UNIVERSIDAD CATÓLICA SEDES SAPIENTIAE

FACULTAD DE INGENIERÍA

Implementación de un Sistema de Automatización de Pruebas con Karate Framework para Mejorar el Proceso de Testing en una Empresa del Rubro Bancario, Lima, 2023

TRABAJO DE SUFICIENCIA PROFESIONAL PARA OPTAR EL TÍTULO PROFESIONAL DE INGENIERO DE SISTEMAS

AUTOR

Gherard Anthony Chipana Quiñones

REVISOR

Silvia Mariana Montoya Saldaña

Lima, Perú
2023

METADATOS COMPLEMENTARIOS

Datos del autor

Nombres	GHERARD ANTHONY
Apellidos	CHIPANA QUIÑONES
Tipo de documento de identidad	DNI
Número del documento de identidad	72930812
Número de Orcid (opcional)	

Datos del asesor

Nombres	SILVIA MARIANA
Apellidos	MONTOYA SALDAÑA
Tipo de documento de identidad	DNI
Número del documento de identidad	09994755
Número de Orcid (obligatorio)	0009-0009-2843-8155

Datos del Jurado

Datos del presidente del jurado

Nombres	
Apellidos	
Tipo de documento de identidad	DNI
Número del documento de identidad	

Datos del segundo miembro

Nombres	
Apellidos	
Tipo de documento de identidad	DNI
Número del documento de identidad	

Datos del tercer miembro

Nombres	
Apellidos	
Tipo de documento de identidad	DNI
Número del documento de identidad	

Datos de la obra

Materia*	Testing, Framewok karate, Desarrollo de software, Automatización, Métodos Agiles
Campo del conocimiento OCDE Consultar el listado: enlace	https://purl.org/pe-repo/ocde/ford#2.00.00
Idioma (Normal ISO 639-3)	SPA - español
Tipo de trabajo de investigación	Trabajo de Suficiencia Profesional
País de publicación	PE - PERÚ
Recurso del cual forma parte (opcional)	
Nombre del grado	Ingeniero de Sistemas
Grado académico o título profesional	Título Profesional
Nombre del programa	Ingeniería de Sistemas
Código del programa Consultar el listado: enlace	612076

*Ingresar las palabras clave o términos del lenguaje natural (no controladas por un vocabulario o tesauro).

FACULTAD DE INGENIERÍA

ACTA N° 003-2023-UCSS-FI/TPISIS

TRABAJO DE SUFICIENCIA PROFESIONAL PARA OBTENER EL TÍTULO PROFESIONAL DE INGENIERO DE SISTEMAS

Los Olivos, 21 de abril de 2023

Siendo el día viernes 21 de abril de 2023, en la Universidad Católica Sedes Sapientiae, se realizó la evaluación y calificación del siguiente informe de Trabajo de Suficiencia Profesional.

"Implementación de un Sistema de Automatización de Pruebas con Karate Framework para Mejorar el Proceso de Testing en una Empresa del Rubro Bancario, Lima, 2023"

Presentado por el bachiller en Ciencias con mención en Ingeniería de Sistemas de la Sede Lima:

CHIPANA QUIÑONES, GHERARD ANTHONY

Ante la comisión evaluadora de especialistas conformado por:

MSc. GUERRA GUERRA, JORGE LEONCIO
Mg. RAMIREZ ROMERO, BRANDON VICENTE

Luego de haber realizado las evaluaciones y calificaciones correspondientes la comisión lo declara:

APROBADO

En mérito al resultado obtenido se expide la presente acta con la finalidad que el Consejo de Facultad considere se le otorgue al Bachiller CHIPANA QUIÑONES, GHERARD ANTHONY el Título Profesional de:

INGENIERO DE SISTEMAS

En señal de conformidad firmamos,

MSc. GUERRA GUERRA, JORGE LEONCIO Evaluador especialista 1	Mg. RAMIREZ ROMERO, BRANDON VICENTE Evaluador especialista 2

Anexo 2

CARTA DE CONFORMIDAD DEL ASESOR(A) DE TESIS / INFORME ACADÉMICO/ TRABAJO DE INVESTIGACIÓN/ TRABAJO DE SUFICIENCIA PROFESIONAL CON INFORME DE EVALUACIÓN DEL SOFTWARE ANTIPLAGIO

Los Olivos, 18 de setiembre de 2023

Señor
Marco Antonio Coral Ygnacio
Coordinador del Programa de Estudios de Ingeniería de Sistemas e Informática
Facultad de Ingeniería
Universidad Católica Sedes Sapientiae

Reciba un cordial saludo.

Sirva el presente para informar que informe de Trabajo de Suficiencia Profesional, bajo mi asesoría, con título: "**Implementación de un Sistema de Automatización de Pruebas con Karate Framework para Mejorar el Proceso de Testing en una Empresa del Rubro Bancario, Lima, 2023",** presentado por CHIPANA QUIÑONES, GHERARD ANTHONY con código 2014100007 y DNI 72930812 para optar el título profesional de Ingeniero de Sistemas, ha sido revisado en su totalidad por mi persona y **CONSIDERO** que el mismo se encuentra **APTO** para ser publicado.

Asimismo, para garantizar la originalidad del documento en mención, se le ha sometido a los mecanismos de control y procedimientos antiplagio previstos en la normativa interna de la Universidad, **cuyo resultado alcanzó un porcentaje de similitud de 3%.** * Por tanto, en mi condición de asesor, firmo la presente carta en señal de conformidad y adjunto el informe de similitud del Sistema Antiplagio Turnitin, como evidencia de lo informado.

Sin otro particular, me despido de usted. Atentamente,

Montoya Saldaña, Silvia Mariana
Docente Revisor
DNI N° 09994755
ORCID: 0009-0009-2843-8155
Facultad de Ingeniería – UCSS

* De conformidad con el artículo 8°, del Capítulo 3 del Reglamento de Control Antiplagio e Integridad Académica para trabajos para optar grados y títulos, aplicación del software antiplagio en la UCSS, se establece lo siguiente:

Artículo 8°. Criterios de evaluación de originalidad de los trabajos y aplicación de filtros

El porcentaje de similitud aceptado en el informe del software antiplagio para trabajos para optar grados académicos y títulos profesionales, será máximo de veinte por ciento (20%) de su contenido, siempre y cuando no implique copia o indicio de copia.

Resumen

El proyecto tiene como objetivo optimizar los procesos de pruebas o testing en procesos de desarrollo de software, para lo cual se implementa un sistema de automatización de pruebas basado en el framewok karate para una empresa del rubro bancario.

La implementación del sistema de automatización de pruebas, utiliza diversas técnicas y tecnologías que pueden ser utilizados en procesos de desarrollo que utilicen métodos ágiles, se genera un API Testing, donde se consideran 70 casos de pruebas logrando automatizar 60. Para el proceso Load Testing, se considera los 70 casos del proceso de API Testing, lográndose automatizar 50 pruebas. El benefició de automatización es de 86% para el proceso de API Testing, y para el proceso de Load Testing de 71%; de forma general el proceso de pruebas o testing obtuvo un 61% de mejora en el proceso siendo positivo para la empresa.

Palabras Clave: Testing, Framewok karate, Desarrollo de software, Automatización, Métodos Agiles.

Abstract

The objective of the project is to optimize the testing or testing processes in software development processes, for which a test automation system based on the framework karate is implemented for a banking company.

The implementation of the test automation system uses various techniques and technologies that can be used in development processes that use agile methods, an API Testing is generated, where 70 test cases are considered, automating 60. For the Load Testing process, the 70 cases of the API Testing process are considered, managing to automate 50 tests. The automation benefit is 86% for the API Testing process, and 71% for the Load Testing process; In general, the testing process obtained a 61% improvement in the process, being positive for the company.

Palabras Clave: Testing, Framewok karate, Software development, Automation, Agile Methods.

Índice de Contenido

1. Introducción

Una empresa bancaria es una institución u organización financiera que tiene como principal tarea la administración del dinero de sus clientes, tanto en inversiones como en préstamos. Además, estas empresas son parte de un sistema bancario y brinda servicios de banca, ya sea en plataformas digitales o a través de entornos presenciales.

En la actualidad, las empresas que hacen uso de las tecnologías de la información, tienen como prioridad la reducción de tiempos y costos en el ciclo del desarrollo de software: planificación, análisis y requerimientos, diseño, implementación, pruebas, despliegue y mantenimiento. Cada etapa define flujos internos que deben ser respetados y generan prototipos o artefactos según sea el caso. En este contexto se vuelve importante el proceso de pruebas también conocido como testing, ya que utiliza un mayor porcentaje de tiempo, debido a los procesos manuales utilizados, incrementándose más cuando se efectúan cambios en el proceso. Adicionalmente los procesos de desarrollo utilizan metodologías ágiles o frameworks que permitan agilizar los procesos del ciclo de vida de desarrollo de software, teniendo tiempos cortos para los procesos de pruebas, es por ello que se propone la automatización de pruebas, |a fin de encontrar la mayor cantidad de incidencias en la menor cantidad de tiempo, logrando mejores productos sin perjudicar el tiempo de entrega.

Las empresas que pertenecen al rubro bancario y que tiene como especialidad los servicios digitales, generan una mayor interacción a través de distintos canales y plataformas con sus clientes. Es por ello, que realizan gran cantidad de actualizaciones en los productos informáticos, lo cual demanda que todo el ciclo de vida del desarrollo de software tenga que estar en constante actualización, y que cada proceso tenga tiempos más ajustados. Por este motivo, la

automatización de pruebas o procesos de testing se vuelven relevantes, permite mejorar y agilizar los procesos de actualización o creación de nuevos servicios y aplicaciones para los clientes, priorizando las pruebas funcionales, las pruebas de carga y las pruebas de regresión.

Además, genera un mayor impacto al área de calidad, debido a que brinda una mayor cobertura en los casos de prueba, reduciendo tiempos y costos lo que permite reutilizar casos de pruebas, evitando que se tenga que volver a testear las funcionalidades una y otra vez.

El proceso de testing tiene como prioridad la detección de defectos, fallas, incidencias y brindar soporte a las funcionalidades de los sistemas que son desarrollados por la empresa, también brinda confiabilidad y, sobre todo, calidad en los productos, además, asegura que se cumplan con los requerimientos solicitados por el negocio, permitiendo una fácil toma de decisiones al momento de realizar los despliegues o pases a producción del software desarrollado por parte del equipo.

Un sistema de automatización de pruebas es un conjunto de herramientas programadas que permiten automatizar el proceso de testing; estos sistemas de automatización de pruebas son requeridos en proyectos de desarrollo de software que utilizan metodologías ágiles, son útiles para procesos y funcionalidades repetitivas, que suelen ser publicadas con versiones o reléase candidato, según lo que el equipo de desarrollo establezca.

El objetivo de este trabajo es la optimización del proceso de pruebas o testing en el ciclo de vida del desarrollo de software en las empresas del rubro bancario, para ello se utiliza un sistema de automatización de pruebas bajo el framework karate contruido en java, gherkin para los casos de pruebas, cucumber para detallar el comportamiento del software, gatlin para las pruebas de rendimiento, jenkins como base para la automatización y scala para programar

patrones, se trabaja bajo un enfoque ágil con scrum; asimismo se impulsa el uso de nuevas metodologías o marcos de trabajo para el área de negocio.

También, se implementará herramientas y módulos complementarios para poder realizar mejoras en la implementación del sistema de automatización de pruebas del ciclo de vida del desarrollo de software de las empresas del rubro bancario que se establece en este proyecto de automatización de pruebas, las cuales serán indicadas a lo largo del presente informe y establecidos en los siguientes capítulos a elaborar.

2. Trayectoria del autor

2.1. Descripción de la Empresa

Toda organización cuenta con un rubro de generar capital, tecnología y trabajo, que sean fundadas para hacer tareas determinadas, ya sea con o sin fines de lucro. Las empresas bancarias son organizaciones que tienen como objetivo el de poder involucrarse en los negocios netamente bancarios, cuya función principal es la de obtener depósitos de clientes externos que brinden financiamiento o inversiones en sus negocios, además de requerir préstamos o créditos financieros, Además, se consideran como organizaciones de trabajo, capital y tecnología.

Este rubro bancario es de lo más antiguos, por ello, toda persona natural y jurídica conoce el flujo básico que es el de solicitar dinero a un plazo fijo y con un tipo de interés determinados, según la empresa lo estime. Con esto, las empresas bancarias, venden este dinero a un precio e interés mayor, así podrán generar sus ganancias correspondientes y devolviendo el monto que anteriormente se había solicitado.

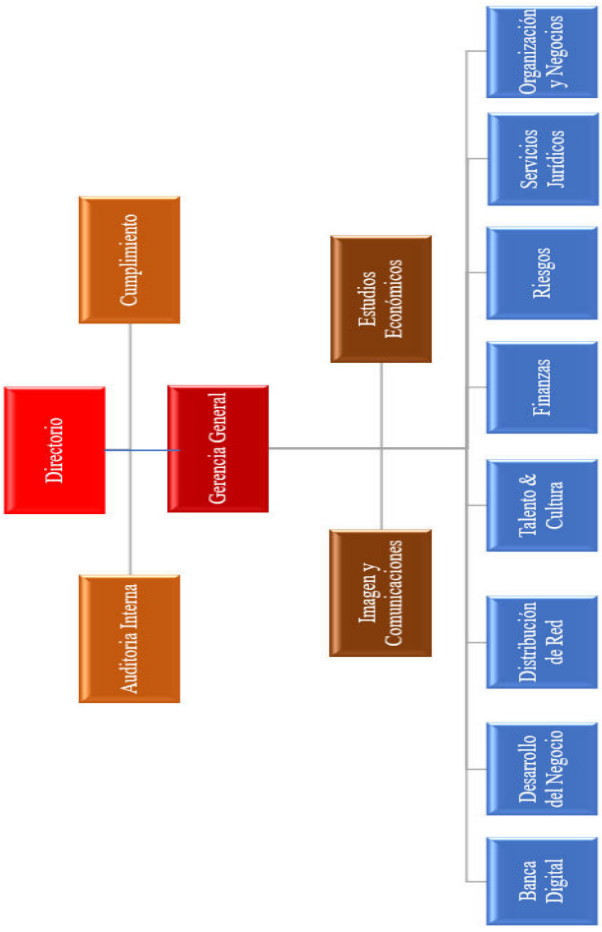
El negocio que tienen las empresas bancarias, no es bien visto desde un balance con fondos negativos, por lo cual, para poder crecer exponencialmente, usan de forma exclusiva la confianza y fidelidad que lograr generar en sus clientes, esto se logra debido a la buena gestión y trato que el personal y directivos brinden a sus clientes y también a sus aliados estratégicos del rubro bancario en el que se encuentran.

Existen diferentes tipos de entidades bancarias, las cuales son: banca minorista, banca de empresas, banca de inversión y la banca privada. La banca minorista se basa en personas naturales, donde los clientes adquieren servicios como cuentas corrientes, depósitos a plaza fijos,

tarjetas, préstamos hipotecarios, etc. La banca de empresas se basa en personas jurídicas, es decir, en empresas, que adquieren servicios como nóminas, transferencias masivas, tpv virtuales, etc. La banca de inversión se basa en el asesoramiento de personas con respecto a inversiones y adquisiciones; por último, la banca privada se basa en servicios de fortunas y grandes patrimonios.

Figura 1

Organigrama de la empresa



Nota: El gráfico representa la estructura organizativa que tienen las empresas y organizaciones del rubro bancario en el 2023. Elaboración propia.

2.2. Áreas y Funciones Desempeñadas

El área de Directorio es aquella área donde se encuentre todo el personal directivo de las empresas bancarias a nivel global, es decir, el representante por cada país de la entidad bancaria y su función es la de brindar el reporte de la organización durante el año en el país, además de facilitar a su equipo en lo que necesite. El área de auditoría interna se encarga de verificar que no existe ningún tipo de irregularidad en los procesos de negocio dentro de la organización. El área de cumplimiento es aquella área que vela por el logro de metas y objetivos trazados por la organización durante el año y que estos sean cumplidos en un gran porcentaje según lo acordado. El área de Gerencia General es donde se encuentran los diferentes gerentes por sede, es decir, los líderes de la organización en diferentes regiones y sedes del país, el cual se encarga por velar por sus colaboradores, y también que se cumplan los objetivos planteados en la sede correspondiente. El área de Imagen y Comunicaciones está encargada del tema de marketing y publicidad, donde actualmente están interactuando con las redes sociales debido a la gran demanda que está teniendo. El área de estudios económicos se encarga de velar por las finanzas de la organización, que se cumpla con lo proyectado, de esta manera, poder evitar que la organización pueda caer en quiebra o banca rota. El área de Banca Digital es la encargada de todo el tema tecnológico, donde se encuentra el tema de desarrollo de software, pruebas de software, despliegues de software, entre otros, además, de realizar los productos y servicios digitales de la organización. El área de desarrollo del negocio se encarga de encontrar oportunidades de negocio y plantearlas en la organización. El área de Distribución de Red es quien se encarga de gestionar la expansión en diferentes regiones de la organización. El área de Talento & Cultura es la parte de recursos humanos, donde se vela por los colaboradores de la empresa. El área de Finanzas permite la administración de todos los procesos financieros. El área de riesgos permite mitigar los riesgos

que se detectan en los proyectos analizados. El área de Servicios Jurídicos vela por toda la parte legal de la organización y así evitar problemas de fraudes o contratos fantasmas. El área de Organización y Negocios es aquella área donde se busca alianzas estratégicas con microempresarios, pequeños negocios y emprendedores.

