UNIVERSIDAD DE EL SALVADOR

FACULTAD DE INGENIERÍA Y ARQUITECTURA

ESCUELA DE INGENIERÍA DE SISTEMAS INFORMÁTICOS

CURSO DE ESPECIALIZACIÓN EN INGENIERÍA DE LA CALIDAD

IMPLEMENTACIÓN DE PROCESO DE PRUEBAS AL SITIO WEB CONDUITAPP Y EJECUCIÓN DE PRUEBAS AUTOMATIZADAS EN HERRRAMIENTA KARATE FRAMEWORK

PRESENTADO POR:

CARPIO CARDONA, FÉLIX ALFREDO

CASTRO ECHEVERRÍA, ELSY MARGOTH

VILLEDA ROMERO, ALFREDO ALEXANDER

PARA OPTAR AL TÍTULO DE:

INGENIERO(A) DE SISTEMAS INFORMÁTICOS

Docente Asesor:

ING. CARLOS EDUARDO CONTRERAS GÁLVEZ

CUIDAD UNIVERSITARIA, FEBRERO DE 2023

UNIVERSIDAD DE EL SALVADOR

RECTOR:

MSc. ROGER ARMANDO ARIAS ALVARADO

SECRETARIO GENERAL:

ING. FRANCISCO ANTONIO ALARCON SANDOVAL

FACULTAD DE INGENIERIA Y ARQUITECTURA

DECANO:

PhD. EDGAR ARMANDO PEÑA FIGUEROA

SECRETARIO:

ING. JULIO ALBERTO PORTILLO

ESCUELA DE INGENIERIA DE SISTEMAS INFORMATICOS

DIRECTOR:

ING. RUDY WILFREDO CHICAS VILLEGAS

UNIVERSIDAD DE EL SALVADOR

FACULTAD DE INGENIERIA Y ARQUITECTURA

ESCUELA DE INGENIERIA DE SISTEMAS INFORMATICOS

Curso de Especialización previo a la opción al Grado de:

INGENIERO(A) DE SISTEMAS INFORMATICOS

CURSO DE ESPECIALIZACIÓN EN INGENIERÍA DE LA CALIDAD

IMPLEMENTACIÓN DE PROCESO DE PRUEBAS AL SITIO WEB CONDUITAPP Y EJECUCIÓN DE PRUEBAS AUTOMATIZADAS EN HERRRAMIENTA KARATE FRAMEWORK

Presentado por:

CARPIO CARDONA, FÉLIX ALFREDO

CASTRO ECHEVERRÍA, ELSY MARGOTH

VILLEDA ROMERO, ALFREDO ALEXANDER

Curso de Especialización aprobado por:

Docente Asesor:

ING. CARLOS EDUARDO CONTRERAS GÁLVEZ

SAN SALVADOR, FEBRERO 2023

Curso de Especialización aprobado por:

Docente Asesor:

ING. CARLOS EDUARDO CONTRERAS GÁLVEZ

Contenido

Introducción

El presente documento ha sido elaborado con el propósito de dar a conocer las diferentes metodologías y tipos de pruebas para aplicarlos a un caso práctico en particular, que es ConduitApp, sitio que es un espacio para que los usuarios pueden compartir con otros, pensamientos y conocimiento a través de sus publicaciones. Su objetivo es ser el medio en el que las personas interactúan a través de publicaciones y comentarios sobre las mismas.

La elaboración de un plan de pruebas resulta de mucho valor al equipo de pruebas, ya que es una guía para documentar correctamente los casos y escenarios de prueba. Por lo que, se ha elaborado un plan de ejecución de las pruebas para tener un mayor control respecto a las fechas y los avances de cada uno de los miembros del equipo. Se realizan pruebas funcionales para verificar que la API se encuentra disponible; pruebas de seguridad para garantizar que se utiliza algún tipo de token y que solo se permita el ingreso a los usuarios registrados.

Para el desarrollo de las pruebas automatizadas se ha preparado el ambiente con anticipación que y se han diseñado casos de prueba para obtener resultados de mayor calidad. Como software de pruebas automatizadas, se utiliza Karate que es un framework desarrollado en java, el cual permite la automatización de pruebas de APIs y de rendimiento.

Además de Karate, se han configurado otras herramientas como prerrequisito, Java 8 o superior, Maven, Visual Studio Code, con sus extensiones: Cucumber (Gherkin) Full Support, Karate Runner, Java Extension Pack y Postman que se utiliza como apoyo para realizar algunas pruebas manuales, previo a la automatización para garantizar una mayor cobertura en las pruebas. Como ambiente de pruebas se tienen los navegadores, Google Chrome y Mozilla Firefox en sistema operativo Windows.

Una de las bondades de este framework es que la sintaxis utilizada para los escenarios es BDD (Behavior Driven Development), con lenguaje Gherkin, un DSL (Domain Specific Languaje). Con la creación de escenarios de pruebas que pueden ser ejecutados muchas veces, se está ahorrando tiempo en la ejecución de las pruebas y en la generación de informes, ya que luego de correr un caso de prueba, se genera el reporte de Cucumber. de los resultados obtenidos.

Se trabaja en la identificación de riesgos y el plan de contingencia para cada uno, pues no se puede garantizar que no habrá dificultades durante el diseño y ejecución de las pruebas. Así mismo, se da seguimiento a los resultados de pruebas, con el monitoreo y se realiza la administración de defectos, respetando las categorías de prioridad baja, prioridad media y alta

Objetivos

Objetivo General

- Implementar el proceso de pruebas para el sitio de ConduitApp con los apartados vistos durante la especialización, realizando una investigación de la herramienta Karate Framework, para la ejecución de pruebas automatizadas en el sitio según los casos de prueba que se estimen y prioricen por el equipo.

Objetivos Específicos

- Desarrollar un marco teórico acerca del proceso de pruebas que se implementará en este proyecto, detallando cada sección con sus conceptos respectivos para dar a conocer lo realizado en nuestro plan y estrategia de pruebas.
- Realizar pruebas exploratorias en el sitio de ConduitApp para definir las funcionalidades principales y crear las historias de usuario necesarias para poder puntuarlas según su importancia.
- Documentar el proceso de pruebas realizado por el equipo con los apartados más importantes, haciendo el análisis respectivo en cada sección para dar contexto de todo el proceso desarrollado.
- Investigar la herramienta asignada llamada Karate Framework para dar un marco teórico del proceso de automatización de pruebas y las ventajas que esta trae para un proyecto.
- Crear los casos de pruebas priorizados por el equipo según el análisis realizado con respecto a las historias de usuario principales para su futura automatización con la herramienta Karate Framework.
- Codificar los casos de pruebas documentados que han sido priorizados según el análisis realizado anteriormente para poder ser ejecutados con la herramienta utilizada de Karate Framework.
- Analizar los resultados de la ejecución las pruebas automatizadas con Karate Framework y de los reportes que genera dicha herramienta para dar las conclusiones como equipo acerca del uso de la automatización en el proyecto.

Planteamiento del Problema

La calidad del software es un factor crítico en el desarrollo de un proyecto de software exitoso. Una forma de asegurar la calidad del software es mediante la implementación de pruebas de software. Las pruebas son un proceso sistemático de evaluación y verificación del software para detectar errores, defectos y problemas de rendimiento. Sin embargo, muchas veces se descuida la implementación de pruebas debido a la falta de tiempo y recursos.

ConduitApp es un proyecto de desarrollo de una plataforma de publicación en línea especializada en la publicación de blogs, artículos y ensayos en una variedad de categorías. El sitio permite agregar etiquetas a los contenidos publicados, de manera que se pueda buscar contenido por medio de etiquetas. Además, cuenta con un sistema de recomendación de contenido para ayudar a los lectores a descubrir historias relevantes y de interés, y un sistema de comentarios y retroalimentación para fomentar la discusión y la colaboración.

La necesidad de implementar un proceso de pruebas en un proyecto de desarrollo del software del sitio ConduitApp, para garantizar la calidad del software y asegurar que se cumplan los requisitos del sistema. El sitio web de ConduitApp es un proyecto complejo que requiere una gran cantidad de pruebas para garantizar la calidad del software y asegurar que se cumplan los requisitos del sistema. Sin embargo, a menudo se descuida la implementación de pruebas debido a la falta de tiempo y recursos. Es importante investigar cómo implementar de manera efectiva un proceso de pruebas para garantizar la calidad del software en el proyecto de desarrollo del sitio web ConduitApp. Esto permitiría detectar problemas temprano en el proceso de desarrollo y reducir los costos, mejorando la experiencia del usuario y la satisfacción del cliente.

Importancia

El desarrollo de pruebas de software es de vital importancia en el ciclo de vida de cualquier proyecto informático, pues permite la detección temprana de errores y la reducción de los mismos para mejorar la calidad del producto que se está construyendo; si estas se realizan desde etapas tempranas del proyecto, es decir, desde el levantamiento de requerimientos.

Sin embargo, como lo dice el séptimo principio de las pruebas, “la ausencia de errores es una falacia”, en ese sentido, lo que se espera con la elaboración de la estrategia de pruebas y el plan de pruebas para ConduitApp, sitio seleccionado para la realización de las pruebas, es la reducción de defectos tanto en los requerimientos, diseño o desarrollo. Documentando a su vez, los casos de prueba a ejecutar y el seguimiento de los defectos que se reportan.

Con el apoyo de las herramientas necesarias y un equipo de pruebas con sus roles definidos se facilita el aseguramiento de la calidad del producto a entregar. Para el presente proyecto, se ha elaborado lo que es una estrategia de pruebas, un plan de pruebas y pruebas automatizadas en Karate Framework, una herramienta enfocada principalmente en las pruebas de API.

Las pruebas automatizadas en Karate, suman valor al proyecto, ya que la acción de crear escenarios de pruebas y guardarlos para ejecutarlos en cualquier momento del día, sin tener que escribirlos una y otra vez, ahorra tiempo y esfuerzo por parte del equipo de pruebas. Lo importante es saber definir qué casos de prueba son factibles para pruebas manuales o automatizadas, para tener la certeza de que se puede probar si el software cumple con los requerimientos del cliente y reducir el riesgo de encontrar defectos en producción.

Justificación

El proyecto que se plasma en el presente documento fue realizado por el equipo que ha cursado la especialización de Calidad, la cual tiene como finalidad obtener conocimientos en el área específica de calidad de software durante todo el desarrollo del curso. Dichos conocimientos tienen como objetivo principal sentar una base para poder desenvolverse y aplicar todos los conceptos vistos ya en un área profesional.

Para cumplir este objetivo, el equipo ha elaborado este proyecto para plasmar dicho conocimiento, principios y conceptos vistos para reforzarlos en un hipotético desarrollo de un sistema. A su vez, también el equipo utilizó los principios de la metodología SCRUM para poder llevar a cabo el desarrollo de las fases del proceso de pruebas. También el equipo aplicó las partes principales del curso como lo son la estrategia y el plan de pruebas que son los artefactos más importantes en el área de calidad y los que más se utilizan.

Aparte de lo antes mencionado, al equipo se le asignó una herramienta para ser investigada la cual es Karate Framework. Con esta herramienta se completaría una fase de investigación y una fase de aplicación dentro del proyecto.

Con estos puntos, se justifica el proyecto ya que en él se aplica tanto la parte realizada en el curso de especialización como en el uso de la herramienta asignada por parte del equipo.

Limitaciones

Dentro del desarrollo del proyecto, las limitaciones más importantes son las siguientes:

- El poco conocimiento previo a el curso acerca del área de calidad para poder desarrollar de mejor manera los temas.
- El equipo ha utilizado la herramienta de Karate Framework por primera vez y debe haber una etapa de investigación previa, igualmente del manejo y conceptos de automatización.
- El equipo no cuenta con una infraestructura y con todos los roles del equipo SCRUM completos para simular completamente el manejo de un proyecto real.
- En relación al manejo de pruebas con API, el equipo ha tenido conocimientos básicos a nivel académico.

Alcances

Durante la especialización de Calidad, se han desarrollados todos los temas relacionados al área de QA, los fundamentos de ISTQB y fundamentos básicos para la automatización que estaban planeados en el cronograma del curso. Para la realización de este proyecto, el equipo aplicó los conceptos vistos anteriormente, y ha documentado todas las partes y puntos más importantes que una persona del área de Calidad debe conocer para desenvolverse en este ámbito. En dicho sentido, el equipo ha tomado como referencia un sitio web ConduitApp que es de fácil acceso y uso libre para implementar la teoría de la estrategia y plan de pruebas.

Como alcance en este punto, se menciona principalmente que el equipo no ha tenido contacto con los desarrolladores o personas encargadas de la implementación y mantenimiento del sitio. Éste se ha tomado únicamente para fines académicos para aplicar la estrategia y plan de pruebas. Por esta razón, durante el desarrollo del proyecto se ha documentado el plan y estrategia de pruebas en base a pruebas exploratorias que el equipo ejecutó en el sitio ConduitApp para poder sacar las funcionalidades principales y a partir de estas, el equipo ha creado las historias de usuario para poder realizar las pruebas necesarias.

El equipo ha creado los escenarios y casos de prueba respectivos, se ha documentado toda la estrategia y plan de pruebas agregando los puntos principales tratados en la especialización como los tipos de prueba, niveles de prueba, herramientas para documentar y ejecución de pruebas.

El equipo ha elegido dicha página para poder también desarrollar la parte de automatización. Se investigó la herramienta que fue asignada durante el curso de especialización llamada Karate Framework. Se ha realizado la investigación respectiva y se ha desarrollado algunas pruebas automatizadas para aplicar los conceptos de dicha herramienta. El alcance de la automatización ha sido codificar y ejecutar algunos casos de pruebas siempre aplicando los conceptos vistos en la investigación.

Finalmente, a modo de resumen, como alcance nosotros como equipo tenemos los puntos antes mencionados:

- Desarrollo de estrategia y plan de pruebas
- Aplicación de conceptos de calidad vistos en el curso
- Investigación de herramienta de automatización
- Codificación y ejecución de pruebas automatizadas con la herramienta

Marco teórico

Qué es poner a prueba

La prueba de software es el proceso de evaluar y verificar que un producto o aplicación de software hace lo que se supone que debe hacer. Los beneficios de las pruebas incluyen la prevención de errores, la reducción de los costos de desarrollo y la mejora del rendimiento (Certified Tester Foundation Level Syllabus, 2018, pág. 13).

Proceso de prueba

No existe un proceso de prueba de software universal, pero existen conjuntos comunes de actividades de prueba sin las cuales la prueba tendrá menos probabilidades de alcanzar sus objetivos establecidos. Estos conjuntos de actividades de prueba constituyen un proceso de prueba. El proceso de prueba de software adecuado y específico en cualquier situación depende de muchos factores. Las actividades de prueba que están involucradas en este proceso de prueba, la manera en que se implementan estas actividades y cuándo ocurren, pueden discutirse en la estrategia de prueba de una organización.

Los factores contextuales que influyen en el proceso de prueba para una organización incluyen, entre otros, los siguientes:

- Modelo de ciclo de vida de desarrollo de software y metodologías de proyectos que se utilizan
- Los niveles y los tipos de prueba que se tienen en cuenta
- Riesgos del producto y del proyecto
- Dominio empresarial
- Las limitaciones operacionales, que incluyen, pero no se limitan a:
    - Presupuestos y recursos
    - Plazos de entrega
    - Complejidad
    - Requisitos contractuales y regulatorios
- Políticas y prácticas organizativas
- Normas internas y externas requeridas

La norma ISO (ISO/IEC/IEEE 29119-2) tiene más información sobre estos procesos de prueba (Certified Tester Foundation Level Syllabus, 2018, págs. 17,18)

Productos de trabajo de la prueba

La prueba de productos de trabajo de se crea como parte del proceso de prueba. Así como existe una variación significativa en la forma en que las organizaciones implementan el proceso de prueba, también existe una variación significativa en los tipos de productos de trabajo creados durante ese proceso, en la forma en que esos productos de trabajo se organizan y administran, y en los nombres que se usan para los mismos.

La norma ISO (ISO/IEC/IEEE 29119-3) también puede servir como una guía para la prueba de productos de trabajo (Certified Tester Foundation Level Syllabus, 2018, pág. 22).

Las pruebas durante el ciclo de vida de software

Un modelo de ciclo de vida de desarrollo de software describe los tipos de actividad que se realizan en cada etapa de un proyecto de desarrollo de software, y cómo las actividades se relacionan entre sí de manera lógica y cronológica. Hay una serie de modelos diferentes de ciclo de vida de desarrollo de software, cada uno de los cuales requiere diferentes enfoques para la prueba (Certified Tester Foundation Level Syllabus, 2018, pág. 28).

Una parte importante del papel de un probador es familiarizarse con los modelos comunes de ciclo de vida de desarrollo de software para que puedan llevarse a cabo las actividades de prueba adecuadas.

Independientemente del modelo de ciclo de vida de desarrollo de software que se elija, las actividades de prueba deben comenzar en las primeras etapas del ciclo de vida, siguiendo el principio de pruebas de pruebas temprana. En el programa de Probador certificado a nivel básico de ISQTB se clasifican como modelos comunes de ciclo de vida de desarrollo de software los siguientes:

- Modelos de desarrollo secuencial
- Modelos de desarrollo iterativo e incremental

Niveles de prueba

Los niveles de prueba son grupos de actividades de prueba que se organizan y administran juntas. Los niveles de prueba están relacionados con otras actividades dentro del ciclo de vida del desarrollo de software.

Los niveles de prueba utilizados en el programa de tester nivel básico son:

- Pruebas de componente
- Pruebas de integración
- Pruebas de sistema
- Pruebas de aceptación

Los niveles de prueba se caracterizan por los siguientes atributos:

- Objetivos específicos
- Base de prueba, referenciada para derivar casos de prueba
- Objetos de prueba (es decir, lo que se está probando).
  Defectos y fallos típicos.
- Enfoques y responsabilidades específicas

Para cada nivel de prueba, se requiere un entorno de prueba adecuado. En las pruebas de aceptación, por ejemplo, un entorno de prueba similar a la producción es ideal, mientras que en las

pruebas de componente los desarrolladores suelen utilizar su propio entorno de desarrollo (Certified Tester Foundation Level Syllabus, 2018, pág. 30).

Pruebas de componentes

Las pruebas de componente (también conocidas como pruebas de unidad o módulo) se enfocan en componentes que se pueden probar por separado. Los objetivos de las pruebas de componente incluyen:

- Reducir el riesgo
- Verificar si los comportamientos funcionales y no funcionales del componente son como se diseñaron y especificaron.
- Crear confianza en la calidad del componente
- Encontrar defectos en el componente
- Prevenir que los defectos escapen a niveles de prueba más elevados

En algunos casos, especialmente en los modelos de desarrollo incrementales e iterativos (p. ej., Ágil) donde los cambios de código son continuos, las pruebas de regresión automatizada de componentes desempeñan un papel clave en crear confianza en que los cambios no han roto los componentes existentes.

Las pruebas de componente a menudo se realizan en forma aislada del resto del sistema, según el modelo de ciclo de vida del desarrollo del software y el sistema, que puede requerir objetos simulados, virtualización del servicio, arneses, stubs y controladores (Certified Tester Foundation Level Syllabus, 2018, pág. 31).

Pruebas de integración

Las pruebas de integración se centran en las interacciones entre componentes o sistemas. Los objetivos de las pruebas de integración incluyen:

- Reducir el riesgo.
- Verificar si los comportamientos funcionales y no funcionales del componente son como se diseñaron y especificaron.
- Crear confianza en la calidad de las interfaces.
- Encontrar defectos (que pueden estar en las interfaces o en los componentes o sistemas).
- Prevenir que los defectos escapen a niveles de prueba más elevados.

Al igual que con las pruebas de componente, en algunos casos, las pruebas de regresión de integración automatizadas brindan confianza en que los cambios no han roto las interfaces, componentes o sistemas existentes (Certified Tester Foundation Level Syllabus, 2018, pág. 32).

Pruebas de sistema

Las pruebas de sistema se centran en el comportamiento y las capacidades de todo un sistema o producto, a menudo considerando las tareas de extremo a extremo que el sistema puede realizar y los comportamientos no funcionales que muestra al realizar esas tareas. Los objetivos de las pruebas de sistema incluyen:

- Reducir el riesgo.
- Verificar si los comportamientos funcionales y no funcionales del sistema son como se diseñaron y especificaron.
- Validar que el sistema está completo y funcionará como se espera.
- Crear confianza en la calidad del sistema como un todo.
- Encontrar defectos.
- Prevenir que los defectos escapen a niveles de prueba o producción más elevados.

Las pruebas del sistema a menudo producen información que usan las partes interesadas para tomar decisiones de entrega. Las pruebas de sistema también pueden satisfacer requisitos o normas legales o reglamentarios (Certified Tester Foundation Level Syllabus, 2018, págs. 34,35).

Pruebas de aceptación

Las pruebas de aceptación, como las pruebas de sistema, generalmente se enfocan en el comportamiento y las capacidades de todo un sistema o producto. Los objetivos de las pruebas de aceptación incluyen:

- Crear confianza en la calidad del sistema como un todo.
- Validar que el sistema está completo y funcionará como se espera.
- Verificar que los comportamientos funcionales y no funcionales del sistema son los especificados.

Las pruebas de aceptación pueden generar información para evaluar la preparación del sistema para la implementación y el uso por parte del cliente (usuario final). Los defectos se pueden detectar durante las pruebas de aceptación, pero detectar defectos a menudo no es un objetivo, y encontrar un número significativo de defectos durante las pruebas de aceptación en algunos casos puede considerarse un riesgo importante para el proyecto. (Certified Tester Foundation Level Syllabus, 2018, pág. 36).

Tipos de pruebas (Certified Tester Foundation Level Syllabus, 2018, págs. 39-41)

Pruebas funcionales

Las pruebas funcionales de un sistema involucran pruebas que evalúan las funciones que debe realizar el sistema. Los requisitos funcionales pueden describirse en productos de trabajo, como son especificaciones de requisitos comerciales, epopeyas, historias de usuario, casos de uso o especificaciones funcionales, o pueden no estar documentados. Las funciones son “lo que” el sistema debe hacer.

Pruebas no funcionales

Las pruebas no funcionales de un sistema evalúan las características de los sistemas y el software, como son la usabilidad, la eficiencia del rendimiento o la seguridad. Consulte la norma ISO (ISO/IEC 25010) para obtener una clasificación de las características de calidad del producto de software. La prueba no funcional es la prueba de “qué tan bien” se comporta el Sistema.

Pruebas de caja blanca

Las pruebas de caja blanca derivan pruebas basadas en la estructura interna o la implementación del sistema). La estructura interna puede incluir código, arquitectura, flujos de trabajo y/o flujos de datos dentro del sistema.

El diseño y la ejecución de pruebas de caja blanca pueden involucrar habilidades o conocimientos especiales, como la forma en que se construye el código (p. ej., para usar herramientas de cobertura de códigos), cómo se almacenan los datos (p. ej., para evaluar posibles consultas de bases de datos) y cómo usar herramientas de cobertura e interpretar correctamente sus resultados.

Pruebas relacionadas al cambio

Cuando se realizan cambios en un sistema, ya sea para corregir un defecto o debido a una funcionalidad nueva o cambiante, se deben realizar pruebas para confirmar que los cambios han corregido el defecto o implementado la funcionalidad correctamente, y no han causado consecuencias adversas imprevistas.

Pruebas estáticas (Certified Tester Foundation Level Syllabus, 2018, pág. 46)

En contraste con las pruebas dinámicas, que requieren la ejecución del software que se está probando, las pruebas estáticas se basan en el examen manual de los productos de trabajo (es decir, revisiones) o la evaluación del código u otros productos de trabajo (es decir, el análisis estático). Ambos tipos de pruebas estáticas evalúan el código u otro producto de trabajo que se está probando sin ejecutar realmente el código o el producto de trabajo que se está probando.

Técnicas de prueba

- **Categorías de técnicas de pruebas**

El propósito de una técnica de prueba, es ayudar a identificar las condiciones de prueba, los casos de prueba y los datos de prueba.

El uso de técnicas de prueba en el análisis de prueba, el diseño de prueba y las actividades de implementación de prueba pueden variar desde muy informales (poca o ninguna documentación) hasta muy formales. El nivel adecuado de formalidad depende del contexto de las pruebas, incluida la madurez de los procesos de prueba y desarrollo, las limitaciones de tiempo, la seguridad o los requisitos reglamentarios, el conocimiento y las habilidades de las personas involucradas y el modelo de ciclo de vida del desarrollo de software que se está siguiendo (Certified Tester Foundation Level Syllabus, 2018, pág. 56).

- **Técnicas de prueba de caja negra**

Pruebas que analizan la funcionalidad del software, sin conocer la estructura interna.

- **Segmentación de equivalencia**

La segmentación de equivalencia divide los datos en segmentos (también conocidos como clases de equivalencia) de tal manera que se espera que todos los miembros de un segmento dado se

procesen de la misma manera. Existen segmentaciones de equivalencia para valores válidos y no válidos. (Certified Tester Foundation Level Syllabus, 2018, pág. 58)

- **Técnicas de prueba de caja blanca**

La prueba de caja blanca se basa en la estructura interna del objeto de prueba. Las técnicas de prueba de caja blanca se pueden usar en todos los niveles de prueba, pero las dos técnicas relacionadas con el código que se analizan en esta sección son las más utilizadas en el nivel de prueba de componentes. (Certified Tester Foundation Level Syllabus, 2018, pág. 61)

- **Técnicas de prueba basadas en la experiencia**

Al aplicar técnicas de prueba basadas en la experiencia, los casos de prueba se derivan de la habilidad y la intuición del probador, y de su experiencia con aplicaciones y tecnologías similares. Estas técnicas pueden ser útiles para identificar pruebas que no se identificaron fácilmente con otras técnicas más sistemáticas. (Certified Tester Foundation Level Syllabus, 2018, pág. 61)

Gestión de la prueba. Planificación y Estimación de la prueba

(Certified Tester Foundation Level Syllabus, 2018, págs. 67,68)

- **Propósito y contenido de un plan de prueba**

Un plan de prueba describe las actividades de prueba para proyectos de desarrollo y mantenimiento. La planificación está influenciada por la política de prueba y la estrategia de prueba de la organización, los ciclos de vida y los métodos de desarrollo que se utilizan, el alcance de las pruebas, los objetivos, los riesgos, las limitaciones, la criticidad, la capacidad de ser probado y la disponibilidad de recursos.

