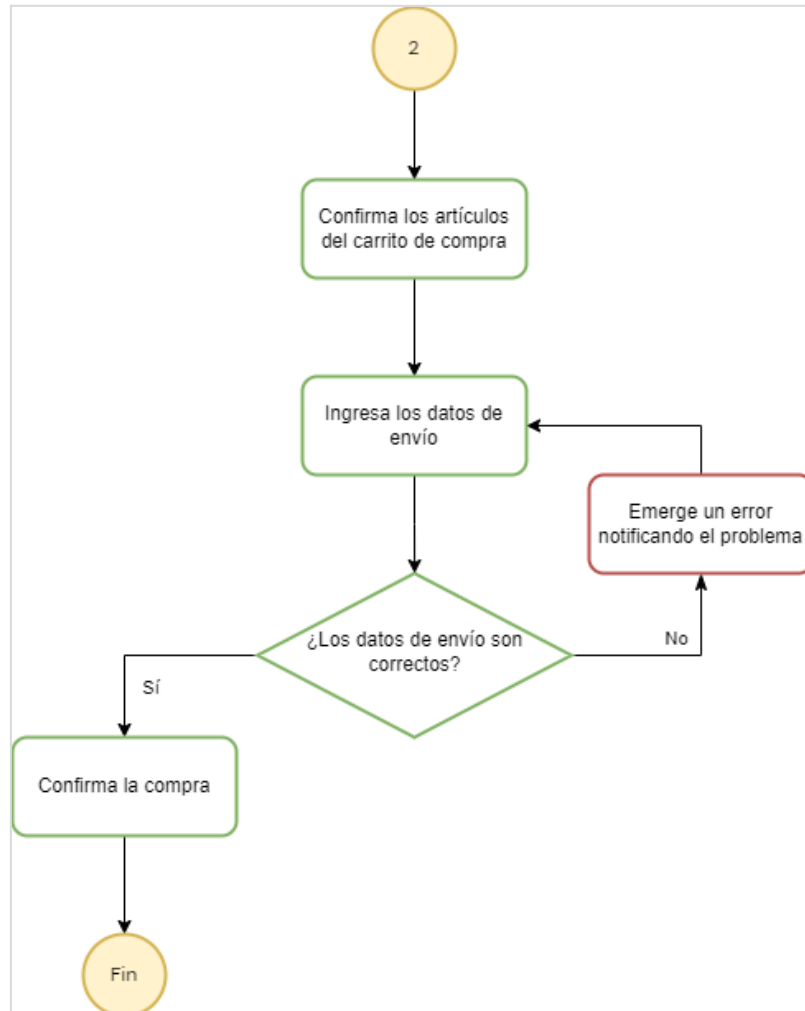
Carrito de compra



Esta funcionalidad cuenta con el siguiente escenario:

- Agregar artículo Sauce Labs Fleece Jacket al carrito: Una vez autenticado el usuario la plataforma le debe permitir navegar entre los artículos, seleccionar el artículo de interés y agregarlo al carrito, además debe validarse en el carrito el artículo agregado.

Confirmación de pedido



Los dos escenarios descritos para esta funcionalidad que deben probarse son:

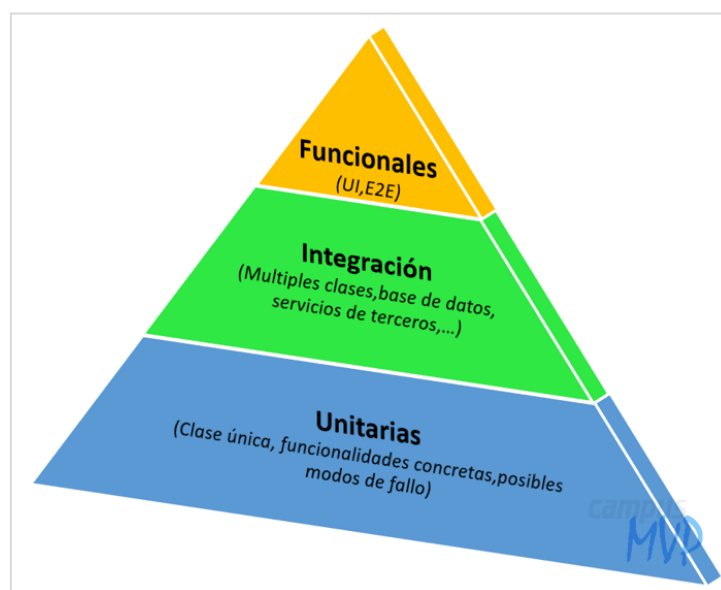
- Confirmar compra con datos de envío correctos: Debe permitir que el usuario confirme la compra y termine el proceso.
- Confirmar compra con datos de envío incorrectos: Debe emitir un mensaje de error informando al usuario qué ha sucedido.