2.3. Experiencia Profesional Realizada en la Organización

El área de calidad de software, que se encuentra dentro de banca digital, tiene como funciones el de poder brindar la entrega del producto con la mínima cantidad de incidencias posibles, es decir, que se encargue de validar que todas las funcionalidades requeridas no tengan o presenten ninguna falla, sobre todo, en las de mayor prioridad por los usuarios, de esta forma, evitar que el cliente presente su inconformidad con el producto y sobre todo con la empresa. El equipo de testing usa diferentes herramientas y metodologías para poder realizar las pruebas de funcionalidades del software, con lo cual, brinda una mejor experiencia de usuario al ser desplegada y entregada.

A lo largo de estos más de 6 años de experiencia profesional en el rubro de tecnología, pasando por áreas como base de datos, programación, gestión y ahora calidad de software, ha tocado en esta oportunidad estar a cargo de liderar el equipo de calidad de software, también llamando, el equipo QA o equipo de Testing, donde se realiza planes y estrategias de pruebas manuales y automatizadas, se entrega propuestas de uso de herramientas, metodologías, frameworks, entre otros. Además, la de brindar capacitación al equipo de pruebas con los nuevos sistemas que se presenten en el área. Por último, la de validar que todas las pruebas y evidencias realizadas se encuentren elaboradas correctamente y en función de lo solicitado.

Se ha realizado en primera instancia un plan de pruebas, en el cual, debido a que se viene trabajando con metodologías ágiles, especialmente, con el marco de trabajo Scrum, se ha adecuado un cronograma por sprint, los cuales, duran dos semanas cada uno, y dentro de cada sprint, se establece la realización de la estrategia de pruebas automatizadas, elaboración de casos de prueba, configuración de ambientes, ejecución de pruebas manuales, realización y elaboración de script de automatización, ejecución de pruebas automatizadas, despliegue de pruebas automatizadas en el ambiente y entorno correspondiente y la generación de reporte de pruebas automatizadas. Una vez que las pruebas automatizadas ya fueron culminadas, se procede a dar paso con la aprobación del equipo de testing al equipo Devops para su respectivo despliegue y pase al ambiente de producción, de esta forma, el software se utilice de forma pública.

3. Problemática

3.1 Planteamiento del Problema o Realidad Problemática

Actualmente, a nivel global, las empresas que elaboran plataformas digitales, requieren el uso de automatización de pruebas, con lo cual, se busca mejorar y agilizar el proceso de testing del ciclo de vida del desarrollo de software, de esta forma, evitar la repetición de pruebas manuales o demora en las pruebas de regresión ante cualquier cambio o actualización que pueda realizarse en la construcción del software elaborado.

Hoy en día, un gran porcentaje de empresas del rubro bancario están adoptando el uso de metodologías ágiles, esto debido a que la demanda de productos está incrementando exponencialmente. Debido a ello, la construcción de software se ha tenido que adaptar a esta forma de trabajo. Es por ello, que en el proceso de Testing, se requiere mejorar el flujo que se realiza de forma manual, agilizándolo y evitando convertirse en cuello de botella en la construcción del producto o software.

Para Diaz (2021), los consumidores digitales se han vuelto más exigentes con respecto a las herramientas tecnológicas que usan, por ello, un gran porcentaje de clientes rechaza los productos que no logran cumplir con sus expectativas de funcionalidad, lo cual, genera que el producto desarrollado tenga una baja estabilidad, volviéndose obsoleto para la empresa que la desarrolló. Así mismo, esta situación propone que el producto se innove y deba pasar todos los filtros de desarrollo de software, entre ellos, el proceso de pruebas, ya sea funcionales o de regresión.

Según Alégroth, Feldt y Kolström (2016), hacen mención sobre un beneficio mejorar el proceso de testing, realizando automatización de pruebas, ya se para plataformas web, para apis, para aplicativos móviles u pruebas de performance. Sin embargo, como indican Cabrera y Pareja (2021), al implementar este sistema, se debe considerar una elevación de costos, una mejora en la arquitectura y la incorporación de colaboradores con habilidades de programación y automatización, así, la implementación del sistema podrá tener un mejor rendimiento al momento de realizar la construcción de scripts y ejecución de pruebas automatizadas.

Para Rivera (2018), el problema que atraviesan las empresas del rubro digital hoy en día se basa en la demora de entrega del producto, esto debido a que aún existe gerentes que siguen estancados en la idea de trabajar bajo en modelo cascada, el cual, genera demasiada dependencia por parte de cada área y proceso, es decir, si un área no culmina sus actividades, la otra área no podrá avanzar. Este problema se genera en gran porcentaje en el proceso de testing, en el cual, por estar dedicados a realizar pruebas de forma manual, el tiempo de entrega de evidencias es mayor, lo cual, en muchas ocasiones puede generar que se estanque para el flujo de forma general.

Además, como menciona Gallego (2022), el área de calidad de software de las empresas del rubro digital tiene un problema de migración de procesos, metodologías y herramientas, por lo que están acostumbrados a trabajar con entornos de cascada, algunos siguen utilizando documentación extensa o diagramas de Gantt para un proyecto que requiere constantes cambios, sin necesidad de esperar a que el producto final tenga una fecha de entrega completa, al mismo tiempo, las diferentes áreas que interactúan para realizar el ciclo de vida de desarrollo de software se encierran en sus procesos cascada, lo cual, genera dependencia de una y de la otra, provocando que se creen retrasos y muchas veces, pérdida de tiempo en productividad, por lo cual, se

propone, principalmente para el área de calidad de software realizar implementaciones de metodologías ágiles y también sistemas de automatización de pruebas, aprovechando que existe una variedad de herramientas de libre código, lo cual, no generará un crecimiento de gastos, todo lo contrario, provocará que los gastos se reduzcan por el uso de nuevas tecnologías y que son completamente preparadas para el uso de colaboradores con conocimientos o no de programación.

Banda (2022) menciona que los problemas principales se dan en la revisión detallada de casos de pruebas, es decir, en el caso de UI por ejemplo, debe ser la validación de campo por campo y funcionalidad por funcionalidad, lo cual, genera que se extienda el tiempo de entrega de reporte de pruebas, y por ello es que se propone realizar la automatización de pruebas, para que, en el momento de realizar las pruebas de regresión, no se tenga que validar uno por uno, todo lo contrario, solo se ejecuten las pruebas automatizadas y se generen las evidencias y reportes de prueba.

3.2 Determinación del Problema

Problema Principal

¿Cómo la automatización de pruebas con karate mejora el proceso de testing en una empresa del rubro bancario?

Problemas Secundarios

¿Cómo la automatización de pruebas con karate agiliza el proceso de testing en una empresa del rubro bancario?

¿Cómo la automatización de pruebas con karate mejora el proceso de API testing en una empresa del rubro bancario?

¿Cómo la automatización de pruebas con Karate mejora el proceso de Load Testing en una empresa del rubro bancario?

3.3. Objetivo General

Determinar si el sistema de automatización de pruebas con karate mejora el proceso de testing en una empresa del rubro bancario.

3.4. Objetivos Específicos

Determinar si el sistema de automatización de pruebas con karate agiliza el proceso de testing en una empresa del rubro bancario.

Determinar si el sistema de automatización de pruebas con karate mejora el proceso de API Testing en una empresa del rubro bancario.

Determinar si el sistema de automatización de pruebas con karate mejora el proceso de Load Testing en una empresa del rubro bancario.

3.5. Justificación

La calidad de los servicios digitales es de suma importancia para las empresas bancarias, por ello, es que se busca poder realizar un excelente filtro en el proceso de testing, de esta manera, poder evitar alguna falla al momento de entregar el producto a los usuarios y estos puedan brindar excelente comentarios y recomendaciones.

Esta investigación tiene como finalidad la de poder realizar una buena implementación de un sistema de automatización de pruebas, y poder lograr con esto la mejora del proceso de testing, la agilidad del flujo de pruebas y la reducción de tiempos, con lo cual, se reducirán costos de servicios externos a la plataforma.

Al diseñar el sistema de automatización de pruebas, se busca realizar una estructura integrada de pruebas, es decir, que pueda brindar no solo testing para un área, sino, que pueda integrar la automatización para todo tipo de pruebas, siendo las principales las de Api y web, que son las más solicitadas en el rubro bancario. Además, al integrar todo ello, se podrá visualizar las pruebas manuales que se realizan en el proceso de testing, la cual, bajo el enfoque ágil y marco de trabajo scrum, toma demasiado tiempo validar, y con esto perjudica el pase a producción del software, por ello, con la automatización, se dará una mejora de toma de decisiones, con evidencias, reportes y soluciones de forma rápida y organizada en el área de calidad. Las API Testing también son importantes para comprobar que las funcionalidades independientes tengan un correcto uso, por ello se busca automatizar estos flujos. Según Lopez, Segura y Ruiz (2022), mencionan que probar API REST ful es importante en la integración de sistema. Existen herramientas para elaborar automatización de pruebas de API como postman, karate, entre otros. Las Load Testing tienen una importancia mayor en sistemas que tienen una interacción de grandes cantidades de usuarios. Según Zuluaga y Bedoya (2018), las pruebas de carga son las encargadas de validar el rendimiento del sistema y así poder evaluar el comportamiento que tenga ante una gran interacción de usuarios.

3.6. Alcances y Limitaciones

Según Garousi, Küçük y Felderer (2019), la automatización de pruebas no siempre tendrá los recursos adecuados para desarrollar scripts de todo tipo de pruebas, debido a que existen algunas restricciones para realizarlas, por ejemplo, la de interactuar con un robot aleatorio, o un código QR o un Capcha. Además, menciona que se debe instalar herramientas open source, en las cuales, se deben encontrar las versiones actualizadas que vendrán con los componentes más recientes.

Para River (2018), las pruebas manuales son útiles en caso se requiera un mayor detalle de las pruebas, es decir, en caso se solicite ver temas de headers, request-body, response-body, entre otros, sin embargo, al realizar las pruebas de regresión, estas no serán útiles, debido al retraso y demora de tiempos que generan, por ello se propone que la automatización de pruebas sea la solución, sin embargo, de no contar con un equipo capacitado en programación básica o estructuras como gherkin, la creación del sistema será más complicada para la empresa de tecnología.

En caso de no contar con recursos para realizar despliegues en la nube como Jenkins u otro software, el reporte de pruebas de automatización en la nube será complicado de realizarse, según Fernández (2018), las herramientas de automatización son indispensables para realizar la implementación, es decir, que se debe cumplir con software como gradle o maven, que tengan un versionamiento que sea compatible con los framework, librerías, etc. que se requieran utilizar para elaborar un sistema de automatización de pruebas y este no genere problemas al ejecutarse.

Chávez (2021) menciona que las pruebas automatizadas tienen ciertos criterios y limitaciones con respecto a sus ejecuciones, esto debido a que en muchas ocasiones se suelen dar actualizaciones de los componentes, librerías, entre otros, lo cual, genera que el sistema de

automatización se pueda volver obsoleto en ciertas funcionalidades realizadas, por ello, es importante estar en constante mantenimiento en estos sistemas de automatización y así evitar que se generen problemas de componentes que ya no existan en los repositorios o dependencias públicas.

Cubas (2019) hace referencia a las limitaciones que pueden existir con respecto a la automatización de pruebas, según la herramienta que se utilice, ya sea selenium, Katalon, playwright entre otros para el frontend y karate, rest assured, postman para la parte backend, además, hace mención de appium, como herramienta de automatización para aplicativos móviles, en todos estos casos, se requiere utilizar ya sea maven o gradle, y como cada herramienta tiene diferentes comunidades, todos de software libre, se implementan mejoras cada cierta cantidad de meses, lo cual, provoca que se cree un archivo de configuración para mantener actualizado los repositorios correspondientes.

En el caso de la planificación o estrategia, según lo que se realice, Kaplan (2018) menciona que es importante crear solo un documento inicial, esto debido a que se debe desde allí utilizar como base la construcción de scripts para automatizar, lo cual, en diferentes ocasiones, puede generar un grado de conflicto, debido a que los colaboradores deben estar en constante capacitación sobre la herramienta y sobre las dependencias que se indiquen en el documento, de no ser así, el flujo de automatización de pruebas puede tener ciertas deficiencias, debido a que el equipo de testing o el tester asignado no pueda realizar los scripts en el tiempo estimado.

Alcances

Como alcance se tiene que pueda ser utilizado para todos los procesos de testing de las empresas bancarias, y el cual puede ser utilizado en diferentes lenguajes de programación y

sistemas operativos, como también pueda tener adaptabilidad con diferentes proveedores de nubes y sea de fácil uso tanto para programadores y equipo técnico como para el área de negocio y proyectos.

Limitaciones

Las limitaciones que puede presentar es la del sector geográfico, debido a que se basa en estándares de fechas y horas locales, además, de tener dependencias en las versiones de los diferentes componentes como librerías o módulos, esto debido a que, como se trabaja con software libre, tendrá una serie de cambios de mejora de forma constante por parte de la comunidad de desarrolladores.

4. Marco Teórico

4.1. Antecedentes Bibliográficos

Borio y Paterno (2021) investigaron e instrumentaron acerca del proceso de pruebas de regresión en el ciclo de vida del desarrollo del software a través del testing automático. Hoy en día, las empresas que brindan servicios y productos tecnológicos, suelen realizar constantes cambios y actualizaciones en las funcionalidades, esto debido a que los usuarios suelen encontrar fallas o suelen solicitar nuevos requerimientos en los aplicativos, además, los actuales usuarios exigen que las tecnologías cumplan con parámetros de eficiencia y calidad en la ejecución de sus funcionalidades. Es por ello que se realizó una investigación al proceso de pruebas, contando con un sistema de gestión financiera, el cual, gestiona comprobantes de una variedad de negocios y que ha sido desarrollado con Java; además, cada versión de su software es desplegada en un ambiente de calidad, y así poder ser sujeto a pruebas. Como se detectó que las pruebas que se realizan son de forma manual, a fin de poder optimizar tiempos, mejorar la calidad y mitigar errores, se creó un proyecto para automatizar las pruebas funcionales, que eran realizados en las pruebas de regresión. Este proyecto se desarrolló con la idea de poder implementar una innovadora forma de ejecución de pruebas, y así poder aumentar la cobertura de casos de pruebas que se elaboren para las funcionalidades de los sistemas. Los resultados obtenidos con el desarrollo de este sistema de automatización de pruebas funcionales enfocadas en regresión fueron óptimos, debido a que se logró reducir los tiempos de prueba en un 70%, además, se logró implementar nuevas tecnologías y herramientas que hoy en día son solicitadas en el mercado de tecnología, con lo cual, permite que ante cualquier metodología, estructura o cambio en gestión que se pueda realizar, el proceso de testing se adapte de forma inmediata, sin generar retrasos ni convirtiéndose en cuello de botella. En conclusión, se permitió demostrar el potencial que tienen

la automatización de pruebas en el proceso de testing, las cuales, también pueden ser adaptables para el equipo de desarrollo, debido a que se toca código fuente de diferentes lenguajes de programación y tiene una mejor comunicación con el equipo de negocio, utilizando escenarios o puntos claves en los requerimientos del sistema.

Diaz (2021) realizó un seguimiento al proceso de pruebas que se realizaban en las diferentes empresas de tecnología en Sevilla debido a la demanda de tester automatizadores en el rubro tecnológico. En la actualidad, las empresas que están dedicadas al rubro tecnológico, están realizando búsquedas de tester automatizadores, los cuales, aparte de tener habilidades para realizar pruebas, también deben tener habilidades de programación, sobre todo, en lenguajes de programación como Java, Python o JavaScript e interacción con diferentes bases de datos relaciones y no relacionales, así como integración continua, para poder realizar una completa automatización de pruebas. Es por ello, que se realizó un sistema basado en la automatización y mejora del proceso de pruebas, utilizando las herramientas de Selenium, Maven, TestNg, Relevantcodes y Log4j, además, de usar el patrón de diseño Page Object, para poder organizar el proyecto en paquetes, recursos y configuraciones según los requerimientos establecidos por el cliente, además, para poder llevar una mejor gestión y documentación, se aplicó los estándares establecidos por el ISTQB y el framework scrum, de esta manera, poder tener una mejor optimización en el proceso de pruebas en el desarrollo del sistema. Con este seguimiento y aplicación, se logró obtener la mejora en el proceso de pruebas, optimizando recursos tecnológicos y humanos en un 50%, reduciendo tiempos establecidos en más del 60% y logrando una mayor detección de incidencias reportadas de los sistemas elaborados, además, esto ha permitido que el ciclo de vida de desarrollo del software pueda acelerar en su desarrollo y dando oportunidad a implementar nuevas funcionalidades en los sistemas elaborados; también el de

poder interactuar con el negocio y el usuario final, para que puedan dar su conformidad a las funcionalidad testeadas a través de la automatización de pruebas. En conclusión, la automatización de pruebas de regresión tiene una gran importancia, porque permite conocer la aplicación web a probar, la api en caso se requiera, la integración de ambas partes y el despliegue e identificación de casos de prueba; en un principio, se puede ver de forma fácil y sencilla, pero con el pasar del proyecto, se pudo demostrar la dificultad de la implementación, debido a su variedad de tecnologías e integración de módulos y paquetes, ya que estos mismos tienen ciertas dependencias que están en constante actualización.