A medida que avanza la planificación del proyecto y la prueba, se dispone de más información y se pueden incluir más detalles en el plan de prueba. La planificación de pruebas es una actividad continua y se realiza a lo largo del ciclo de vida del producto.

- **Estrategia de prueba y enfoque de la prueba**

Una estrategia de prueba proporciona una descripción general del proceso de prueba, generalmente a nivel de producto u organización. Los tipos comunes de estrategias de prueba incluyen:

- **Analítica:** Este tipo de estrategia de prueba se basa en un análisis de algún factor (p. ej., requisitos or riesgos). Las pruebas basadas en el riesgo son un ejemplo de un enfoque analítico, donde las pruebas se diseñan y priorizan según el nivel de riesgo.
- **Basadas en modelos:** En este tipo de estrategia de prueba, las pruebas se diseñan en función de algún modelo de algún aspecto requerido del producto, como una función, un proceso de negocios, una estructura interna o una característica no funcional (p. ej., confiabilidad).
- **Metódica**: Este tipo de estrategia de prueba se basa en hacer un uso sistemático de un conjunto predefinido de pruebas o condiciones de prueba, como una taxonomía de tipos comunes o probables de fallas, una lista de características de calidad importantes, o el

aspecto y normas de aspecto visual y operacional para aplicaciones móviles o páginas Web para toda la empresa.

- **Compatibles con los procesos (o compatibles con las normas)**: Este tipo de estrategia de prueba implica analizar, diseñar e implementar pruebas basadas en reglas y normas externas, como las especificadas por normas específicas de la industria, por documentación del proceso, mediante la identificación rigurosa y uso de la base de prueba, o por cualquier proceso o norma impuesta a o por la organización.
- **Dirigidas (o consultivas):** Este tipo de estrategia de prueba se basa principalmente en el asesoramiento, la orientación o las instrucciones de las partes interesadas, los expertos en el dominio de negocios o los expertos en tecnología, que pueden estar fuera del equipo de prueba o fuera de la organización.
- **Adversas a la regresión:** Esta estrategia de prueba incluye la reutilización del software de prueba existente (especialmente los casos de prueba y los datos de prueba), la automatización extensiva de las pruebas de regresión y los juegos de prueba estándar.
- **Reactivas:** En este tipo de estrategia de prueba, la prueba es reactiva al componente o sistema que se está probando, y los eventos que ocurren durante la ejecución de la prueba, en lugar de ser planificados previamente (como lo son las estrategias anteriores).

Una estrategia de prueba adecuada se crea a menudo combinando varios de estos tipos de estrategias de prueba. Por ejemplo, las pruebas basadas en el riesgo (una estrategia analítica) se pueden combinar con las pruebas exploratorias (una estrategia reactiva); se complementan entre sí y pueden lograr pruebas más efectivas cuando se usan juntas.

Técnicas de estimación de prueba

Hay una serie de técnicas de estimación utilizadas para determinar el esfuerzo requerido para realizar pruebas adecuadas. Dos de las técnicas más utilizadas son:

- La técnica basada en métricas: Estimar el esfuerzo de prueba basado en métricas de proyectos similares anteriores, o en valores típicos.
- La técnica basada en expertos: La estimación del esfuerzo de prueba en base a la experiencia de los propietarios de las tareas de prueba o de expertos. (Certified Tester Foundation Level Syllabus, 2018, pág. 70).

Monitorización y control de la prueba

El propósito de la monitorización de pruebas es recopilar información y brindar retroalimentación y visibilidad sobre las actividades de prueba. El control de prueba describe cualquier guía o acción correctiva tomada como resultado de la información y las métricas recopiladas y (posiblemente) informadas. Las acciones pueden cubrir cualquier actividad de prueba y pueden afectar cualquier otra actividad del ciclo de vida del software. (Certified Tester Foundation Level Syllabus, 2018, pág. 71)

Gestión de la configuración

El propósito de la gestión de la configuración es establecer y mantener la integridad del componente o sistema, el software de prueba y sus relaciones entre sí a través del ciclo de vida del

proyecto y del producto. Para apoyar adecuadamente las pruebas, la gestión de la configuración puede implicar asegurar lo siguiente:

- Todos los elementos de prueba se identifican de forma única, se controlan las versiones, se rastrean los cambios y se relacionan entre sí.
- Todos los elementos del software de prueba se identifican de forma única, se controlan las versiones, se rastrean los cambios, se relacionan entre sí y se relacionan con las versiones de los elementos de prueba, de modo que la trazabilidad se pueda mantener durante todo el proceso de prueba.
- Todos los documentos y elementos de software identificados se mencionan de forma inequívoca en la documentación de prueba. (Certified Tester Foundation Level Syllabus, 2018, pág. 73)

Los riesgos y las pruebas (Certified Tester Foundation Level Syllabus, 2018, pág. 73)

Definición de riesgo

El riesgo implica la posibilidad de un evento en el futuro que tenga consecuencias negativas. El nivel de riesgo está determinado por la probabilidad del evento y el impacto (el daño) de ese evento.

Riesgos del producto y del proyecto

El riesgo del producto implica la posibilidad de que un producto de trabajo (p. ej., una especificación, componente, sistema o prueba) no logre satisfacer las necesidades legítimas de sus usuarios y/o partes interesadas. Cuando los riesgos del producto se asocian con características de calidad específicas de un producto (p. ej., idoneidad funcional, confiabilidad, eficiencia de rendimiento, facilidad de uso, seguridad, compatibilidad, mantenibilidad y portabilidad), los riesgos del producto también se denominan riesgos de calidad.

Gestión de defectos

Dado que uno de los objetivos de las pruebas es encontrar defectos, se deben registrar los defectos encontrados durante las pruebas. La forma en que se registran los defectos puede variar, según el contexto del componente o sistema que se esté probando, el nivel de prueba y el modelo de ciclo de vida de desarrollo de software. (Certified Tester Foundation Level Syllabus, 2018, pág. 76)

Herramientas de apoyo a las pruebas

Consideraciones sobre las herramientas de prueba

Las herramientas de prueba se pueden usar para apoyar una o más actividades de prueba. Tales herramientas incluyen:

- Herramientas que se utilizan directamente en las pruebas, como las herramientas de ejecución de pruebas y las herramientas de preparación de datos de prueba.

- Herramientas que ayudan a administrar los requisitos, casos de prueba, procedimientos de prueba, guiones de prueba automatizados, resultados de prueba, datos de prueba y defectos, y para informar y monitorizar la ejecución de la prueba.
- Herramientas que se utilizan para la investigación y evaluación.
- Cualquier herramienta que ayude en las pruebas (una hoja de cálculo también es una herramienta de prueba en este sentido). (Certified Tester Foundation Level Syllabus, 2018, pág. 79)

Clasificación de las herramientas de pruebas

Las herramientas se pueden clasificar según varios criterios, como el propósito, la fijación de precios, el modelo de licencia (p. ej., el código comercial o código abierto) y la tecnología utilizada. Las herramientas se clasifican en este programa de acuerdo con las actividades de prueba que apoyan.

Las herramientas se pueden clasificar según varios criterios, como el propósito, la fijación de precios, el modelo de licencia (p. ej., el código comercial o código abierto) y la tecnología utilizada. (Certified Tester Foundation Level Syllabus, 2018, pág. 79)

Herramientas de automatización de pruebas

La simple adquisición de una herramienta no garantiza el éxito. Cada nueva herramienta introducida en una organización requerirá un esfuerzo para lograr beneficios reales y duraderos. Existen beneficios y oportunidades potenciales con el uso de herramientas en las pruebas, pero también existen riesgos. Esto es particularmente cierto en el caso de las herramientas de ejecución de pruebas (que a menudo se conoce como automatización de pruebas). (Certified Tester Foundation Level Syllabus, 2018, pág. 81)

Principios principales para la selección de herramientas, (Certified Tester Foundation Level Syllabus, 2018, pág. 83)

Las principales consideraciones al seleccionar una herramienta para una organización incluyen:

- Evaluar la madurez de una organización, sus fortalezas y debilidades.
- Identificar las oportunidades para un proceso de prueba mejorado apoyado por herramientas.
- Comprender las tecnologías utilizadas por e0l (los) objeto (s) de prueba para seleccionar una herramienta que sea compatible con esa tecnología.
- Las herramientas de compilación e integración continua ya en uso dentro de la organización, para garantizar la compatibilidad e integración de la herramienta.
- Evaluar la herramienta contra requisitos claros y criterios objetivos.
- Tener en cuenta si la herramienta está disponible o no para un período de prueba gratuito (y por cuánto tiempo).
- Evaluación del proveedor (incluidos aspectos de capacitación, soporte y comerciales) o soporte para herramientas no comerciales (p. ej., de código abierto).

- Identificar los requisitos internos para el adiestramiento y tutoría en el uso de la herramienta.
- Evaluar las necesidades de capacitación, teniendo en cuenta las habilidades de prueba (y automatización de prueba) de quienes trabajarán directamente con la (s) herramienta (s).
- Considerar las ventajas y desventajas de varios modelos de licencia (p. ej., comercial o de código abierto) Estimación de una relación costo-beneficio basada en un estudio de viabilidad comercial concreto (si es necesario)

Métodos de pruebas ágiles

Existen determinadas prácticas de pruebas que pueden seguirse en cualquier proyecto de desarrollo (ágil o no) para producir productos de calidad. Entre ellas se encuentra escribir pruebas con antelación para expresar el comportamiento correcto, centrarse en la prevención, detección y eliminación temprana de defectos, y asegurar que se ejecutan los tipos de pruebas adecuados en el momento idóneo y como parte del nivel de prueba correcto. (Plan de estudios de Nivel Básico Probador Ágil, 2014, pág. 31)

Cuadrantes de pruebas, niveles de pruebas y tipos de pruebas (Plan de estudios de Nivel Básico Probador Ágil, 2014, págs. 32,33)

Los cuadrantes de pruebas, definidos por Brian Marick, alinean los niveles de pruebas con los tipos de pruebas adecuados en la metodología ágil. El modelo de cuadrantes de pruebas, y sus variantes, ayuda a garantizar que todos los tipos de pruebas y niveles de pruebas importantes se incluyen en el ciclo de vida del desarrollo. En los cuadrantes de pruebas, las pruebas pueden referirse al negocio (usuario) o la tecnología (desarrollador). Los cuatro cuadrantes son los siguientes:

El cuadrante Q1 es el nivel unitario, relativo a la tecnología, y da soporte al equipo. Este cuadrante contiene pruebas unitarias. Estas pruebas deben automatizarse e incluirse en el sistema de integración continua.

El cuadrante Q2 es el nivel de sistema, orientadas al negocio y confirma el comportamiento del producto. Este cuadrante contiene pruebas funcionales, ejemplos, pruebas de historias, prototipos de experiencia del usuario y simulaciones. Estas pruebas comprueban los criterios de aceptación y pueden ser manuales o estar automatizadas.

El cuadrante Q3 es el nivel de sistema o aceptación de usuario, relativo al negocio, y contiene pruebas que evalúan el producto mediante escenarios y datos realistas. Este cuadrante incluye pruebas exploratorias, escenarios, flujos de proceso, pruebas de usabilidad, pruebas de aceptación de usuario, pruebas alfa y pruebas beta

El cuadrante Q4 es el nivel de sistema o aceptación operativa, relativo a la tecnología, y contiene pruebas que evalúan el producto. Este cuadrante contiene pruebas de rendimiento, carga, estrés y escalabilidad, pruebas de seguridad, pruebas de mantenibilidad, gestión de la memoria, compatibilidad e interoperabilidad, migración de datos, infraestructura y recuperación. A menudo están automatizadas.

Evaluar riesgos de calidad y estimar el esfuerzo de prueba (Plan de estudios de Nivel Básico Probador Ágil, 2014, pág. 35)

Uno de los muchos retos de las pruebas es la correcta selección, asignación y priorización de las condiciones de prueba. Esto incluye determinar la cantidad de esfuerzo que debe dedicarse para cubrir cada condición con pruebas, y secuenciar las pruebas resultantes de tal manera que se optimice la eficacia y eficiencia del trabajo de pruebas a realizar.

El riesgo es la posibilidad de un resultado o evento, negativo o no deseado. El nivel de riesgo se establece evaluando la probabilidad de que el riesgo suceda y su impacto. Cuando el efecto principal del problema potencial se refiere a la calidad del producto, los problemas potenciales se denominan riesgos de calidad o riesgos de producto.

Diseño de pruebas de caja negra funcionales y no funcionales

En las pruebas ágiles, los probadores crean las pruebas en paralelo a las actividades de programación de los desarrolladores. Al igual que los desarrolladores programan en base a las historias de usuario y a los criterios de aceptación, los probadores crean pruebas en base a las historias de usuario y a sus criterios de aceptación. (Plan de estudios de Nivel Básico Probador Ágil, 2014, pág. 40)

Diseño/Evaluación de requisitos

Como se mencionó en el alcance inicial del proyecto, como equipo hemos hecho pruebas exploratorias en el sitio de ConduitApp. Dichas pruebas constan de explorar las funcionalidades que existen y listarlas para obtener un backlog y así poder obtener historias de usuario para luego definir los escenarios y casos de uso.
A partir de esas pruebas, se ha definido las siguientes historias:

Historias de usuario

Código	Nombre
HDU-01	Iniciar sesión
HDU-02	Cerrar sesión
HDU-03	Mostrar pantalla home
HDU-04	Registrar usuario
HDU-05	Configurar perfil
HDU-06	Editar perfil
HDU-07	Ver perfil
HDU-08	Cambiar contraseña
HDU-09	Agregar artículo a favoritos
HDU-10	Publicar nuevo artículo
HDU-11	Editar artículo
HDU-12	Eliminar artículo
HDU-13	Ver etiquetas populares
HDU-14	Publicar comentario en artículo
HDU-15	Eliminar comentario en artículo

Descripción de historias de usuario

Código:	HDU-01	**Nombre:**	Iniciar sesión
Prioridad en el negocio:	Alta	**Analista asignado:**	Elsy Castro
Prioridad en el desarrollo:	Alta		
Descripción: Yo como usuario, quiero acceder al sitio para poder visualizar la pantalla home, ver artículos publicados.			
Criterio de aceptación: • Dada la pantalla de inicio, cuando el ingreso de usuario y contraseña son correctos, entonces el sistema permitirá el ingreso.			

Código:	HDU-02	**Nombre:**	Cerrar sesión
Prioridad en el negocio:	Alta	**Analista asignado:**	Elsy Castro
Prioridad en el desarrollo:	Alta		
Descripción: Yo como usuario, quiero poder dar por terminada mi sesión y salir de manera segura del sitio y ser redireccionada a la pantalla de login.			
Criterio de aceptación: • Dada la opción cierre de sesión, cuando presiono el botón click here to logout, entonces el sistema redirecciona automáticamente al login.			

Código:	HDU-03	**Nombre:**	Mostrar pantalla home
Prioridad en el negocio:	Media	**Analista asignado:**	Elsy Castro
Prioridad en el desarrollo:	Alta		
Descripción: Yo como usuario, quiero que se muestre la pantalla principal, ver el global feed, las opciones de crear un nuevo artículo y acceder a la configuración.			
Criterio de aceptación: • Dada la pantalla de home, cuando seleccione una de las opciones, entonces me muestre el contenido correspondiente a la que elegí.			

Código:	HDU-04	**Nombre:**	Registrar usuario
Prioridad en el negocio:	Media	**Analista asignado:**	Elsy Castro
Prioridad en el desarrollo:	Alta		
Descripción: Como usuario, quiero poder registrarme como nuevo usuario para poder tener acceso a todas las opciones del sitio.			
Criterio de aceptación: • Dada la pantalla de inicio de sesión, cuando selecciono la opción de Need an account, entonces se despliega un formulario para el registro, el cual posee los campos de nombre de usuario, correo electrónico y contraseña, presiono sign up y me carga la pantalla de home.			

Código:	HDU-05	**Nombre:**	Configurar perfil
Prioridad en el negocio:	Media	**Analista asignado:**	Elsy Castro
Prioridad en el desarrollo:	Media		
Descripción: Yo como usuario, quiero poder configurar mi perfil para tener actualizada mi información.			
Criterio de aceptación: • Dada la pantalla de configuración de perfil, cuando modifico la url de la imagen de perfil, nombre, breve descripción sobre mí, email o contraseña, entonces al presionar sobre el botón update, settings, se guarda los datos actualizados.			

Código:	HDU-06	**Nombre:**	Editar perfil
Prioridad en el negocio:	Media	**Analista asignado:**	Alexander Villeda
Prioridad en el desarrollo:	Alta		
Descripción: Yo como escritor de blogs quiero poder editar mi perfil en el sitio para tener mi información pública y datos actualizados.			
Criterio de aceptación: • Que me permita actualizar mi foto de perfil por medio de una URL • Que pueda cambiar los siguientes valores de mi perfil: nombre de usuario, biografía (descripción acerca de mí), correo electrónico			

Código:	HDU-07	**Nombre:**	Ver perfil
Prioridad en el negocio:	Alta	**Analista asignado:**	Alexander Villeda
Prioridad en el desarrollo:	Media		
Descripción: Yo como escritor de blogs quiero que se muestre mi perfil público en la plataforma para que otras personas puedan ver mi trabajo publicado.			
Criterios de aceptación: • Que se muestre un listado de mis publicaciones hechas • Que se muestra en una pestaña adjunta un listado de publicaciones que he marcado como favoritas			

Código:	HDU-08	**Nombre:**	Cambiar contraseña
Prioridad en el negocio:	Media	**Analista asignado:**	Alexander Villeda
Prioridad en el desarrollo:	Alta		
Descripción: Yo como escritor de blogs quiero poder cambiar mi contraseña de acceso para que mi cuenta esté segura.			
Criterio de aceptación: • Que, al ser actualizada la contraseña, la aplicación me redirija a mi perfil público.			

Código:	HDU-09	**Nombre:**	Agregar artículo a favoritos
Prioridad en el negocio:	Baja	**Analista asignado:**	Alexander Villeda
Prioridad en el desarrollo:	Baja		
Descripción: Yo como escritor de blogs quiero agregar un artículo, escrito por otro usuario, a favoritos.			
Criterio de aceptación: • Que aparezca un botón para agregar a favoritos, en el artículo visualizado. • Que, al darle clic al botón para agregar a favoritos, este se convierta en un botón para quitar el artículo de favoritos y viceversa.			

Código:	HDU-10	**Nombre:**	Publicar nuevo artículo
Prioridad en el negocio:	Alta	**Analista asignado:**	Alexander Villeda
Prioridad en el desarrollo:	Alta		
Descripción: Yo como escritor de blogs quiero poder publicar artículos en mi perfil de usuario.			
Criterio de aceptación: • Que, al publicar un nuevo artículo, la aplicación redirija a este para visualizarlo.			

Código:	HDU-11	**Nombre:**	Editar artículo
Prioridad en el negocio:	Alta	**Analista asignado:**	Félix Carpio
Prioridad en el desarrollo:	Alta		
Descripción: Yo como usuario quiero editar el artículo que he publicado anteriormente y ver reflejado los cambios para actualizar cambios.			
Criterio de aceptación: • Al editar el artículo y guardar los cambios, se debe reflejar inmediatamente si alguien consulta el artículo.			

Código:	HDU-12	**Nombre:**	Eliminar artículo
Prioridad en el negocio:	Alta	**Analista asignado:**	Félix Carpio
Prioridad en el desarrollo:	Alta		
Descripción: Yo como usuario quiero eliminar algún artículo que haya publicado o editado anteriormente.			
Criterio de aceptación: • Al entrar al artículo debo tener la opción de eliminar el artículo.			

- Se mostrará un botón de eliminar el cual, al presionarlo, borrará la publicación de forma inmediata.

Código:	HDU-13	**Nombre:**	Ver etiquetas populares
Prioridad en el negocio:	Alta	**Analista asignado:**	Félix Carpio
Prioridad en el desarrollo:	Alta		

Descripción: Yo como usuario quiero visualizar las etiquetas existentes.

Criterio de aceptación:

- Al crear o en la pantalla principal del sitio, deseo visualizar el listado de etiquetas que se encuentran en eso.
- Deseo poder categorizar por etiquetas los artículos.
- Las etiquetas deben poder ser editables en cualquier momento.

Código:	HDU-14	**Nombre:**	Publicar comentario en artículo
Prioridad en el negocio:	Alta	**Analista asignado:**	Félix Carpio
Prioridad en el desarrollo:	Alta		

Descripción: Yo como usuario quiero comentar en los artículos que yo desee.

Criterio de aceptación:

- En los artículos existentes, debe haber una opción para poder agregar un comentario y también debo poder editarlo las veces que sean necesarias.
- Debe haber número indefinido de comentarios en el artículo.
- Puedo comentar mi propio artículo.

Código:	HDU-15	**Nombre:**	Eliminar comentario en artículo
Prioridad en el negocio:	Alta	**Analista asignado:**	Félix Carpio
Prioridad en el desarrollo:	Alta		

Descripción: Yo como usuario quiero editar el artículo que he publicado anteriormente y ver reflejado los cambios para actualizar cambios**.**

Criterio de aceptación:

- Al editar el artículo y guardar los cambios, se debe reflejar inmediatamente si alguien consulta el artículo.

Refinamiento

Código:	HDU-01	**Nombre:**	Iniciar sesión
Prioridad en el negocio:	Alta	**Analista asignado:**	Elsy Castro
Prioridad en el desarrollo:	Alta		
Descripción: Yo como usuario, quiero acceder al sitio para poder visualizar la pantalla home, ver artículos publicados.			
Criterio de aceptación: • Dada la pantalla de inicio, cuando el ingreso de usuario y contraseña son correctos, entonces el sistema permitirá el ingreso.			
Reglas del negocio: • Como usuario quiero que al iniciar sesión con correo electrónico y contraseña correctos me muestre la pantalla principal inicialmente mis publicaciones para poder ver mis artículos publicados. • Como usuario quiero que al iniciar sesión con correo electrónico y contraseña correctos se muestre la pantalla principal con la opción de muro global para poder consultar los artículos publicados por otros. • Como usuario quiero que al iniciar sesión con correo electrónico y contraseña correctos se muestre la pantalla principal con la opción de seleccionar una etiqueta popular para ver solo artículos relacionados a la etiqueta seleccionada.			

Código:	HDU-02	**Nombre:**	Cerrar sesión
Prioridad en el negocio:	Alta	**Analista asignado:**	Elsy Castro
Prioridad en el desarrollo:	Alta		
Descripción: Yo como usuario, quiero poder dar por terminada mi sesión y salir de manera segura del sitio y ser redireccionada a la pantalla de login.			
Criterio de aceptación: • Dada la opción cierre de sesión, cuando presiono el botón click here to logout, entonces el sistema redirecciona automáticamente al login.			
Reglas del negocio: • Como usuario, quiero poder cerrar mi sesión al ingresar al menú herramientas y seleccionar el botón click here to logout para dar por terminada mi sesión.			

Código:	HDU-03	**Nombre:**	Mostrar pantalla home
Prioridad en el negocio:	Media	**Analista asignado:**	Elsy Castro
Prioridad en el desarrollo:	Alta		
Descripción: Yo como usuario, quiero que se muestre la pantalla principal, ver el global feed, las opciones de crear un nuevo artículo y acceder a la configuración.			
Criterio de aceptación: • Dada la pantalla de home, cuando seleccione una de las opciones, entonces me muestre el contenido correspondiente a la que elegí.			
Reglas del negocio: • Como usuario, quiero que se muestre la pantalla principal, cuando he iniciado sesión para ver las opciones de global feed, crear un nuevo artículo y acceder a la configuración. • Como usuario, quiero que se muestre la pantalla principal, cuando no he iniciado sesión para ver todos los artículos publicados en el muro global que es público.			

Código:	HDU-04	**Nombre:**	Registrar usuario
Prioridad en el negocio:	Media	**Analista asignado:**	Elsy Castro
Prioridad en el desarrollo:	Alta		
Descripción: Como usuario, quiero poder registrarme como nuevo usuario para poder tener acceso a todas las opciones del sitio.			
Criterio de aceptación: • Dada la pantalla de inicio de sesión, cuando selecciono la opción de Need an account, entonces se despliega un formulario para el registro, el cual posee los campos de nombre de usuario, correo electrónico y contraseña, presiono sign up y me carga la pantalla de home.			
Reglas del negocio: • Como usuario, quiero poder registrarme para tener una cuenta de usuario con la que pueda realizar publicaciones y agregar artículos a favoritos.			

Código:	HDU-05	**Nombre:**	Configurar perfil
Prioridad en el negocio:	Media	**Analista asignado:**	Elsy Castro
Prioridad en el desarrollo:	Media		
Descripción: Yo como usuario, quiero poder configurar mi perfil para tener personalizada mi información.			

Criterio de aceptación:

- Dada la pantalla de configuración de perfil, cuando modifico la url de la imagen de perfil, nombre, breve descripción sobre mí, email o contraseña, entonces al presionar sobre el botón update, settings, se guarda los datos actualizados.

Reglas del negocio:

- Como usuario quiero poder configurar mi perfil para agregar una foto de perfil.
- Como usuario quiero poder configurar mi perfil para escribir una breve biografía sobre mí.

Código:	HDU-06	**Nombre:**	Editar perfil
Prioridad en el negocio:	Media	**Analista asignado:**	Alexander Villeda
Prioridad en el desarrollo:	Alta		

Descripción: Yo como escritor de blogs quiero poder editar mi perfil en el sitio para tener mi información pública y datos actualizados.

Criterio de aceptación:

- Que me permita actualizar mi foto de perfil por medio de una URL
- Que pueda cambiar los siguientes valores de mi perfil: nombre de usuario, biografía (descripción acerca de mí), correo electrónico.

Reglas del negocio:
Como usuario quiero poder editar mi perfil para actualizar mi nombre de usuario y cambiar foto de perfil.
Como usuario quiero poder editar mi perfil para poder modificar mi cuenta de correo electrónico y contraseña.

Código:	HDU-07	**Nombre:**	Ver perfil
Prioridad en el negocio:	Alta	**Analista asignado:**	Alexander Villeda
Prioridad en el desarrollo:	Media		

Descripción: Yo como escritor de blogs quiero que se muestre mi perfil público en la plataforma para que otras personas puedan ver mi trabajo publicado.

Criterios de aceptación:

- Que se muestre un listado de mis publicaciones hechas
- Que se muestra en una pestaña adjunta un listado de publicaciones que he marcado como favoritas.