Estos tres escenarios aunque serán probados de forma independiente corresponden a un flujo completo de experiencia de usuario, por ende cada funcionalidad debe considerar los pasos de la funcionalidad anterior como se ve en los diagramas de flujo.

5.2. Diseño

5.2.1. Tipo de pruebas

Para seleccionar las pruebas a realizar se tuvo en cuenta la pirámide de Cohn, se decidió que las pruebas a realizar son las del tercer nivel, de forma que serán de tipo funcionales automatizadas y validarán de extremo a extremo el comportamiento de la plataforma en la interacción con el usuario.



Pirámide de Cohn mostrando sus tres niveles

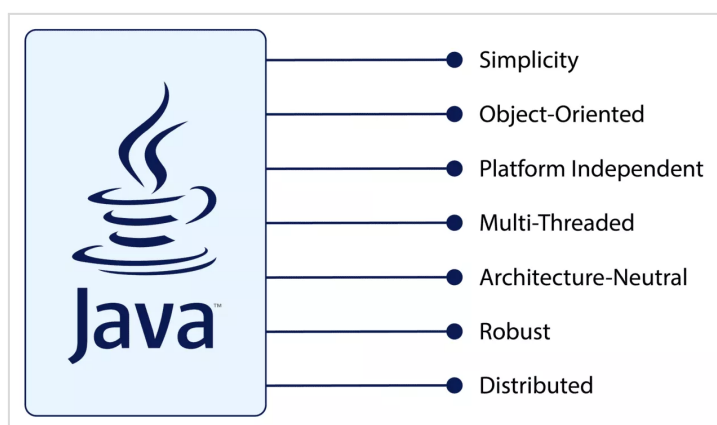
5.2.2. Tecnologías a usar

Para realizar unas pruebas funcionales automatizadas de calidad las tecnologías y herramientas seleccionadas debían cumplir con los siguientes requisitos:

- El lenguaje de programación debía permitir el paradigma de orientación a objetos
- El framework de pruebas debía ser compatible con el lenguaje de programación seleccionado
- El framework de pruebas debía ser compatible con el estilo BDD y el patrón ScreenPlay
- El framework de pruebas debía contar con soporte para algún empaquetador del lenguaje de programación seleccionado
- El framework de pruebas debía ser compatible con el lenguaje Gherkin, ya que en este lenguaje serían provistos los escenarios por el equipo de negocio

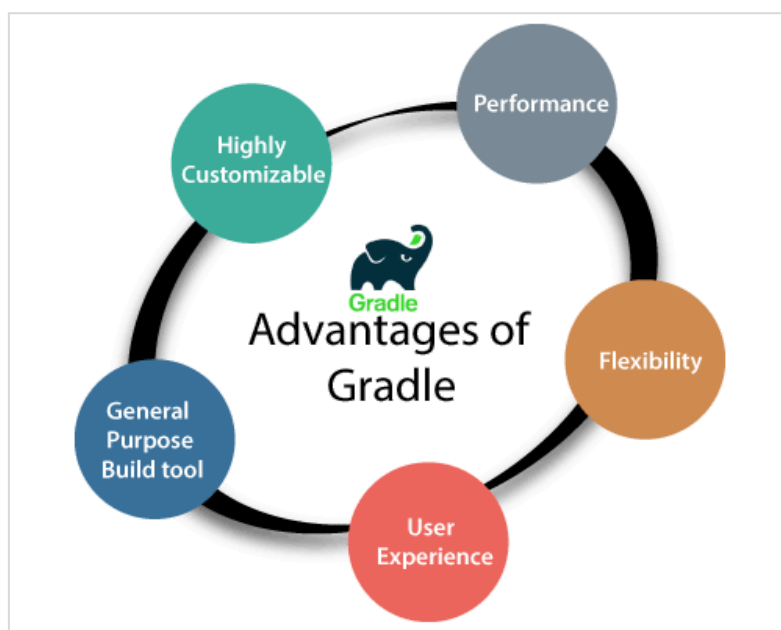
Las siguientes son herramientas y tecnologías que cumplen con estos requisitos y que serán usadas para la realización de las pruebas:

Java 11: Al tener soporte para el paradigma de programación orientada a objetos, ser multiplataforma, contar con una amplia documentación y una variedad de frameworks de pruebas se designa como lenguaje de programación.