Fernández (2018) diseño, elaboró, construyó e implementó un proceso de pruebas automatizadas basadas en las funcionalidades creadas y por crear. Las empresas de la actualidad suelen tener dificultades al momento de implementar nuevos cambios o actualizaciones en sus aplicativos tecnológicos, esto debido a que, los equipo de desarrollo y pruebas aún están aplicando un método cascada, lo que genera que se presenten retrasos en cada etapa del ciclo de desarrollo del software, sobre todo en el área de testing, que se convierte en un cuello de botella, por realizar pruebas manuales repetitivas, y que en muchas ocasiones, presentan nuevas fallas o incidencias, generando una nueva validación y corrección por parte del equipo de desarrollo. Por eso mismo, se implementó las herramientas de Selenium Web Driver, TestNg y SoapUI, con las cuales, se elaboraron pruebas automatización partiendo de los requerimientos básicos establecidos por el negocio y usuarios, además, al automatizar las pruebas funcionales, también se elabora scripts de automatización para las nuevas funcionalidades establecidas y así no generar demora en la entrega de requerimientos ya establecidos en anteriores versiones; además, se implementó metodologías ágiles, de esta manera, dejar de lado el método cascada y poder optimizar tiempos de entrega, tanto por parte del equipo de Desarrollo como por parte del equipo

de Testing. Como resultado, se logró obtener una mejora del proceso de testing y también de desarrollo, gracias a la detección eficaz y rápida de incidencias y errores en los aplicativos, además, de evitar los cuellos de botella en el proceso de testing, ya que se logró evitar volver a repetir pruebas ya realizadas en versiones anteriores, esto debido a los scripts desarrollados y ejecutados por el equipo de pruebas automatizadas, y solo teniendo que elaborar scripts de las nuevas funcionalidades a probar. Como conclusión, se establece que el desarrollo de este sistema de automatización de pruebas de regresión, basado en funcionalidades antiguas y actuales ha permitido enfocar y tener como prioridad la detección de incidencias de forma rápida y eficaz, como también, de evitar que se presenten nuevos errores en funcionalidades ya validadas en procesos anteriores de validación, así, poder agilizar este flujo y poder dar una mejor experiencia de usuario, tanto al negocio como a los clientes; con esto, poder determinar la entrega de un excelente aplicativo para la empresa y para los consumidores finales.

Cabrera y Pareja (2021) realizaron una investigación sobre el área de calidad de software en una empresa del rubro de Retails. En la actualidad, toda empresa que brinda servicios digitales tecnológicos, conseguirá que su área de calidad de software sea de las más fortalecidas, para que los productos y servicios tecnológicos que brinden, no presenten ningún problema con respecto a la experiencia de usuario o cliente, además, de esta forma, se podrá obtener una mejor calificación tanto por parte del cliente como por la parte empresarial de negocio. Es por ello, que se realizó una investigación cuantitativa, en la cual, se usó diferentes herramientas open source y la herramienta de automatización web selenium, además del framework ágil scrum, con las cuales, se pudo estimar el tiempo que toma la realización de pruebas funcionales y de performance en el proceso de testing, como también, poder evaluar la cantidad de defectos encontrados en los softwares desarrollados. Con esto, se logró determinar que la propuesta de

automatización en la empresa que investigaron, permitió la mejora del área de pruebas en un 60%, tanto en la optimización de tiempos, detección de fallas, coberturas de escenarios y esfuerzo humano, este último debido a que ya no se requiere estar realizando pruebas de regresión manuales, porque con la automatización de pruebas, estas ya están construidas a través de scripts elaborados en el primer pase por calidad, además, permitió realizar una mejora en los diseños de casos de prueba, esto debido a que se utilizó un lenguaje específico de dominio denominado gherkin, con el cual, se interactúa tanto negocio como el área técnica, y así poder todos llegar a un consenso con respecto a los requerimientos que el usuario solicita en el software; otro de los aspectos que se logró determinar es que con la automatización de pruebas, se pudo aplicar el framework scrum, debido a que ahora se está empezando a implementar las metodologías ágiles, por ello, se pudo realizar la adaptación de estas metodologías con la implementación del sistema de automatización de pruebas web con Selenium en los proyectos realizados en la empresa del rubro Retails. En conclusión, se corroboró que la automatización permite mejorar el proceso de pruebas del área de calidad de software en los escenarios ejecutados, tanto funcionales como de regresión, logrando reducir los tiempos de pruebas en un 60% en la parte funcional y en un 61% en la parte de regresión dentro de los registros encontrados en la investigación realizada por el equipo en la empresa del rubro Retails.

Chávez (2021) realizó una automatización de pruebas utilizando la herramienta de Selenium y testNG y así mejorar el proceso de pruebas en un sistema que brinda servicios RPO. En estos tiempos, los proyectos que brindan servicios RPO, tienen una gran cartera de clientes que llegan a enormes cantidades de compañías, los cuales, cuentan con sitios web que tienen diferentes idiomas, es decir, son plataformas multilenguaje, por ello mismo, es importante elaborar las pruebas correspondientes de las páginas web, para poder comprobar que no tenga

ningún problema al momento de ser utilizada por los usuarios consumidores. Por ello, como primera instancia, se ha establecido utilizar estándares basados en el ISTQB, de esta forma, poder aplicar buenas prácticas en la elaboración y ejecución de casos de pruebas, además, se utiliza herramientas como Postman y Selenium, para realizar la automatización tanto de Apis de idiomas como la automatización del funcionamiento en las plataformas web, y como herramienta de reporte, se utiliza el cucumber report. Los resultados obtenidos fueron los de lograr la comprobación de idiomas sin ninguna dificultad, a través de Json de comparación y automatización las APIS con Newman en Postman, asimismo, se logró agilizar las pruebas funcionales en un 50%, detectando las incidencias presentadas en las plataformas y sistemas web, y logrando reducir el tiempo de entrega o publicación para el público en la web, otro logro obtenido fue el de realizar pruebas en tiempo real, esto debido a que, al estar automatizadas las pruebas, solo se deben realizar la ejecución cambiando la URL o dominio de pruebas al público, y dar la oportunidad de corregir lo que fue detectado por el equipo de pruebas, y los tester automatizadores. Como conclusión, se logró implementar las pruebas automatizadas de web, integrándolas con Jenkins y slack, además, se aumentó las instancias establecidas de los clientes en las pruebas de smoke, logrando un 100% de funcionalidades probadas, y así, cubrir las expectativas que tiene el cliente; también se logró la elaboración de reportes más ordenados y estructurados, con mejor comprensión y entendimiento para el negocio; se logró reducir costos de tiempo, aumentar la velocidad del proceso de pruebas y desarrollar un crecimiento en los conocimientos de testing y programación, elaborando sistemas de pruebas para Web o para APIs, según lo que solicite el cliente o el negocio establecidos por la empresa.

4.2. Bases Teóricas

Para ISTQB (2018), Testing es el proceso que tiene como concepto a las actividades de la construcción del software, tanto de forma estáticas como de forma dinámicas, además, hace referencia a la planificación, elaboración y evaluación de software desarrollado. Además, según Toledo (2017), el testing es una técnica de investigación que se realiza para promover la información que requieren los clientes sobre la calidad del software o del servicio digital elaborado.

Kaplan, Panessi y Ortiz (2018), definen las pruebas funcionales como el punto de partida del proceso de testing, debido a que es allí donde se realiza la validación principal de la calidad del producto, con lo cual, el cliente puede indicar si se encuentra conforme con lo desarrollado o no dentro del software. Por otra parte, Medina (2020), indican que las pruebas funcionales tienen como prioridad la comprobación de las funciones en los aplicativos de software, el cual debe cumplir ciertos estándares y requisitos solicitados por el cliente, además, demuestra lo que el sistema realiza y que se encuentre funcionando perfectamente.

Banda (2022), indica que las UI Testing son aquellas pruebas que se realizan a las plataformas web, identificando la perfecta integración en cada una de sus funcionalidades, gráficos, idiomas, presentación, etc. Por otro lado, Fernández (2018), indican que las UI Testing tienen como principal función la de comprobar que los estándares de calidad tecnológicos establecido por el ISTQB se cumplan, indicando los botones, cajas de texto, diseños, funcionalidades, etc; con esto, poder entregar un buen producto a los usuarios consumidores de las plataformas y sistemas web.

Rivera (2021) menciona que las API Testing son pruebas que se realizan netamente a los microservicios, los cuales, independizan las funcionalidades que son desarrolladas, para así poder

evitar caídas totales de los sistemas y estos puedan ser vista de formas independientes al ser consumidos por el equipo de desarrollo frontend. Las pruebas que se realizan a las API son de suma importancia debido a los métodos que presentan para realizar automatización de escenarios de prueba.

De tal manera Lopez, Segura y Ruiz (2020), hace mención que las API al ser probadas son importantes para la integración que se realiza con otros procesos del software. Los frameworks que están creados se utilizan para automatizar una variedad de casos de prueba, los cuales, son desarrollados para un fácil uso por parte del programador o tester. Además, se adaptan de forma sencilla a los nuevos requerimientos que puedan presentar en la empresa o en las reglas del negocio. La ventaja que se tiene en algunos frameworks es la variedad de lenguajes que se usan para programar y es versátil, además, las sintaxis planteadas son sencillas para una persona que programa a nivel avanzado como para alguien que es un principiante. Como indica Lopez (2020), estas pruebas son para lograr detectar errores que se puedan presentar en las API o microservicios desarrollados, además, hoy en día, las API son muy común para sus usos y también criticas al momento de testear. Por ello, con la automatización se busca que se pueda realizar el debuger en tiempo real.

Para automatizar pruebas, se debe contar con un nivel de experiencia en pruebas, ya sea manual o automatizada que sea mínima, esto debido a que se requiere tener una visión de las reglas del negocio y los escenarios o casos de prueba a realizar, así como menciona Garousi, Küçük y Felderer (2019), que una automatización no siempre generar scripts tal cual lo que se requiere y allí es donde se debe tener los conocimientos básicos para poder adaptar a la realidad del negocio que se necesita, además, es importante tener conocimientos de BDD o Gherkin, así

como mencionan los autores, ya que son herramientas de grandes beneficios y comunicación entre el área de negocio y el área técnica del proyecto que se estaría realizando.

Es importante utilizar metodologías para automatizar, así como mencionan Keleş, Balaman, Özdemir (2021), se ha realizado una serie de metodologías que tienen forma pragmática, con las cuales, se puede realizar pruebas a niveles altos de estándares, con capacidades de madurez y herramientas comerciales de código abierto, para realizar una serie de automatizaciones en conjunto, se debe tener una mayor cobertura de escenarios y casos de prueba, de esta forma, poder encontrar los bug en tiempo real y se pueda corregir de forma ágil. Es por ello que se define los casos a probar en tiempos establecidos, para así poder tener un mejor reporte de evidencias.

4.3. Definición de Términos Básicos

El testing o pruebas de software es un proceso crítico que se utiliza para garantizar que el software se comporte de acuerdo a lo esperado y cumpla con los requisitos de los usuarios. Consiste en ejecutar un conjunto de pruebas automatizadas o manuales para identificar errores, problemas de rendimiento y seguridad. El testing es esencial para mejorar la calidad del software, detectar problemas temprano en el ciclo de vida del software y reducir el riesgo de fallos costosos. Un enfoque adecuado de testing implica una combinación de técnicas de prueba, herramientas de automatización, diseño, planificación, ejecución y seguimiento para garantizar que el software sea confiable, escalable y fácil de mantener.

Las pruebas funcionales son un tipo de pruebas de software que se centran en evaluar el comportamiento del software en función de los requisitos funcionales establecidos. Estas pruebas se realizan para garantizar que el software cumpla con las expectativas de los usuarios y se ajuste

a las especificaciones del proyecto. Las pruebas funcionales se basan en el análisis de los casos de uso y escenarios de usuario, y se centran en la validación de las funcionalidades clave del software. Las pruebas funcionales pueden ser manuales o automatizadas, y se pueden realizar en diferentes niveles, desde la unidad hasta las pruebas de aceptación del usuario.

Las pruebas de regresión son un tipo de pruebas de software que se utilizan para asegurarse de que las modificaciones o cambios realizados en el software no hayan introducido errores o problemas en el funcionamiento de funcionalidades existentes. Estas pruebas se realizan después de realizar cambios en el software y se centran en la validación de que las funcionalidades que antes funcionaban correctamente, siguen funcionando correctamente después de los cambios. Las pruebas de regresión pueden ser manuales o automatizadas, y se pueden realizar en diferentes niveles, desde la unidad hasta las pruebas de aceptación del usuario. Las pruebas de regresión son esenciales para garantizar la calidad y estabilidad del software a medida que se va desarrollando.

Las UI Testing son un tipo de pruebas de software que se utilizan para evaluar la interfaz de usuario de una aplicación o sitio web. Estas pruebas se enfocan en la funcionalidad y la apariencia visual de la interfaz, incluyendo la navegación, el diseño, la legibilidad, la interacción, el tiempo de espera y la usabilidad en general. Las pruebas de UI pueden ser manuales o automatizadas y pueden involucrar a usuarios reales o simulados para evaluar la facilidad de uso y la experiencia del usuario final. Las pruebas de UI son importantes para asegurar que la interfaz sea intuitiva, fácil de usar y cumpla con las expectativas de los usuarios. Estas pruebas tienen como ventaja que tiene variedad de herramientas de automatización.

Las API testing son un tipo de pruebas de software que se utilizan para evaluar la funcionalidad y el rendimiento de las interfaces de programación de aplicaciones. Estas pruebas se enfocan en la comunicación y la integración entre diferentes componentes de software, incluyendo servidores, bases de datos, aplicaciones y servicios web. Las pruebas de API pueden involucrar la validación de los datos de entrada y salida, la autenticación y autorización, el manejo de errores y la escalabilidad del sistema. Estas pruebas pueden ser manuales o automatizadas y son importantes para garantizar la calidad, la eficiencia y la fiabilidad de los sistemas de software.

Las pruebas de carga son un tipo de pruebas de software que se utilizan para medir el comportamiento y rendimiento del sistema cuando se somete a una carga de trabajo. Estas pruebas simulan la actividad de usuarios reales y miden la capacidad del sistema para manejar una carga de trabajo determinada sin afectar negativamente el tiempo de respuesta o la calidad del servicio. Las pruebas de carga son importantes para detectar cuellos de botella, errores de rendimiento y problemas de escalabilidad en el sistema, y ayudan a identificar la capacidad máxima del sistema. Estas pruebas pueden ser realizadas de manera manual o automatizada, y se pueden realizar en diferentes niveles, desde la unidad hasta las pruebas de aceptación del usuario. El objetivo de las pruebas de carga es garantizar que el sistema pueda manejar la carga de trabajo prevista de manera efectiva y sin interrupciones para los usuarios finales.

5. Propuesta de Solución

5.1. Metodología de la Solución

Para poder realizar la mejora en el proceso de testing del área de calidad de software, se establecerá el uso de Karate DSL como framework para la implementación de la automatización de pruebas. Así que, para poder visualizar los resultados obtenidos en esta mejora, se realizará dos cuadros, los cuales serán para realizar la comparación del proceso de testing antes de la implementación del sistema y los resultados después de implementar el sistema de automatización de pruebas.

Parte de la propuesta que se está dando para la solución es el uso de agile, con su metodología, donde, a través de scrum se organizará la estructura a realizar en el proyecto, permitiendo de esta forma un entregable seguro y de calidad. Trabajar con agile permite una colaboración eficaz en equipo, con lo cual, viene la autoorganización de cada colaborador, además, de la comunicación asertiva, la atención al cliente y entrega de valor. Con este framework se logra rescatar la labor de los equipos frente a cualquier dificultad que se presente dentro del proyecto a través de la entrega incremental y los ciclos que se puedan dar. Es por ello que hoy en día los negocios y proyectos se enfocan en adaptar sus procesos al uso de esta metodología, por su variedad de soluciones que presenta ante la implementación o cambios de algún flujo ya se tecnológico, de negocio o empresarial.

Scrum está basado en ciclo denominados sprint (los cuales pueden ser conocidos en el método clásico como un cronograma), del cual, se generan valores, buenas prácticas y pilares que se determinan en el manifiesto ágil. Este framework es adaptable a cualquier tipo de modelo de gestión de proyectos, sobre todo, los que sean de relevancia para los negocios digitales. Un sprint

es un período de tiempo fijo y limitado en el que un equipo de desarrollo de software trabaja en tareas específicas para lograr objetivos concretos. Cada sprint dura un aproximado de dos semanas, o a veces puede durar hasta 4 semanas, según la dificultad que presente el proyecto y también los acuerdos que se establezcan por parte de negocio.

5.2. Desarrollo de la Solución

Planificación de Pruebas Automatizadas

La metodología que se viene trabajando es la ágil, utilizando como marco de trabajo a Scrum, por lo cual, se presentará el siguiente desarrollo de solución basado en la estructura que este marco de trabajo plantea; es por ello, que se dividirá la solución por sprint, y así poder determinar las fechas y flujos para la implementación. Se debe dejar en claro que esta propuesta de solución tiene como indicio la conformidad del equipo de desarrollo, de esta forma elaborar el trabajo en equipo.