Reglas del negocio:

- Como usuario quiero que se muestre mi perfil para que otras personas puedan ver mis artículos publicados y agregados a favoritos.
- Como usuario quiero que se muestre mi perfil para poder ver las publicaciones que he agregado a mi muro.
- Como usuario quiero que se muestre mi perfil para poder ver la cantidad de me gusta que tienen mis publicaciones.

Código:	HDU-08	**Nombre:**	Cambiar contraseña
Prioridad en el negocio:	Media	**Analista asignado:**	Alexander Villeda
Prioridad en el desarrollo:	Alta		

Descripción: Yo como escritor de blogs quiero poder cambiar mi contraseña de acceso para que mi cuenta esté segura.

Criterio de aceptación:

- Que, al ser actualizada la contraseña, la aplicación me redirija a mi perfil público.

Reglas del negocio:

- Como usuario quiero poder cambiar mi contraseña para que al actualizar la contraseña en el próximo inicio de sesión se autorice el ingreso con la contraseña más reciente.

Código:	HDU-09	**Nombre:**	Agregar artículo a favoritos
Prioridad en el negocio:	Baja	**Analista asignado:**	Alexander Villeda
Prioridad en el desarrollo:	Baja		

Descripción: Yo como escritor de blogs quiero agregar un artículo, escrito por otro usuario, a favoritos.

Criterio de aceptación:

- Que aparezca un botón para agregar a favoritos, en el artículo visualizado.
- Que, al darle clic al botón para agregar a favoritos, este se convierta en un botón para quitar el artículo de favoritos y viceversa.

Reglas del negocio:

- Como usuario quiero agregar un artículo de las publicaciones globales a favoritos para que al agregarlo aumente en uno la cantidad de favoritos de la publicación.
- Como usuario quiero agregar un artículo de las publicaciones globales a favorito en mi perfil para que aparezca en las publicaciones favoritas.

Código:	HDU-10	**Nombre:**	Publicar nuevo artículo
Prioridad en el negocio:	Alta	**Analista asignado:**	Alexander Villeda
Prioridad en el desarrollo:	Alta		
Descripción: Yo como escritor de blogs quiero poder publicar artículos en mi perfil de usuario.			
Criterio de aceptación: • Que, al publicar un nuevo artículo, la aplicación redirija a este para visualizarlo.			
Reglas del negocio: • Como usuario quiero poder publicar artículos en mi perfil, escribiendo el titulo del artículo, breve descripción del artículo y el cuerpo del artículo, así como palabras clave o etiquetas para que otros puedan velos en mi muro y buscarlos por etiquetas populares.			

Código:	HDU-11	**Nombre:**	Editar artículo
Prioridad en el negocio:	Alta	**Analista asignado:**	Félix Carpio
Prioridad en el desarrollo:	Alta		
Descripción: Yo como usuario quiero editar el artículo que he publicado anteriormente y ver reflejado los cambios para actualizar cambios.			
Criterio de aceptación: • Al editar el artículo y guardar los cambios, se debe reflejar inmediatamente si alguien consulta el artículo.			
Reglas del negocio: • Como usuario quiero editar artículos que he publicado anteriormente para actualizar el título del artículo, descripción del artículo, contenido o palabra clave y que se muestre la publicación actualizada.			

Código:	HDU-12	**Nombre:**	Eliminar artículo
Prioridad en el negocio:	Alta	**Analista asignado:**	Félix Carpio
Prioridad en el desarrollo:	Alta		
Descripción: Yo como usuario quiero eliminar algún artículo que haya publicado o editado anteriormente.			
Criterio de aceptación: • Al entrar al artículo debo tener la opción de eliminar el artículo.			

- Se mostrará un botón de eliminar el cual al presionarlo borrará la publicación de forma inmediata.

Reglas del negocio:

- Como usuario quiero eliminar artículos que he publicado anteriormente para que ya no se encuentren visibles en mi perfil.

Código:	HDU-13	**Nombre:**	Ver etiquetas populares
Prioridad en el negocio:	Alta	**Analista asignado:**	Félix Carpio
Prioridad en el desarrollo:	Alta		

Descripción: Yo como usuario quiero visualizar las etiquetas existentes.

Criterio de aceptación:

- Al crear o en la pantalla principal del sitio, deseo visualizar el listado de etiquetas que se encuentran en eso.
- Deseo poder categorizar por etiquetas los artículos.
- Las etiquetas deben poder ser editables en cualquier momento.

Reglas del negocio:

- Como usuario quiero visualizar las etiquetas populares cuando seleccione una de las palabras en el recuadro de etiquetas populares.

Código:	HDU-14	**Nombre:**	Publicar comentario en artículo
Prioridad en el negocio:	Alta	**Analista asignado:**	Félix Carpio
Prioridad en el desarrollo:	Alta		

Descripción: Yo como usuario quiero comentar en los artículos que yo desee.

Criterio de aceptación:

- En los artículos existentes, debe haber una opción para poder agregar un comentario y también debo poder editarlo las veces que sean necesarias.
- Debe haber número indefinido de comentarios en el artículo.
- Puedo comentar mi propio artículo.

Reglas del negocio:

- Como usuario quiero comentar los artículos cuando seleccione uno de los artículos del muro global para realizar aportes a las publicaciones de otros e inclusive las propias.

Código:	HDU-15	**Nombre:**	Eliminar comentario en artículo
Prioridad en el negocio:	Alta	**Analista asignado:**	Félix Carpio
Prioridad en el desarrollo:	Alta		
Descripción: Yo como usuario quiero editar el artículo que he publicado anteriormente y ver reflejado los cambios para actualizar cambios**.**			
Criterio de aceptación: • Al editar el artículo y guardar los cambios, se debe reflejar inmediatamente si alguien consulta el artículo.			
Reglas del negocio: • Como usuario quiero eliminar comentarios que he realizado a publicaciones de otros o propias para que mi comentario desaparezca al eliminarlo.			

Estimación de requisitos

Planning Póker

El planning poker (o póquer de planificación) es un método de estimación que ayuda al equipo ágil a calcular la cantidad de esfuerzo que se necesita para completar una historia de usuario en un backlog del producto. Se utiliza a menudo en metodologías ágiles de gestión de proyectos, y a veces también se lo conoce como "Scrum poker" o "pointing poker". El término "póker" del nombre hace referencia a las cartas que cada miembro del equipo usa durante el proceso.

Cualquier especialista puede participar en este procedimiento: programadores, probadores, ingenieros técnicos, gerentes de proyectos, administradores de sistemas, diseñadores y otras personas que estén involucradas en este proyecto.

Dado que diferentes miembros del equipo del proyecto participan en este proceso, dicho método permite obtener una estimación independiente compleja al resolver tareas técnicas y prácticas difíciles.

¿Cómo es el proceso del planning póker?

El proceso de planning poker se realiza al principio en el proceso de planificación del sprint, de modo que los Scrum masters y los gerentes de producto puedan tener una idea precisa de cuánto trabajo puede completarse en cada sprint.

Reglas del Planning Póker

Este póquer se lleva a cabo durante las reuniones antes de empezar a trabajar en el proyecto o cuando se despliega una nueva funcionalidad en el producto ya desarrollado.

Cada participante recibe una baraja especial. Hay dos barajas más populares.

Primera baraja

Las tarjetas se basan en una escala de valoración especializada de Mike Cohn. Estas tarjetas se basan en la secuencia de Fibonacci (números 0, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89 + tarjeta con el signo "?" + tarjeta con la imagen del café + tarjeta con el signo del infinito).

Interpretación de las cartas

Tarjeta	Significado
0	Una tarea que ya está hecha o es muy sencilla.
1-3	Tareas pequeñas.
5-13	Tareas de prioridad media.
21-55	Tareas largas y complejas.
89	Tareas globales
∞	La tarea es tan grande que no tiene sentido darle un número.
?	Proporciona información adicional del propietario del producto. Algunos jugadores pueden utilizar esta tarjeta para negarse a estimar el sprint actual (pero no pueden negarse a participar en la discusión y estimación).
Taza de café	Uno de los participantes solicita tomar un descanso para ir por café o té.

En la siguiente valoración se ha considerado que cada punto es el equivalente a dos horas de trabajo del lado de QA.

Código	Nombre	Valoración
HDU-01	Iniciar sesión	2
HDU-02	Cerrar sesión	2
HDU-03	Mostrar pantalla home	2
HDU-04	Registrar usuario	8
HDU-05	Configurar perfil	3
HDU-06	Editar perfil	3
HDU-07	Ver perfil	3
HDU-08	Cambiar contraseña	3
HDU-09	Agregar artículo a favoritos	5
HDU-10	Publicar nuevo artículo	8
HDU-11	Editar artículo	5
HDU-12	Eliminar artículo	5
HDU-13	Ver etiquetas populares	5
HDU-14	Publicar comentario en artículo	3
HDU-15	Eliminar comentario en artículo	3

Estrategia de pruebas

Con la estrategia de pruebas se busca definir y planificar las pruebas que serán realizadas por el equipo. Se automatizarán pruebas de caja negra a funcionalidades seleccionadas de acuerdo a criterios.

Se realizarán las siguientes pruebas:

Pruebas funcionales
Se utilizarán para validar los códigos de estado y verificar que la API se encuentra disponible. Inicialmente se realizarán pruebas manuales y conforme se tenga más estabilidad en las funcionalidades se empezará a automatizar casos de prueba.

Pruebas de seguridad
Con las pruebas de seguridad se probará la autenticación, se verificará si utiliza algún tipo de token y que solo puedan ingresar a la API usuarios autorizados.

Métricas
Se documentará el total de casos de prueba ejecutados, cantidad de incidentes detectados y cantidad de incidentes resueltos.

Riesgos y planes de contingencia

No	Riesgos	Probabilidad de ocurrencia (1-5)	Impacto (1-5)	Severidad (Prob*Impacto)	Plan de contingencia
1	Alcances mal definidos	2	3	6	Redistribuir las funcionalidades que serán probadas.
2	No disponibilidad del entorno de pruebas	3	5	15	Contar con un segundo entorno de prueba configurado.
3	Equipo de pruebas no capacitado para realizar las pruebas al sitio.	2	4	8	Capacitar al equipo de testing para fortaleces sus capacidades
4	No cumplimiento de los entregables programados en el cronograma.	2	3	6	Realizar estudio para establecer prioridades en las pruebas a realizar.
5	Cambios en funcionalidades que ya están automatizadas las pruebas	3	4	12	Realizar ajustes en la calendarización para hacer nuevamente los scripts de pruebas.

Ambiente y herramientas de pruebas

Herramientas de pruebas

Herramienta	Función
Java 8	Permite que algunas aplicaciones escritas en lenguaje de programación Java se inicien a través de ciertos navegadores.
Maven v3.8.7	Herramienta de construcción de código que permite compilar cualquier tipo de proyecto, gestiona dependencias entre módulos y distintas versiones de librerías.
Postman v10.9.0	Herramienta que permite realizar peticiones para hacer pruebas de APIs de tipo REST.
Visual Studio Code v1.75.1	Editor de código fuente, con soporte para la depuración, control integrado de Git, refactorización de código, etc.
VS Code Plugins	
Cucumber (Gherkin) Full Support v2.15.2	Permite el soporte de lenguaje Gherkin y manejo de ficheros.
Karate Runner v1.2.3	Extensión utilizada para automatizar los casos de prueba.
Java Extension Pack v17.791.533	Facilita la escritura, prueba y debug de aplicaciones.

Arquitectura del framework de automatización

Karate es la única herramienta de código abierto que combina la automatización de las pruebas de la API, los mocks, las pruebas de rendimiento e incluso la automatización de la interfaz de usuario en un único marco unificado. La sintaxis BDD, popularizada por Cucumber, es neutra en cuanto al lenguaje y fácil de usar incluso para los no programadores. Las aserciones y los informes HTML están incorporados, y se pueden ejecutar las pruebas en paralelo para aumentar la velocidad.

También hay un ejecutable independiente, no es necesario compilar el código. Sólo hay que escribir pruebas con una sintaxis sencilla y legible, cuidadosamente diseñada para HTTP, JSON, GraphQL y XML. Y se puede combinar la automatización de pruebas de la API y de la interfaz de usuario dentro del mismo script de prueba.

También existe una API de Java para quienes prefieren integrar mediante programación las capacidades de automatización y afirmación de datos de Karate.

Ambiente de pruebas

Navegador	Sistema Operativo
Google Chrome 110.0	Windows 10
Mozilla Firefox 110.0	

Planificación de ejecución de pruebas

A partir de las historias de usuario definidas anteriormente, se han definido los siguientes casos de prueba:

Código	Nombre del caso de prueba	Responsable	Fecha	Prioridad
CP-01	Registro de nuevo usuario exitoso	Elsy Castro	08/08/2022	Alta
CP-02	Registro de nuevo usuario con nombre de usuario ya existente.	Elsy Castro	09/08/2022	Alta
CP-03	Registro de usuario con email faltante.	Elsy Castro	10/08/2022	Alta
CP-04	Registro de nuevo usuario con contraseña en blanco.	Alexander Villeda	11/08/2022	Alta
CP-05	Registro de nuevo usuario con nombre de usuario vacío.	Alexander Villeda	12/08/2022	Medio
CP-06	Crear nuevo artículo exitoso.	Alexander Villeda	15/08/2022	Medio
CP-07	Eliminar artículo exitoso.	Félix Carpio	16/08/2022	Medio
CP-08	Consultar artículos de la página.	Félix Carpio	17/08/2022	Bajo
CP-09	Ver las etiquetas.	Félix Carpio	18/08/2022	Bajo

Plan de pruebas

Control de versiones

Versión	Autor(es)	Descripción	Fecha
1.0	Félix Carpio	Creación del documento	Agosto 2022
1.1	Félix Carpio	Modificación de tipos de prueba	Septiembre 2022

Introducción

El Plan de pruebas de ConduitApp identifica las tareas y actividades que serán llevadas a cabo de manera que todos los aspectos del sistema sean probados adecuadamente y que el sistema sea exitosamente implementado. Este plan documenta el enfoque, metodología, secuencia, gestión, y responsabilidades para las actividades de prueba.

Contexto de negocio

ConduitApp es una aplicación solicitada con el fin de que los colaboradores de la organización puedan tener un espacio para poder publicar sus pensamientos, ideas y artículos de carácter profesional y personal; y que estos puedan ser distribuidos, vistos y compartidos por y para otros colaboradores y potenciales clientes, y que a su vez estos puedan expresar comentarios acerca de las publicaciones realizadas.

Esto ayudará a la organización a tener un espacio de referencia para compartir contenido que puedan fácilmente observados por una amplia audiencia, puedan servir como medio en el cual la organización pueda expresar sus valores alineados en las publicaciones de sus colaboradores y a que su vez promuevan los diversos productos y servicios ofrecidos a clientes actuales y potenciales.

Objetivos de las pruebas

El objetivo de las pruebas es verificar la funcionalidad de ConduitApp de acuerdo a las especificaciones.

Las pruebas:

- Ejecutarán y verifican los scripts de pruebas
- Identificarán, buscarán que se reparen y se re-ejecuten las pruebas en los defectos de encontrados.
- Estarán orientadas a entregar un producto de alta calidad y confiabilidad.

El producto final de las pruebas será un software listo para producción y un conjunto de scripts de pruebas que puedan ser reutilizados para ejecuciones de pruebas de funcionales y de aceptación.

Alcance de las pruebas

El documento se enfoca principalmente en la realización de pruebas API manualmente y de manera automatizada probando y validando los datos obtenidos como salida de la aplicación según lo indicado en los requerimientos provistos por el cliente.

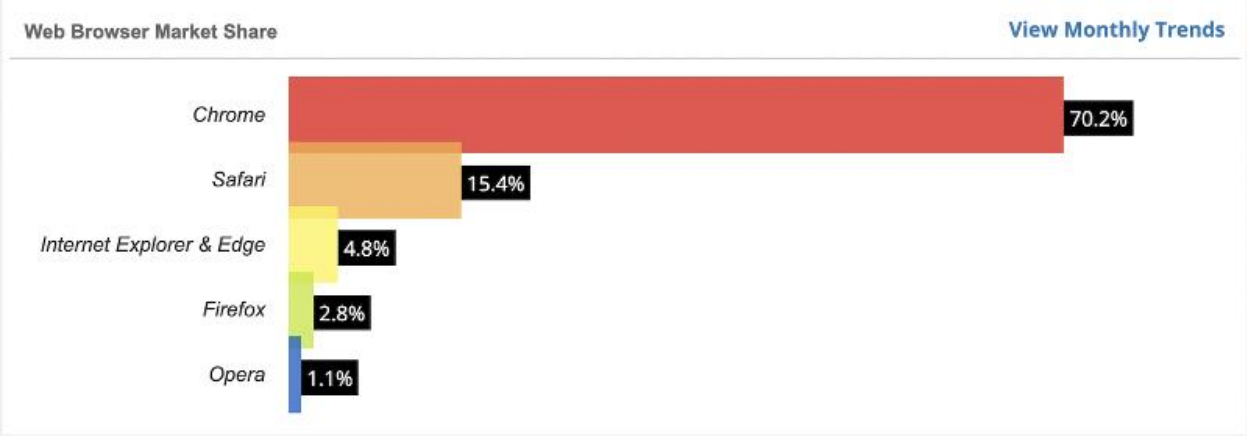
Para las pruebas de ConduitApp se probarán los siguientes módulos:

- Módulo de gestión de usuarios y perfiles
- Módulo de gestión de artículos, comentarios y etiquetas

Los navegadores y sistemas operativos para realizar las pruebas son los siguientes:

Navegador	**Sistema Operativo**
Google Chrome	Windows 10
Mozilla Firefox	

Para determinar en qué navegadores haremos nuestras pruebas, investigamos los navegadores que más se usan actualmente. A continuación, una gráfica de los navegadores más usados.



A tomar en cuenta que se eligieron los navegadores de Chrome y Firefox debido a su compatibilidad con otros sistemas operativos aparte de Windows como se muestra a continuación:

FUNCIONES	Chrome	Safari	Firefox	Edge	Opera
Sincronización en la nube	Sí	Sí	Sí	Sí	Sí
Gestor de descargas	Sí	Sí	Sí	Sí	Sí
Navegación privada	Sí	Sí	Sí	Sí	Sí
Modo de pantalla completa	Sí	Sí	Sí	Sí	Sí
Pestañas verticales	Sí	No	Sí	Sí	Sí
Extensiones personalizadas	Sí	Sí	Sí	Sí	Sí

COMPATIBILIDAD	Chrome	Safari	Firefox	Edge	Opera
Windows	Sí	No	Sí	Sí	Sí
macOS	Sí	Sí	Sí	Sí	Sí
Linux	Sí	No	Sí	Sí	Sí
Android	Sí	No	Sí	Sí	Sí
iOS	Sí	Sí	Sí	Sí	Sí

No se contempla el uso y prueba de aplicaciones móviles ya que la el software en cuestión no tiene versión para dispositivos móviles.

Entre las pruebas no funcionales se realizarán pruebas de seguridad y rendimiento a la aplicación.

Riesgos de calidad

No	Riesgos	Probabilidad de ocurrencia (1-5)	Impacto (1-5)	Severidad (Prob*Impacto)	Plan de contingencia
1	El software no está acorde a las especificaciones del cliente	2	5	10	Reorganizar la programación del proyecto y proceder a rediseñar e implementar las funcionalidades de acuerdo a lo indicado por el Product Owner

2	El rendimiento del software no es el esperado	3	3	9	Realizar las debidas pruebas de rendimiento en los componentes que no cumplen con lo esperado con el fin de determinar las causas del bajo rendimiento actual.
3	Se percibe la aplicación con un nivel de usabilidad baja	3	2	6	Realizar un estudio con profesionales del área para identificar y solventar los puntos de mejora.

Tipos de pruebas

Las pruebas a realizarse serán:

Pruebas funcionales
Se utilizarán para validar los códigos de estado y verificar que la API se encuentra disponible. Inicialmente se realizarán pruebas manuales y conforme se tenga más estabilidad en las funcionalidades se empezará a automatizar casos de prueba.

Pruebas de seguridad
Con las pruebas de seguridad se probará la autenticación, se verificará si utiliza algún tipo de token y que solo puedan ingresar a la API usuarios autorizados.

Pruebas automatizadas
Tipo de pruebas basadas en secuencias de comandos ejecutadas por herramientas de automatización de pruebas con el fin de ahorrar tiempo y esfuerzo y verificar la integridad de los componentes de software y la aplicación en general sin intervención humana.

Recursos y preparación para las pruebas

Las herramientas necesarias para las pruebas son:

PROCESO	HERRAMIENTAS
Ejecución de casos de prueba	Postman
Reporte de pruebas	PDF, HTML

Herramientas para ambiente de pruebas automatizadas

HERRAMIENTA	FUNCIÓN
Java 8	Permite que algunas aplicaciones escritas en lenguaje de programación Java se inicien a través de ciertos navegadores.
Maven v3.8.7	Herramienta de construcción de código que permite compilar cualquier tipo de proyecto, gestiona dependencias entre módulos y distintas versiones de librerías.
Visual Studio Code v1.75.1	Editor de código fuente, con soporte para la depuración, control integrado de Git, refactorización de código, etc.
VS Code Plugins	
Cucumber (Gherkin) Full Support v2.15.2	Permite el soporte de lenguaje Gherkin y manejo de ficheros.
Karate Runner v1.2.3	Extensión utilizada para automatizar los casos de prueba.
Java Extension Pack v17.791.533	Facilita la escritura, prueba y debug de aplicaciones.

Organización del equipo de pruebas

Para el proyecto se cuenta con 3 analistas de calidad, cuyas responsabilidades principales son las siguientes:

- Planificar estrategia de pruebas y documentación del plan de pruebas
- Crear los escenarios y casos de prueba
- Revisar los escenario y casos de prueba
- Preparar los datos de prueba
- Coordinar con el Líder de QA acerca de cualquier problema encontrado durante la preparación o ejecución de las pruebas

Equipo de pruebas:

ROL	NOMBRE	CONTACTO
Líder de QA	Félix Alfredo Carpio	cc14025@ues.edu.sv
Analista de Calidad	Elsy Margoth Castro	ce12006@ues.edu.sv
Analista de Calidad	Alfredo Alexander Villeda	vr15004@ues.edu.sv

Enfoque/Adaptación de las pruebas

De los 2 navegadores para las pruebas se utilizará principalmente Google Chrome debido a su mayor adopción en el mercado comparado a Mozilla Firefox.

En las pruebas que así lo requieran se utilizará una o más de una de las siguientes técnicas de prueba:

- **Técnica de partición de equivalencia**
  - Se utilizará esta técnica cuando se necesite agrupar conjuntos de valores en rangos válidos e inválidos para su evaluación lógica.

- **Técnica de análisis de valores frontera**
  - Se utilizará está técnica unida a la técnica de partición de equivalencia para evaluar los valores máximos y mínimos de las tablas de equivalencias que creen y verificar si estos cumplen con el resultado esperado o no.

- **Técnica de transición de estado**
  - Se utilizará esta técnica cuando se requiera evaluar diferentes combinaciones de condiciones (descritas en los requerimientos) que generen diferentes resultados.

- **Técnica de prueba y cobertura de decisión**
  - Se utilizará esta técnica cuando se requiera verificar las transiciones válidas y las potencialmente inválidas entre estados, así como eventos.

Los niveles de prueba que con que se trabajarán serán:

- **Pruebas unitarias:** el equipo de desarrollo debe cumplir con el requisito de pruebas unitarias a su código. Dicha cobertura de pruebas tendrá un porcentaje para luego poder ser aprobadas y validadas por el equipo de QA.

- **Pruebas de integración:** en esta etapa, el equipo de QA hará pruebas de integración entre endpoints para completar flujos. Dichas pruebas se harán manuales en las etapas tempranas y una vez ya se haya tenido madurez de pruebas en los desarrollos, se podrá pasar escenarios a la etapa de automatización.

- **Pruebas de sistemas:** en esta etapa, el equipo de QA tratará de cubrir el mayor número de escenarios y casos de prueba tanto para validar la parte a nivel de servicios y endpoints como también en pruebas end to end y pruebas manuales a la parte web de la aplicación.

Administración de defectos

Dentro del proyecto, como equipo de QA se tiene la siguiente clasificación de los defectos con su respectiva descripción:

- Prioridad baja: El defecto no afecta la funcionalidad nueva ni las ya existentes. Por ejemplo: el color de un header en el sitio web no es el correcto.

- Prioridad media: La funcionalidad desarrollada se ve comprometida, pero se pueden realizar los flujos principales sin problemas. Por ejemplo: una nueva configuración para el proceso de pago donde se ha puesto una URL mala para ir al proveedor equivocado, pero siempre se puede procesar pago como se debe.
- Prioridad alta: La funcionalidad desarrollada tiene un comportamiento no deseado en cualquier de sus flujos lo cual puede ocasionar una falla en el sistema. Por ejemplo: existe un servicio que envía un contrato a un endpoint del sitio web, pero el contrato sufrió cambios solo de 1 lado y la página no carga por la incoherencia del contrato.

Administración de la configuración

Para nuestro proyecto, tendremos un solo ambiente que será ya la propia aplicación web en línea de ConduictApp:

- Ambiente de QA: En este ambiente se probarán todas las funcionalidades y serán validadas por el equipo de QA. En nuestro caso como equipo, ya tenemos el sitio de ConduitApp en línea para realizar las pruebas, por lo tanto, ese será nuestro único ambiente.

Entregables de pruebas

Dentro de nuestros entregables, se darán los siguientes reportes:

- Reporte de fallos detectados: En cada sprint, se dará a conocer en un reporte el listado de bugs reportados según el nivel de criticidad. También se dará a conocer su estado a la hora de haber cerrado el sprint.
- Reporte del plan de pruebas: Se entregará cuando se considere necesario o sea solicitado un reporte del avance del plan de pruebas para validar que el proyecto continúa dentro del rango de tiempo establecido, el estado de las pruebas, etc.
- Reporte de estrategia de pruebas: Al inicio de proyecto, se dará a conocer el estado de la estrategia de pruebas como equipo de QA para validar que se han tomado todas las consideraciones y alcanzar el éxito del proyecto en el tiempo determinado.