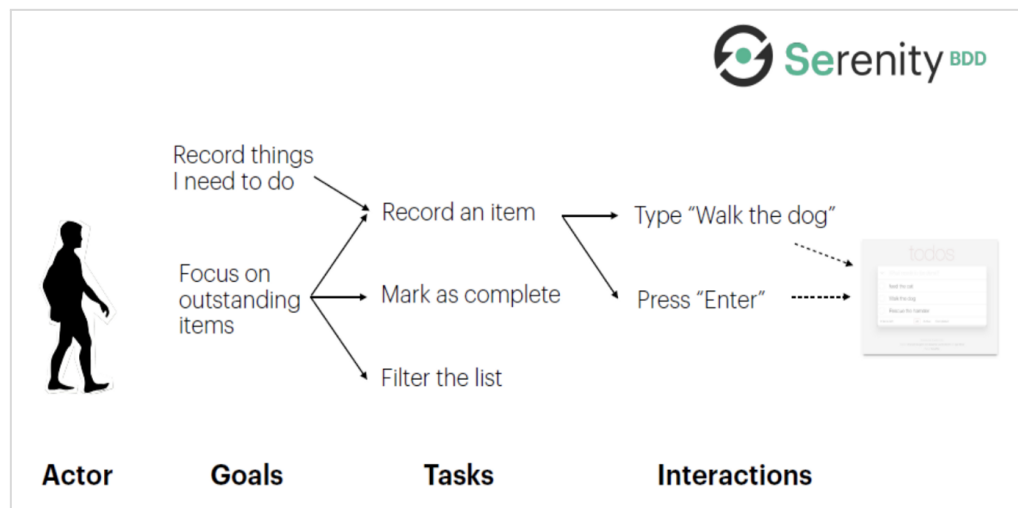
Algunas de las características principales de Java

Gradle 7.6: Por ser ampliamente extendido entre las librerías y herramientas del ecosistema de Java, fácil de implementar y poderoso para tareas de compilación y empaquetado



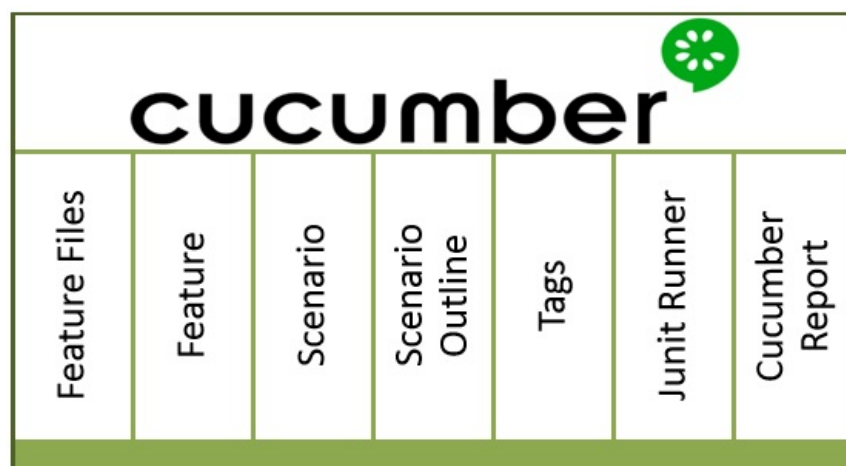
Algunas ventajas de Gradle

Serenity BDD 3.5.0: Este framework de pruebas ha sido seleccionado porque cumple con todos los requisitos mencionados anteriormente, además que cuenta con poderosas herramientas para generación de reportes.



Secciones del patrón ScreenPlay junto a Serenity BDD

Cucumber 3.5.0: Para poder comprender y definir en pasos las historias escritas en Gherkin se usará esta herramienta, que integrada con Serenity provee una enorme cantidad de implementaciones al lenguaje para realizar pruebas automatizadas de alta calidad.



Cucumber y sus características

5.2.3. Escenarios en Gherkin a validar

Las siguientes son las historias que comparte el equipo de negocio para cada funcionalidad y sus escenarios, estas historias tienen una sintaxis Gherkin y serán usadas para la implementación de los pasos en las pruebas automatizadas.