Figura 2

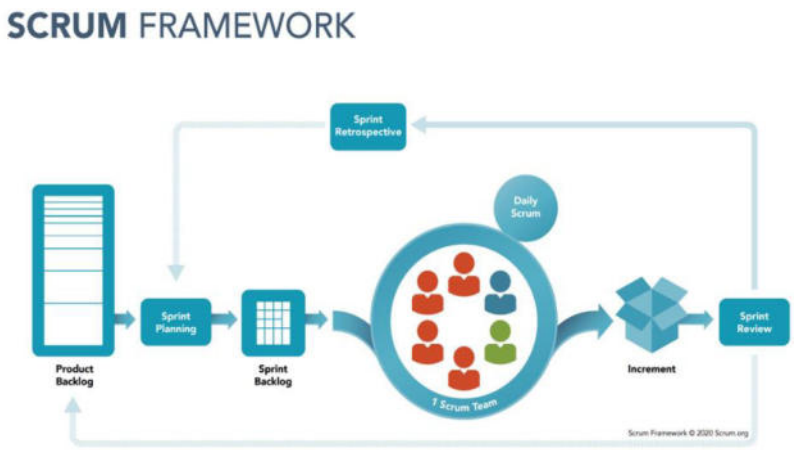
Metodología Ágil

Nota: El gráfico es la metodología Agile en el 2023. Tomado de ebf.com.es

Para empezar, se deberá realizar la planificación de los sprint, los cuales, tendrán cada uno una duración de dos semanas, teniendo como Sprint inicial (Sprint 0) toda la parte de estrategia de pruebas automatizadas de API, construcción de script, ejecución de script, integración de Karate con Jenkins y Docker, por último, el reporte de pruebas de API; en el Sprint 1 se priorizará las pruebas de carga, para lo cual, se realizará la estrategia de pruebas automatizadas de Carga, la construcción de script, la ejecución y reporte de pruebas de Carga; por último, en el Sprint 3 se realizará la ejecución de pruebas automatizadas de Regresión, con modificaciones en los scripts.

Figura 3

Framework Scrum

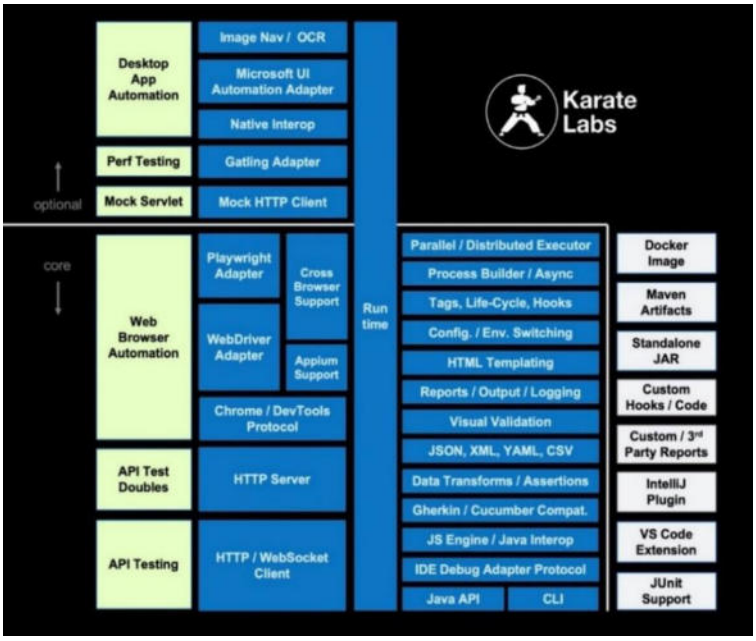


Nota: El gráfico es el framework Scrum en el 2023. Tomado de Scrum.org

Para poder realizar la automatización de pruebas, se hará uso del framework Karate, el cual, es una herramienta open source, que tiene una gran sencillez y facilidad de uso, debido a que viene integrado con código java basado en la estructura del lenguaje específico de dominio (DSL) llamado Gherkin.

Figura 4

Arquitectura de Karate



Nota: El gráfico es la arquitectura de Karate. Tomado de Github.com/karatelabs

Sprint 1: Pruebas Automatizadas de API

Para la propuesta de solución presentada en este informe, se utilizarán datos referenciales públicos, esto debido a que las empresas bancarias solicitan un alto grado de confidencialidad, además, como indica el código de ética para la investigación de la Universidad Católica Sedes Sapientiae (véase anexo 1) en el capítulo II de Principios éticos para la investigación, punto 2 de Respeto de la Privacidad, se deberá respetar el derecho a no exponer información que no sea permitida por las organizaciones.

Estrategia de Pruebas Automatizadas de API

Para iniciar con la implementación de pruebas automatizadas, se debe realizar un documento de estrategias de pruebas automatizadas (véase anexo 2), en el cual, principalmente se debe detallar las herramientas y metodología de trabajo.

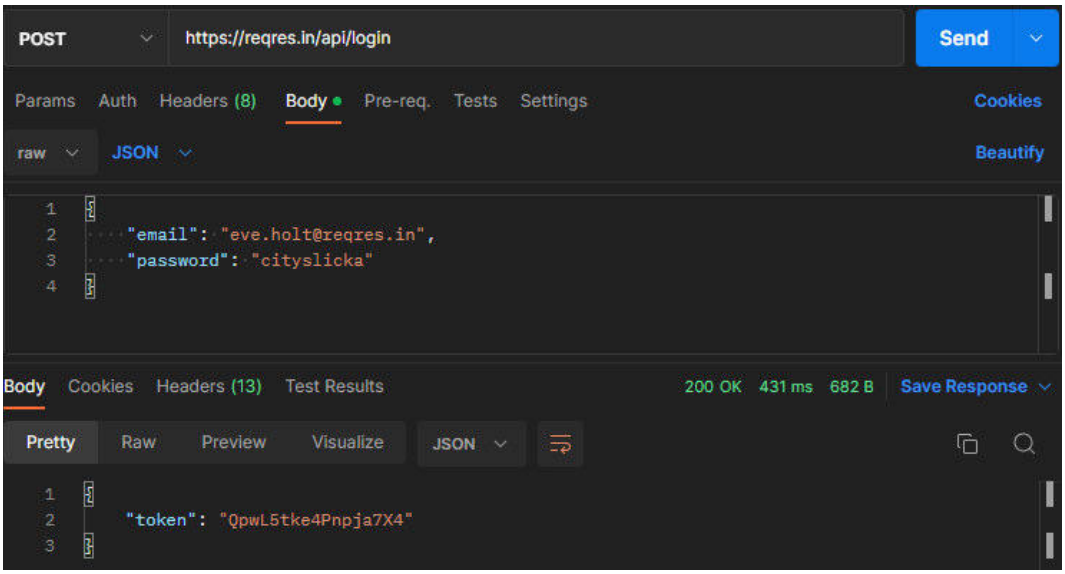
Ejecución de Pruebas Automatizadas

Construcción de Scripts

Para este caso, se utilizarán 70 pruebas, de las cuales, 10 serán de forma manual con la herramienta postman (se mostrará solo 1) y 60 se deberán automatizar con karate, que trabaja con Gherkin y Java para la elaboración de scripts de pruebas.

Figura 5

Prueba Manual de login Successful



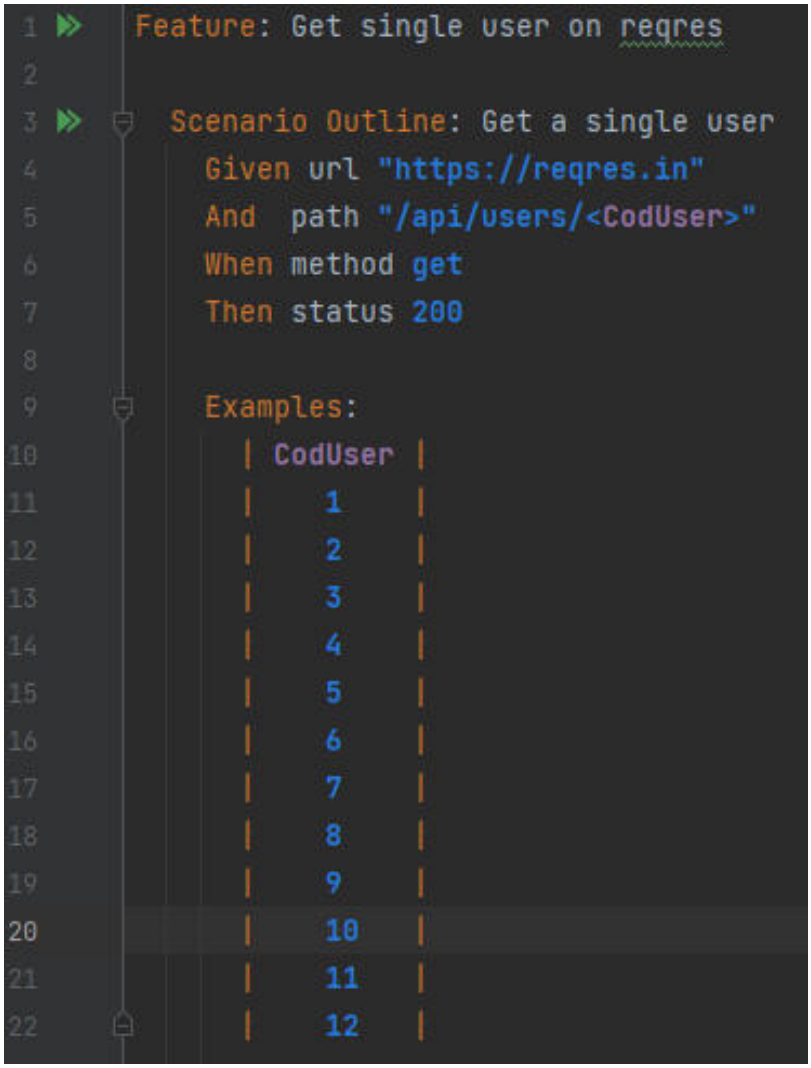
Nota: Prueba manual en postman. Elaboración propia

Get single use on reqres

Este es un escenario de consulta de usuarios, a través del cual se busca realizar un método get, para poder traer la información de los diferentes usuarios según se establezca en los casos de prueba aprobados por el negocio.

Figura *6*

Feature Get single use on reqres



Nota: Script del Scenario Get a single user. Elaboración propia

Get list users on reqres

Este es un escenario de consulta de listas, a través del cual se busca realizar un método get, para poder traer los diferentes listados de usuarios según se establezca en los casos de prueba aprobados por el negocio.

Figura 7

Feature Get list users on reqres

Nota: Script del Scenario Get a list user. Elaboración propia

Post user on reqres

Este es un escenario de creación de usuarios, a través del cual se busca realizar un método post, para poder enviar un request body como parámetro de entrada de los usuarios a crear según establezca el negocio.

Figura 8

Feature Post user on reqres

```
Feature: Post user on reqres

  Background:
    * url "https://reqres.in"
    * path "/api/users"
    * request { "name": "#(name)", "job": "#(job)" }

  Scenario Outline: Post a user
    When method post
    Then status 201

    Examples:
      | name     | job      |
      | Juan     | Leader   |
      | Gustavo  | Doctor   |
      | Maria | Engineer |
      | Roberto | Administrator |
      | Ana | Teacher |
      | Roger | Lawyer |
      | Luis | Manager |
      | Marco | Scientific |
      | Miguel | Sportsman |
      | Antonio | Writer |
      | Arturo | Lawyer |
      | Nataniel | Manager |
      | Angela | Scientific |
      | Jesus | Sportsman |
      | Cristian | Writer |
```

Nota: Script del Scenario Post a user. Elaboración propia

Put user on reqres

Este es un escenario de actualización de datos, a través del cual se envía como parámetros de entrada un código de usuario y los datos a actualizar utilizando el método put, y así poder tener el api operativo según requerimientos de negocio.

Figura 9

Feature Put user on reqres

```
Feature: Put user on reqres

  Scenario Outline: Put a user
    Given url "https://reqres.in"
    And path "/api/users/<CodUser>"
    And request { "name": <name>, "job": <job> }
    When method put
    Then status 200

    Examples:
      | CodUser |  name    | job     |
      |    1    | Elias    | Doctor  |
      |    2    | Ruben    | Administrator  |
      |    3    | Marco | Writer |
      |    4    | Silvia | Manager |
      |    5    | Roxana | Sportsman |
      |    6    | Mateo | Scientific |
      |    7    | Marcos | Lawyer |
      |    8    | Maribel | Teacher |
      |    9    | Natalia | Engineer |
      |    10   | Jorge | Writer |
      |    11   | Oscar | Sportsman |
      |    12   | Orlando | Lawyer |
      |    13   | Jeni | Engineer |
      |    14   | Claudia | Manager |
      |    15   | Deysy | Administrator |
```

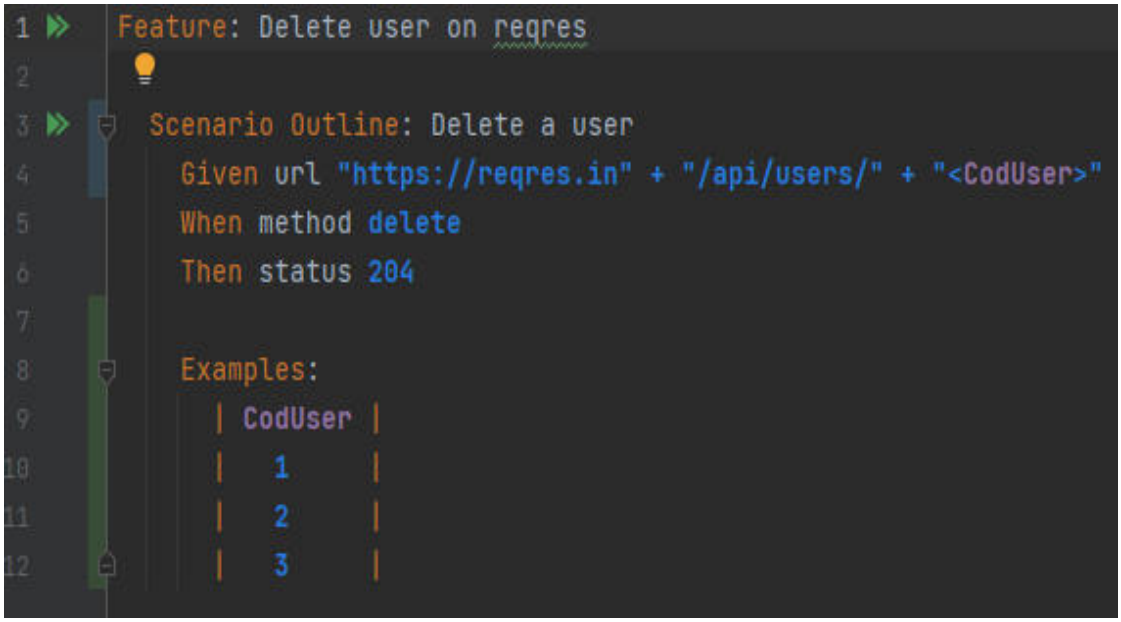
Nota: Script del Scenario Put a user. Elaboración propia

Delete user on reqres

Este es un escenario de eliminación de usuarios, a través del cual, se usa el método Delete y el código de usuario como parámetros de entrada y así poder eliminar el usuario solicitado según requerimientos de negocio.

Figura 7

Feature Delete user on reqres

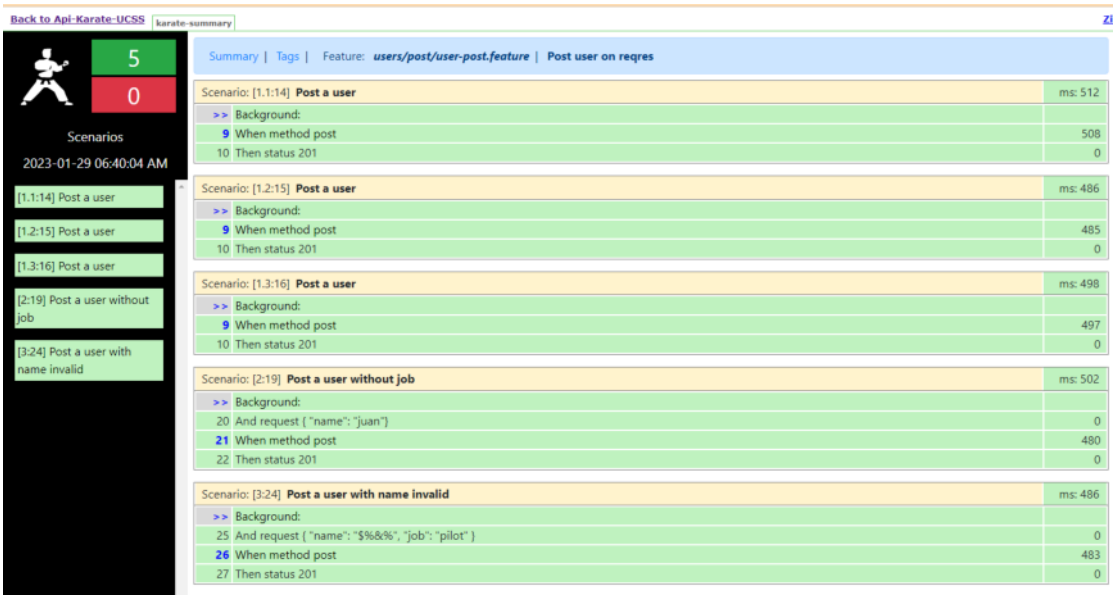


Nota: Script del Scenario Delete a user. Elaboración propia

Reporte de Pruebas Automatizadas

Figura 11

Reporte Prueba Automatizador



Nota: Reporte de Pruebas Automatizadas. Elaboración propia

5.3. Factibilidad Técnica - Operativa

En la factibilidad técnica, se indicará los requisitos que tiene la implementación del sistema de automatización de pruebas tanto en hardware, software y parte humana, para poder iniciar con el presente proyecto. Las entidades bancarias cuentan con PC o laptops, las cuales, son utilizadas por todo el personal, especialmente, por los colaboradores que están enfocados en la parte de tecnología, finanzas y negocio. Además, hoy en día se viene realizando el trabajo remoto, por lo cual, es necesario tener equipos técnicos que cumplan con características mínimas.

Tabla 1

Requisitos de Hardware

Cantidad	Descripción
1	Procesador Intel Core i7
1 TB	Disco Duro

250 GB	Disco Solido SSD

Nota: La tabla representa los requisitos de hardware. Elaboración propia

Respecto al Software, se hace una reseña de las principales herramientas informáticas utilizadas en el proceso, a continuación, se hace una referencia a cada una de ellas.