Métricas de pruebas

Métrica de fuga de errores: Esta métrica mide el nivel de fuga de bugs encontrados en algún ambiente, es decir, el porcentaje de bugs que aún no se han resuelto. Por ejemplo, si se encuentran cierta cantidad de bugs y algunos no han sido resueltos, podemos tener el porcentaje de cobertura de bugs. A continuación, se muestra la fórmula:

$$\frac{Bugs\ resueltos\ en\ ambiente\ actual}{Bugs\ pendientes\ en\ ambiente\ actual + Bugs\ resueltos\ en\ ambiente\ actual} * 100\%$$

Métrica de pruebas automatizadas: esta métrica indicará el nivel de aceptación de las pruebas automatizadas. En nuestro caso, se evaluará el número de ejecuciones erróneas de alguna prueba con respecto a las ejecuciones exitosas de las pruebas. Si hay más ejecuciones con fallos que exitosas, se debe tomar en cuenta para poder hacer las correcciones necesarias.

Criterios de salida

- Criterios de pruebas automatizadas: Como criterio de salida respecto a las pruebas automatizadas se ha estipulado que debe cumplirse una cobertura del 80% de dichas pruebas para poder cerrar la etapa de automatización.

Monitoreo y control

Durante la ejecución de las pruebas manuales, se monitorearon las actividades y defectos que se han trabajado sobre las funcionalidades encontradas. Con respecto a la métrica de fuga de defectos, se han reportado los siguientes números de defectos en ambiente QA:

- Bugs reportados de nivel alto: 1
- Bugs reportados de nivel medio: 1
- Bugs reportados de nivel bajo: 1

Calculando la fuga de bugs sería:

$$Porcentaje\ de\ fuga = \frac{1}{1+0} * 100 = 100\%$$

En la fórmula, se utilizaron el número de bugs de nivel alto. En este caso, nosotros no tenemos forma de poder reportar dichos bugs que nosotros consideramos como mejoras o correcciones, pero el equipo a partir de las pruebas manuales realizadas, estimó ciertos puntos donde la aplicación podría dar más valor a su sitio haciendo las correcciones de los bugs reportados.

Dichos bugs se tratarán en un apartado más adelante, y este cálculo ha servido como ejemplo de la métrica que se llevaría.

Análisis de pruebas

Para el proyecto se tienen planeados los siguientes escenarios de pruebas:

Código	Nombre de escenario	Prioridad
EP-01	Registro exitoso de nuevo usuario.	Alto
EP-02	Registro de usuario fallido con datos erróneos	Alto
EP-03	Registro de usuario fallido con campos vacíos	Alto
EP-04	Creación exitosa de nuevo artículo	Alto
EP-05	Creación de nuevo artículo con campos erróneos	Medio

EP-06	Creación de nuevo artículo con campos vacíos	Medio
EP-07	Eliminación de artículo de forma exitosa.	Medio
EP-08	Consulta de artículos de forma exitosa.	Bajo
EP-09	Consulta de etiquetas de forma exitosa.	Bajo

Diseño de las pruebas

A continuación, se detallarán los casos de prueba del sprint1 a partir de nuestros escenarios de prueba:

Caso de Prueba: Registro de un nuevo usuario de manera exitosa.

Objetivo de la prueba	Validar que se pueda realizar un registro de un nuevo usuario de manera exitosa con datos correctos.
Identificador	CP-01
Nombre del caso	Nuevo usuario registrado
Prioridad del caso	Alta
QA Asignado	Elsy Castro
Datos de entrada	Username: test123 Email: test221312@das.com Password: test123
Precondiciones	• Que el usuario pueda acceder a la sección de “Sign Up” y cargue el formulario para registrar usuario.
Paso	Resultado esperado
1. Entrar al sitio web y dar clic en la opción de Sign Up.	Carga correctamente el formulario con los campos a llenar para registrar usuario nuevo.
2. Se llenan todos los campos de forma correcta.	
3. Se da clic en el botón “Sign Up” para poder registrar usuario.	Se registra el usuario correctamente y carga el feed principal ya automáticamente con la sesión iniciada.

Caso de Prueba: Registro de un nuevo usuario fallido con datos erróneos

Objetivo de la prueba	Validar que no se pueda registrar un usuario nuevo con un username ya existente.
Identificador	CP-02
Nombre del caso	Registro de un nuevo usuario con username ya existente.
Prioridad del caso	Alta
QA Asignado	Elsy Castro
Datos de entrada	Username: test123 Email: test221312@das.com

	Password: test123
Precondiciones	• Que el usuario pueda acceder a la sección de “Sign Up” y cargue el formulario para registrar usuario.
Paso	Resultado esperado
1. Se ingresa al sitio y se da clic en la opción de “Sign Up”	Carga correctamente el formulario con los campos a llenar para registrar usuario nuevo.
2. Se llenan los campos correctamente y en el campo de nombre se ingresa un username ya existente.	
3. Se da clic en el botón de “Sign Up” para intentar registrar el usuario.	Se muestra un mensaje de error que menciona “username has already been taken” indicando que ya existe ese nombre de usuario.

Caso de Prueba: Registro de un nuevo usuario fallido con campos vacíos

Objetivo de la prueba	Validar que el formulario no acepte campos vacíos en el flujo de registro de usuario.
Identificador	CP-03
Nombre del caso	Registro de un nuevo usuario con campo de email vacío.
Prioridad del caso	Alta
QA Asignado	Elsy Castro
Datos de entrada	Username: test1234 Email: Password: test123
Precondiciones	• Que el usuario pueda acceder a la sección de “Sign Up” y cargue el formulario para registrar usuario.
Paso	Resultado esperado
1. Se ingresa al sitio y se da clic en la opción de “Sign Up”	Carga correctamente el formulario con los campos a llenar para registrar usuario nuevo.
2. Se llenan los campos correctamente excepto el campo de email.	Al dejar vacío un campo, se deshabilita el botón de “Sign up” para registrar el usuario.

Caso de Prueba: Registro de un nuevo usuario fallido con campos vacíos

Objetivo de la prueba	Validar que el formulario no acepte campos vacíos en el flujo de registro de usuario.
Identificador	CP-04
Nombre del caso	Registro de un nuevo usuario con campo de contraseña vacío.
Prioridad del caso	Alta
QA Asignado	Elsy Castro
Datos de entrada	Username: test123 Email: tes4324234@gmail.com Password:

Precondiciones	• Que el usuario pueda acceder a la sección de “Sign Up” y cargue el formulario para registrar usuario.
Paso	Resultado esperado
1. Se ingresa al sitio y se da clic en la opción de “Sign Up”	Carga correctamente el formulario con los campos a llenar para registrar usuario nuevo.
2. Se llenan los campos correctamente excepto el campo de contraseña.	Al dejar vacío un campo, se deshabilita el botón de “Sign up” para registrar el usuario.

Caso de Prueba: Registro de un nuevo usuario fallido con campos vacíos

Objetivo de la prueba	Validar que el formulario no acepte campos vacíos en el flujo de registro de usuario.
Identificador	CP-05
Nombre del caso	Registro de un nuevo usuario con campo de nombre de usuario vacío.
Prioridad del caso	Alta
QA Asignado	Alexander Villeda
Datos de entrada	Username: Email: test42423@das.com Password: test123
Precondiciones	• Que el usuario pueda acceder a la sección de “Sign Up” y cargue el formulario para registrar usuario.
Paso	Resultado esperado
1. Se ingresa al sitio y se da clic en la opción de “Sign Up”	Carga correctamente el formulario con los campos a llenar para registrar usuario nuevo.
2. Se llenan los campos correctamente excepto el campo de nombre de usuario.	Al dejar vacío un campo, se deshabilita el botón de “Sign up” para registrar el usuario.

Caso de Prueba: Creación de nuevo artículo de forma exitosa

Objetivo de la prueba	Validar que el formulario de nuevo artículo funcione correctamente y se agregue el artículo exitosamente.
Identificador	CP-06
Nombre del caso	Creación de nuevo artículo de forma exitosa.
Prioridad del caso	Alta
QA Asignado	Alexander Villeda
Datos de entrada	Título: Ejemplo artículo Descripción: Descripción de prueba Cuerpo del artículo: Este artículo de ejemplo es para el caso de prueba número 6 Tag: Prueba, test
Precondiciones	• El usuario debe haber iniciado sesión para crear el artículo.

Paso	Resultado esperado
1. Una vez iniciado sesión y se entra a la página de inicio, se da clic en la opción de "New Article" en la parte superior derecha.	Carga correctamente el formulario para ingresar el nuevo artículo.
2. Se llenan los campos correctamente.	
3. Al terminar de llenar los campos, se da clic en el botón de "Publish Article".	Se crea correctamente el artículo y se visualiza el artículo creado para poder añadir un comentario. También se visualiza botones para poder editar o eliminar el artículo.

Caso de Prueba: Eliminar artículo de forma exitosa

Objetivo de la prueba	Validar que se pueda eliminar un artículo en específico en forma exitosa.
Identificador	CP-07
Nombre del caso	Eliminar artículo de forma exitosa.
Prioridad del caso	Medio
QA Asignado	Alexander Villeda
Datos de entrada	Artículo anteriormente creado
Precondiciones	• El usuario debe haber iniciado sesión para visualizar el artículo. • Debe haber creado un artículo anteriormente.
Paso	Resultado esperado
1. Una vez iniciado sesión y se entra a la página de inicio, se da clic en la opción de "Global Feed" para visualizar el listado de artículos.	Cargan los artículos creados hasta el momento.
2. Una vez encontrado el artículo a eliminar, se da clic sobre el título de este.	Se visualizar toda la información del artículo y los botones para editar y eliminar artículo.
3. Se da clic en el botón "Delete Article".	Una vez se da clic, se redirecciona a la página principal y al visualizar los artículos creados, ya no está el artículo eliminado.

Caso de Prueba: Consultar artículos creados

Objetivo de la prueba	Validar que se pueda visualizar los artículos creados anteriormente.
Identificador	CP-08
Nombre del caso	Consultar artículos creados
Prioridad del caso	Medio
QA Asignado	Félix Carpio
Datos de entrada	Artículos anteriormente creados

Precondiciones	• El usuario debe haber iniciado sesión para visualizar el artículo. • Debe haber creado un artículo anteriormente.
Paso	Resultado esperado
1. Una vez iniciado sesión y se entra a la página de inicio, se da clic en la opción de "Global Feed" para visualizar el listado de artículos.	Cargan los artículos creados hasta el momento.

Caso de Prueba: Visualizar etiquetas utilizadas

Objetivo de la prueba	Validar que se pueda visualizar las etiquetas más usadas en los artículos.
Identificador	CP-09
Nombre del caso	Visualización de etiquetas utilizadas
Prioridad del caso	Bajo
QA Asignado	Félix Carpio
Datos de entrada	Artículos y etiquetas creadas anteriormente.
Precondiciones	• El usuario debe haber iniciado sesión para visualizar el artículo. • Debe haber creados artículos y etiquetas creadas anteriormente.
Paso	Resultado esperado
1. Una vez iniciado sesión y se entra a la página de inicio.	Cargan los artículos creados hasta el momento y en la parte derecha se visualizan las etiquetas más utilizadas.

Implementación de las pruebas

Se presenta en la siguiente tabla toda la preparación de herramientas previo a la ejecución de las pruebas.

Herramientas de pruebas

Nombre y versión	Función	Estado
Java 8	Permite que algunas aplicaciones escritas en lenguaje de programación Java se inicien a través de ciertos navegadores	
Maven v3.8.7	Herramienta de construcción de código que permite compilar cualquier tipo de proyecto,	

	gestiona dependencias entre módulos y distintas versiones de librerías	
Postman v10.9.0	Herramienta que permite realizar peticiones para hacer pruebas de APIs de tipo REST.	
VS Code Plugins		
Visual Studio Code v1.75.1	Editor de código fuente, con soporte para la depuración, control integrado de Git, refactorización de código, etc	Instalado
Cucumber (Gherkin) Full Support v2.15.2	Permite el soporte de lenguaje Gherkin y manejo de ficheros.	
Karate Runner v1.2.3	Extensión utilizada para automatizar los casos de prueba.	
Java Extensio Pack v17.791.533	Facilita la escritura, prueba y debug de aplicaciones.	

Ambiente de pruebas

Navegador	
Ambiente	**Estado**
Google Chrome 110.0.5481.177	Instalado
Mozilla Firefox 110.0	
Sistema Operativo	
Windows 10	Instalado
Equipo informático y repositorio de pruebas	
PC con especificaciones mínimas de Procesador i3 y Memoria RAM de 8 GB	Listo

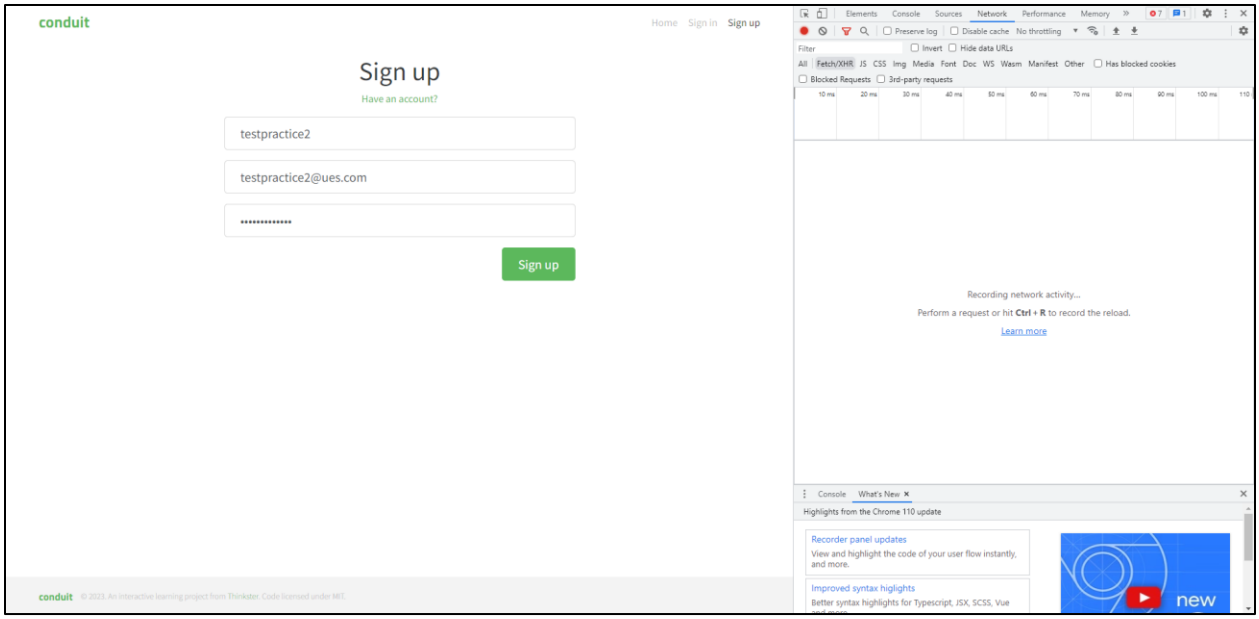
Ejecución de las pruebas

Pruebas manuales

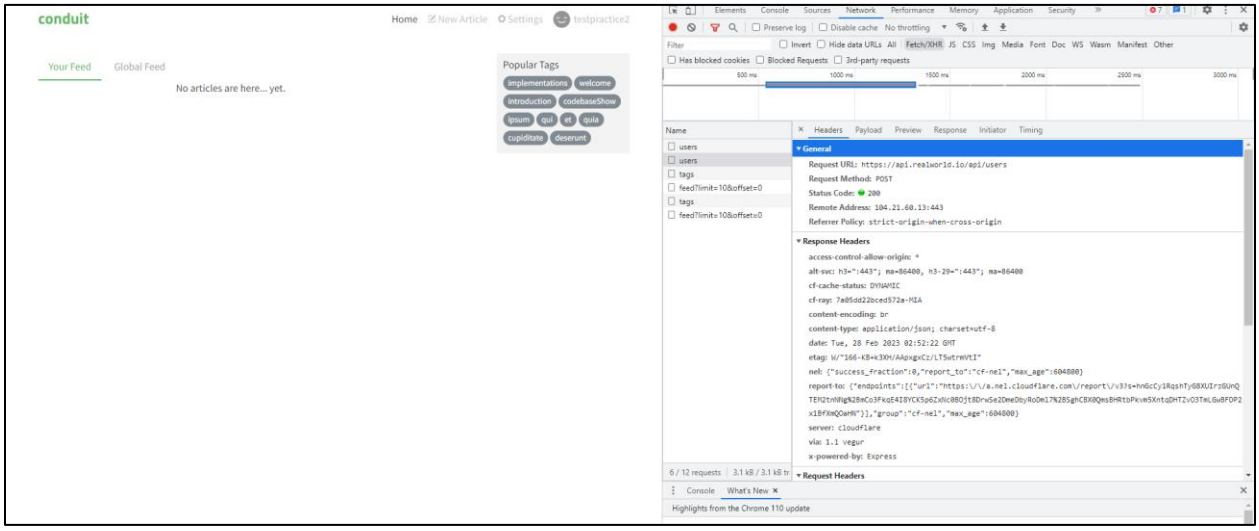
Dentro de las pruebas exploratorias al sitio, como las más importantes fueron las siguientes:

- **Crear nuevo usuario**

En esta prueba, nosotros entramos al sitio y creamos un nuevo usuario. Utilizamos las herramientas del navegador para poder capturar el contrato de entrada y salida que se usa.



Acá nosotros ponemos la información para crear usuario y damos clic en el botón de “Sign up”. En la parte derecha vemos la herramienta de inspección del navegador. A continuación, esto aparece cuando ya creamos el usuario.



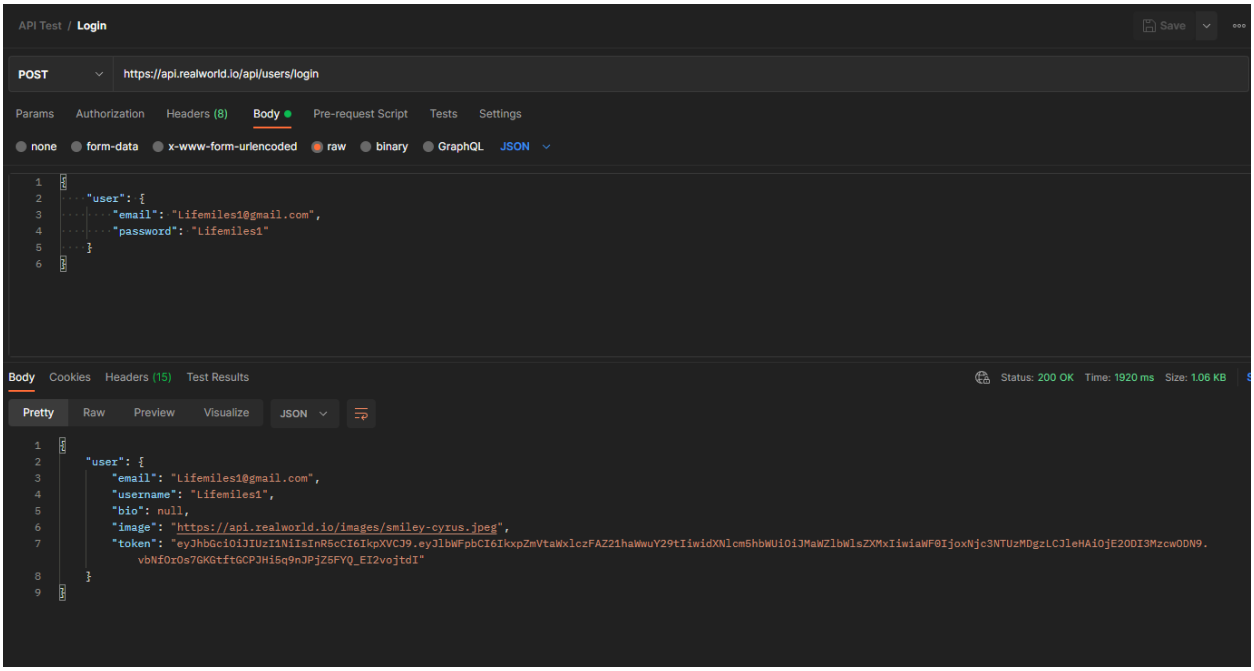
En la pestaña Headers, vemos la url del endpoint para crear usuario. En la pestaña de Preview y Response, podemos obtener los contratos de entrada y salida para hacer la prueba en Postman.

Podemos ver que tanto la petición de Preview como la de respuesta, vienen en formato JSON. Ahora nosotros tomamos esto y lo llevamos a Postman para poder ejecutar la prueba a nivel de endpoint.

- **Iniciar sesión**

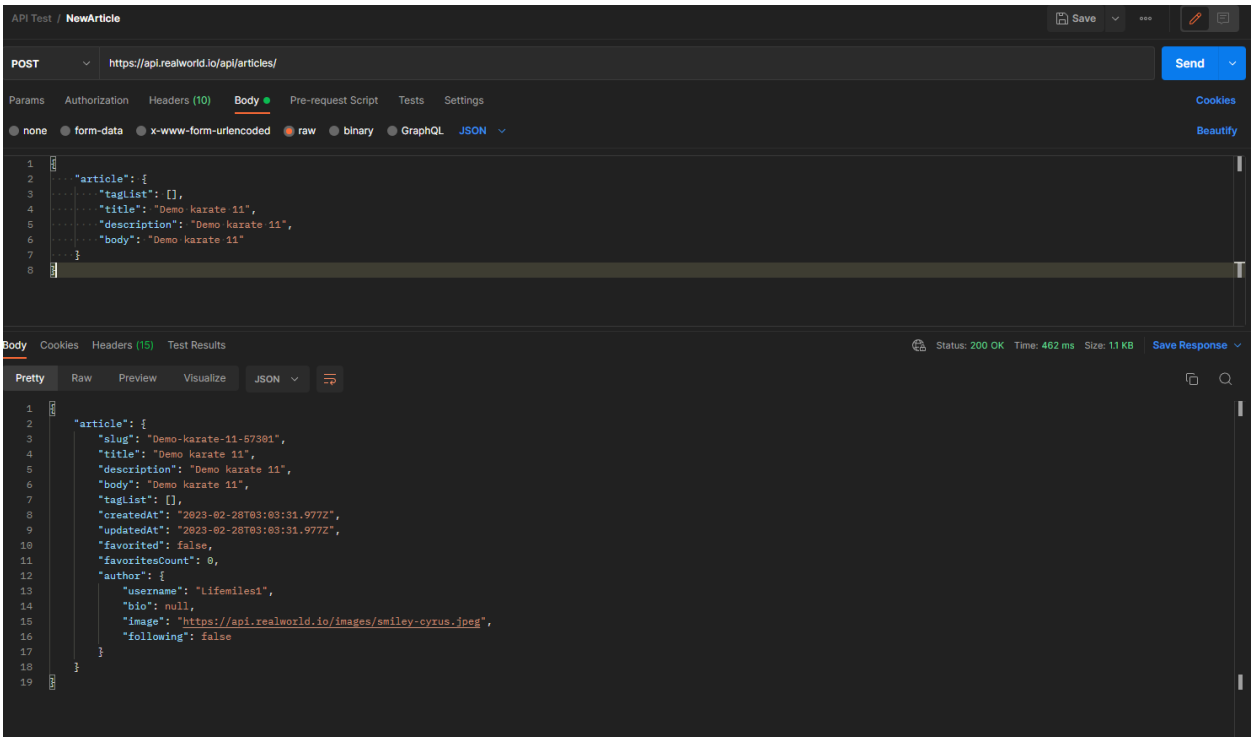
Otra prueba que realizamos fue iniciar sesión con un usuario creado previamente. El proceso que se siguió fue el mismo del caso anterior, se realizó la prueba manualmente en el sitio y se capturaron los contratos para usarlos en Postman.

A continuación, del ejemplo en Postman del inicio de sesión. Como vemos es un endpoint tipo POST, agregamos la URL y el contrato de entrada, es decir, la petición de Request. Enviamos la petición y obtenemos la respuesta en la imagen con el token de sesión.



- **Crear nuevo artículo**

También como un caso de prueba importante que se incluyó fue el de crear el nuevo artículo. Acá podemos ver el ejemplo siempre en Postman, con su URL y Request respectivo y luego de enviar la petición, su respuesta correspondiente.



Reportes

REPORTE DE FALLOS DETECTADOS

- Bugs de criticidad alta: 1
  - DFCT-01 – Cuando se edita un artículo, el formulario acepta campos vacíos.

- Bugs de criticidad media: 1
  - DFCT-02 – El formato del correo en el formulario para crear usuario no está validado.

- Bugs de criticidad baja: 1
  - DFCT-03 – En los campos de formulario para agregar crear usuario, no se valida la longitud de los campos.

REPORTE DE ESTADO DE PRUEBAS

Al terminar de ejecutar los casos de pruebas anteriormente documentados, se determina que:

- Se ejecutaron 9 casos de prueba.
- 0 casos de pruebas quedaron abiertos.
- 9 casos de prueba quedaron cerrados.
- De los casos cerrados 4 son de prioridad alta.
- De los casos cerrados 3 son de prioridad media.
- De los casos cerrados 2 son de prioridad baja.

Defectos

Se encontraron los siguientes defectos en desarrollo de las pruebas en el ambiente QA

Reporte de defecto	
Identificador	DFCT-01
Título	Cuando se edita un artículo, el formulario acepta campos vacíos
Elemento de prueba	Módulo de artículo
Entorno de prueba	Ambiente QA
Descripción	Cuando se desea editar un artículo en específico, al borrar algún campo del formulario para su edición, al dejar vacío y dar clic en el botón para guardar los cambios, el sitio no tira ningún error de validación de campos que no van llenos. Pasos para replicar el bug: 1. Se inicia sesión en la aplicación y se selecciona el artículo a editar. 2. Una vez seleccionado el artículo, se borra algún campo y se da clic en el botón para guardar cambios.
Reportado por	Elsy Castro
Prioridad	Alta

Reporte de defecto	
Identificador	DFCT-02
Título	El formato del correo en el formulario para crear usuario no está validado.
Elemento de prueba	Módulo de gestión de usuario
Entorno de prueba	Ambiente QA
Fase de pruebas	Fase de pruebas
Descripción	Al intentar crear un usuario nuevo, cuando se digita los datos en el formulario, el campo de email no valida el formato correo, es decir, que contenga arroba y el punto del .com o .net por ejemplo. Pasos para replicar bug: 1. Se entra al sitio de ConduitApp y se registra nuevo usuario. 2. Se agregar los datos necesarios y se da clic para crear usuario.