Autenticación de usuario

Feature: Login to SwagLabs dashboard

Scenario Outline: Login with valid credentials to <type> user

Given Marianella is on login page

When she enter "<username>" as username

And she enter "secret_sauce" as password

And she clicks the login button

Then she should be able to see the "<message>"

Examples:

| type | username | message |

Scenario Outline: Login with invalid credentials to <type> user

Given Marianella is on login page

When she enter "<username>" as username

And she enter "<password>" as password

And she clicks the login button

Then she will receive the error message "<message>"

Examples:

| type | username | password | message |

Carrito de compra

Feature: Add article to cart

Scenario Outline: Add article to cart

Given Marianella is on login page

When she enter "<username>" as username

And she enter "<password>" as password

And she clicks the login button

And she clicks on "<article>" article

And she clicks on add to cart button

And she clicks on cart icon

Then she should be to see her cart

And she should be to see "<article>" article

Examples:

| username | password | article |

Confirmación de pedido

Feature: Confirm purchase of items added to cart

Scenario Outline: Confirm the purchase of items added to the cart by adding correct shipping information

Given Marianella is on login page
When she enter "<username>" as username
And she enter "<password>" as password
And she clicks the login button
And she clicks on "<article>" article
And she clicks on add to cart button
And she clicks on cart icon
And she clicks on checkout button
And she enter "<first_name>" as first name
And she enter "<last_name>" as last name
And she enter "<postal_code>" as postal code
And she continue to confirm purchase
And she finish the purchase
Then she should be able to see the "<message>"

Examples:

username	password	article	first_name	last_name	postal_code	message

Scenario Outline: Try to confirm the purchase without entering the <type_field> in the shipping information

Given Marianella is on login page
When she enter "standard_user" as username
And she enter "secret_sauce" as password
And she clicks the login button
And she clicks on "Sauce Labs Fleece Jacket" article
And she clicks on add to cart button
And she clicks on cart icon
And she clicks on checkout button
And she enter "<first_name>" as first name
And she enter "<last_name>" as last name
And she enter "<postal_code>" as postal code
And she continue to confirm purchase
Then she will receive the error message "Error: <empty_field> is required"

Examples:

type_field	first_name	last_name	postal_code	empty_field

En estos escenarios existe una tabla con los ejemplos para cada caso de variables, tal es el caso de los diferentes usuarios que van a probarse en el sistema, durante la construcción de las pruebas estas tablas serán nutridas para su correcta ejecución.

5.3. Reporte

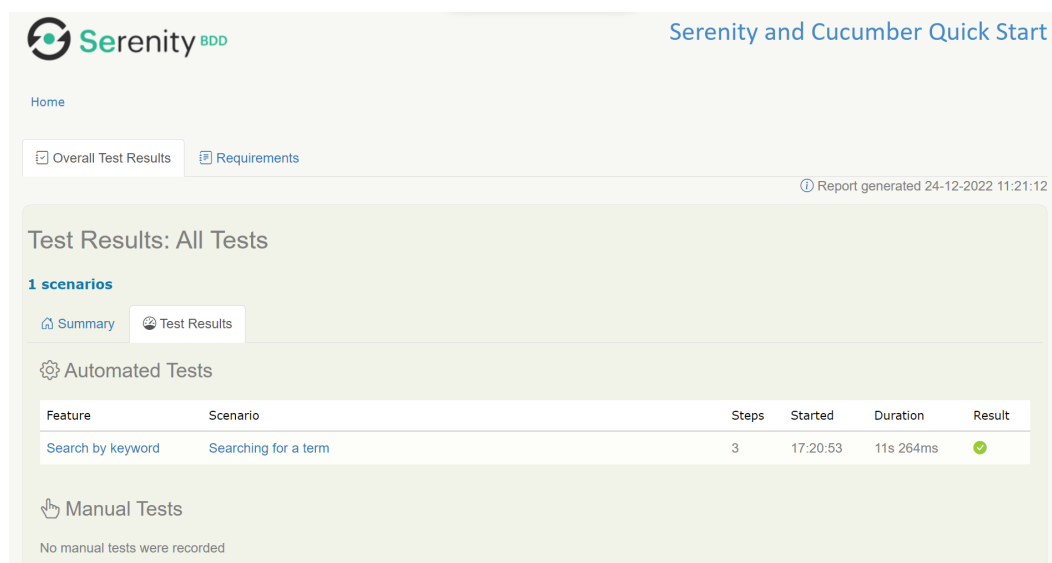
Serenity BDD permite construir reportes completos de cada prueba realizada, estos se construyen luego de ejecutar el comando del punto anterior y permiten desde el resultado de cada escenario probado hasta el paso a paso de las pruebas que se han construido. Para acceder a este reporte solo falta ubicarse en la siguiente ruta:

```
~my-project/target/site/serenity/index.html
```

