Tecnológico Nacional de México

Instituto Tecnológico de Zacatecas

Tesis de Maestría

IMPLEMENTACIÓN DE PRUEBAS BASADAS EN ISTQB PARA EL ANÁLISIS Y COMPARACIÓN DE FRAMEWORKS BACKEND EN APLICACIONES WEB

Presentada por

Berenice Ortiz Torres

como requisito para obtención del grado de

Maestra en Sistemas Computacionales

Director de tesis

Dr. Jaime Iván López Veyna

Codirector de tesis

MSC. Luis Enrique Márquez Martínez

Zacatecas, Zacatecas, México Octubre 2024

Dependencia: III
Sección: III-9
Expediente: CI/División de Estudios de Posgrado e Investigación
No. De Oficio: 0144

Zacatecas, Zac., 08 de octubre de 2024

Asunto: Autorización de impresión

C. Berenice Ortiz Torres
Egresada de la Maestría en Sistemas Computacionales
Presente

Por este conducto me permito comunicar a usted, que tiene autorización para llevar a cabo la impresión de su Trabajo Profesional denominado:

"IMPLEMENTACIÓN DE PRUEBAS BASADAS EN ISTQB PARA EL ANÁLISIS Y COMPARACIÓN DE FRAMEWORKS BACKEND EN APLICACIONES WEB."

Que de acuerdo con la opción I (**TESIS**) presenta para poder obtener el grado de **MAESTRA EN SISTEMAS COMPUTACIONALES**

Se le informa que deberá entregar a este departamento 2 (dos) copias impresas y 5 (cinco) digitalizadas de su trabajo profesional, si contiene fotografías deberán de ser a color y en original. Así mismo integrar toda la información digitalizada, ordenada y completa.

A T E N T A M E N T E
"Ciencia y Tecnología Binomio De Progreso Universal"

Ma. De Lourdes Balderas Castañeda
Jefa de la División de Estudios de Posgrado e Investigación

Vo.Bo.
Roberto Ortiz Delgadillo
Director

EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA
TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE ZACATECAS
DIRECCIÓN

AGRADECIMIENTOS

Al concluir esta tesis de posgrado, anhelo expresar mis más sinceros agradecimientos, aquellas personas que estuvieron en toda la trayectoria de este logro, en mi vida.

Primero, quiero agradecer a Dios, El Señor De La Ascensión, mi consuelo y mi mejor aliado, en quien dejo a su voluntad mis acciones. Le agradezco por la salud que me otorga, por mi familia y mi trabajo. Por todas las bendiciones que me obsequia.

A mi madre, Anita Torre Gonzales, por ser el ejemplo inquebrantable de humildad y honradez. A ella le doy mi agradecimiento eterno por todo su apoyo y su sacrificio, para apoyarme en mi formación académica. Gracias por enseñarme a trabajar para alcanzar mis metas y proyectos. A mi madre le dedico todos mis logros y muy en especial, esta tesis. A mi padre Felipe Ortiz Lumbreras, lo honro por hacer lo mejor que pudo, como padre.

A mis hermanas, Cecilia Ortiz Torres y Emmy Loou Ortiz Torres, a ellas les debo mi gratitud por ser mis cómplices. La confianza que han puesto en mí y su apoyo incondicional en los momentos difíciles, ha sido un sustento en mi vida.

A mis sobrinos Ángel Guadalupe Pérez Ortiz, Erwin Adriel Pérez Ortiz, Tadeo Emiliano Cardona Ortiz, a quienes les agradezco por sus ideas tan elocuentes, sus sonrisas, sus juegos. su forma de ver la vida, que ha transformado mi persona y me ayudado a ser mejor de lo que antes fui.

A todos mis profesores y compañeros, que me han dado su apoyo y me alentaron a seguir en mis estudios, les agradezco por su guía, sus enseñanzas, su buena voluntad así mi persona, su dedicación y todo su esfuerzo para ayudar a cultivarme como perfeccionar.

Con gran respeto y humildad, Berenice Ortiz Torres.

Índice

Índice de Figuras

Índice de Gráficas

Índice de Tablas

RESUMEN

Esta investigación muestra un análisis comparativo de frameworks backend Spring Boot, Django y Laravel, conforme a la implementación de pruebas de software, basadas en la organización de ISTQB *(International Software Testing Qualifications Board*), con el propósito de determinar cuál de ellos ofrece una alternativa adecuada para la construcción de una aplicación *web*.