Reportado por	Félix Carpio
Prioridad	Medio

Reporte de defecto	
Identificador	DFCT-03
Título	En los campos de formulario para agregar crear usuario, no se valida la longitud de los campos
Elemento de prueba	Módulo de gestión de usuarios
Entorno de prueba	Ambiente QA
Fase de pruebas	Fase de pruebas
Descripción	Al intentar crear un usuario nuevo, cuando se digita los datos en el formulario, los campos necesarios que se solicitan no están validados en cuanto a la longitud aceptada. Pasos para replicar bug: 1. Se entra al sitio de ConduitApp y se registra nuevo usuario. 2. Se agregar los datos necesarios y se da clic para crear usuario.
Reportado por	Félix Carpio
Prioridad	Bajo

Evaluación de criterios de salida y reportes

EVALUACION DE CRITERIOS DE SALIDA

- **Criterio para dar por cerrada la ejecución de pruebas**:
  - ¿Hay un 100% de casos de pruebas ejecutados?
    - SÍ
  - ¿Ha habido casos de prueba que hayan tenido algún problema para ejecutarse?
    - NO
  - ¿Se ejecutaron los casos de prueba más prioritarios encontrados por el equipo?
    - SÍ
  - ¿Se documentaron todos los casos de prueba en el plan de pruebas?
    - SÍ

REPORTE DE FALLOS DETECTADOS

- Bugs de criticidad alta: 1

    - DFCT-01 - Cuando se edita un artículo, el formulario acepta campos vacíos.

- Bugs de criticidad media: 1
    - DFCT-02 - El formato del correo en el formulario para crear usuario no está validado.

- Bugs de criticidad baja: 1
    - DFCT-03 - En los campos de formulario para agregar crear usuario, no se valida la longitud de los campos.

REPORTE DE ESTADO DE PRUEBAS

Se terminó la ejecución de los casos de prueba previstos.

- Se ejecutaron 9 casos de prueba.
- 0 casos de pruebas quedaron abiertos.
- 9 casos de prueba quedaron cerrados.
- De los casos cerrados 4 son de prioridad alta.
- De los casos cerrados 3 son de prioridad media.
- De los casos cerrados 2 son de prioridad baja.

Cierre de actividades de pruebas

Con respecto al proceso de pruebas que ha sido ejecutado se comenta lo siguiente:

- Se detectaron defectos durante la validación de las actividades por parte del equipo de QA. Dichos defectos han sido documentados como mejoras que el equipo considera que el sitio debe modificar para dar más valor al producto.

- Los defectos documentados han sido sacados de los casos de prueba ejecutados en el sitio de ConduitApp. Dado que no tenemos acceso al código o comunicación con programadores del sitio, solamente se ha documentado todos el proceso y plan de pruebas realizado aplicando todo lo visto en la especialización de calidad.

Definición de especialización de pruebas

¿Qué es automatizar?

La calidad no es una moda. Aplicada a los procesos en una empresa asegura la satisfacción del cliente tanto interno como externo. Un flujo de trabajo ejecutado con calidad evita pérdidas. Un servicio de calidad fideliza a los clientes. Hablando de tecnología esto no es diferente. QA son las siglas de Quality Assurance, es decir, aseguramiento de calidad. Específicamente, que el software se ajuste a los requisitos del proyecto, considerando los factores de tiempo, costo y -por supuesto- calidad. Para que esto ocurra, el software debe ser sometido a pruebas, las cuales pueden automatizarse. Esto es, diseñar, desarrollar y ejecutar dichas pruebas, remplazando las tareas repetitivas. Al reducir la intervención humana, se elimina la posibilidad de error.
Con la automatización se logra reducir la cantidad de recursos asignados al testeo. Hay ahorro de tiempo una vez que la automatización está completa y así las empresas pueden dedicar mayor tiempo a tareas que sí requieran intervención de personas. Como consecuencia directa, hay reducción en el costo de mano de obra.

Específicamente con la automatización de pruebas de software, se pueden optimizar resultados de una forma muy rápida y eficaz para entregar al cliente soluciones probadas que permitan al negocio brindar seguridad e implementar una mejora continua en su propuesta de valor. Sus principales ventajas son:

- Mayor capacidad de ejecución de pruebas: Es posible realizar un gran número de pruebas en un breve período de tiempo. Las mismas pueden ser ejecutadas durante las 24 horas, los 7 días de la semana.
- Contribuye integración continua y DevOps: Las tendencias actuales para los proyectos de testing, van de la mano con la adopción de procedimientos DevOps ya que permite acelerar todo el proceso de entrega de software a producción y responder rápidamente ante las exigencias de los negocios.
- Ahorra tiempo y recursos: Las pruebas automatizadas multiplican la capacidad de los equipos de testing y las unidades de calidad de las organizaciones. Minimizan sustancialmente la ejecución de las pruebas manuales, requiriéndose estas últimas solo en escenarios especiales. Con herramientas de automatización de pruebas hay ahorros de tiempo hasta del 65%.
- Incrementa la productividad de todos los equipos: Con una herramienta de automatización, los perfiles funcionales pueden probar de forma directa las aplicaciones de su negocio, sumando eficiencia en los procesos de validación y permitiendo que el testing aporte desde fases tempranas del proceso.
- Pruebas repetibles: El trabajo que implica desarrollar una prueba automatizada es recompensado por la gran cantidad de veces que será ejecutada. Para test de regresión, las

pruebas automatizadas se podrán ejecutar una y otra vez a medida que el software evolucione, asegurando consistencia y que todo lo que funcionaba en la versión anterior, seguirá funcionando en la nueva.

- Una única herramienta para todas las plataformas: Usando herramientas de automatización, una plataforma de testing automatizado, se puede gestionar la automatización para aplicaciones móviles, web, GUI y sistemas legados.

Mayor precisión: Las pruebas automatizadas facilitan la precisión a la hora de diagnosticar la falla detectada. Proveen la evidencia mediante una serie de reportes. Además, permite mayor cubrimiento de código, requerimientos funcionales y de casos de prueba a considerar.

¿Qué son las pruebas automatizadas?

Las pruebas automatizadas son las tareas manuales que realiza una persona de QA en su rutina diaria la cual se puede correr de forma automática con ayuda de alguna herramienta.

¿Por qué realizar estas pruebas?

Realizar pruebas permite a las empresas mejorar la calidad, evitar reprocesos y acortar los ciclos de desarrollo. De igual forma, permite comparar los resultados obtenidos, frente a los esperados y los requeridos, para identificar las mejoras y las acciones a tomar para el crecimiento y fortalecimiento de la organización. Las pruebas automáticas no pretenden erradicar pruebas manuales, sino más bien, reducir el número de casos de prueba que se ejecutan manualmente, una de sus principales ventajas.

¿Para qué hacer pruebas de automatización?

La tarea manual se realiza de manera exhaustiva también, sin embargo, las pruebas automatizadas promueven más beneficios que las primeras, a continuación, se listan algunos motivos para implementarlas:

- Las tareas, flujos de trabajo o campos son fáciles de realizar en la automatización, ya que ahorra tiempo y dinero, en lugar de hacerlo manualmente.
- Se pueden ejecutar sin supervisión, es decir sin intervención humana.
- Tiene mayor cobertura de la prueba y no es propenso a errores.
- La velocidad de pruebas también se mejora cuando se realiza a través de un proceso automatizado.
- Se debe contar con un proceso para realizar pruebas de automatización, un plan y ejecución

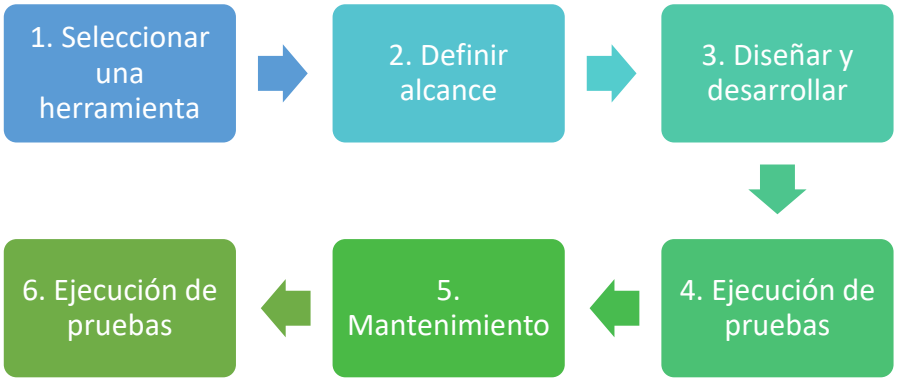
La automatización y la agrupación de los elementos correctos demostrarán ser eficientes en cuanto a costos y tiempo, además de proporcionar todos los beneficios de la automatización. “Las pruebas manuales y de automatización van de la mano para garantizar que las pruebas se realicen correctamente”. Gracias a las pruebas se puede ahorrar dinero y garantizará que todo esté a tiempo.

Para realizar pruebas automatizadas se debe tener en cuenta a la hora de seleccionar una herramienta de automatización lo siguiente:

- Definir su alcance
- Planificación, diseño y desarrollo
- Ejecución de la prueba
- Realización de mantenimiento

Proceso para automatizar una prueba

Automatizar una prueba consiste en hacer uso de herramientas diseñadas para desarrollar casos de prueba automatizados, que normalmente se realizarían de forma manual. Esto se logra generando una prueba mediante programación, replicando el flujo que normalmente se realizaría para una prueba manual. Para generar las pruebas, previo a la codificación se debe tener en mente ciertos factores para comenzar con este proceso los cuales son:



1. Seleccionar una herramienta

Existen diferentes herramientas para la creación y desarrollo de pruebas, sin embargo, algunas tendrán más o menos opciones que otras. Debemos de tener presente desde un inicio sobre qué tipo de sistema automatizará, evidentemente no es lo mismo automatizar una plataforma, instalada de manera local a uno que esté sobre una plataforma web. Por lo tanto, se debe conocer bien la base del sistema en el cual se realizarán las pruebas.

Algunas de las herramientas más famosas para automatizar en diferentes escenarios son:

- Sistemas web
    - Selenium
    - Cypress
    - Mocha

- Sistemas de escritorio
    - Winium
    - WinAppDriver
    - Ranorex

2. Definir el alcance de las pruebas

Tal vez esta es la parte más importante de todas, ya que aquí se define lo productivo que se será con el diseño y el desarrollo, hay que recordar que estas, tienen un objetivo corto y conciso, no se debe extender demasiado con lo que sé está llevando a cabo, ya que entre más se extienda una prueba, existe una mayor probabilidad de que esta falle.

3. Diseñar y desarrollar

Una vez que ya se cuenta con el alcance, se comienza a diseñar la automatización, hay que definir los pasos a seguir y hay que saber que no se puede encontrar al momento de estar en ejecución, cada factor, se debe tomar en cuenta para cumplir con un diseño claro.

Desarrollarlo es una parte importante del proceso, este es el punto donde se replica cómo se comportaría una persona en el sistema, cuánto tardaría en pasar de X a Y elemento o qué haría si se encuentra con algún elemento extraño. El mayor problema que se puede llegar a tener desarrollando las pruebas, es no apegarse lo suficiente al diseño antes planeado, lo que ocasionará que las pruebas no de los resultados esperados.

4. Ejecución de pruebas

Ya con todas las pruebas desarrolladas en su totalidad, se puede comenzar con la ejecución, manejándolas de forma unitaria para que cuando se realice un cambio en un módulo especifico, se ejecute solo esa prueba o bien de manera integrada para comprobar si algún cambio realizado en el sistema no afecta alguna otra parte.

5. Mantenimiento

Después de cada prueba se obtiene una retroalimentación, esta servirá para saber si la prueba se realizó de forma correcta o si bien, hay que modificar algo en el diseño para poder mejorar el porcentaje de éxito. No se debe olvidar que cada cambio en el sistema puede afectar las pruebas, por lo tanto, es necesario darles mantenimiento constante para confirmar que todo sigue en orden.

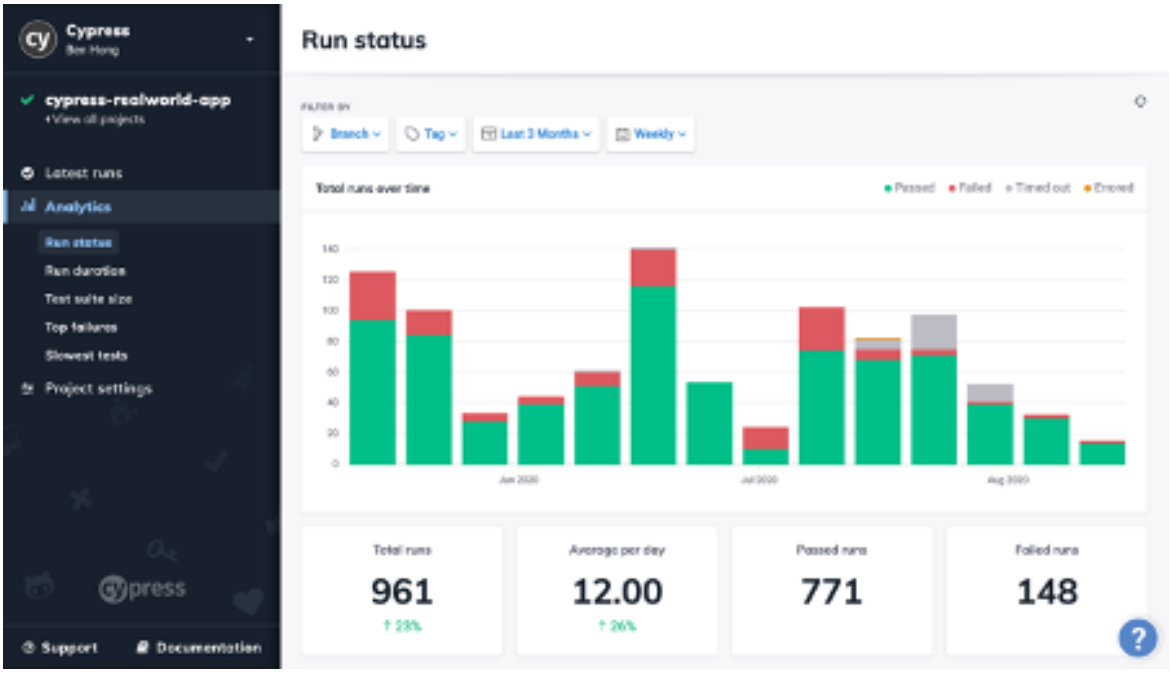
6. Implementación

Este puede llegar a ser un proceso totalmente opcional, sin embargo, puede todavía aún más trabajo.

Suponiendo que se tiene un sistema web, el cual tiene sus 4 principales ramas: Desarrollo, QA, Producción y Diseño, existen dashboards (dependiendo de la herramienta que utilicemos) los cuales, permitirán hacer la sincronización con el repositorio, con el fin de centralizar las pruebas a ejecutar en una sola herramienta.

Los dashboards son herramientas que ayudarán a visualizar de manera clara las diferentes pruebas que se ejecutan en el sistema, el tiempo promedio de ejecución y hasta la capacidad de dividir la carga de trabajo, en diferentes nodos con el objetivo de mejorar, el tiempo que lleve a cabo realizar las pruebas, de esta manera cada vez que se detecte un cambio, en cualquiera de las ramas.

Las pruebas previamente definidas y desarrolladas se ejecutarán, dando esta retroalimentación inmediata, sin tener que realizar pruebas manuales o ejecutar la automatización por cuenta propia, generando así una integración continua con lo que se esté desarrollando.



Las pruebas automatizadas permiten ahorrar tiempo en la creación de una aplicación, tiempo que la mayoría de las veces es necesario. De la misma forma que se menciona al inicio, se debe recordar que esto no es una batalla entre pruebas manuales y automatizadas, es buscar generar un nexo entre estas dos para poder tener una mayor productividad y fiabilidad.

Diferencia entre prueba manual y prueba automatizada

¿Cuándo se deben implementar las Pruebas Manuales?

Sabiendo que las pruebas manuales son parte importante durante todo el ciclo de vida del desarrollo y tomando en cuenta que se necesitan habilidades como experiencias humanas para su planificación y ejecución, resultan apropiadas para los siguientes casos: pruebas exploratorias, pruebas de usabilidad, pruebas ad-hoc entre clasificaciones de pruebas relacionadas.

Ventajas del Testing Manual

- No se requiere conocimiento sobre programación o desarrollo de software.
- Se obtienen resultados desde una perspectiva humana en condición de usuario final.
- Ejecución de pruebas exploratorias (estas son ejecutadas sólo de forma manual).
- Comprensión y percepción directa del problema.
- Ejecución de pruebas en dispositivos móviles utilizando gestos.
- Recomendable para diseños GUI que cambian o actualizan de forma dinámica.
- El costo generado en las pruebas manuales es menor, ya que no se requieren herramientas y procesos de automatización.

¿Cuándo se deben implementar las Pruebas Automatizadas?

Teniendo en cuenta que existen diversos contextos respecto a cuándo y qué se debe automatizar, en este caso hablaremos del tipo de pruebas más relevantes para implementarla:

- Pruebas de Regresión: para asegurar que las mejoras y correcciones realizadas en el software funcionan de manera eficiente y no afectan la funcionalidad existente del software.
- Pruebas de Carga: para simplificar las complicaciones involucradas en la generación de la carga y la simulación de las condiciones de carga de datos.
- Ejecución Repetitiva: para las pruebas que implican la ejecución de una sola tarea una y otra vez
- Pruebas de Rendimiento: para simular la interacción de miles de usuarios simultáneos a la vez.

Ventajas del Testing Automatizado

- Fiabilidad técnica en procesos y en operación de equipos.
- Reducción del tiempo empleado en procesar la información.
- La automatización no requiere de intervención humana, por lo que se puede ejecutar la prueba automatizada de forma desatendida.
- Aumenta la velocidad de ejecución de la prueba.

- Es lo más recomendable cuando se necesitan ejecutar diversos casos de prueba de manera repetitiva y por un período de tiempo extenso.
- Pueden ejecutarse una y otra vez luego de ser creadas.
- Minimiza los errores humanos al momento de la ejecución del testing.
- Facilita la integración de trabajo entre desarrolladores y testers.

Importancia de automatización de pruebas[1]

La automatización de pruebas puede traer consigo una serie de beneficios que podrían ayudar a elevar el nivel de calidad del producto software y también a disminuir su coste. Los más destacados podrían ser:

- Se pueden ejecutar un número mayor de pruebas.
- Una vez que se ha automatizado un determinado caso de prueba, es fácil ejecutarlo sucesivas veces variando solo sus datos de entrada.
    - **Consecuencia**: La cobertura aumenta, se prueban muchas más combinaciones, aumentando el nivel de confiabilidad del producto software.
- Se pueden ejecutar pruebas de forma desatendida.
    - La ejecución se puede lanzar a cualquier hora del día, durante la noche, en periodos no laborables, etc.
    - **Consecuencia:** Se reduce considerablemente el tiempo de la fase de ejecución de pruebas. Menor coste.
- Reducción de errores durante la ejecución de las pruebas.
- Una prueba automatizada se ejecuta siempre de la misma manera, mientras que en una ejecución manual se pueden cometer errores, sobre todo, dependiendo del nivel del tester que realice dicha ejecución y de la complejidad de la prueba
    - **Consecuencia:** Aumenta la calidad de las pruebas. Posible aumento también de la calidad general del producto software.
- Ayuda a estandarizar procesos.
    - Implantar un sistema de automatización de pruebas pasa por analizar bien los procesos que se realizan durante la ejecución de pruebas, lo que ayuda a estandarizarlos y a crear una fase de pruebas más consistente.
    - **Consecuencia:** Aumenta la calidad de las pruebas. Posible aumento también de la calidad general del producto software.

[1] Testing y calidad de software, Automatización de Pruebas con Selenium WebDriver. Rafael Cubas Montenegro. Capítulo 3.3, beneficios de la automatización de pruebas, pág. 24 https://oa.upm.es/49320/1/PFC_RAFAEL_CUBAS_MONTENEGRO.pdf

- Facilita las pruebas de regresión.
    - La automatización de pruebas favorece que las pruebas de regresión se realicen sin apenas coste, ya que la repetición de pruebas no supone una inversión grande de tiempo. Consecuencia: Se reduce el coste durante las fases de mejora o evolución del producto software.

También hay que tener en cuenta diversos factores antes de decidir si merece la pena invertir tiempo y dinero en automatizar pruebas. Si el proceso de automatización no se realiza adecuadamente muchos de los posibles beneficios que en teoría debería aportar se pueden volver en nuestra contra.

Por ejemplo, seguramente se necesite personal más cualificado para realizar la automatización. La falta de experiencia puede provocar que las pruebas no estén bien diseñadas, que surjan problemas técnicos que alarguen el proceso de creación de los test automáticos, etc., provocando que el nivel de calidad de las propias pruebas sea insuficiente, aportando mucha incertidumbre sobre el nivel de calidad del producto software final.

Otro problema puede venir de una mala decisión sobre lo que se automatizará y sobre lo que no. Existen algunos procesos que por su complejidad es posible que sea más costoso automatizarlos que realizarlos manualmente. Es necesario saber bien dónde invertir los esfuerzos de automatización.
Lo principal para que la automatización pueda contribuir al éxito del proyecto, es elegir una buena estrategia de pruebas, identificando las partes donde puede suponer un beneficio el realizar procesos automáticos. Por ello, es necesario tener en cuenta que:

- No todo se puede automatizar. Un objetivo realista podría ser entorno al 50 % de las pruebas.
- Comenzar automatizando las tareas repetitivas más básicas que consumen una cantidad significativa de tiempo.
- La automatización se basa en la reutilización, si un test se diseña para ser ejecutado una sola vez, no tiene sentido automatizarlo.
- Los test que necesitan ejecutarse para cada compilación son los mejores candidatos para ser automatizados.
- Los test que utilizan múltiples valores para las mismas tareas, también son buenos candidatos.
- En cada caso, es necesario realizar una valoración de si realmente la automatización puede acarrear algún beneficio frente al testing manual.
- Los test con resultados NO predecibles deben ser descartados para el proceso de automatización.

Pruebas automatizadas dentro del proceso de pruebas

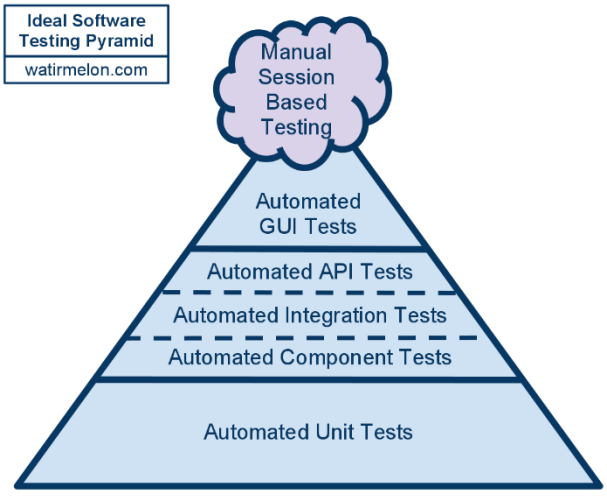
Mike Cohn describe en su libro “Succeeding with Agile”, una pirámide de pruebas que se divide en tres niveles de automatización, en los que va disminuyendo la complejidad las pruebas ya que la construcción de los escenarios ya no será tan rápida y el tiempo de ejecución irá aumentando.

Esta también representa el esfuerzo o la cantidad de test automáticos que debería tener cada nivel, empezando por el más bajo, donde se debería concentrar el mayor número de test automáticos, hasta llegar al tercer nivel, el de la interfaz de usuario, donde el número de pruebas, por su complejidad, debería ser menor.

Los niveles de prueba antes mencionado son:

- **Automatización de pruebas unitarias:** Ubicadas en la base de la pirámide y que representan la parte primordial para detectar defectos y donde debemos enfocar las pruebas.
- **Pruebas de servicio:** Permite probar la comunicación entre los diferentes componentes de una aplicación, por ejemplo, como interactúa con una base de datos, con una API externa o con diferentes módulos de la aplicación misma.

- **Pruebas de interfaz de usuario.** Comprende pruebas de un recorrido de un usuario final por toda la aplicación con el fin de asegurar que la aplicación funcione de acuerdo a lo esperado.



¿Qué es Karate DSL?

Karate es una herramienta de código abierto, fácil de manejar para personas que no poseen conocimientos de programación. Esta herramienta combina la automatización de API y las pruebas de rendimiento en un marco único. No requiere compilar código y permite mezclar la automatización de pruebas de API con UI en el mismo script de prueba.
Además, Karate tiene una API de Java que permite volver más potentes las capacidades de automatización, escribiendo las pruebas en una sintaxis simple diseñada para HTTP, JSON, GraphQL y XML.

Descripción de la herramienta

- Hace uso del lenguaje Gherkins fácil de entender.
- No requiere conocimientos de programación técnica como Java.
- Se basa en los estándares populares de Cucumber.
- Fácil de crear un marco.
- Las pruebas paralelas son la funcionalidad principal que proporciona el mismo Karate, por lo tanto, no necesitamos depender de Maven, Gradle , etc.
- Interfaz de usuario para depurar la prueba.
- Llamar a un archivo de características desde otro archivo.
- Proporciona soporte para las pruebas de controladores de datos que se construyen internamente, por lo que no es necesario depender de marcos externos.
- Informes de descanso nativos integrados. Además, se puede integrar con Cucumber para obtener mejores informes de IU y más claridad.
- Proporciona soporte interno para cambiar la configuración en diferentes entornos de prueba (QA, Stage, Prod, Pre-Prod).
- Soporte perfecto para la integración de CI / CD que puede resultar útil.
- Capaz de manejar varias llamadas HTTP:
    - Soporte Web Socket
    - Solicitud SOAP
    - HTTP
    - Manejo de cookies del navegador
    - HTTPS
    - Datos en formato HTML
    - Solicitud XML

Behavior Driven Development (BDD)

El desarrollo basado en el comportamiento (BDD) es una metodología ágil de desarrollo de software en la que una aplicación se documenta y diseña en torno al comportamiento que un usuario espera experimentar al interactuar con ella. Al centrarse los desarrolladores en los

comportamientos solicitados de una aplicación o programa, BDD ayuda a evitar la sobrecarga, el código excesivo, las funciones innecesarias o la falta de atención. Esta metodología combina, aumenta y refina las prácticas utilizadas en el desarrollo basado en pruebas (TDD) y las pruebas de aceptación.

Un proyecto típico que utiliza el desarrollo basado en el comportamiento comenzaría con una conversación entre los desarrolladores y el cliente para formar una imagen general de cómo se pretende que funcione un producto. Las expectativas para el comportamiento del producto se establecen como objetivos para los desarrolladores y, una vez que se superan todas las pruebas de comportamiento, el producto cumple con sus requisitos y está listo para su entrega al cliente.