Gradle

Es una herramienta de construcción de software que se utiliza para automatizar el proceso de construcción, prueba y distribución de aplicaciones. Permite manejar proyectos de cualquier tamaño y complejidad, y es altamente personalizable. Además, es compatible con múltiples lenguajes de programación y proporciona una sintaxis de script fácil de usar para los desarrolladores y tester.

Karate

Es un framework de automatización de pruebas de software escrito en Java, que se enfoca en pruebas de API REST y servicios web. Se caracteriza por su simplicidad, rapidez y la inclusión de todas las herramientas necesarias para la escritura y ejecución de pruebas. Karate también ofrece funcionalidades avanzadas, como la generación de datos de prueba, la verificación de resultados y la integración con otras herramientas de automatización y gestión de pruebas. En resumen, Karate es una herramienta muy útil para la automatización de pruebas de software, especialmente para pruebas de API.

Java

Es un lenguaje de programación que sirve para desarrollar aplicaciones de software en diferentes plataformas y sistemas operativos. Es conocido por ser un lenguaje simple, seguro y portátil, lo que significa que el código escrito en Java puede ejecutarse en diferentes dispositivos sin necesidad de adaptaciones. Además, cuenta con una gran cantidad de bibliotecas y herramientas que facilitan el desarrollo de software. Java es muy popular en el desarrollo empresarial y de aplicaciones críticas gracias a su gestión automática de memoria y características de seguridad.

Cucumber

Es una herramienta de automatización de pruebas de software que se enfoca en el lenguaje natural y la colaboración entre equipos de desarrollo y de pruebas. Permite escribir pruebas en un lenguaje natural y estructurado en forma de escenarios, utilizando la metodología de BDD (Behavior Driven Development). Cucumber también incluye la funcionalidad de ejecutar y generar informes detallados de las pruebas. En resumen, Cucumber es una herramienta poderosa para la automatización de pruebas, que fomenta la colaboración y el entendimiento común entre desarrolladores y testers

Gherkin

Es un lenguaje de especificación utilizado en el desarrollo de software para describir el comportamiento esperado de un sistema en un formato legible para las personas. Gherkin se utiliza comúnmente en la metodología de desarrollo de software BDD (Desarrollo Dirigido por Comportamiento) y permite a los desarrolladores, testers y stakeholders colaborar y entender el comportamiento esperado del software de una manera clara y concisa.

Gatling

Es una herramienta de pruebas de carga y rendimiento para aplicaciones web. Permite simular el comportamiento de usuarios concurrentes en un sitio web. Es una herramienta de código abierto escrita en Scala. Proporciona informes detallados y métricas sobre el rendimiento del sitio web bajo carga. Es una herramienta popular en el desarrollo de software y en la industria de pruebas de software.

Jenkins

Es una herramienta de automatización de integración y entrega continua de código abierto, que se utiliza para construir, probar y desplegar software de manera automatizada. Se integra fácilmente con diversas herramientas de desarrollo y control de versiones, y permite la configuración de pipelines de CI/CD, que automatizan el proceso de construcción y despliegue de aplicaciones. Jenkins también incluye un amplio conjunto de plugins que permiten personalizar y extender su funcionalidad, tiene reportes html para diferentes herramientas de automatización, además de una interfaz gráfica de usuario fácil de usar. En conclusión, Jenkins es una herramienta esencial para equipos de desarrollo de software que buscan automatizar y acelerar el proceso de construcción, pruebas y despliegue de sus aplicaciones.

IntelliJ IDEA

Esta herramienta es un entorno integrado para desarrolladores, que se utiliza para poder crear aplicaciones web, móviles o de microservicios, además de tener una versatilidad de lenguajes de programación, siendo Java el más importante. Se caracteriza por su interfaz de usuario intuitiva, facilidad de uso y funcionalidades avanzadas, como la depuración,

refactorización, integración de control de versiones y soporte para diferentes marcos y herramientas de desarrollo. En resumen, IntelliJ IDEA es una herramienta muy popular y poderosa para desarrolladores de software en diversos lenguajes de programación para web, móvil y servicios.

Scala

Es un lenguaje de programación de propósito general que se ejecuta en la plataforma Java. Combinas características de la programación orientada a objetos y la programación funcional. Es un lenguaje de código abierto y altamente escalable. Scala es popular en el desarrollo de aplicaciones web, big data y procesamiento de lenguaje natural. Tiene como ventaja una sintaxis sencilla de utilizar, con la cual, se puede digitar un código fácil y legible para el programador.

JSON

JSON es un formato de intercambio de datos ligero y fácil de leer para la comunicación entre aplicaciones. Se basa en sintaxis de objetos y arrays en JavaScript y se utiliza para representar datos estructurados. Los datos se organizan en pares clave-valor y se pueden anidar para formar estructuras complejas. JSON es ampliamente utilizado en aplicaciones para transmitir datos entre el cliente y el servidor.

5.4. Cuadro de Inversión

Se realizó un presupuesto basado en recolección de datos utilizados para la realización del proyecto elaborado en el presente informe.

Tabla 8

Presupuesto de Datos Obtenidos

Presupuesto de Datos Obtenidos			
Descripción	Cantidad	Precio (Soles)	Subtotal (Soles)
Laptop	1	3500	3500
Folder	1	4	4
Hojas bond	10	1	10
Internet	1	200	200
Cuadernos	1	5	5
Lapiceros	3	9	27
Impresoras	1	700	700
Reuniones	1	50	50
Total			S/. 4496.00

Nota: El gráfico es presupuesto de datos obtenidos. Elaboración propia

Costos de Equipo de Automatización de Pruebas

A continuación, se establece los costos obtenidos en la implementación de la automatización para estas pruebas realizadas, dentro de las cuales, se establece el salario de cada colaborador que es parte del equipo del proyecto.

Tabla 3

Cuadro de Costos de Equipo de Automatización de Pruebas

Costos de Equipo de Automatización de Pruebas			
Descripción	Cantidad	Salario	Subtotal
Project Manager	1	S/ 12,000.00	S/ 12,000.00
Product Owner	1	S/ 10,000.00	S/ 10,000.00
Scrum Master	1	S/ 7,000.00	S/ 7,000.00
DevOps	2	S/ 9,000.00	S/ 18,000.00
Senior QA Automation	1	S/ 9,000.00	S/ 9,000.00
QA Automation	2	S/ 6,000.00	S/ 12,000.00
Senior Backend Developer	1	S/ 11,000.00	S/ 11,000.00
Backend Frontend	3	S/ 7,000.00	S/ 21,000.00

Analista de Datos	1	S/	6,000.00	S/	6,000.00
UX/UI	1	S/	4,000.00	S/	4,000.00
Total				S/	110,000.00

Nota: La tabla muestra el presupuesto automatizado. Elaboración propia

6. Análisis de Resultados

6.1. Análisis Costo - Beneficio

Con respecto a los resultados, primero se hará una evaluación de cómo se encontraba el proceso de testing antes de la propuesta de solución implementada, debido a que se detectó una gran cantidad de retrasos en los entregables de pruebas solicitados.

Debido a los retrasos generados y la detección de incidencias de funcionalidades en producción, se hizo una evaluación de la situación en contexto, donde se llegó a la conclusión que la metodología de trabajo tradicional que se estaba utilizando no era la adecuada para el proceso de testing, debido a que no lograba cumplir con las expectativas que el cliente o usuario requerían cubrir con respecto a los sistemas.

El equipo de testing trabajaba bajo una metodología de tipo cascada a través de un diagrama de Gantt, en la cual, se elaboraba en un Project un cronograma de actividades (véase anexo 3), a través del cual, se enfocaban en la finalización del proyecto, sin contar los feedback que puedan obtenerse durante la construcción del producto, lo cual, generaba que el sistema presente fallas al momento de ser desplegado en producción.

Para el diseño y elaboración de casos de pruebas de api manuales, se debía trabajar con un formato establecido de casos de pruebas (véase anexo 4), donde se debía completar los campos de fecha, estado de prueba, resumen de funcionalidad, prioridad, pre condiciones, datos de ingreso, paso a paso de la prueba, resultado esperado y ambiente de prueba; lo cual, generaba un retraso en el proceso de API Testing.

Además, para la realización de las pruebas de carga se esperó que se realicen las pruebas manuales de las API y que estas sean aprobadas, luego se procede a desplegar en producción, una vez completado el proceso, se podía realizar las pruebas de carga de cada API, lo cual, generaba otro retraso en el proceso de Load Testing.

Para poder obtener un detalle de los resultados logrados tras realizar la implementación del sistema de automatización de pruebas, se determina si el sistema de automatización de pruebas con karate mejora el proceso de testing en una empresa bancaria.

Para este proyecto, se realiza la ejecución de 250 pruebas funcionales, de las cuales 70 fueron automatizadas, tomando un tiempo de 10 minutos por cada caso de prueba, y en tiempo total sumando 2865 minutos.

Tabla 4

Cuadro de Mejora del proceso de Testing

Mejora del proceso de Testing					
Casos de Prueba	Cantidad de Pruebas	Tiempo en minutos			
		Prueba Manual	Prueba Automatizada	Ejecución de Prueba Manual	Ejecución de Pruebas Automatizada
Post login Successful	25	10	0	250	0
Get single use on reqres	50	30	12	1500	600
Get list users on reqres	50	35	15	1750	750
Post user on reqres	48	30	15	1440	720
Put user on reqres	47	32	15	1504	705
Delete user on reqres	30	15	3	450	90

Total	250	25.33	10.00	6894	2865

Porcentaje Tiempo de Pruebas Funcionales (TPF% = (EPM / EPA) * 100
41.56
Mejora del Proceso de Testing (100% - TPF%)
58.44

Nota: Detalle de la mejora del proceso. Elaboración propia

El resultado obtenido en la mejora del proceso de testing es de un 61%, esto debido a que, a pesar de la gran cantidad de casos de prueba que se encuentran en listado, con la implementación del sistema de automatización de pruebas se logra realizar una reducción de tiempo por encima del que se solía utilizar en procesos anteriores. Por ello, se comprende que la implementación del sistema de automatización de pruebas con Karate ha logrado generar un gran impacto y mejora exponencial en el proceso testing del ciclo de vida de desarrollo de software del área de calidad de software.

También se determina si el sistema de automatización de pruebas con karate agiliza el proceso de testing en una empresa del rubro bancario, se determinarán que a través del tiempo obtenido en minutos y transformado a horas, el sistema de automatización de pruebas permite la agilización del proceso de testing del área de calidad de software. Para realizar la ecuación, se toma el tiempo de duración de un sprint equivalente a dos semanas, en total 10 días hábiles (lunes a viernes), convertido a horas tenemos 240, por otra parte, se usará el tiempo de pruebas manuales que es de 114 horas y el tiempo de pruebas automatizadas que es de 47 horas según los resultados obtenidos. Se utilizan las siguientes variables para la ecuación:

Variables de Formula

TS: Tiempo Solicitado

TPM: Tiempo de Pruebas Manuales

TPA: Tiempo de Pruebas Automatizadas

TEPM: Tiempo Entrega Pruebas Manuales

TEPA: Tiempo Entrega Pruebas Automatizadas

Formula y Aplicación

TEPM = TS - TPM = 240 - 114 = 126

TEPA = TS - TPA = 240 - 47 = 193

Con la formula aplicada, donde se estiman las 240 horas que dura un sprint y con los datos obtenidos de la mejora del proceso de testing, donde las pruebas manuales toma alrededor de 114 horas para culminar sus pruebas, a diferencia de las pruebas automatizadas que solo toman alrededor de 47 horas, dejando un restante de 193 horas como beneficio para otras actividades, que, a diferencia de las pruebas manuales, da solo 126 horas de diferencia, lo que genera una agilización de 67 horas que pueden ser utilizadas para otra actividades del área de calidad de software, con lo cual, se demuestra la agilización del proceso de testing con respecto a las pruebas funcionales.

Se determina que el sistema de automatización de pruebas con karate mejora el proceso de API Testing, para ello se presenta la mejora que obtuvo el proceso API Testing en la empresa del rubro bancario a través de la implementación del sistema de automatización de pruebas con karate. Dentro de cada sprint se realiza una evaluación del proceso de testing, estas son realizadas con el framework Karate, lográndose evitar los flujos y procesos repetitivos, se verifica la mejora

del proceso al realizar las pruebas de regresión según las especificaciones del cliente para las funcionalidades requeridas.

Para poder realizar una mejor evaluación de resultados, se opta por realizar ecuaciones lineales, lográndose mejoras en el proceso API Testing del área de calidad de software; además, los resultados obtenidos han sido positivos, demostrándose una mejora significativa con la propuesta.

La ecuación demuestra el porcentaje de mejora que la implementación del sistema de automatización de pruebas ha logrado sobre el proceso de API Testing. Para ello, se ha establecido un conjunto de 70 casos de pruebas manuales de forma inicial, de las cuales, se automatizarán 60 para evaluar la mejora,

Variables de Formula

PM: Pruebas Manuales

PA: Pruebas Automatizadas

CFPM: Cantidad final de pruebas manuales

Formula

CFPM = PM - PA

Aplicación de Formula

10 = 70 - 60

La aplicación de la formula, se observa que, de 70 pruebas manuales que se solicitaron en un inicio, al realizar la automatización de pruebas a 60 evaluadas, la cantidad final de pruebas manuales que quedan para testear es de solo 10.

En el proceso de API Testing, se realiza la entrega de 70 pruebas manuales con sus respectivas evidencias, al automatizar 60 se logra determinar la mejora que se da dentro del proceso.

Tabla 5

Mejora del proceso de API Testing

Mejora del Proceso de API Testing				
Cantidad de Pruebas Manuales	Cantidad de Pruebas Automatizadas	Fórmula de Cálculo de Porcentaje	Cálculo de Porcentaje	Resultado Obtenido
PM = 70	PA = 60	(PA/PM) * 100	(60/70) * 100	**85.71 %**

Nota: Representación de mejora del proceso de Test. La tabla es elaboración propia

El resultado obtenido a través de la automatización de 60 casos de prueba es de un 85.71% de mejora en el proceso de API Testing, con lo cual, se demuestra un mejor rendimiento de ejecuciones de las pruebas en el área de calidad de software de la empresa del sector bancario. Además, se logra identificar que, con esta implementación del sistema de automatización de pruebas con karate, el proceso de testing logra agilizar su flujo en la realización de pruebas funcionales y de regresión, logrando de esta manera entregar reportes de pruebas organizados y seguros, de acuerdo a los requerimientos que el área de negocio ha solicitado.

Se Determinar también que el sistema de automatización de pruebas con karate mejora el proceso de Load Testing, para este objetivo específico, se muestra la mejora que se obtuvo del proceso de Load Testing con la implementación del sistema de automatización de pruebas con karate. Una vez realizado las pruebas automatizadas de API, en el siguiente sprint se elabora las pruebas de

carga, con las cuales, se determinará el rendimiento de soporte que tendrá cada api, y a través de la automatización de pruebas se podrá evaluar el porcentaje de mejora en su rendimiento del proceso de testing dentro del área de calidad de software.

Por ello, para poder obtener los resultados del proceso evaluado, se opta por realizar ecuaciones lineales, de esta manera, se logrará obtener los resultados generados por la implementación del sistema de automatización de pruebas sobre la mejora del proceso de Load Testing del área de calidad de software. Por otra parte, se ha logrado visualizar que los resultados obtenidos con la implementación del sistema son positivos, con lo cual, se comprende que se ha logrado una mejora exponencial con la propuesta de solución del sistema de automatización de pruebas para el proceso de Load Testing del ciclo de vida de desarrollo del software en el área de calidad de software.

Con la ecuación que se presenta, se demuestra el porcentaje de mejora que la implementación del sistema de automatización de pruebas ha logrado sobre el proceso de Load Testing. Para este caso, se ha tomado los 70 casos de pruebas manuales del proceso de API Testing, de las cuales, se automatizarán 50 para evaluar la mejora, y resultará la cantidad de pruebas manuales que quedarán como entrega final.

Variables de Formula

PMC: Pruebas Manuales de Carga

PAC: Pruebas Automatizadas de Carga

CFPMC: Cantidad final de pruebas manuales

Formula

CFPMC = PMC - PAC

Aplicación de Formula: 20 = 70 – 50

En la aplicación de la formula, se logra detectar que, de 70 pruebas manuales que se solicitaron en un inicio, al realizar la automatización de pruebas a 60 evaluadas, la cantidad final de pruebas manuales que quedan para testear es de solo 10.

En el proceso de API Testing, se realiza la entrega de aproximadamente 70 pruebas manuales con sus respectivas evidencias, sin embargo, al automatizar 60, se logrará determinar la mejora que se da dentro del proceso.

Tabla 6

Mejora del proceso de Load Testing

Mejora del Proceso de Load Testing				
Cantidad de Pruebas Manuales de Carga	Cantidad de Pruebas Automatizadas de Carga	Fórmula de Cálculo de Porcentaje	Cálculo de Porcentaje	Resultado Obtenido
PM = 70	PA = 50	(PAC/PMC) * 100	(50/70) * 100	**71.43 %**

Nota: Representación de mejora de Load Testing. La tabla es elaboración propia

El resultado obtenido con la automatización de 50 casos de prueba nos da un 71.43% de mejora en el proceso de Load Testing, demostrándose la mejora en el rendimiento de ejecuciones de las pruebas en el área de calidad de software de la empresa del sector bancario. Además, se identificó con esta implementación del sistema de automatización de pruebas con karate, que el proceso de testing logra agilizar su flujo en la realización de pruebas de rendimiento, logrando de esta manera entregar reportes de pruebas estructurados, organizados y seguros, de acuerdo a los requerimientos que el área de negocio ha solicitado.