Para este propósito, se implementó un contexto experimental que se desempeñó en la construcción de una aplicación *web*, conformada principalmente por APIs REST (*Representational State Transfer*) desarrolladas en Spring Boot, Django y Laravel. El análisis comparativo empleó los programas de estudio de la organización ISTQB conforme al proceso de prueba, tipos de pruebas y técnicas de prueba para la elección de pruebas no funcionales; con ello se obtuvieron las pruebas de rendimiento (carga, comportamiento temporal, utilización de recursos y estrés) y fiabilidad (capacidad de recuperación). Como instrumento para la recolección de datos y métricas entre las pruebas de software aplicadas en los frameworks backend se utilizó la herramienta Apache JMeter.

Los resultados en la ejecución de las pruebas indicaron que el framework backend Django obtuvo mejores desempeños tanto en las pruebas de rendimiento como de fiabilidad. El framework backend Spring Boot mostró una ventaja en los resultados en las pruebas de estrés por lo que se posicionó como framework intermedio. El framework backend Laravel tuvo un rendimiento un poco más bajo, sin embargo, en las pruebas de utilización de recursos y capacidad de recuperación mantuvo puntuaciones óptimas.

ABSTRACT

This research shows a comparative analysis of Spring Boot, Django and Laravel backend frameworks, according to the implementation of software testing, based on the organization of ISTQB (International Software Testing Qualifications Board), with the purpose of determining which of them offers an alternative. suitable for building a web application.

For this purpose, an experimental context was implemented that was used to build a web application, made up mainly of REST APIs (Representational State Transfer) developed in Spring Boot, Django and Laravel. The comparative analysis used the study programs of the ISTQB organization according to the testing process, types of tests and testing techniques for the selection of non-functional tests; With this, performance tests (load, temporal behavior, resource use and stress) and reliability (recovery capacity) were obtained. As an instrument for collecting data and metrics between the software tests applied in the backend frameworks, the Apache JMeter tool was used.

The results of the test execution indicated that the Django backend framework obtained better performances in both performance and reliability tests. The Spring Boot backend framework showed an advantage in the results in the stress tests, which is why it was positioned as an intermediate framework. The Laravel backend framework performed a little lower, however, in the resource utilization and resilience tests it maintained optimal scores.

INTRODUCCIÓN

El uso de las aplicaciones *web* se ha convertido en una parte integral e indispensable para las actividades cotidianas de multitudes. Hoy en día, una gran cantidad de usuarios ha utilizado por lo menos un software para comprar algún producto en línea, enviar un correo, realizar una transacción bancaria, hacer algún trabajo, entre otros servicios que se podrían nombrar. Estas aplicaciones *web* son desarrolladas por programadores que se encargan del buen funcionamiento de muchos aspectos técnicos y lógicos que los usuarios ajenos a la construcción de un software, no perciben.

Una aplicación o software *web* está conformada por varios factores en los que se encuentran las capas frontend y backend. El frontend es la capa superficial de una aplicación, su función es interactuar directamente con los usuarios; esta puede incluir desde botones, imágenes, menús, entre otros componentes distintos. Mientras que el backend es la parte del lado del servidor que implementa respuestas, accediendo a la información por medio de la aplicación para luego responder a las peticiones de los clientes. Debido a lo cual constituye el motor y parte del desarrollo lógico, que ejecuta funciones y procesos para que un sistema *web* funcione de forma correcta, por lo que se considera una de las capas más cruciales.

Dado que existen frameworks backend que facilitan el cumplimiento de este criterio, en esta investigación se hace el uso de estos, puesto que cuentan con múltiples programas, herramientas, librerías, lenguajes de programación y funcionalidades de alto rendimiento que se encuentran a disposición de quienes laboran en la construcción de un software.

Es relevante comprender y definir las necesidades de un software *web* a implementar; una vez teniendo claro esto, se pasa a un punto crucial, ¿Cuál framework backend es el más indicado para solucionar y satisfacer cada una de

las necesidades de un software *web*? Por esta razón, para esta investigación, es de gran utilidad el probar y evaluar frameworks backend. Las pruebas de software sirven para mostrar la presencia de defectos. No obstante, pueden ser empleadas para la recolección y obtención de datos, en relación a las peculiaridades y propiedades que estos proporcionan. Por este motivo se considera productivo y benéfico la implementación de pruebas basadas en ISTQB para el análisis y comparación de frameworks backend en aplicaciones *web*.

En el desarrollo de esta investigación se utilizaron los datos de los sitios *web*: ***Statista*** [1, 2]**,** ***Stack OverFlow*** [3]**,** ***Acropolim*** [4]**,** ***Radix*** [5] **y** ***Wearedevelopers*** [6] para seleccionar los frameworks backend con mayor tendencia durante el presente año. Después se implementó una aplicación *web* como entorno de experimentación y se establecieron pruebas de software para la evaluación de los mismos, dichas pruebas se categorizaron en: rendimiento (carga, el comportamiento temporal, la utilización de recurso y el estrés) y fiabilidad (capacidad de recuperación). Estas pruebas son descritas por la ISTQB y conforme a estas, se diseñaron los planes, escenarios, procedimientos y estructura de los casos de prueba; así como reportes de pruebas que se utilizaron. Reflejando en ellos los resultados obtenidos, para su análisis y comparación.