BDD ofrece la capacidad de ampliar el grupo de aportes y comentarios para incluir a las partes interesadas del negocio y a los usuarios finales que pueden tener poco conocimiento sobre desarrollo de software. Debido a este ciclo de retroalimentación ampliado, los equipos de desarrollo pueden usar BDD más fácilmente en entornos de integración continua y entrega continua.

BDD y Ágil

Si el equipo ya está utilizando algún tipo de metodología ágil, planificando el trabajo en pequeños incrementos de valor como Historias de usuarios. BDD no reemplaza su proceso ágil existente, lo mejora.

Se debe pensar en BDD como un conjunto de complementos para un proceso existente que hará que el equipo sea más capaz de cumplir las promesas de agilidad: lanzamientos oportunos y confiables de software funcional que satisfaga las necesidades cambiantes de su organización, lo que requiere algo de disciplina y esfuerzo de mantenimiento.

Gherkin, el lenguaje para BDD

El lenguaje usado para BDD es Gherkin, ya que es un lenguaje fácil de usar, pues es similar al lenguaje natural de las personas. Se puede utilizar tanto para la realización de pruebas como para escribir historias de escenarios de usuarios.

Por su simplicidad, el formato Gherkin, puede ser escrito por individuos sin grandes experiencias en el área de programación. Además, a pesar de que es común ver procesos realizados en Ingles, Gherkin permite más de treinta idiomas para trabajar.

Generalmente las pruebas realizadas con Gherkin son almacenadas en archivos “.Feature”. Estos archivos deben ser versionados junto al código fuente de la aplicación que se está testeando.

Gherkin está compuesto por varios elementos que permiten esta comunicación entre perfiles de negocio y técnicos de forma sencilla. Gherkin tiene elementos de tipo características, de tipo

comportamientos, de tipo acciones. Pero todo en lenguaje natural, que un perfil de negocio entienda y un técnico pueda trasladar a código.

Estos archivos contienen los siguientes términos:

- **Feature**: Es el nombre de la función que se va a testear. También se puede considerar el nombre de la prueba.
- **Scenario**: Es el escenario de la función a poner a prueba.
- **Given**: Son las condiciones previas a la acción de la función.
- **When**: El conjunto de acciones que se ejecutarán
- **Then**: Los resultados que se esperan con el testeo. A partir de ellos, se realizan las validaciones.

Ejemplo

- **Given**: Dado que me estoy registrando para una prueba gratuita (Contexto)
- **When**: Cuando envío los detalles requeridos (Sucede una acción)
- **Then**: Entonces recibo un enlace a la descarga. (Resultados)

Usualmente debe haber solo una función por archivo y en ella puede haber diversos escenarios de prueba.

Beneficios de la herramienta de pruebas con Karate Framework

Ventajas y desventajas

Ventajas

- Fácil de comenzar con poco conocimiento de código. Debido a que utiliza un lenguaje con estilo similar a Gherkin de fácil aprendizaje.
- Soporte Nativo para JSON. Se puede escribir expresiones JSON en los archivos requerido y serán automáticamente reconocidos por el Framework
- Validaciones JSON muy poderosas
- Soporte para Java y JavaScript
- Ejecución paralela multihilo
- Reportería y reportes detalladas
- Pruebas de rendimiento con Gatling. Karate posee una excelente integración esta herramienta para pruebas de rendimiento.

Desventajas

- Propio lenguaje de script (de estilo similar a Gherkin)
- No tiene soporte para ningún tipo de sensibilidad inteligente.
- Puede ser difícil detectar los errores manualmente.

Comparativa con herramienta Cucumber

Característica	CUCUMBER	KARATE DSL
Definiciones de pasos incorporadas	❌ NO. Debe seguir implementándolos a medida que crece su funcionalidad. Esto puede volverse muy tedioso, especialmente porque para la inyección de dependencia, estás solo.	✅ SI. No se necesita código Java adicional.
Capa única de código para mantener	❌ NO. Hay 2 capas. Los archivos spec o feature de Gherkin forman una capa, y también tendrá las definiciones de pasos de Java correspondientes.	✅ Si. Solo 1 capa de Karate-script (basado en Gherkin).
Especificación legible	✅ SI. Cucumber se leerá como un lenguaje natural si implementa correctamente las definiciones de pasos.	❌ NO. Aunque Karate es simple y un verdadero DSL, en última instancia es un mini lenguaje de programación. Pero es perfecto para probar servicios web a nivel de solicitudes y respuestas HTTP.
Reutilizar archivos de funciones	❌ NO. Cucumber no admite la posibilidad de llamar (y, por lo tanto, reutilizar) otros archivos *.feature desde un script de prueba.	✅ SI
Pruebas dinámicas basadas en datos	❌ NO. El Esquema de Escenario de Cucumber espera que los Ejemplos contengan un conjunto fijo de filas.	✅ SI: El soporte de Karate para llamar a otros archivos *.feature le permite usar una matriz JSON como fuente de datos y puede usar JSON o incluso CSV directamente en un archivo Scenario Outline.
Ejecución en Paralelo	❌ NO	✅ SI. Karate ejecuta incluso scripts de Scenario en paralelo, no solo scripts de Feature-s.
Ejecute las rutinas de 'Configuración' solo una vez	❌ NO. Cucumber tiene una limitación en la que los pasos en segundo plano se vuelven a ejecutar para cada Scenario y peor aún para cada fila Examples dentro de un archivo Scenario Outline.	✅ SI
Motor de JavaScript integrado	❌ NO. Y debe implementar su propio enfoque para la configuración específica del entorno y preocuparse por la inyección de dependencia.	✅ SI. Defina fácilmente todos los entornos en un solo archivo y comparta variables en todos los escenarios. Capacidad completa de secuencias de comandos a través de la interoperabilidad JS o Java.

Ejemplo de escenario en Cucumber:

```
Feature: Contract test to payment-coordinator-svc

  @payment-coordinator-svc @pay @ID-1
  Scenario Outline: I want to validate the contract of the next service payment-coordinator-svc
    When I made the service call service payment-coordinator-svc in the endpoint pay "payment-coordinator-svc/pay" with next parameters "<request>"
    Then Verify the contract of "paySchema"

    Examples:
      | request          |
      #| pay-RedeemCOM    |
      | pay-BuyPointsCOM |
      | pay-TransferCOM  |
```

Implementación de las pruebas

Justificación de elección de casos de pruebas para automatización

Como equipo, se realizó un análisis para seleccionar los casos de prueba que se automatizarían. A continuación, se detallan los criterios que se tomaron en cuenta:

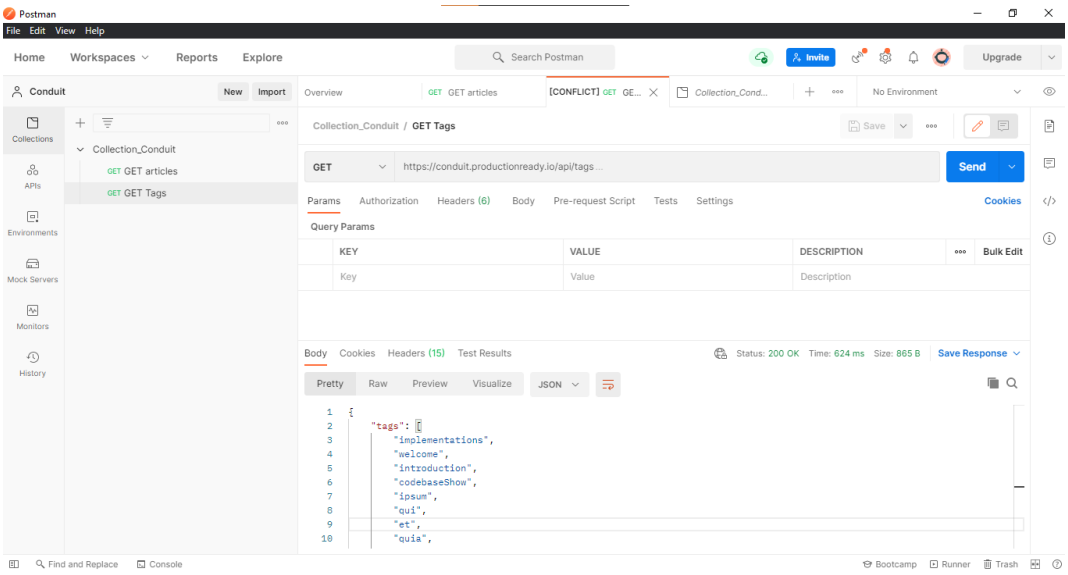
- Se decidieron automatizar los siguientes casos debido a la prioridad que el equipo estimó, la mayoría son acerca de las funcionalidades principales como el inicio de sesión o la gestión de los artículos.
- Otro criterio que se tuvo fue la facilidad para codificar y automatizar. Los casos de prueba que se automatizaron son posibles de codificar y pueden ser ejecutados las n veces que se desee.
- Los casos de pruebas que se automatizaron en un futuro si se quiere implementar esta herramienta de Karate en un ambiente más completo, dichas pruebas automatizadas pueden usarse para pruebas de regresión de las funcionalidades actuales, si en su momento se desean hacer cambios al sitio de ConduitApp

Se adjunta enlace para el repositorio de pruebas automatizadas:
https://github.com/AlexVilleda/Karate-Tests

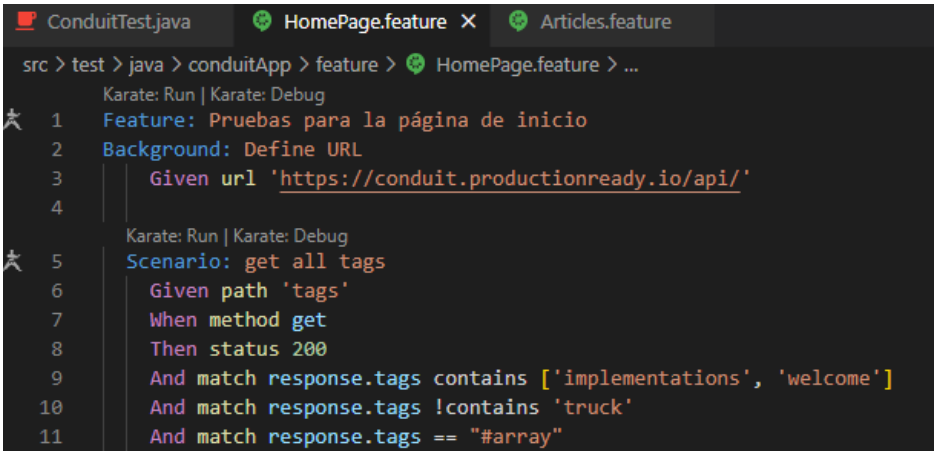
A continuación, se presentan algunos de los casos de prueba ejecutados, los demás pueden ser consultados en el repositorio antes mencionado.

Obtener etiquetas

En la siguiente imagen se presenta la estructura del Get Request para obtener el listado de las etiquetas.



En Karate se escribe el escenario de la siguiente forma:

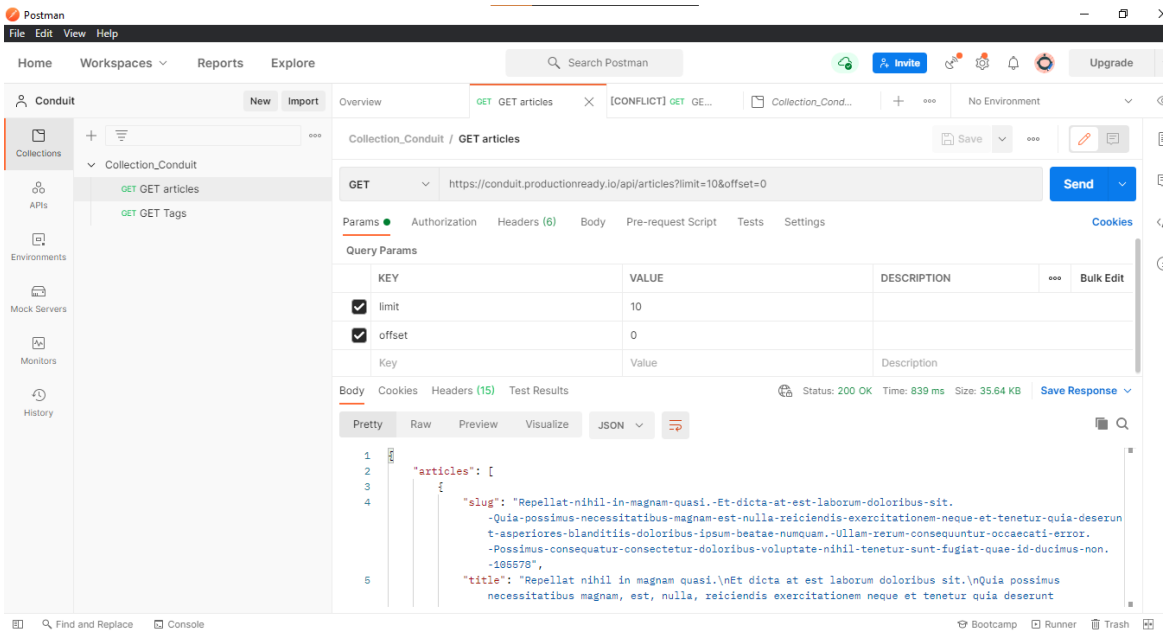


Al correr el escenario, en la terminal podemos visualizar el resultado obtenido.

```
PROBLEMS 4   OUTPUT   TERMINAL   DEBUG CONSOLE                                  powershell

2 < NEL: {"success_fraction":0,"report_to":"cf-nel","max_age":604800}
2 < Server: cloudflare
2 < CF-RAY: 76ef0bb61b7667c0-MIA
2 < alt-svc: h3=":443"; ma=86400, h3-29=":443"; ma=86400
{"tags":["implementations","welcome","introduction","codebaseShow","ipsum","qui","et","quia","cupiditate","deserunt"]}

-------------------------------------------------------
feature: classpath:conduitApp/feature/HomePage.feature
scenarios:  1 | passed:  1 | failed:  0 | time: 1.6402
-------------------------------------------------------

21:26:20.332 [pool-2-thread-1] INFO  com.intuit.karate.Suite - <<pass>> feature 1 of 1 (0 remaining) classpath:conduitApp/feature/HomePage.featu
re
Karate version: 1.2.0
======================================================
elapsed:   4.82 | threads:    5 | thread time: 1.64
features:     1 | skipped:    0 | efficiency: 0.07
scenarios:    1 | passed:     1 | failed: 0
======================================================

HTML report: (paste into browser to view) | Karate version: 1.2.0
file:///C:/Users/margh/Desktop/Karateproject/target/karate-reports/karate-summary.html
======================================================================

[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 5.597 s - in conduitApp.ConduitTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
```

Mostrar 10 artículos

Se presenta el GET Request en Postman, la estructura del escenario en Karate y el resultado de la ejecución.



```
Karate: Run | Karate: Debug
Scenario: Obtener 10 artículos de la página
    Given params { limit: 10, offset: 0}
    Given path 'articles'
    When method get
    Then status 200
    And match response.articles == '#[10]'
```

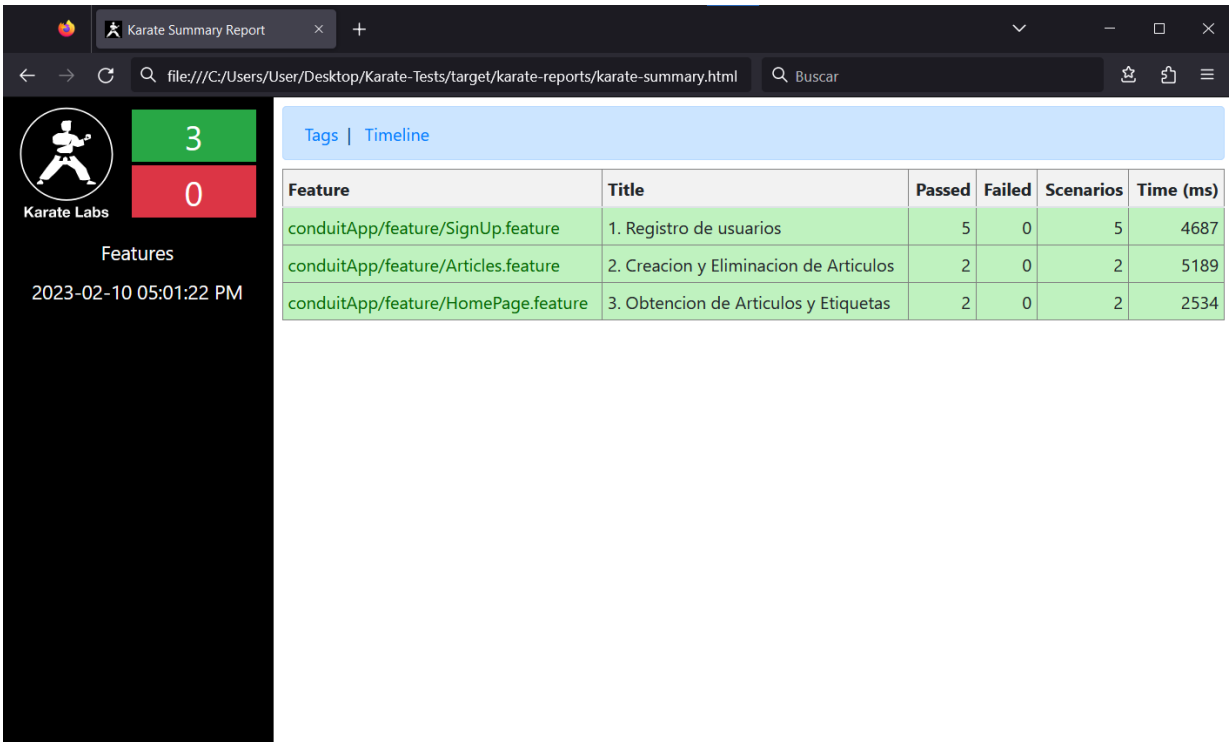
```
PROBLEMS 4  OUTPUT  TERMINAL  DEBUG CONSOLE                    powershell

scenarios:    2 | passed:    2 | failed: 0
======================================================

HTML report: (paste into browser to view) | Karate version: 1.2.0
file:///C:/Users/margh/Desktop/Karateproject/target/karate-reports/karate-summary.html
------------------------------------------------------------------------

[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 5.435 s - in conduitApp.ConduitTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
```

Reporte de ejecución de pruebas automatizadas generados por Karate Framework

Al finalizar la ejecución de las pruebas automatizadas con el comando **mvn test,** se muestra los resultados de las pruebas indicando valores como el número de escenarios de prueba ejecutados, la cantidad de escenarios ejecutados con éxito y fallidos, así como el tiempo de ejecución total de las pruebas, en segundos.

```
TERMINAL  PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN                    bash

16:39:30.635 [pool-2-thread-1] INFO  com.intuit.karate.Suite - <<pass>> feature 2 of 3 (1 remaining) classpath:conduitApp/feature/HomePage.feature
feature: classpath:conduitApp/feature/SignUp.feature
scenarios:  5 | passed:  5 | failed:  0 | time: 6.5726

16:39:31.926 [pool-2-thread-1] INFO  com.intuit.karate.Suite - <<pass>> feature 3 of 3 (0 remaining) classpath:conduitApp/feature/SignUp.feature
Karate version: 1.2.0
elapsed:  22.51 | threads:    5 | thread time: 16.44
features:     3 | skipped:    0 | efficiency: 0.15
scenarios:    9 | passed:     9 | failed: 0

HTML report: (paste into browser to view) | Karate version: 1.2.0
file:///C:/Users/Alex%20Villeda/Desktop/Karate-Tests/target/karate-reports/karate-summary.html

[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 24.069 s - in conduitApp.ConduitTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  34.009 s
[INFO] Finished at: 2023-02-21T16:39:34-06:00
[INFO] ------------------------------------------------------------------------

Lín. 1, col. 1  Espacios: 4  UTF-8  CRLF  Karate
```

En la misma ventana también se provee un enlace con el reporte de resultados de las pruebas en formato HTML, el cual puede ser visualizado pegando el directorio mostrado en la consola dentro de la barra de direcciones del navegador web de preferencia y presionando Enter.



Se muestran los conjuntos de pruebas realizados indicando la ruta donde se encuentran, el título dado a ese conjunto de pruebas, la cantidad de escenarios en total, la cantidad de escenarios ejecutados con éxito, la cantidad de escenarios ejecutados con error y el total de tiempo en que se corrió ese conjunto de pruebas, medidos en milisegundos.

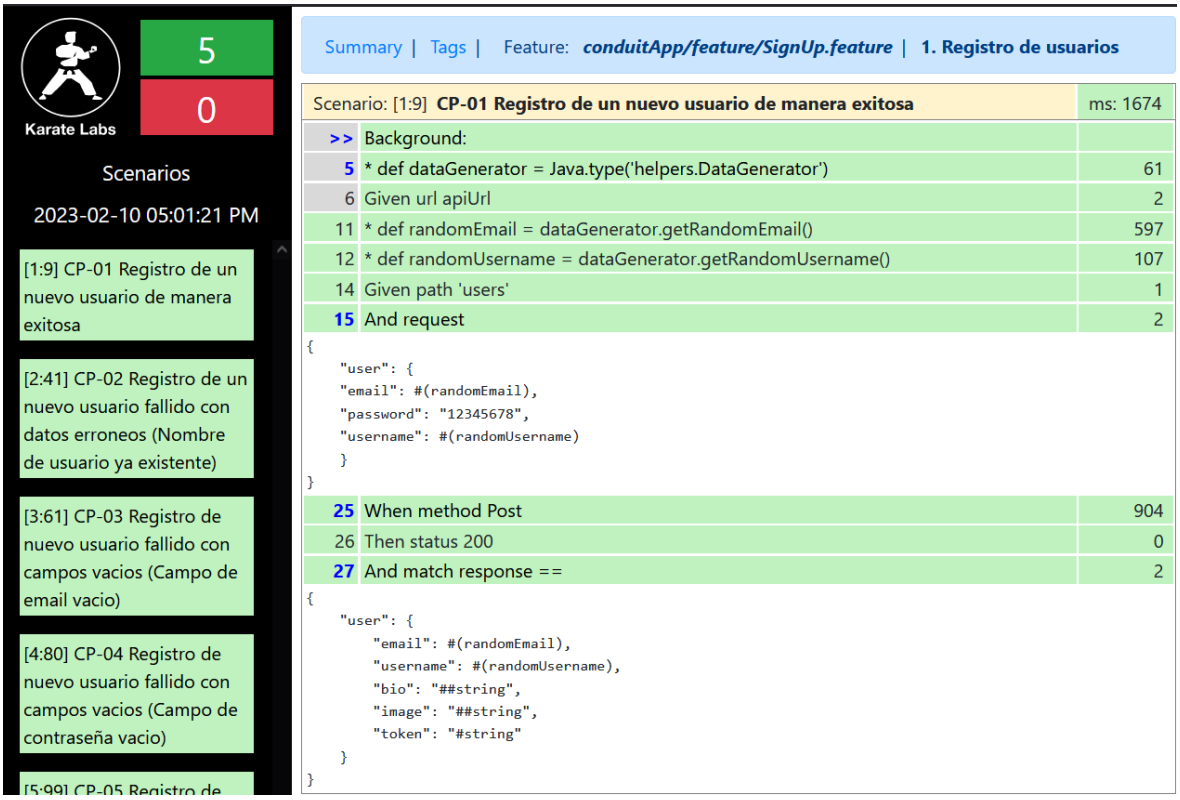
En el listado de pruebas se muestran agrupadas por el archivo en que fueron contenidas.
Teniendo en total 3 conjuntos de pruebas que contienen 9 escenarios de prueba:

- SignUp.feature: Registro de usuarios
  - CP-01 Registro de un nuevo usuario de manera exitosa
  - CP-02 Registro de un nuevo usuario fallido con datos erróneos (Nombre de usuario ya existente)
  - CP-03 Registro de nuevo usuario fallido con campos vacíos (Campo de email vacío)
  - CP-04 Registro de nuevo usuario fallido con campos vacíos (Campo de contraseña vacío)

    - CP-05 Registro de nuevo usuario fallido con campos vacíos (Campo de nombre de usuario vacío)
- Articles.feature: Creación y eliminación de artículos
    - CP-06 Creacion de nuevo artículo de forma exitosa
    - CP-07 Eliminar artículo de forma exitosa
- HomePage.feature: Obtención de artículos y etiquetas
    - CP-08 Consultar artículos creados
    - CP-09 Visualizar etiquetas utilizadas

Al seleccionar un conjunto de pruebas se amplía el resultado de ejecución de los escenarios de pruebas contenidos dentro del conjunto.