7. Aportes más Destacables a la Empresa

A lo largo del proyecto, se realiza una evaluación del proceso de pruebas del área de calidad de software, el cual se encuentra en un proceso de migración respecto a su metodología de trabajo, es necesario por ello innovar el flujo de testing. En este contexto se implementa el sistema de automatización de pruebas para los procesos de testing.

Entre los principales aportes que se realizó a la empresa se genera un impulso por la innovación y el uso de nuevas herramientas, las cuales, están dando resultados positivos en otros países con tecnología avanzada, respecto a la cultura organizativa, se aporta una constante capacitación para el equipo, donde el clima laboral mejoró, debido a que se creó comunidades de estudio y aprendizaje de nuevas tecnologías.

En este sentido los colaboradores que tienen conocimientos de programación y testing, asumen los roles de liderazgo, al poder comprender de forma rápida la implementación del sistema de automatización de pruebas, siendo ellos los que impulsan el uso del sistema para la mejora y agilización del flujo. Los colaboradores se encargan de brindar capacitaciones al personal del área de testing, con lo cual, se puede dar un avance en la productividad de la elaboración y generación de reportes de pruebas.

En cuanto al trabajo en equipo, las relaciones humanas lograron un crecimiento exponencial, se creó una mayor integración por parte de los equipos de desarrollo, testing y devops, con el equipo de negocio en la empresa.

Se tuvo como principal reto la migración de la herramienta, de un entorno manual a uno automático, lo cual representa el éxito del proyecto.

8. Conclusiones

Se evidencia que el sistema de automatización de pruebas basado en el framework karate logra mejorar el proceso de testing en la empresa bancaria. Según R. Cubas (2017), la automatización de pruebas es aquel paso de conversión de la ejecución de pruebas manuales a la ejecución desatendida de pruebas automatizadas, es decir, sin necesidad de que un tester tenga que intervenir, ya que solo es cuestión de presionar un botón para que todas las pruebas automatizadas puedan ser ejecutada, además, de permitir ejecutar un número de pruebas mayor, detección y reducción de bug en la ejecución de pruebas y mejora estandarizada del proceso de testing. Además, S. Fernández (2018), hace mención que las pruebas automatizadas permiten evolucionar de un proceso cascada de pruebas a un proceso ágil de pruebas, esto debido a que ya no se debe volver a repetir la construcción de casos de pruebas y la ejecución uno por uno de estos, sino, solo deberá ejecutarse en las llamadas pruebas de regresión; así mismo, en caso se presente alguna modificación o se agregue alguna funcionalidad en el sistema, el equipo de testing, al ya tener automatizado todos los escenarios de prueba, solo deberá realizar un mínimo esfuerzo de modificar o agregar código al que ya existe, y así poder ejecutar la automatización, logrando una mejora en el proceso de testing y dando como resultado la reducción del tiempo de entrega. En conclusión, la implementación de un sistema de automatización de pruebas permite mejorar el proceso de testing, logrando cumplir los parámetros más importantes, que son: la reducción de tiempo de entrega, la detección eficaz de incidencias o bug en el sistema, la eficacia en la elaboración de casos de prueba y la constante capacitación y crecimiento profesional del equipo de colaboradores en el área de calidad de software.

La siguiente determinación que se dió fue que el sistema de automatización de pruebas con Karate logró agilizar el proceso de testing en la empresa del rubro bancario. Para Garousi,

Küçük y Felderer (2018), la automatización de pruebas en una solución al problema de demora y cuello de botella que se genera en el proceso de testing del área de calidad de software, esto debido a que con la automatización de pruebas, se puede optimizar recursos financieros y humanos, además, de poder agilizar el proceso con los escenarios de prueba que no es necesario describirlos de forma manual, sino, de frente se puede automatizar estos escenarios, enviando los scripts directamente al código según lo requerido por el cliente, de esta manera, poder cumplir con los tiempos solicitados por las metodologías ágiles. Según J Borio y R. Paterno (2021), las pruebas de regresión son parte fundamental del ciclo de vida del desarrollo de software, más aun implementando metodologías ágiles, esto debido a que deben realizar pruebas de nuevas funcionalidades creadas o añadidas en el sistema, es por ello que la automatización de pruebas permite acoplar este proceso de testing en el área de calidad de software a cualquier metodología ágil que se pueda presentar, ya que permite cumplir con los tiempos estimados y estructura secuencial que el framework scrum propone para el desarrollo de proyectos en tecnología, además, permite adaptar la integración de nuevas herramientas y tecnologías, ya sea para el desarrollo del sistema de automatización o para la ejecución de pruebas automatizadas en los diferentes ambientes que se de en el proyecto, como por ejemplo desarrollo, uat y producción. En conclusión, al implementar el sistema de automatización de pruebas con karate, se agiliza el proceso de testing, esto debido a que se logra cumplir con la estimación de recursos, ya sea en tiempo o entregables que se solicita en las metodologías ágiles, además, la herramienta karate permite realizar una interacción comprensible entre la parte del negocio y la parte de desarrollo o técnica aplicando Gherkin con el respectivo formato de interacción “Given - When - Then”.

Se determinó que el sistema de automatización de pruebas con karate mejoró el proceso de API Testing en la empresa del rubro bancario. C. Rivera (2018), indica que las pruebas de API

se realizan de forma independiente a través de microservicios, con lo cual, se da prioridad a la funcionalidad por parte del backend, además, cuando se realiza la automatización de pruebas, se puede especificar diferentes escenarios, donde se da la funcionalidad principal del servicio; lo que permite mejorar el proceso de pruebas de API, detectar de forma más rápida las incidencias que puedan presentarse y agilizar sus entregables tanto para el cliente como para el negocio. En el caso de P. Chávez (2021), menciona que la automatización de pruebas de API permiten mejorar la detección de incidencias que se puedan presentar, es decir, reconocer a tiempo los bug detectados y de esta forma darle una pronta solución a las funcionalidades del sistema; con ello, poder entregar un sistema que logre cumplir las exigencias y expectativas que tengan los usuarios consumidores, además, de poder generar confianza tanto al equipo de desarrollo, pruebas, despliegue y negocio; así mismo, de poder agilizar y mejorar el flujo del proceso de pruebas de API. En conclusión, implementando el sistema de automatización de pruebas con karate, se logra mejorar el proceso de API testing, esto debido a que logra tener un mayor alcance en la detección de incidencias en funcionalidades principales, permite obviar la redacción de casos o escenarios de prueba de forma manual y permite la reducción de tiempos tanto para la validación de pruebas como para la entrega de los reportes.

Se determinó que, a través de la automatización con Karate Framework, el proceso de Load Testing ha mejorado, debido a su agilidad al presentar los resultados obtenidos y no presentar problemas de timeout. Según el ISTQB (2018), este proceso mejora el proceso de pruebas de carga, con lo cual, con los scripts ya elaborados, solo es cuestión de correr las pruebas en el sistema y validar que incidencias se puedan presentar. Además, según la configuración que se realice, esta prueba se orientara a cada API desarrollada, para poder determinar que no existe algún problema de demora. Por otra parte, W. Gallego (2022), hace mención a la importancia de

la pruebas de carga, con lo cual, se puede identificar el soporte que tendrá el sistema con respecto al consumo que tenga, es decir, que podrá visualizar que el sistema no tenga problemas de caída o congelamiento con respecto al servidor cuando se reciba una gran cantidad de solicitudes por parte de los usuarios, y para poder armarlo, se podrá automatizar estas pruebas construyendo diferentes scripts orientados a la funcionalidad, y así, se podrá ir mejorando la ejecución de las Load Test que el cliente solicite. En conclusión, con la implementación del sistema de automatización de pruebas con karate, se logra mejorar el proceso de Load Testing, debido a que se logra simular la cantidad de usuarios y el tiempo que toma la interacción de los mismos con respecto a las funcionalidades del sistema, como también el de poder agilizar el proceso de entrega de evidencias, a través de reportes generados con la automatización de pruebas según la herramienta que se pueda utilizar y de preferencia que sea una herramienta como Gatling o Artillery, que hoy en día son de las más utilizadas por el open source.

9. Recomendaciones

Como recomendación, se da la de capacitar de manera constante al equipo de colaboradores que trabaja en el área de calidad de software, sobre todo en programación y herramientas de automatización, esto servirá para que el equipo tenga una mayor interacción con el sistema de automatización y pueda sacar un mayor provecho, además, en caso de contratar a nuevos colaboradores, que en los perfiles se solicite de preferencia que sean tester con conocimientos en gherkin, karate, gatling, manejo básico de programación, de preferencia, en el lenguaje Java; de esta forma, se podrán familiarizar de forma rápida con el sistema de automatización de pruebas.

Además, se recomienda poder contar con un equipo multidisciplinario, el cual, pueda manejar conceptos básicos de metodologías ágiles, de preferencia, que cumplan con experiencia bajo el marco de trabajo scrum, de esta forma, podrán familiarizarse con el proceso de pruebas que se está implementando y no tendrán dificultades para adaptarse al funcionamiento del sistema, lo cual, permitirá que tengan un mejor desempeño con el proceso ágil y uso del sistema de automatización de pruebas.

Dado los resultados obtenidos en la mejora del proceso de API testing, se recomienda poder hacer uso de herramientas open source, además, de contar con un equipo enfocado a las pruebas de API y que tengan conocimientos básicos de maven y gradle, de esta forma, podrán adaptarse al sistema de automatización de pruebas y si en caso algún componente se vuelve obsoleto, con los conocimientos de dichas herramientas, podrán detectar las actualizaciones correspondientes para el sistema.

En el caso del proceso de Load Testing, se recomienda poder contar con un equipo que tenga conocimiento básicos de matrices y estándares de calidad, de esta forma, poder armar los scripts de automatización de pruebas enfocados en las pruebas de carga, así mismo, poder identificar que patrones se deben de mejorar en los sistemas, según el uso y consumo de interacción que las funcionalidades presenten por parte de los usuarios, además, que los colaboradores del área de calidad cuenten con constantes capacitaciones y conocimientos básicos en Gatling y Artillery.

10. Referencias

Gallego Durango, W. (2022). *Investigación y desarrollo de pruebas de Rendimiento automatizadas (Medellín).* Universidad de Antioquía.

Rivera Martínez, C. (2018). *Automatización de pruebas de regresión (Santiago de Chile).* Universidad de Chile.

Banda Torpoco, R. (2022). *Automatización de pruebas de calidad de código para las aplicaciones web con código abierto (Lima).* Universidad San Ignacio de Loyola.

Garousi, V. et. al (2018). *Lo que sabemos sobre los olores en el código de prueba de software (Karlskrona).* Instituto de Tecnología de Blekinge.

Fernández Vera, S. (2018). *Herramientas de automatización para pruebas de software (Valparaíso)*. Universidad Técnica Federico Santa María.

Chávez Lazarte, P. (2021). *Automatización de casos de pruebas usando Selenium y TestNG Framework para mejorar el proceso de pruebas de un sistema que brinda servicio RPO, 2020 (Lima).* Universidad Nacional Mayor de San Marcos.

Diaz Asencio, E. (2021). *Automatización de pruebas de regresión (Sevilla)*. Escuela Técnica Superior de Ingeniería de la Universidad de Sevilla.

Kaplan, G. et. al (2018). *Planificación de las pruebas de software (Buenos Aires).* Departamento de Ciencias Básicas de la Universidad Nacional de Luján.

Cabrera Serna, J. y Pareja Verastegui, C. (2021). *Automatización de pruebas funcionales web para mejorar el área de calidad de software en una empresa del rubro de retails en el año 2021 (Lima).* Universidad Tecnológica del Perú.

Alégroth, E. et. al (2016). *Mantenimiento de suites de pruebas automatizadas en la industria: Un estudio empírico sobre pruebas de GUI visual (Gotemburgo).* Universidad Tecnológica de Chalmers

Anexos

Anexo 1. Código de Ética de Investigación UCSS

CÓDIGO DE ÉTICA PARA LA INVESTIGACIÓN

Universidad Católica Sedes Sapientiae

Aprobado el 07 de diciembre de 2016 – Mediante Resolución N° 065-2016-UCSS-AG/GC

Modificado el 29 de mayo de 2018 – Mediante Resolución Nº 063-2018-UCSS-AG/GC

Modificado el 20 de junio ed 2018 – Mediante Resolución Nº 087-2018-UCSS-AG/GC

CAPÍTULO I: DISPOSICIONES GENERALES

El presente código tiene por finalidad establecer normas éticas que garanticen conducta responsable en la investigación, en la Universidad Católica Sedes Sapientiae (UCSS), que se canaliza a través de la Dirección General de Investigación (DGI).

Base legal:

a. Constitución Política del Perú.

b. La Ley Universitaria, Ley N°30220, Artículos: 3º; 6º, 6.5; 7º, 7.2; 48º; 49º; 50º; 53º; 63º, 86º.

c. Reglamento de Calificación y Registro de Investigadores en Ciencia y Tecnología del Sistema Nacional de Ciencia, Tecnología e Innovación Tecnológica – SINACYT, con Resolución de Presidencia nº 198-2017-CONCYTEC-P.

d. Decreto Ley Nº 23211.

e. El Estatuto de la UCSS, artículos. 4º; 7º; 31º; 39°; 41º; 62º; 83º; 84º; 85°; 86°; 87°; 88°; 89°; 90°.

f. Reglamento de organización y funciones de la Dirección General de Investigación aprobado con Resolución Nro. 024-2017-UCSS-AG/CG.

CAPITULO II: INDICACIONES Y PRINCIPIOS ÉTICOS

Indicaciones generales de valor ético:

a. Adherir a una metodología de la investigación caracterizada por el rigor científico y ofrecer resultados de elevada calidad científica.

b. Reconocer que la ciencia y la tecnología deben estar al servicio de la persona humana en el respeto pleno de su dignidad y de sus derechos.

c. Reconocer que, a cada ser humano, desde el primer momento de su existencia y hasta su muerte, va garantizado el respeto pleno e incondicionado que se le debe a razón de su peculiar dignidad y que existe un modo de vivir de acuerdo con esa trascendente dignidad inalienable de cada ser humano, siempre al servicio del bien común de la sociedad.

d. Reconocer la vocación del ser humano.

Principios éticos para la investigación

En el desarrollo de las actividades de investigación se debe tener como premisa básica la ética. Por ello, se presentan cinco principios éticos de la investigación que se aplican a todas las áreas en las que la Universidad hace investigación.

1. Respeto de la confidencialidad y política de protección de datos

a) La aplicación y administración de los cuestionarios se realizará respetando la confidencialidad. Los datos personales de los encuestados serán protegidos y no divulgados.

b) Indicación y explicación a los participantes de la finalidad y del uso que se le dará a la información resultante.

c) Los datos de los participantes serán accesibles solo a un restringido grupo identificable de personas: investigador-tesista en el caso de trabajos de investigación curriculares de pregrado y postgrado, o por el equipo de trabajo compuesto por el investigador principal (Coordinador científico) y los investigadores colaboradores en trabajos de investigación extracurriculares.

d) Los datos generados a partir de la fase de obtención de la información y el libro de variables serán, a su vez, guardados en la computadora del investigador-tesista o del coordinador científico utilizando una contraseña personal.

e) Los materiales de la investigación no digitales serán conservados y protegidos en un lugar adecuado por un periodo de siete años.

2. Respeto de la privacidad

a) Respeto del derecho de los sujetos, que participan en el estudio, a elegir el tiempo, las circunstancias y la cantidad de información a compartir con los investigadores.

b) Respeto del derecho de los sujetos, que participan en el estudio, a no dar información que no desean compartir.

c) Se garantizará, en la medida que lo permitan las circunstancias, la administración de los cuestionarios en locales que garanticen el respeto de los dos puntos anteriores.

3. No discriminación y libre participación

a) No existirán formas de discriminación en el grupo de sujetos que participen al estudio en cuanto a género, grupo étnico o por condición social, en sintonía con los criterios metodológicos de exclusión e inclusión al estudio.

b) No existirán formas de inducción coercitiva de participación al estudio.

4. Consentimiento informado a la participación en la investigación

a) Se ofrecerá información relevante a los sujetos sobre la finalidad y las características del proyecto de investigación para solicitar el consentimiento informado a la participación al estudio.

b) Se ofrecerá información clara y apropiada a los sujetos involucrados sobre los riesgos y beneficios relativos a la participación al estudio.

c) Se garantizará la obtención del consentimiento informado de los sujetos antes de participar al estudio.

d) Se informará a los futuros encuestados de la posibilidad de abandonar el estudio y de la ausencia de consecuencias derivadas de esta decisión.

5. Respeto por la calidad de la investigación, autoría y uso de los resultados

a) Promoción del valor científico de la investigación representado por la importancia clínica y/o social del estudio.

b) Búsqueda de la validez científica del estudio representado por la creación de un marco teórico suficiente que se basa en documentación científica válida y actualizada, el uso coherente del método de investigación con el problema que se desea dar respuesta, la selección adecuada de la muestra de los sujetos que serán involucrados, una codificación y análisis de los datos que garanticen elevados estándares de calidad y una interpretación crítica de los mismos, uso de un lenguaje adecuado en la comunicación de los resultados de la investigación.

c) Disponibilidad del material físico y de la base de datos elaborada para la revisión del proceso de recolección de información, por parte de la autoridad competente.

d) Se reconoce a los tesistas del equipo de trabajo el derecho de autoría de los productos del estudio, de acuerdo a las normas nacionales e internacionales que regulan la filiación institucional.

e) Queda prohibida la comercialización, negociación y la divulgación indiscriminada del contenido parcial o total de la tesis y de los potenciales resultados futuros por parte del equipo de investigación o de terceros ajenos al mismo.

f) Los miembros del equipo deberán declarar la ausencia de conflicto de intereses en la realización del estudio.