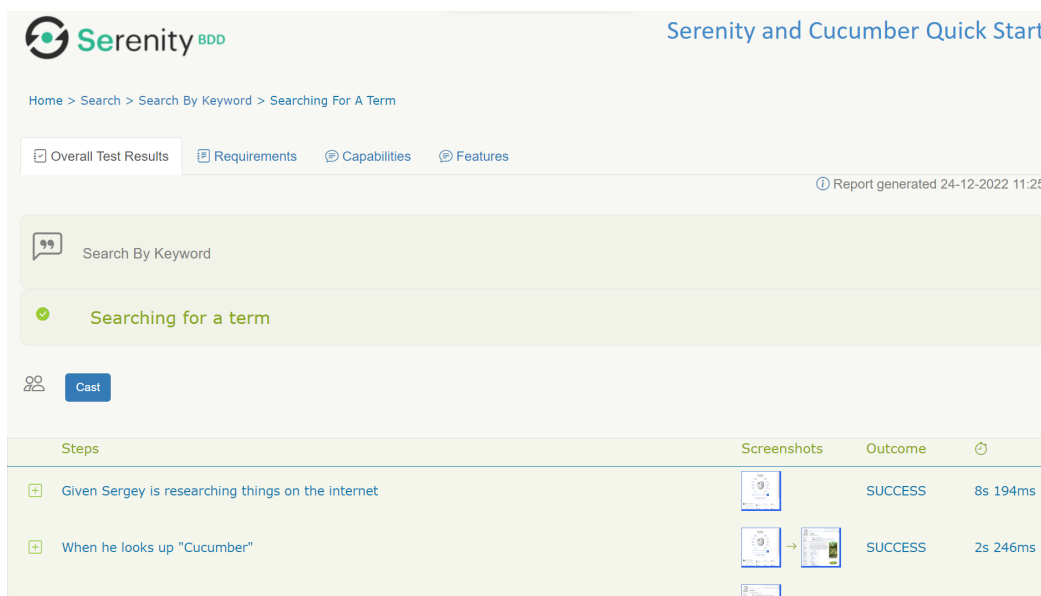
Esta es el reporte generado:



Página principal del reporte



Resultado de las pruebas



Paso a paso del resultado de las pruebas

6. Entregables de pruebas

Este proyecto de pruebas automatizadas genera 4 entregables que son los siguientes:

- **Plan de pruebas:** Corresponde a este documento con la información detallada de las pruebas que serán realizadas.
- **Matriz de riesgos:** Corresponde a una hoja de cálculo donde se plasma la matriz de riesgo con sus niveles por probabilidad e impacto y un listado de riesgos identificados.
- **Repositorio del proyecto:** Corresponde a la implementación en código de las pruebas diseñadas en el presente documento. Se entregará un enlace al repositorio en GitHub donde podrá disponerse libremente de él.
- **Reporte de resultado de pruebas:** Corresponde a un documento con información detallada del resultado de las pruebas para cada funcionalidad así como de unas conclusiones que podrán ser tomadas en cuenta para la corrección o evolución de la plataforma.

7. Matriz de riesgos

Para identificar aquellos elementos que podrían poner en riesgo las pruebas o el funcionamiento normal del sistema se adjunta una matriz de riesgos, esta contiene riesgos que podrían presentarse en cualquiera de las tres capas del proyecto: dominio, aplicación e infraestructura.

La siguiente tabla muestra los niveles de riesgo contemplados en la matriz, de esta manera no solo podrá entenderse el nivel, sino también su prioridad junto a otros riesgos del mismo tipo.

		Impacto				
		Insignificante 1	Menor 2	Significativo 3	Mayor 4	Severo 5
Probabilidad	Casi seguro 5	Medio 5	Alto 10	Muy alto 15	Extremo 20	Extremo 25
	Probable 4	Medio 4	Medio 8	Alto 12	Muy alto 16	Extremo 20
	Moderado 3	Bajo 3	Medio 6	Medio 9	Alto 12	Muy alto 15
	Poco probable 2	Muy bajo 2	Bajo 4	Medio 6	Medio 8	Alto 10
	Raro 1	Muy bajo 1	Muy bajo 2	Bajo 3	Medio 4	Medio 5

Niveles de probabilidad e impacto en la matriz de riesgo

Este es el listado enlazado de los riesgos, para más detalle sobre ellos debe consultar el documento de matriz de riesgos.

Riesgo
Cambio en el modelo de datos del sistema
Cambio en los escenarios requeridos de prueba
Cambio en las opciones de autenticación de usuarios
Cambios en la estructura visual (FrontEnd) de la aplicación
Problemas en el motor de base de datos del sistema
Cambios en las API's de terceros para envíos y pagos
Cambios en la legislación comercial
Indisponibilidad de las instancias de aplicación y base de datos
Vulnerabilidades encontradas en frameworks o librerías
Vencimiento de licencias y dominios