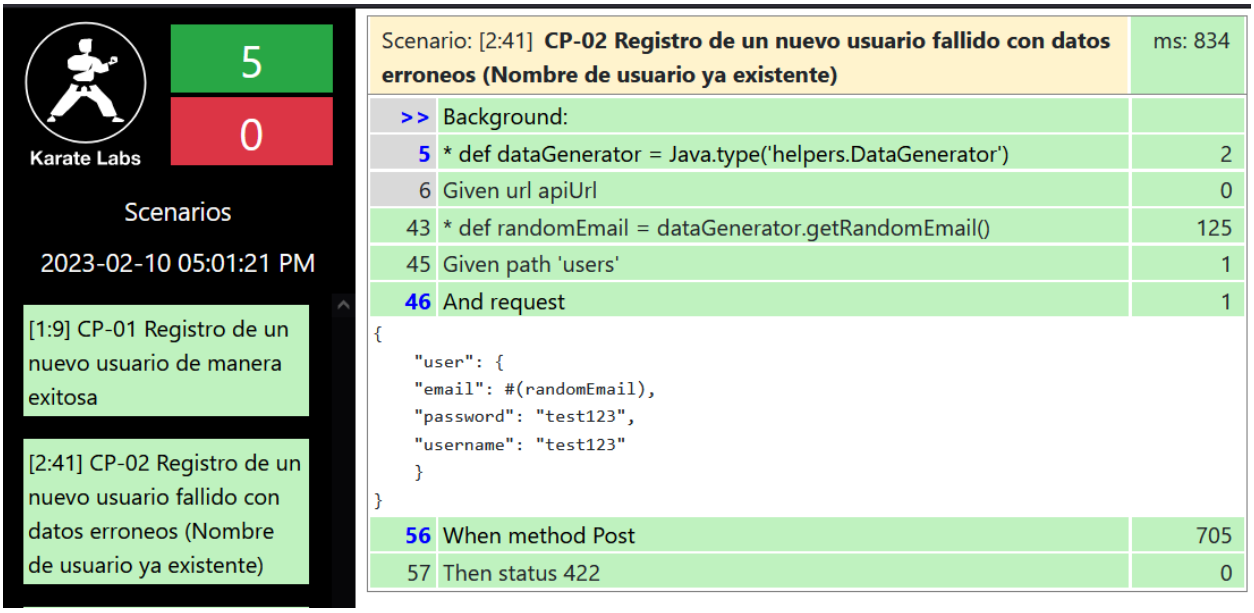
CP-01 Registro de un nuevo usuario de manera exitosa



- Previo a la ejecución del escenario se llamó a la función Java generadora de datos aleatorios y se inicializó la URL de la API de pruebas
- Se generó y definió un correo electrónico aleatorio
- Se generó y definió un nombre de usuario aleatorio
- Se estableció el endpoint ‘users’ para ser consultado

- Se preparó una petición HTTP con el cuerpo de la petición en formato JSON con los datos del nuevo usuario a ser creado:
    - Email: (Correo electrónico aleatorio generado previamente)
    - Password: 12345678
    - Username: (Nombre de usuario aleatorio generado previamente)
- Se envió la petición por medio de un método POST
- Se recibió un status 200 indicando que la solicitud ha tenido éxito
- La respuesta recibida por el servidor cumplió exitosamente con el esquema JSON esperado:
    - Email: El email de la respuesta es el mismo email de la petición enviada
    - Username: El username de la respuesta es el mismo que el de la petición enviada
    - Bio: El de campo de biografía en la respuesta es un campo de tipo String o en su defecto, un campo vacío
    - Image: El campo de imagen de usuario en la respuesta es un campo de tipo String o en su defecto, un campo vacío
    - Token: El campo del token de sesión en la respuesta es un campo de tipo String

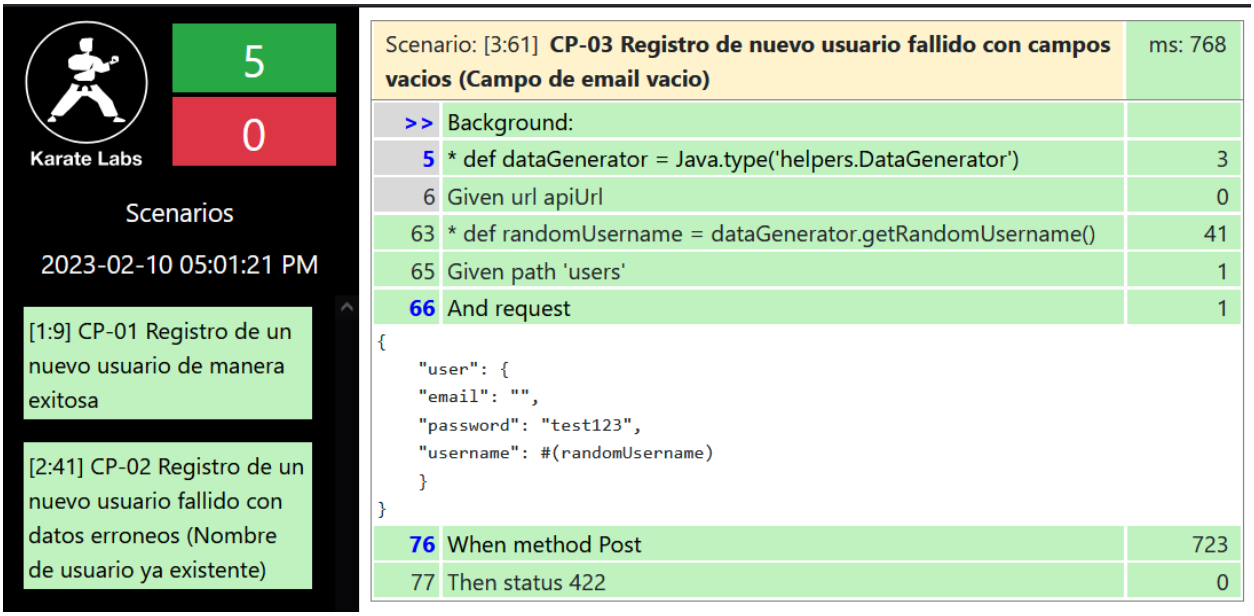
CP-02 Registro de un nuevo usuario fallido con datos erróneos (Nombre de usuario ya existente)



- Previo a la ejecución del escenario se llamó a la función Java generadora de datos aleatorios y se inicializó la URL de la API de pruebas
- Se generó y definió un correo electrónico aleatorio
- Se estableció el endpoint ‘users’ para ser consultado

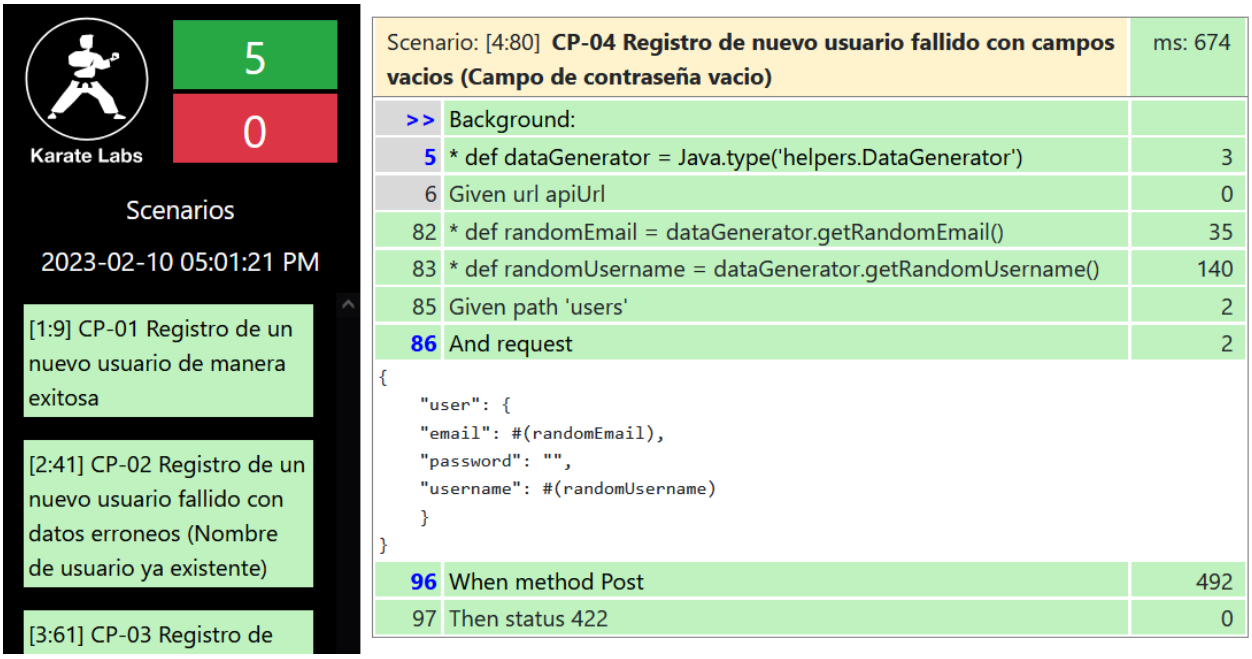
- Se preparó una petición HTTP con el cuerpo de la petición en formato JSON con los datos del nuevo usuario a ser creado:
    - Email: (Correo electrónico aleatorio generado previamente)
    - Password: test123
    - Username: test123
- Se envió la petición por medio de un método POST
- Se recibió un status 422 indicando que hubo un error del lado del cliente, debido al uso de un nombre de usuario ya existente en el sitio

CP-03 Registro de nuevo usuario fallido con campos vacíos (Campo de email vacío)



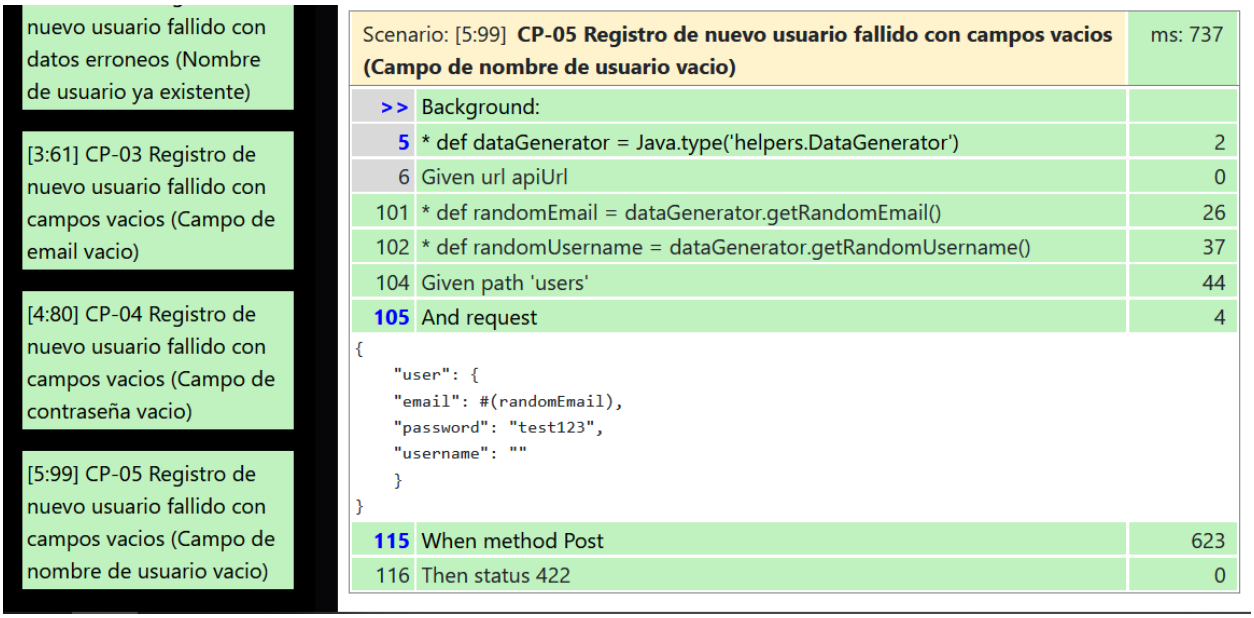
- Previo a la ejecución del escenario se llamó a la función Java generadora de datos aleatorios y se inicializó la URL de la API de pruebas
- Se generó y definió un nombre de usuario aleatorio
- Se estableció el endpoint ‘users’ para ser consultado
- Se preparó una petición HTTP con el cuerpo de la petición en formato JSON con los datos del nuevo usuario a ser creado:
    - Email: (Campo vacío)
    - Password: test123
    - Username: (Nombre de usuario aleatorio generado previamente)
- Se envió la petición por medio de un método POST
- Se recibió un status 422 indicando que hubo un error del lado del cliente, debido a que no se envió correo en la petición de creación de usuario

CP-04 Registro de nuevo usuario fallido con campos vacíos (Campo de contraseña vacío)



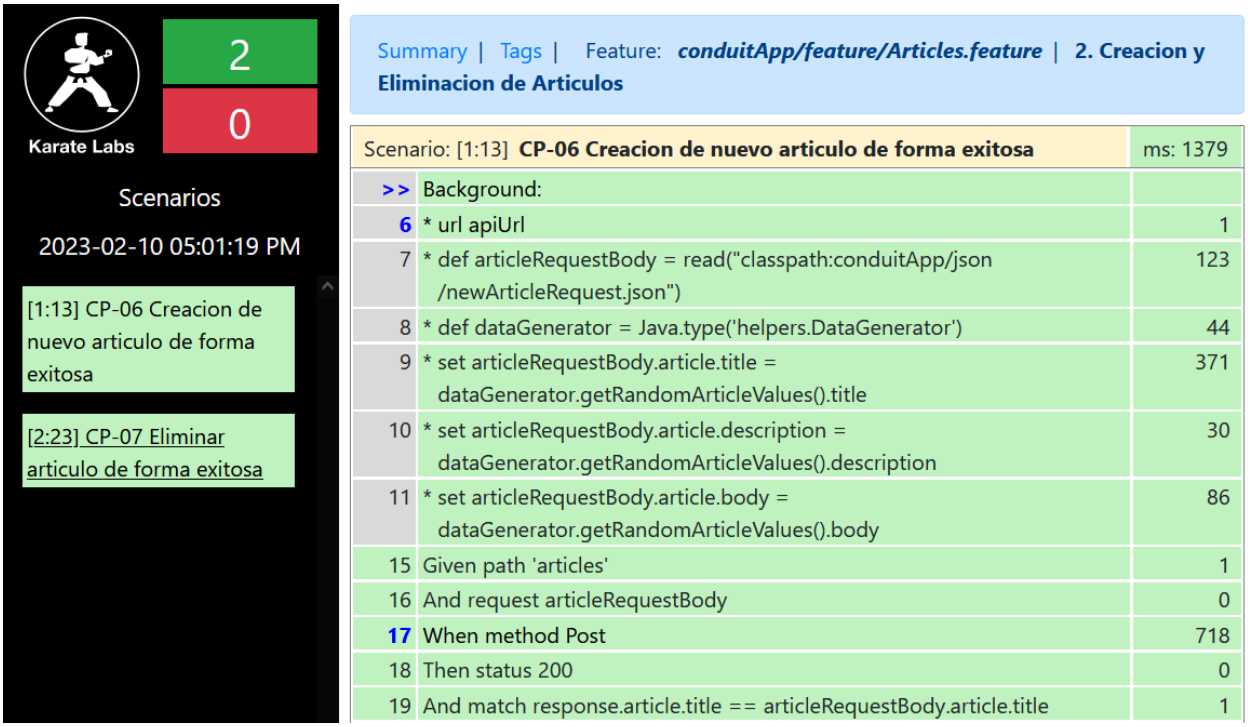
- Previo a la ejecución del escenario se llamó a la función Java generadora de datos aleatorios y se inicializó la URL de la API de pruebas
- Se generó y definió un email aleatorio
- Se generó y definió un correo electrónico aleatorio
- Se generó y definió un nombre de usuario aleatorio
- Se estableció el endpoint ‘users’ para ser consultado
- Se preparó una petición HTTP con el cuerpo de la petición en formato JSON con los datos del nuevo usuario a ser creado:
    - Email: (Correo electrónico aleatorio generado previamente)
    - Password: (Campo vacío)
    - Username: (Nombre de usuario aleatorio generado previamente)
- Se envió la petición por medio de un método POST
- Se recibió un status 422 indicando que hubo un error del lado del cliente, debido a que no se envió una contraseña en la petición de creación de usuario

CP-05 Registro de nuevo usuario fallido con campos vacíos (Campo de nombre de usuario vacío)



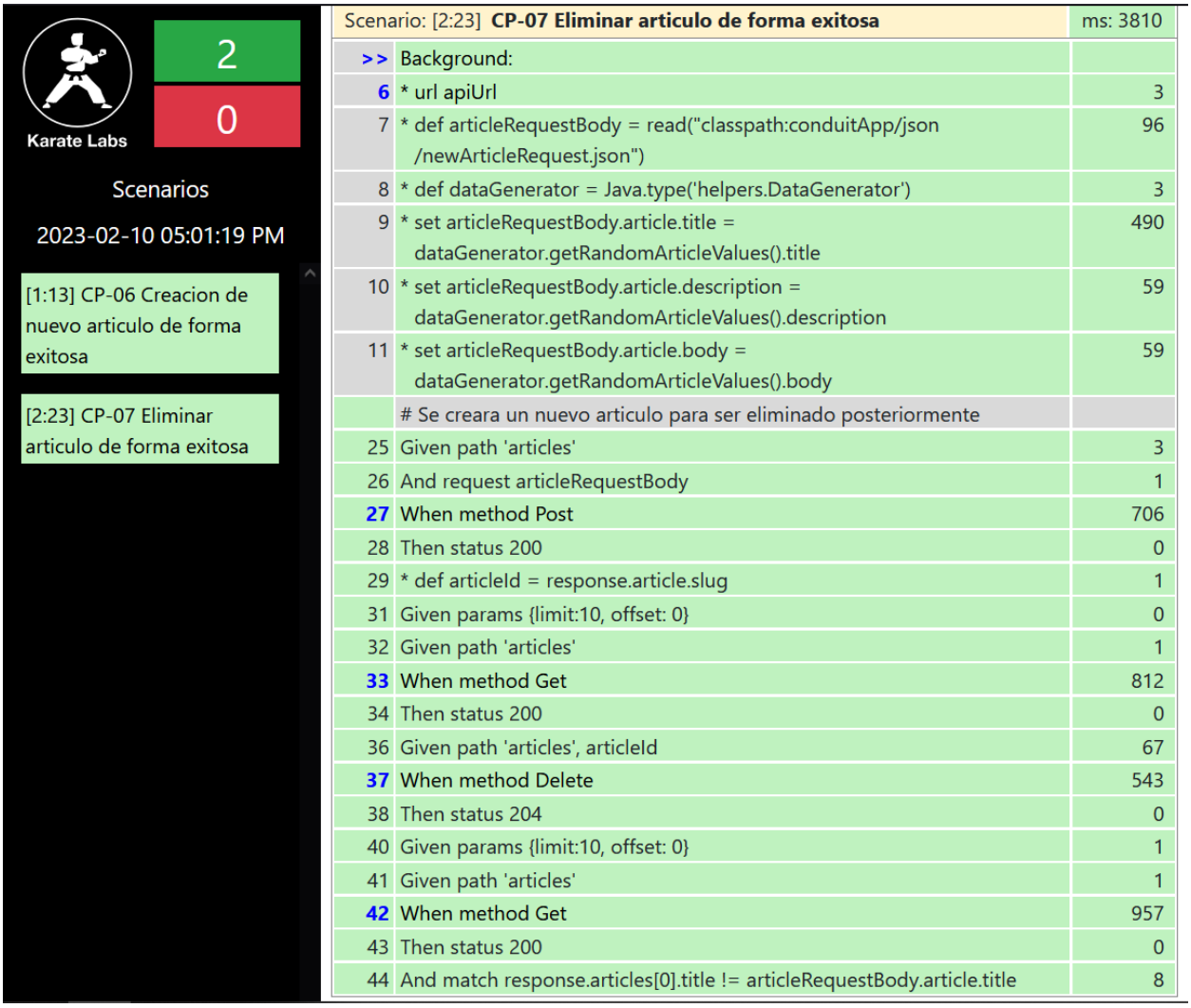
- Previo a la ejecución del escenario se llamó a la función Java generadora de datos aleatorios y se inicializó la URL de la API de pruebas
- Se generó y definió un nombre de usuario aleatorio
- Se estableció el endpoint ‘users’ para ser consultado
- Se preparó una petición HTTP con el cuerpo de la petición en formato JSON con los datos del nuevo usuario a ser creado:
    - Email: (Correo electrónico aleatorio generado previamente)
    - Password: test123
    - Username: (Campo vacío)
- Se envió la petición por medio de un método POST
- Se recibió un status 422 indicando que hubo un error del lado del cliente, debido a que no se envió un nombre de usuario en la petición de creación de usuario

CP-06 Creación de nuevo artículo de forma exitosa



- Previo a la ejecución del escenario:
  - Se inicializó la URL de la API de pruebas
  - Se definió la variable articleRequestBody conteniendo un objeto JSON con la estructura general de un artículo web
  - Se inicializó la función Java generadora de datos aleatorios
  - Se generó y asignó un título aleatorio de artículo a la variable articleRequestBody
  - Se generó y asignó una descripción aleatoria de artículo a la variable articleRequestBody
  - Se generó y asignó un cuerpo de contenido aleatorio de artículo a la variable articleRequestBody
- Se estableció el endpoint ‘articles’ para ser consultado
- Se preparó una petición HTTP con el cuerpo de la petición en formato JSON con los datos del nuevo artículo web a ser creado contenidos en la variable articleRequestBody
- Se envió la petición por medio de un método POST
- Se recibió un status 200 indicando que la solicitud ha tenido éxito
- El título del artículo web en la respuesta recibida por el servidor concordó con el título del artículo web generado aleatoriamente, cumpliendo exitosamente con la solicitud de creación de nuevo artículo web

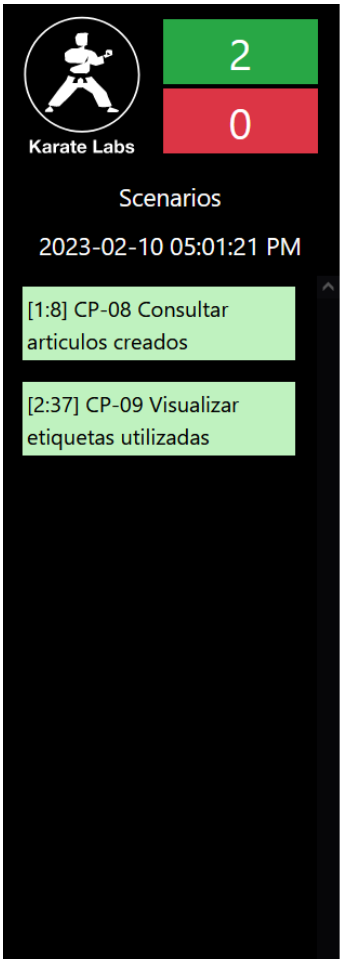
CP-07 Eliminar artículo de forma exitosa



- Previo a la ejecución del escenario:
  - Se inicializó la URL de la API de pruebas
  - Se definió la variable articleRequestBody conteniendo un objeto JSON con la estructura general de un artículo web
  - Se inicializó la función Java generadora de datos aleatorios
  - Se generó y asignó un título aleatorio de artículo a la variable articleRequestBody
  - Se generó y asignó una descripción aleatoria de artículo a la variable articleRequestBody
  - Se generó y asignó un cuerpo de contenido aleatorio de artículo a la variable articleRequestBody
- Se estableció el endpoint ‘articles’ para ser consultado

- Se preparó una petición HTTP con el cuerpo de la petición en formato JSON con los datos del nuevo artículo web a ser creado contenidos en la variable articleRequestBody
- Se envió la petición por medio de un método POST
- Se recibió un status 200 indicando que la solicitud de creación de artículo ha tenido éxito
- Se guardó el identificador único del artículo creado (slug) en la variable articleId
- Se establecieron los siguientes parámetros para realizar una consulta HTTP:
    - Limit: 10
    - Offset: 0
- Se estableció el endpoint ‘articles’ para ser consultado
- Se envió la petición de consulta por medio de un método GET
- Se recibió un status 200 indicando que la solicitud de consulta de artículos ha tenido éxito
- Se estableció el endpoint ‘articles’ y utilizó la variable articleId
- Se envió la petición de eliminación por medio de un método DELETE
- Se recibió un status 204, indicando que la solicitud de eliminación de artículo ha tenido éxito
- Se establecieron los siguientes parámetros para realizar una consulta HTTP:
    - Limit: 10
    - Offset: 0
- Se estableció el endpoint ‘articles’ para ser consultado
- Se envió la petición de consulta por medio de un método GET
- Se recibió un status 200 indicando que la solicitud de consulta de artículos ha tenido éxito
- Se confirmó que el artículo fue eliminado exitosamente al buscar que el título del artículo eliminado no se encuentre la respuesta de artículos obtenida de parte del servidor

CP-08 Consultar artículos creados



Summary | Tags | Feature: ***conduitApp/feature/HomePage.feature*** | **3. Obtencion de Articulos y Etiquetas**

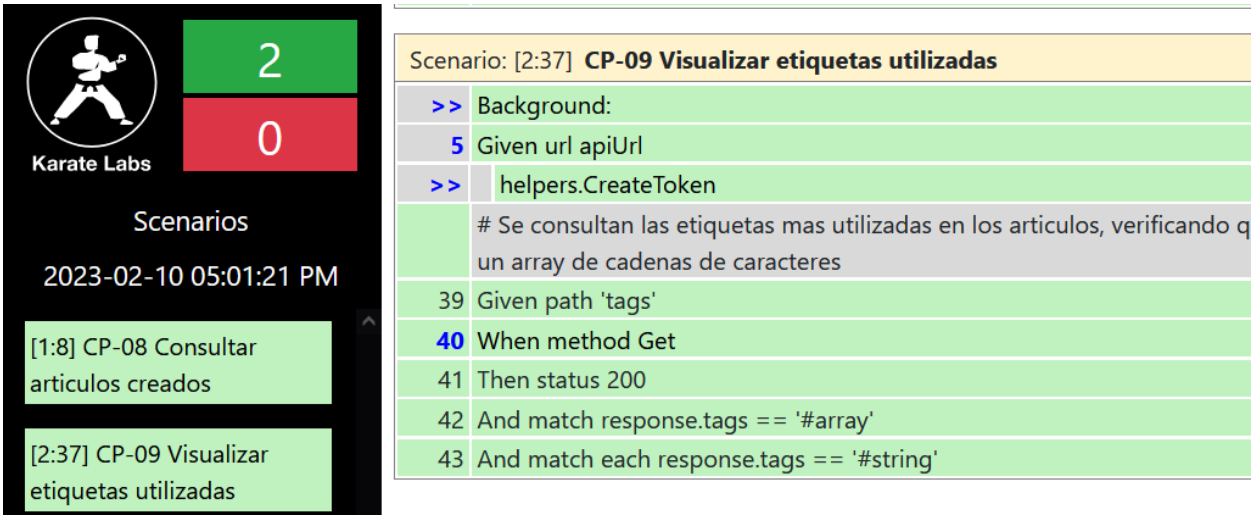
Scenario: [1:8] **CP-08 Consultar articulos creados**		ms: 1499
>>	Background:	
5	Given url apiUrl	1
9	* def timeValidator = read('classpath:helpers/timeValidator.js')	99
11	Given params {limit:10, offset: 0}	2
12	Given path 'articles'	44
13	When method Get	1134
14	Then status 200	0
15	And match response == {"articles":"#[10]", "articlesCount": "#number"}	5
16	And match each response.articles ==	215

```
{
    "slug": "#string",
    "title": "#string",
    "description": "##string",
    "body": "#string",
    "tagList": "#array",
    "createdAt": "#? timeValidator(_)",
    "updatedAt": "#? timeValidator(_)",
    "favorited": false,
    "favoritesCount": "#number",
    "author": {
        "username": "#string",
        "bio": "##string",
        "image": "#string",
        "following": "#boolean"
    }
}
```

- Previo a la ejecución del escenario:
    - Se inicializó la URL de la API de pruebas
- Se llamó a la función JavaScript validadora de formato de hora
- Se establecieron los siguientes parámetros para realizar una consulta HTTP:
    - Limit: 10
    - Offset: 0
- Se estableció el endpoint ‘articles’ para ser consultado
- Se envió la petición de consulta por medio de un método GET
- Se recibió un status 200 indicando que la solicitud de consulta de artículos ha tenido éxito
- Se verificó que la respuesta concordara con lo siguiente:
    - La respuesta tuviera un arreglo llamado ‘articles’ con 10 elementos
    - La respuesta tuviera un contador llamado articlesCount de tipo número
- Se verificó exitosamente que el arreglo ‘articles’ de la respuesta tuviera la siguiente estructura:
    - Slug: Identificador único del artículo, de tipo String

- Title: Título del artículo, de String
- Description: Descripción del artículo, de tipo String
- Taglist: Conjunto de tipo Array de etiquetas del artículo
- CreatedAt: hora y fecha de creación del artículo, validados por la función timeValidator
- UpdatedAt hora y fecha de creación del artículo, validados por la función timeValidator
- favorited: indicador de si el artículo ha sido marcado como favorito, cuyo valor por defecto debe ser 'false'
- favoritesCount: contador de tipo número de veces que el artículo ha sido marcado como favorito
- autor: objeto JSON, aninado, que contiene la estructura siguiente relacionado al autor del artículo:
  - username: nombre de usuario del autor, de tipo 'String'
  - bio: biografía del autor, el cual puede ser un campo de 'String' o vacío
  - image: campo de tipo 'String' que contiene la dirección de la imagen de perfil del autor
  - following: valor de tipo 'boolean' que indica si el autor está siendo seguido por otros usuarios

CP-09 Visualizar etiquetas utilizadas



- Previo a la ejecución del escenario:
  - Se inicializó la URL de la API de pruebas
  - Se llamó la función generadora de token de inicio de sesión
- Se estableció el endpoint ‘tags’ para ser consultado
- Se envió la petición de consulta al endpoint por medio de un método GET
- Se recibió un status 200 indicando que la solicitud de consulta de etiquetas ha tenido éxito
- Se verificó exitosamente que la respuesta concordara con lo siguiente:

- La respuesta tiene un arreglo de etiquetas de artículos
- Que cada uno de los elementos del arreglo es de tipo ‘String’

Comparativa de Pruebas manuales vs Pruebas automatizadas

A partir de los reportes de las pruebas automatizadas mostradas anteriormente, podemos sacar una métrica del tiempo total invertido en la ejecución de las pruebas. Como se mencionó al inicio de este informe, como equipo hicimos pruebas exploratorias para sacar las funcionalidades del sitio ConduitApp. También a partir de obtener la lista de funcionalidades, se hicieron pruebas manuales antes de automatizar.