La aceptación, conjuntamente a la correcta puesta en práctica de estos principios y criterios, es normativa y determina el ingreso y la permanencia de los investigadores en el equipo de estudio.

CAPÍTULO III: CONDUCTAS NO ÉTICAS EN LA INVESTIGACIÓN

1. Plagio

a) El plagio es el acto de apropiarse indebidamente de las ideas de otro para presentarlas como propias. Se considera una práctica profundamente contraria a la moral en cualquier ámbito y absolutamente inaceptable en la investigación ya que con ella se retrasa el progreso.

b) Las ideas pueden haber sido presentadas por el autor propietario de las mismas bajo diversas modalidades tales como textos, tablas, figuras, imágenes, videos, partituras, maquetas, prototipos, entre otros. En ese sentido, no pueden ser tomadas por otro, bajo ninguna modalidad, sin el cuidado de citar adecuadamente la fuente original.

c) La Universidad Católica Sedes Sapientiae considera el plagio como una falta grave la cual está sujeta al procedimiento especial de sanción correspondiente pudiendo ser sujeto a suspensión

definitiva de la institución. De manera excepcional, será considerada como una falta mayor sujeta a amonestación escrita y a la posibilidad de suspensión temporal de la institución.

d) Otras formas de plagio son:

- El autoplagio el cual consiste en no citar la fuente original en donde se publicó anteriormente una idea propia;
- Parafraseo libre de una frase ajena, aun cuando sea menor, sin citar la fuente de inspiración original.
- Citación indirecta, que consiste en tomar o parafrasear un texto citado por un autor sin indicar que la cita no se tomó de modo directo, sino que fue tomada del autor que la citó.

e) Las formas más sutiles de plagio señaladas en el punto anterior, así como el posible descuido en la referencia correcta o completa de una fuente original pueden ser consideradas faltas menores cuando se detectan dentro del proceso de formación de un investigador, sobre todo cuando se trata de un estudiante a quien aún se está entrenando en el adecuado manejo de la información.

f) No obstante, toda posible falta cercana al plagio debe ser detectada y corregida oportunamente por los responsables directos de una investigación y, dependiendo de su tenor - sobre todo si no fue corregida a tiempo o fue detectada posteriormente a su publicación- deberá ser presentada al Comité de Ética para la Investigación de la UCSS para que determine las enmiendas y sanciones correspondientes.

2. Falseamiento de la información y documentación para docentes investigadores o docentes que investigan

a) La Universidad Católica Sedes Sapientiae considera c o m o una falta mayor

la presentación de información y/o documentación no veraz para ser registrados en REGINA. Cuando esta situación se verifique se aplicará lo dispuesto en el art 12 del Reglamento de Calificación y Registro de Investigadores en Ciencia y Tecnología del Sistema Nacional de Ciencia, Tecnología e Innovación Tecnológica – SINACYT.

b) Igualmente, considera c o m o una falta mayor la presentación de información y/o documentación para ser nombrados o renovados como "docente investigador o docente que investiga" en la Universidad.

3. Incumplimiento de contratos y/o convenios

a) La gravedad de la falta respecto al incumplimiento de contratos y/o convenios que generan obligaciones y responsabilidades al docente que realiza investigación en la Universidad será evaluada por el Comité de Ética para la Investigación que decidirá la sanción respectiva a partir de la documentación y argumentación presentada.

4. Conflictos de interés en la investigación

a) Los docentes investigadores y los docentes que investigan tienen un conflicto de interés cuando sus intereses o compromisos pueden afectar sus juicios, sus informes de investigación o sus comunicaciones a sujetos de investigación, participantes, pacientes o clientes.

b) El investigador, en caso se pudiera presentar un conflicto de intereses, lo debe reportar a la Dirección General de Investigación y posteriormente, evaluar con el Comité de Ética para la Investigación si debe retirarse del proceso que genera conflicto.

c) La Universidad Católica Sedes Sapientiae considera como falta mayor cuando como resultado del conflicto de interés se introducen sesgos o mina la validez científica de los resultados del estudio.

d) La Universidad Católica Sedes Sapientiae considera como falta grave cuando, como resultado del conflicto de interés, se produce daño a la salud o la integridad de sujetos participantes en el estudio.

e) La Universidad Católica Sedes Sapientiae considera como falta grave cuando, como resultado del conflicto de interés, se producen daños económicos para la Universidad y/o la entidad que provee financiamiento al estudio.

5. Fabricación y falsificación de resultados de investigación

a) La Universidad Católica Sedes Sapientiae considera como falta mayor cuando los docentes investigadores y los docentes que investigan fabrican y/o falsifican resultados de investigación científica, inventan experimentos o datos, sustituyen u omiten datos negativos, incluyendo efectos secundarios, manipulan o distorsionan malintencionadamente imágenes (gráficos, fotografías, micrografías, radiologías, etc.) y cuando describen metodologías falsas.

6. Falsa autoría y acreditación en las publicaciones

a) La Universidad Católica Sedes Sapientiae considera como falta mayor cuando los docentes investigadores y los docentes que investigan incurren en duplicación de artículos, autoría fantasma u honoraria o excluyen a personas que deberían ser acreditados como coautores.

7. Mal uso de animales y plantas en investigaciones

a) La Universidad Católica Sedes Sapientiae considera como falta menor cuando el investigador utilice un animal que no se adapte a su investigación no teniendo en cuenta los grados sensoriales de la especie.

b) La Universidad Católica Sedes Sapientiae considera como falta menor cuando el investigador no vele para que las condiciones de mantenimiento del animal sean las mejores para aportarle los cuidados necesarios, antes, durante y después de la investigación.

c) La Universidad Católica Sedes Sapientiae considera como falta mayor cuando el investigador no evite al animal de experimentación todo sufrimiento físico inútil o maltrato. Igualmente, cuando no se dispusieran de los métodos adecuados para reducir el riesgo, el dolor, el estrés y la angustia del animal.

d) La Universidad Católica Sedes Sapientiae considera como falta grave cuando el investigador utilice animales y/o plantas en peligro de extinción en su investigación en circunstancias no excepcionales y/o no definidas claramente.

e) La Universidad Católica Sedes Sapientiae considera como falta grave cuando el investigador realice acciones que pongan en serio riesgo la biodiversidad vegetal en sus investigaciones, sin haber evaluado los posibles efectos de la misma.

f) La Universidad Católica Sedes Sapientiae considera como falta grave cuando el investigador emplee en sus investigaciones muestras de especies vegetales o animales provenientes del comercio ilegal o tráfico de especies protegidas.

CAPÍTULO IV: COMITÉ DE ÉTICA PARA LA INVESTIGACIÓN

a) Para una adecuada orientación y vigilancia del sentido ético en la práctica de la investigación, la Universidad Católica Sedes Sapientiae cuenta con un Comité de Ética para la Investigación, el cual se encuentra adscrito a la Dirección General de Investigación de la Universidad Católica Sedes Sapientiae.

b) El Comité está conformado por el Jefe de la Dirección General de Investigación y por docentes representantes de cada una de las Facultades.

c) El objetivo del Comité es velar por el cumplimiento de los principios y las normas del presente Código en todas las prácticas de investigación científica que se realizan en la universidad.

d) El Comité de Ética para la Investigación tiene como política central difundir la importancia y el significado de la ética en la práctica de la investigación científica mediante conferencias, seminarios, talleres, informes, comunicados y cualquier otro medio que sea adecuado para la comunidad académica de la institución.

e) El Comité de Ética para la Investigación tiene como responsabilidad no sólo difundir el Código de Ética para la Investigación de la institución, sino velar para que su relevancia y sentido sean adecuadamente comprendidos y asumidos por toda la comunidad académica.

f) Como procedimiento general, toda práctica contraria a cualquiera de los principios o normas señalados en el presente Código debe ser presentada al Comité de Ética para la Investigación, ya sea por el autor o por quien haya detectado tal práctica inadecuada.

g) Son funciones generales del Comité de Ética para la Investigación:

1. Revisar permanentemente el Código de Ética para la Investigación en la UCSS y proponer las modificaciones que puedan ser pertinentes y aclarar cualquier duda respecto a su interpretación.
2. Revisar y evaluar los casos en los que se genere algún conflicto ético en la labor investigativa de la UCSS.

3. Promover y asegurar el cumplimiento de las buenas prácticas investigativas en la Universidad.

4. Promover en la Comunidad Académica la reflexión sobre los temas relacionadas a la ética de la Investigación.

5. Elaborar un informe anual sobre la labor realizada.

CAPÍTULO V: SANCIONES

Las prácticas contrarias a los principios o normas del presente Código serán sancionadas de la siguiente manera por el Comité de Ética para la Investigación:

1.- Falta menor: el responsable de la falta será advertido o amonestado verbalmente con el objetivo formativo de que corrija inmediatamente la incorrección y no vuelva a cometerla en el futuro.

2.- Falta mayor: el responsable de la falta será amonestado por escrito, se comunicará formalmente a la autoridad máxima de su Escuela, Facultad o Área y podrá ser sujeto de una suspensión temporal de la institución.

3.- Falta grave: el responsable de la falta será sometido a un procedimiento especial cuyo expediente será elevado al Vicerrectorado Académico y a la Dirección General de Investigación para que se determine su posible suspensión definitiva de la institución.

Disposición Final

El presente Código de Ética para la Investigación, cuyo contenido es de obligatorio cumplimiento en la Comunidad Universitaria, entrará en vigencia a partir de su aprobación por la Asamblea General.

RESOLUCIÓN N° 087-2018-UCSS-AG/GC

Los Olivos, 20 de junio de 2018

El Obispo de la Diócesis de Carabayllo Monseñor Lino Mario Panizza Richero, Gran Canciller de la Universidad Católica Sedes Sapientiae

VISTO

Que la Universidad Católica Sedes Sapientiae, es una institución perteneciente a la Iglesia Católica; y, el D.S. 001-97-JUS el cual nombra como Obispo de la Diócesis de Carabayllo a Monseñor Lino Mario Panizza Richero, identificado con DNI N° 40010019; y,

CONSIDERANDO

Que, mediante Resolución N° 065-2016-UCSS-AG/GC de fecha 07 de diciembre de 2016 se ratificó y aprobó el Código de Ética para la Investigación de la Universidad Católica Sedes Sapientiae

Que, mediante Resolución N° 063-2018-UCSS-AG/GC de fecha 29 de mayo de 2018 se modificó el Código de Ética para la Investigación de la Universidad Católica Sedes Sapientiae

Que, con fecha 20 de junio de 2018 se ha llevado a cabo la sesión extraordinaria de Asamblea General de la Universidad Católica Sedes Sapientiae, en la cual el Pleno se pronunció sobre la necesidad de modificar el Código de Ética para la Investigación de la Universidad Católica Sedes Sapientiae.

Que, luego de una breve deliberación la Asamblea General de la Universidad Católica Sedes Sapientiae ha aprobado por unanimidad la propuesta de modificación del Código de Ética para la Investigación de la Universidad Católica Sedes Sapientiae.

De conformidad con lo establecido en el Estatuto de la Universidad Católica Sedes Sapientiae y demás disposiciones legales vigentes.

SE RESUELVE:

Artículo Único: Modificar el Código de Ética para la Investigación de la Universidad Católica Sedes Sapientiae; según el texto que se adjunta y que forma parte de la presente resolución.

Regístrese comuníquese y archívese

Monseñor Lino Mario Panizza Richero
Gran Canciller UCSS

Anexo 2. Estrategia Pruebas Automatizadas API

Estrategia de Pruebas Automatizadas "API" proyecto bancario

HISTORIAL DE REVISIONES

Versión	Autor	Descripción	Fecha	Aprobado Por
1.0	Gherard Chipana	Creación del documento	10/01/2023	Project Manager

TABLA DE CONTENIDO

1. INTRODUCCIÓN

En esta estrategia para la realización de pruebas automatizadas se describe el alcance de las pruebas, los roles y responsabilidades, los riesgos y planes de contingencia, el ambiente de pruebas, los recursos necesarios, las herramientas a utilizar, los criterios de pruebas y los reportes.

2. ALCANCE

Se realizarán pruebas de caja negra automatizadas a las funcionalidades que serán seleccionadas después de realizar las pruebas manuales. Para realizar este proceso de pruebas automatizadas, se deberá seguir las siguientes recomendaciones:

Funcionalidades con tareas repetitivas. Funcionalidades en áreas de riesgo de la aplicación. Funcionalidades que conecten con un gran conjunto de datos. Funcionalidades con rutas críticas del sistema. Funcionalidades con prioridad para el negocio. Funcionalidades con alto grado de error en pruebas manuales. Funcionalidades que requieran ser ejecutadas en diferentes dispositivos, sistemas operativos, ambientes y navegadores.	Proyecto con insuficiente tiempo. Proyecto con insuficientes recursos. Funcionalidades inestables. Funcionalidades sin resultados predecibles, es decir, que no sean detectadas por el robot automatizador; por ejemplo: el capcha, QR, botones random, etc.

3. ROLES Y RESPONSABILIDADES

Roles	Responsabilidades
QA Lead	- Planificación y Monitoreo de las pruebas automatizadas - Reporte de progreso de las pruebas
QA Engineer QA Automation Engineer	- Diseño e Implementación de las pruebas. - Ejecución de las pruebas automatizadas. - Reporte de defectos y resultados de las pruebas.

Product Owner / Stakeholders	-	Toma de decisiones

4. RIESGOS Y PLANES DE CONTINGENCIA

N°	Riesgos	Probabilidad de Ocurrencia (1-5)	Impacto (1-5)	Severidad (Prob*Impacto)	Plan de Contingencia
1	Solicitud de cambios en aquellas funcionalidades que tienen pruebas automatizadas. Esto ocasiona un mayor tiempo de trabajo debido a que se deben actualizar los scripts	3	3	9	Estimar tiempo de cambio y priorizar las funcionalidades que se automatizarán
2	Funcionalidades no terminadas en tiempo estimado no puede ser parte de las funcionalidades planificadas para automatizar.	2	5	10	Re planificar las funcionalidades para ser automatizadas.

5. AMBIENTE Y HERRAMIENTAS DE PRUEBAS

a. Ambiente de Pruebas

UAT	Es el ambiente donde los Tester hacen sus pruebas, después que el equipo de desarrollo haya desplegado sus cambios correspondientes.	https://reqres.in

b. Herramientas de Pruebas

Oracle 19c	Base de Datos de Salida.
Java 1.8	Lenguaje de Programación

Karate DSL	Framework de automatización de pruebas de API.
Gradle 4.6	Sistema de automatización de construcción de código de software
Gherkin	Lenguaje específico de dominio (DSL)
Google Chrome	Navegador de Pruebas
Git	Sistema de Control de Versiones
Bitbucket / Github	Repositorio de Código
Jenkins	Servidor open source para integración continua.
Windows	Sistema Operativo donde se realizará las pruebas.

6. CRITERIOS DE ENTRADA Y SALIDA

Criterios de Entrada

- Las funcionalidades deben estar desplegadas en el ambiente de UAT
- Las funcionalidades deben haber sido probadas de forma manual
- El framework de pruebas debe estar instalado y listo para su ejecución
- El ambiente de UAT debe estar disponible
- Los defectos críticos encontrados en las pruebas manuales deben haber sido resueltos y cerrados.

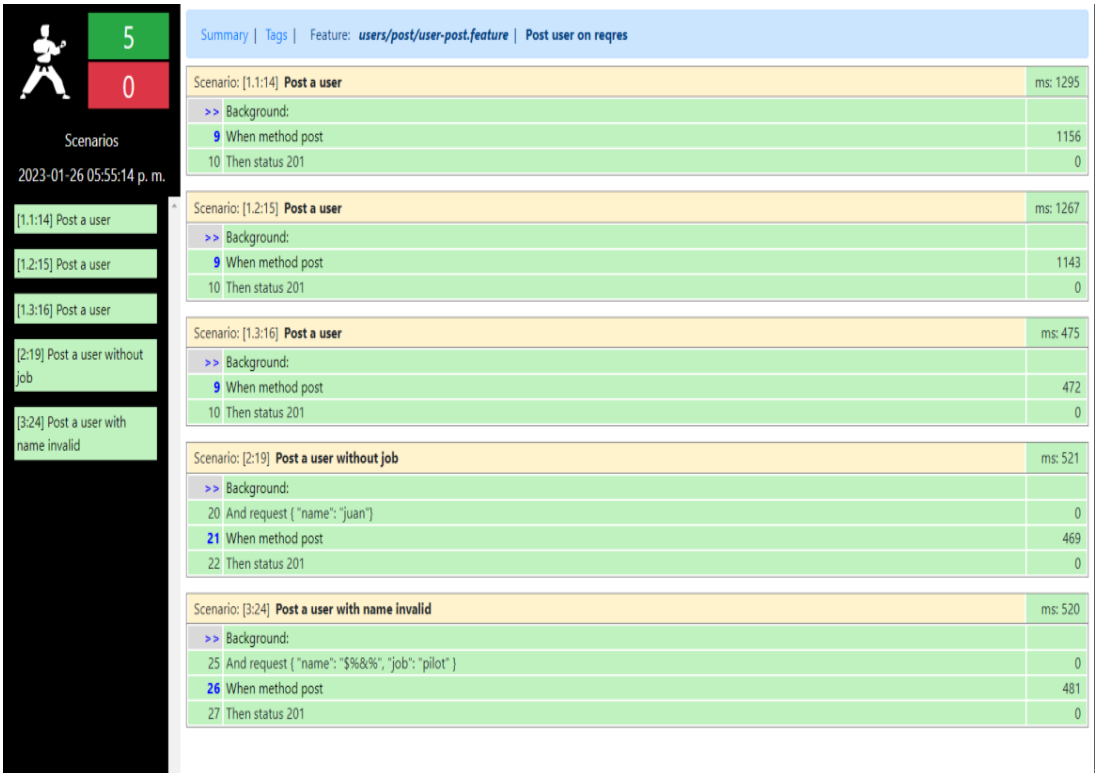
Criterios de Salida

- Ejecución de todos los casos de pruebas automatizadas.
- Se recibe la aprobación de los Stakeholders.
- No debe haber defectos de severidad alta.