A continuación, se describe de manera general la estructura del presente documento: Capítulo 1: Se especifica el ámbito y la problemática de esta investigación, se definen los objetivos y se describe la justificación por la cual se realizó este trabajo de investigación considerando la importancia y utilidad que la misma. Capítulo 2: Se describe la información teórica requerida. Capítulo 3: Muestran los trabajos relacionados con el problema de investigación. Capítulo 4: Se explica el marco metodológico que será utilizado para la ejecución de la investigación. Capítulo 5: Se detalla la implementación del desarrollo de la investigación, incluyendo los entornos y el proceso de prueba. Se muestran los resultados de la ejecución de las pruebas de software y la creación de los

resúmenes de pruebas. Posteriormente, se analizaron y compararon los resultados obtenidos por medio de gráficas, para su respectiva puntualización. Capítulo 6: Se incluyen las conclusiones, recomendaciones obtenidas durante el proyecto de investigación, así como el trabajo futuro.

CAPÍTULO I. PLANTEAMIENTO DEL PROBLEMA / ANTECEDENTES

1.1 Descripción del Problema

En una aplicación o software *web* el backend estructura y ejecuta procesos para el correcto funcionamiento de la aplicación, por lo que representa una capa lógica muy importante. Dentro de la industria del software, existe una gran variedad de frameworks backend, esto produce un impacto positivo para los desarrolladores y la calidad del software.

Sin embargo, el framework backend “perfecto” como tal, no existe. Ciertamente se trabaja con el mas ideal, pues en realidad un programador ajusta y acondiciona las necesidades de una aplicación *web* a un frameworks backend y viceversa. Sin duda, el framework backend elegido debe cumplir con todas las funcionalidades e instrucciones que se codifiquen y esto será de acuerdo a sus propiedades, características y posibilidades.

Es preciso mencionar que en la selección de un framework backend influyen muchos aspectos, como las necesidades del cliente (requisitos funcionales y no funcionales), la conformación de base de datos, la arquitectura del software *web*, la lógica de programación, las pruebas a realizar y límite de tiempo del proyecto. Esto no quiere decir que el o los frameworks backend utilizados sean los más adecuados, idóneos o convenientes para el desarrollo de una determinada aplicación *web*, sino que, en la mayoría de las ocasiones, el programador elije las tecnologías que conoce y domina, debido a la falta de información de otras alternativas tecnológicas de backend, a la carga de trabajo asignada al inicio de un proyecto de desarrollo, o a la falta de planificación de los tiempos de entrega del proyecto.

Estos y otros muchos factores en las etapas del ciclo de vida de desarrollo de software van dejando de lado más opciones de frameworks backend; aunque no es muy recomendable trabajar con las mismas tecnologías por largos periodos de tiempo, ya que puede resultar contraproducente y más, si dichas tecnologías no presentan un soporte o mantenimiento continuo. Por consecuencia esto genera un estado de conformidad negativa.

Es posible, que en un proyecto de software que utilice herramientas de software o librerías deprecadas, disminuya en cierto grado, su calidad, aun y cuando exista un aumento en los recursos técnicos, humanos y económicos. Para las empresas y los desarrolladores autónomos que no actualizan y amplían sus conocimientos y habilidades, se aminoran sus clientes, las oportunidades laborales con mejores ingresos monetarios y las posibilidades de posicionarse como un proveedor altamente competitivo en el mercado.

Es conveniente e importante analizar los frameworks backend para determinar cuál de ellos ofrece una mejor alternativa en la construcción de una aplicación *web* para implementar determinadas herramientas que ofrezcan tecnologías más novedosas y avanzadas que se han estado perfeccionando, moldeando y actualizado a lo largo de los años. En base a estos antecedentes y la problemática planteada anteriormente, se propone la implementación de pruebas de rendimiento y fiabilidad para evaluar los frameworks backend, puesto que las pruebas de software pueden aportar un nivel de cobertura más amplio sobre ciertos aspectos del framework y para obtener mayores datos e información en la elección de una tecnología backend.

Esta investigación se enfoca en la implementación de pruebas basadas en ISTQB para el análisis y comparación de frameworks backend en aplicaciones *web*.

1.2 Preguntas de Investigación

- ¿Qué características debe considerar un desarrollador a la hora de elegir un framework backend?
- ¿Qué beneficios obtendría un desarrollador al contar con información, acerca de pruebas que se han implementado sobre framework backend en aplicaciones *web*?
- ¿Cómo determinar, cuál framework backend es el más indicado o conveniente para solucionar y satisfacer cada una de las necesidades de una aplicación *web* de tamaño pequeña-mediana?