A continuación, se muestra la comparativa de los tiempos que se obtuvieron en las pruebas automatizadas, contra los tiempos (en segundos) que se han en una prueba manual.

	Tiempos de ejecución de prueba	
Caso de Prueba	Manual	Automatizado
CP-01 Registro de un nuevo usuario de manera exitosa	10 s	1.6 s
CP-02 Registro de un nuevo usuario fallido con datos erróneos (Nombre de usuario ya existente)	5 s	0.8 s
CP-03 Registro de nuevo usuario fallido con campos vacíos (Campo de email vacío)	5 s	0.7 s
CP-04 Registro de nuevo usuario fallido con campos vacíos (Campo de contraseña vacío)	5 s	0.6 s
CP-05 Registro de nuevo usuario fallido con campos vacíos (Campo de nombre de usuario vacío)	5 s	0.7 s
CP-06 Creación de nuevo artículo de forma exitosa	15 s	1.3 s
CP-07 Eliminar artículo de forma exitosa	10 s	3.8 s
CP-08 Consultar artículos creados	5 s	1.4 s
CP-09 Visualizar etiquetas utilizadas	3 s	0.5 s
TOTAL	**63 s**	**11.4 s**

Como podemos comparar, existe una diferencia importante entre los tiempos de las pruebas manuales y automatizadas ya que mejora el tiempo de ejecución en un promedio de 5 veces. También tomar en cuenta que, en el caso de prueba de eliminar artículo, aparte de eliminarlo, se crea previamente para después borrarlo.

Con esta comparativa se puede constar la ventaja de las pruebas automatizadas y su importancia ya que, en una prueba de regresión, si tenemos un sistema que es grande con muchas más funcionalidades que nuestro caso, sería mucho más eficiente tener los casos de prueba automatizadas para ahorrar tiempo y poder desplegar cambios mucho más rápido.

Aparte que, si se hacen regresiones manuales, siempre se tiene que tener en cuenta el error humano y puede ser un proceso tedioso. Por lo tanto, como equipo con esta comparativa concluimos que las pruebas automatizadas son de gran utilidad en el área de QA y una gran ventaja dentro de cualquier organización, para ahorrar tiempo y mejorar la calidad del producto.

Conclusiones

La realización de pruebas es un proceso que puede volverse extenso, pero es una labor importante en el desarrollo de software, pues el propósito de desarrollar sistemas es, entregar al usuario o dueño del producto un software funcional y con una cantidad mínima de errores tal como se le planteó que es el producto que se entrega al finalizar la fase de pruebas, pues los errores que son detectados de manera temprana, será menos complejos de reducir.

La elaboración del plan de pruebas es de vital importancia, ya que se toma como referencia y guía a lo largo del proceso o ciclo de vida de las pruebas. Contemplar toda la planificación de manera general y especifica ha sido de gran apoyo para el diseño y ejecución de las actividades de prueba.

La etapa de realización de pruebas es indispensable, durante el desarrollo de software, pues si existe algún tipo de error o defecto, con la ejecución de las pruebas pueden ser corregidos, para la materialización de dichas pruebas se trabajaron las etapas que componen el plan de pruebas, estás van desde el diseño y evaluación de requisitos hasta el cierre del proceso de prueba.

A lo largo del presente documento, se hizo mención de lo que son las pruebas, pero con especial atención a las pruebas automatizadas, que aportan muchas ventajas que se pudieron experimentar con la realización de un caso práctico.

Con las pruebas automatizadas, hay que tener en cuenta el tipo de funcionalidad a probar; funcionalidades críticas del software deben probarse cada vez que se realicen cambios, incluso si estos son pequeños.

En el mercado existen muchas herramientas para la automatización de pruebas, cada una con ventajas y desventajas respecto a otras. En el caso de estudio presentado en este documento se utilizó Karate Framework, herramienta de código abierto que utiliza la sintaxis BDD muy popular en Cucumber, admite escritura en JSON, lo que facilitó la creación de los diferentes escenarios y razón por la cual es más recomendable la utilización de Karate sobre Cucumber.

Los casos de prueba creados por el equipo, que fueron automatizados y ejecutados con la herramienta Karate, generaron reportes por cada ejecución. A partir de esto, se pudo realizar la comparación de los tiempos que tarda cada caso de prueba en correr correctamente contra hacer la prueba manualmente. Con esta comparación, se constató de la ventaja que tiene la automatización de prueba en el ahorro de tiempo.

Bibliografía

Abstracta Team. (2021 de Octubre de 2021). *Testing Manual vs Automatizado: ¿cuál elegir para tu proyecto?* Obtenido de abstracta.us: https://cl.abstracta.us/blog/testing-manual-vs-testing-automatizado/

Board, I. S. (2014). *Plan de estudios de Nivel Básico Probador Ágil.*

Ciberninjas. (8 de septiembre de 2020). *Karate: Automatización de pruebas simples*. Obtenido de ciberninjas.com: https://ciberninjas.com/karate-automatizar-pruebas/

Cubas Montenegro, R. (2016). *Testing y Calidad de Software. Automatización de Pruebas con Selenium WebDriver.* Madrid, España. Obtenido de https://oa.upm.es/49320/1/PFC_RAFAEL_CUBAS_MONTENEGRO.pdf

Hernández Castillo, P. (29 de Junio de 2021). *Pasos para automatizar pruebas de software*. Obtenido de financialsolutions.mx: https://www.financialsolutions.mx/pasos-para-automatizar-pruebas-de-software/

Informáticas, T. -S. (6 de Octubre de 2021). *BDD: ¿En qué consiste y qué hay que tener en cuenta?* Obtenido de Tecnova.cl: https://www.tecnova.cl/2021/10/06/bdd-en-que-consiste-y-que-hay-que-tener-en-cuenta/

International Software Testing Qualifications Board. (2018). *Certified Tester Foundation Level Syllabus.*

Laoyan, S. (6 de Mayo de 2022). *Planning poker: la estrategia integral para la estimación ágil*. Obtenido de Asana.com: https://asana.com/es/resources/planning-poker

Marin Benavides, J. C. (15 de Octubre de 2021). *Manual de automatización de integración (API) con la herramienta Karate*. Obtenido de pragma.com.co: https://www.pragma.com.co/academia/lecciones/manual-de-automatizacion-de-integracion-api-con-la-herramienta-karate

QATechTools. (24 de Mayo de 2020). *What is the difference between Karate and Cucumber*. Obtenido de QATechTools.com: https://qatechtools.com/2020/05/24/what-is-the-difference-between-karate-and-cucumber/

Rubio Abujas, D. (15 de Marzo de 2021). *Karate*. Obtenido de Pandemonio Digital: https://pandemoniodigital.es/testing/2021/03/15/karate.html

Software Testing Bureau. (24 de Noviembre de 2020). *7 razones por las que es importante automatizar pruebas de software*. Obtenido de https://www.softwaretestingbureau.com/siete-razones-para-automatizar-pruebas/

Vargas, C. (7 de Noviembre de 2019). *¿Por qué automatizar las pruebas?* Obtenido de trycore.co: https://trycore.co/gestion-de-negocios/para-que-automatizar-pruebas-qa/

Zabolennyi, S. (18 de Julio de 2019). *Karate Framework: Testeo de APIs de Impacto*. Obtenido de Apimhub: https://apiumhub.com/es/tech-blog-barcelona/karate-framework-testeo-apis/

Glosario de Términos

Account: Una cuenta es un conjunto de información que se utiliza para acceder a un sistema o servicio. En el contexto de desarrollo de software, una cuenta puede ser utilizada para acceder a una aplicación o plataforma, y puede incluir información como un nombre de usuario y una contraseña.

API (Application Programming Interface): Una API es un conjunto de reglas y protocolos que permiten a los desarrolladores acceder a un sistema o servicio. Las APIs permiten a los desarrolladores interactuar con una aplicación o plataforma, y pueden ser utilizadas para acceder a funciones, datos y servicios específicos.

BDD (Behavior Driven Development): Es una metodología de desarrollo de software que se enfoca en el comportamiento de un sistema en lugar de en su implementación técnica. En BDD, se escriben casos de prueba en lenguaje natural para describir el comportamiento esperado de un sistema, y luego se escribe código para cumplir con esos casos de prueba.

Blog: Un blog es un sitio web o una página web donde una o varias personas publican contenido de forma periódica. Pueden incluir noticias, opiniones, tutoriales, entre otros. En el contexto de desarrollo de software, un blog puede ser utilizado para compartir conocimiento, experiencias y actualizaciones sobre proyectos o tecnologías específicas.

Bug: Un bug es un error o defecto en el código de un programa que causa que este no funcione correctamente. Pueden ser causados por errores de lógica, problemas de sintaxis o problemas con los datos de entrada.

CD/CI (Continuous Delivery / Continuous Integration): Es una metodología de desarrollo de software en la que se integran y se entregan cambios al código de forma continua y automatizada. Con CD/CI, los desarrolladores pueden detectar y solucionar problemas de forma temprana, y los usuarios finales pueden obtener nuevas funciones y actualizaciones de forma más rápida.

DevOps: Es una práctica que combina desarrollo de software y operaciones para mejorar la eficiencia y la calidad del software. Incluye técnicas de automatización, monitoreo en tiempo real y colaboración entre desarrolladores y operadores para mejorar el rendimiento del sistema, reducir los tiempos de inactividad y aumentar la confiabilidad y seguridad del software.

Endpoint: Es un punto final de un sistema o servicio que puede ser accedido por un cliente. Puede ser una URL, una dirección IP o un punto de conexión de una API.

Feed/Global feed: Un feed es un formato de datos que permite a los usuarios suscribirse a contenido actualizado en un sitio web o aplicación. Un feed global es un feed que incluye contenido de varias fuentes en un solo lugar.

GraphQL: Es un lenguaje de consulta y una herramienta para interactuar con bases de datos. Permite a los desarrolladores solicitar solo los datos que necesitan, lo que ayuda a reducir el ancho de banda y mejorar la eficiencia de las aplicaciones.

GUI (Graphical User Interface): Es una interfaz de usuario que permite a los usuarios interactuar con un sistema o aplicación a través de elementos gráficos, como botones, menús y ventanas. Es una forma más intuitiva y fácil de usar en comparación con las interfaces de línea de comandos.

Header: Es una sección de una petición o respuesta en un protocolo de red que contiene información adicional sobre la petición o la respuesta. Puede incluir información como el tipo de contenido, la autorización, la ubicación, entre otros.

Home: Es la página principal de un sitio web o aplicación. Es la primera página que se muestra al acceder a un sitio web o aplicación.

HTTP (Hypertext Transfer Protocol): Es el protocolo utilizado para transferir información en la internet. Es responsable de la transferencia de hipertexto, como páginas web, imágenes y otros recursos multimedia.

Issue: Es un problema o una tarea específica que se debe resolver en un proyecto de software. Puede ser un error, una mejora o una nueva característica.

Iteración: Es un ciclo de trabajo en un proceso de desarrollo de software en el que se desarrolla una pequeña parte de un proyecto, se prueba y se realizan cambios antes de continuar con el siguiente ciclo.

JSON (JavaScript Object Notation): Es un formato de intercambio de datos ligero y fácil de leer. Es ampliamente utilizado en aplicaciones web para transmitir datos entre el cliente y el servidor.

Login: Es el proceso de ingresar a un sistema o servicio utilizando un nombre de usuario y una contraseña. Es una forma de autenticar al usuario y permitirle acceder a funciones y datos específicos.

Logout: Es el proceso de salir de un sistema o servicio después de haber ingresado. Es una forma de cerrar la sesión y proteger la información privada.

Mock: Es una representación de un objeto o servicio utilizado en pruebas para simular el comportamiento de un objeto o servicio real. Se utiliza para probar el comportamiento de una aplicación sin depender de los servicios externos.

Pipeline: Es un conjunto de procesos automatizados que se utilizan para llevar a cabo tareas específicas en un proyecto de software. Pueden incluir tareas como la compilación, pruebas, implementación y monitoreo.

Product Owner: Es una persona o un equipo responsable de definir y priorizar las características y requisitos de un producto o proyecto de software. Se encarga de asegurar que el producto se adapte a las necesidades del negocio y del usuario final.

Prueba Ad-Hoc: Es una prueba que se realiza de forma espontánea, sin seguir un plan establecido previamente. Se utiliza para descubrir problemas o comportamientos inesperados en un sistema o aplicación.

Prueba end-to-end: Es una prueba que se realiza para verificar si un sistema o aplicación cumple con los requisitos y funciona correctamente desde el inicio hasta el final del proceso.

QA (Quality Assurance): Es el proceso de verificar y asegurar la calidad de un producto o proyecto de software. Puede incluir tareas como pruebas, revisión de código, monitoreo y mejora continua.

QA Asignee/Asigned: Es la persona o equipo responsable de llevar a cabo las tareas de calidad en un proyecto de software. Puede incluir tareas como planificar, ejecutar y reportar los resultados de las pruebas.

QA Validation: Es el proceso de verificar si un producto o proyecto de software cumple con los estándares y los requisitos establecidos. Puede incluir tareas como verificar la funcionalidad, el rendimiento, la seguridad y la usabilidad.

Refactorización: Es el proceso de modificar el código existente de un programa para mejorar su estructura y su calidad sin cambiar su funcionalidad. Puede incluir tareas como eliminar código duplicado, mejorar la legibilidad y la escalabilidad.

Reporter: Es la persona o equipo responsable de generar y presentar informes sobre el estado de un proyecto de software. Puede incluir tareas como generar informes de pruebas, seguimiento de errores y seguimiento de tiempos.

REST (Representational State Transfer): Es un estilo de arquitectura de software que se utiliza para construir servicios web. Se basa en el uso de métodos HTTP como GET, POST, PUT y DELETE para manipular recursos en un servidor. REST es una forma popular de construir servicios web escalables y fiables.

Retesting: Es el proceso de volver a ejecutar pruebas en un sistema o aplicación después de haber realizado cambios. Se utiliza para asegurar que los cambios no han introducido nuevos errores o problemas.

Settings: Es un menú o una sección de una aplicación o sistema que permite a los usuarios configurar diferentes opciones y ajustes. Puede incluir opciones como la configuración de la privacidad, la configuración de notificaciones, la configuración de idioma, entre otros.

Sign Up: Es el proceso de registrarse en un sistema o servicio. Puede incluir la ingresar información como el nombre de usuario, la contraseña, el correo electrónico, entre otros.

Testing: Es el proceso de verificar y validar si un sistema o aplicación cumple con los requisitos y funciona correctamente. Puede incluir tareas como pruebas unitarias, pruebas de integración, pruebas de aceptación, entre otros.

Token: Es una cadena de caracteres o símbolos que se utiliza para identificar y autenticar a un usuario o un proceso en un sistema de software. Los tokens se utilizan comúnmente para implementar mecanismos de seguridad en aplicaciones de software, como sistemas de autenticación de usuarios y autorización de acceso a recursos protegidos.

UAT (User Acceptance Testing): Es el proceso de verificar si un sistema o aplicación cumplen con las necesidades y los requisitos del usuario final. Es llevado a cabo por los usuarios finales y es la última etapa antes de la implementación en producción.

Update: Es el proceso de actualizar un sistema o aplicación con nuevas funciones, correcciones de errores o mejoras.

URL (Uniform Resource Locator): Es una dirección que se utiliza para acceder a un recurso en internet, como una página web o una imagen.

Username: Es el nombre de usuario utilizado para ingresar a un sistema o servicio. Es una forma de identificar al usuario y permitirle acceder a funciones y datos específicos.

Web: El conjunto de tecnologías y estándares utilizados para crear y presentar contenido en internet. Incluye lenguajes de programación como HTML, CSS y JavaScript, así como protocolos como HTTP y HTTPS.

XML (Extensible Markup Language): Es un lenguaje de marcas utilizado para describir datos y estructurar información. Es similar a HTML, pero más versátil y se utiliza para describir una variedad de tipos de datos.

Anexos

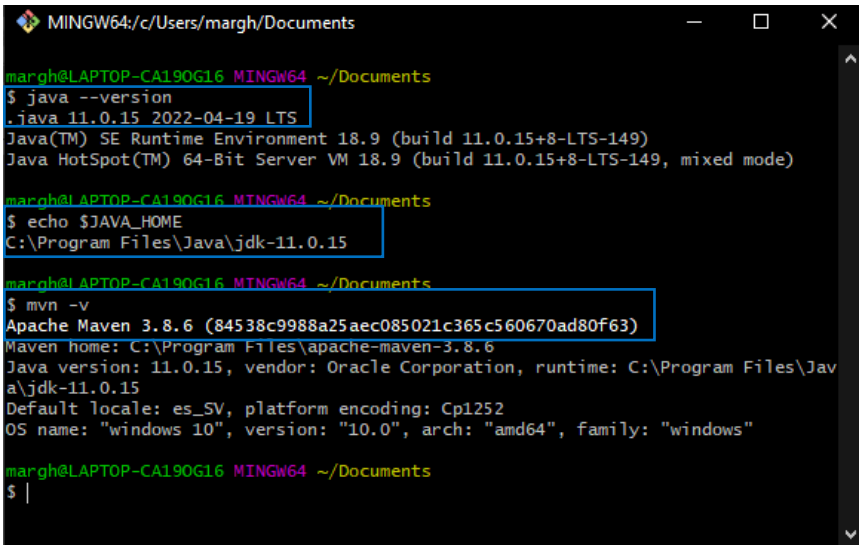
Guía de instalación y configuración de Karate Framework

Se utiliza una extensión de Visual Studio Code para realizar la automatización de pruebas en Karate.

Como primer paso se debe instalar:

- Java 8 o superior
- Maven, guía de instalación en el siguiente enlace: https://maven.apache.org/

Posterior a la instalación y configuración de variables de entorno, se verifica que todo se instaló correctamente.

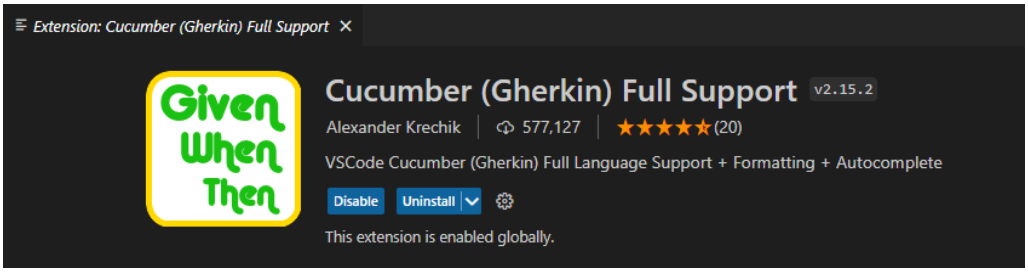


Paso siguiente instalar

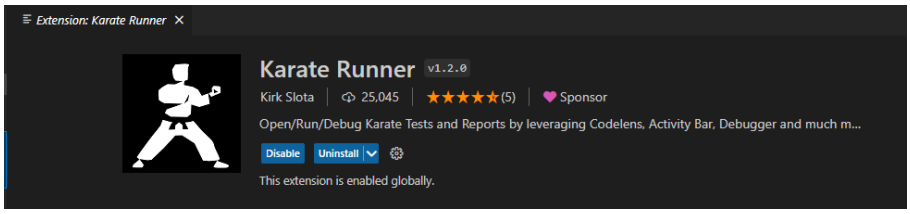
- Node.js
- Visual Studio Code

Extensiones en VScode

Cucumber (Gherkin) Full Support



Karate Runner



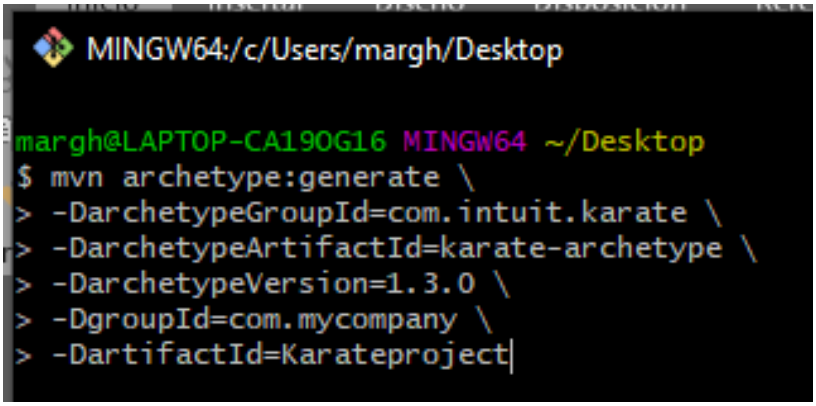
Extensión Pack for Java

Crear proyecto en Karate

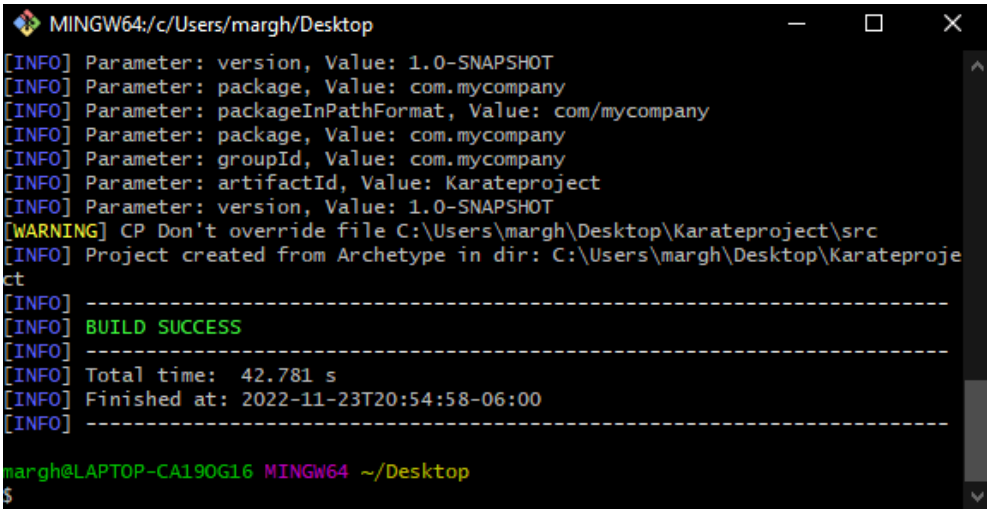
Ingresar a Repositorio de Peter Thomas donde se encuentra la documentación de Karate. https://github.com/karatelabs/karate

En Index seleccionar Opción Quickstart

En la terminal ubicar el proyecto y pegar siguiente código. Y dar enter cuando pregunte para que cargue los valores por defecto.

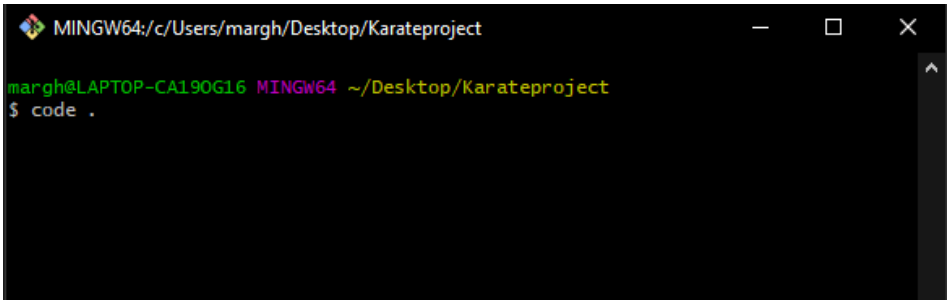


Presionar Enter para todas las preguntas que aparecen



Abrir Visual Studio Code

- Desde la ubicación de la carpeta del proyecto abrir VSCode



Proyecto creado