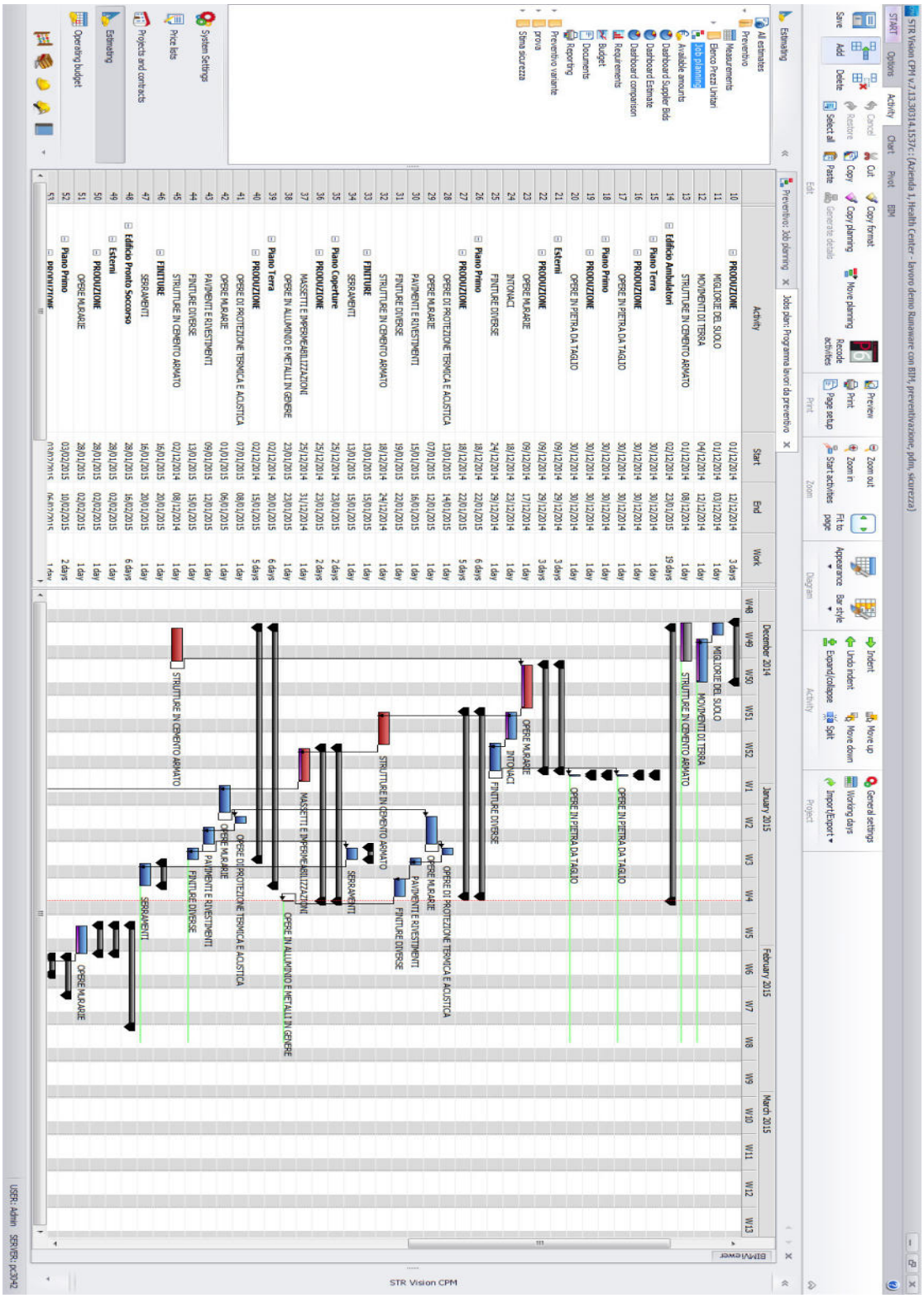
7. REPORTE DE PRUEBAS

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

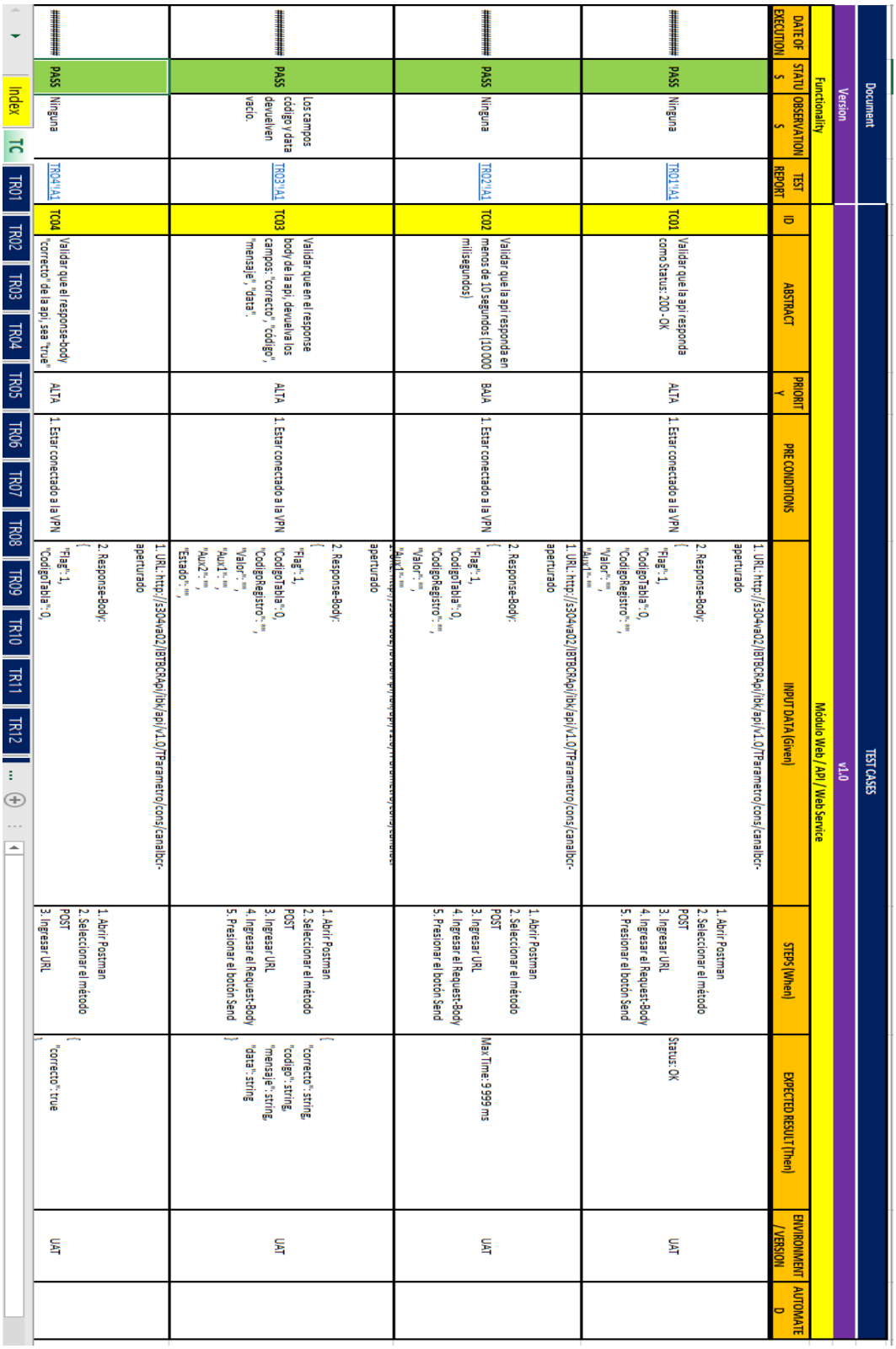
Ejemplo:



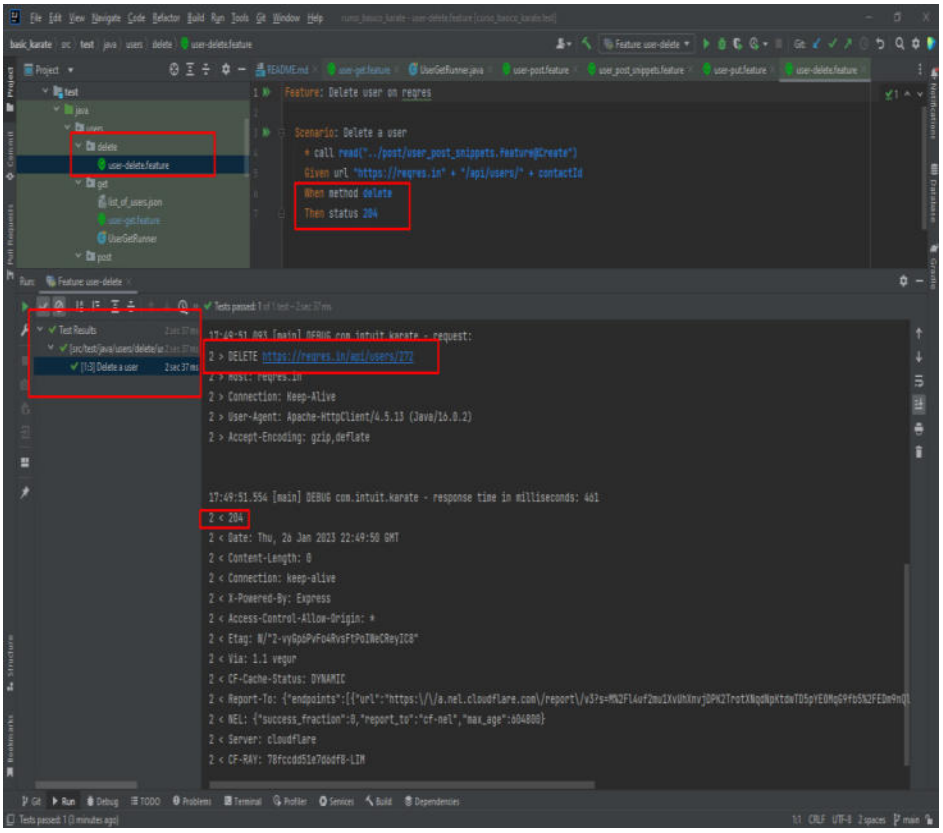
Anexo 3. Ejemplo de Diagrama de Gantt



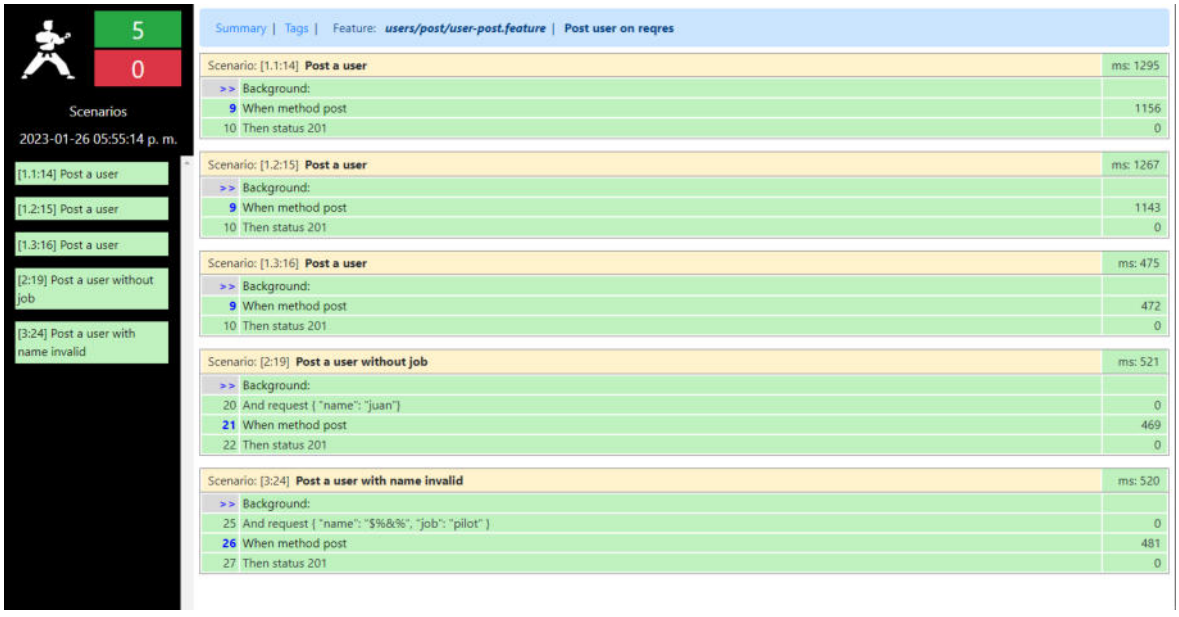
Anexo 4. Casos de Prueba



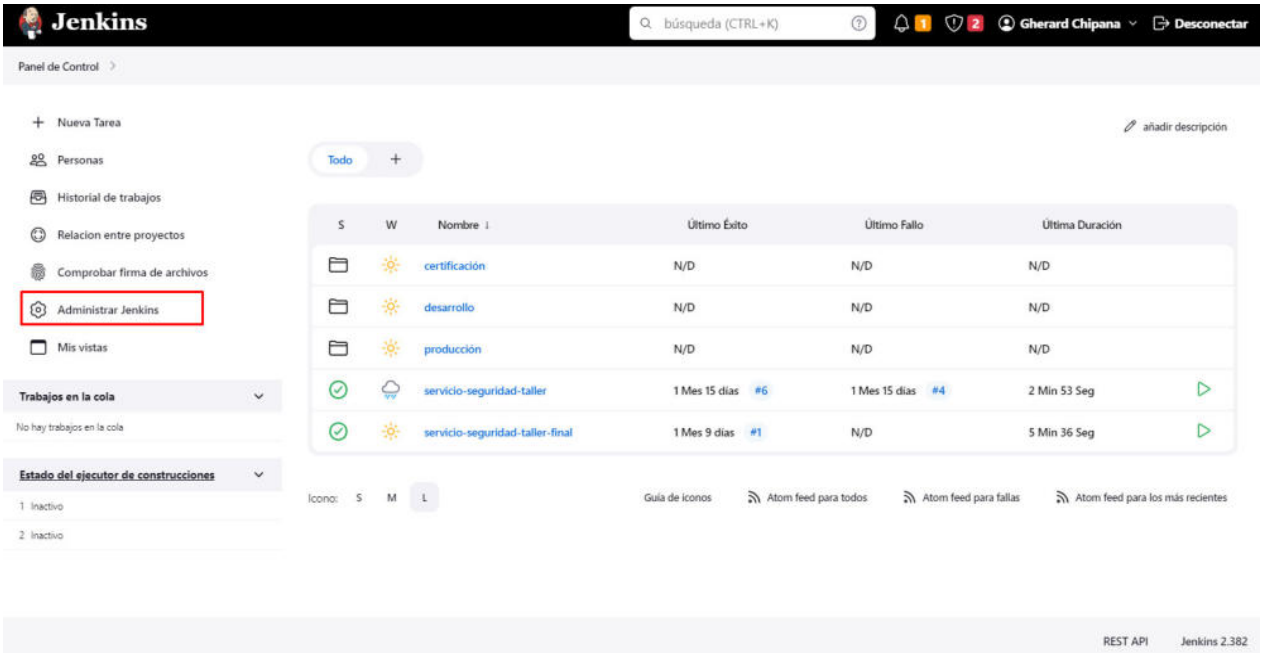
Anexo 5. Ejecución de Karate



Anexo 6. Reporte de Pruebas Local



Anexo 7. Plataforma Jenkins



Anexo 8. Reporte de Pruebas de Karate en Jenkins

Anexo 9. Código Management

```java
package users;

import com.intuit.karate.KarateOptions;
import com.intuit.karate.Results;
import com.intuit.karate.Runner;
import net.masterthought.cucumber.Configuration;
import net.masterthought.cucumber.ReportBuilder;
import org.apache.commons.io.FileUtils;
import org.junit.jupiter.api.Test;

import java.io.File;
import java.util.ArrayList;
import java.util.Collection;
import java.util.List;

2 usages  gherardtest
@KarateOptions(tags = {"~@ignore"})
public class ManagementTest {
    no usages  gherardtest
    @Test
    public void testParallel() {
        System.out.println("getClass:" + getClass());
        Results results = Runner.parallel(getClass(),  threadCount: 1, reportDir: "target/surefire-reports");
        generateReport(results.getReportDir());
    }
    1 usage  gherardtest
    public static void generateReport(String karateOutputPath) {
        Collection<File> jsonFiles = FileUtils.listFiles(new File(karateOutputPath), new String[] {"json"},  recursive: true);
        List<String> jsonPaths = new ArrayList(jsonFiles.size());
        jsonFiles.forEach(file -> jsonPaths.add(file.getAbsolutePath()));
        Configuration config = new Configuration(new File( pathname: "target"),  projectName: "Api-Karate-UCSS");
        ReportBuilder reportBuilder = new ReportBuilder(jsonPaths, config);
        reportBuilder.generateReports();
    }
}
```