1.3 Objetivos

1.3.1 Objetivo General

Implementar pruebas de software basadas en ISTQB para analizar y comparar diversos frameworks backend y determinar cuál de ellos ofrece una mejor alternativa para la construcción de una aplicación *web*.

1.3.2 Objetivos Específicos

- Identificar las tendencias en el uso de frameworks backend para seleccionar aquellos más utilizados.
- Desarrollar el proceso de prueba y determinar las pruebas a ejecutar, definidas por la ISTQB.

- Implementar una aplicación *web* con APIs REST como entornos experimentales en cada uno de los frameworks backend seleccionados.
- Analizar y diseñar la composición de los casos, procedimientos y resúmenes de pruebas que se utilizaran.
- Ejecutar los casos y procedimientos de pruebas en cada entorno de prueba desarrollado.
- Efectuar un análisis comparativo de las evaluaciones obtenidas en los casos y resúmenes de prueba.

1.4 Justificación

La justificación y propósito para efectuar este proyecto de investigación se centra en el hecho de que tanto las organizaciones desarrolladoras de software como los programadores autónomos realizan la mayoría de las etapas del ciclo de vida de desarrollo de software, ya sea de un modo formal o informal. Dentro de estas fases se encuentra el proceso de implementación de software, que engloba parte de las tecnologías que se emplearán en la construcción de la arquitectura de un sistema y otros puntos, en esta parte, se selecciona menor o gran variedad de herramientas, incluyendo los frameworks backend. Del mismo modo, está la fase pruebas, que es una parte elemental para evaluar y verificar el preciso desempeño de un software *web*. Los aspectos anteriores mencionados, van de modo coherente y en comunión con otras fases del desarrollo, ya que todo el proceso dependerá tanto del ámbito, las historias de usuario, la magnitud, el alcance del proyecto y otros aspectos técnicos que se consideran, dado que cada proyecto *web* es diferente.

En el proceso de creación de un software *web*, usar uno o varios frameworks backend resulta muy favorable. Existen una gran variedad de

frameworks backend, que aumentan y facilitan las actividades de desarrollo, en el proceso lógico que debe contribuir el backend. Sin embargo, cada uno de ellos posee cualidades y debilidades, que hace, uno frente al otro más ventajoso y en ocasiones más actualizado. Por estas razones, resulta importante, la implementación de pruebas basadas en ISTQB para el análisis y comparación de frameworks backend en aplicaciones *web*, ya que se pretende contribuir a identificar y examinar las características más destacadas de determinados frameworks backend; así como influir en proceso del desarrollo, aprovechando estas características de forma eficaz y eficiente.

Debido a lo anterior, el propósito de la presente investigación es proporcionar información, que aporte y guie a los desarrolladores en el proceso de selección de uno o más frameworks backend, para determinar la opción más relevante que se adecue y solvente las necesidades de un proyecto de desarrollo de software *web*.

1.5 Alcances y Limitaciones

1.5.1 Alcances

- Aun cuando existen los estándares y normas ISO/IEC 9126, ISO/IEC 25010, ISO/IEC 29119, entre otras para la mejora de la calidad y las prácticas pruebas de software, en este trabajo únicamente se utilizarán las pruebas descritas por la ISTQB.
- Aunque existen frameworks de backend, frontend y fullstack en esta investigación nos enfocaremos única y exclusivamente a la comparativa de frameworks de backend.
- Para comparar frameworks de backend existen diferentes métodos, técnicas, métricas, herramientas para la obtención de datos y

características. Para este trabajo se utilizarán las pruebas de carga, comportamiento temporal, utilización de recursos, estrés y capacidad de recuperación.

1.5.2 Limitaciones

- Para el caso de la gestión e implementación del software se opta por crear un cliente ficticio en un ambiente controlado, dado que la aplicación *web* solo se utilizará para medir criterios de calidad en base a las pruebas.
- Este análisis comparativo de software está diseñado para soluciones de tamaño pequeño-mediana, no está adecuado para soluciones robustas y amplias.
- Las nuevas tecnologías existen en un futuro, respecto a frameworks, herramientas de testing, normas, entre otros, pueden no estar incluidas de momento en esta investigación.

CAPÍTULO 2. MARCO TEÓRICO

En el presente capítulo se presentan las ideas, nociones e investigaciones previas a la realización de este trabajo de investigación.

2.1 Bases teóricas

2.1.1 Aplicación web

Un software que se efectúa como un cliente-servidor remoto en un navegador *web*, se le designa aplicación *web*. Los usuarios o clientes pueden acceder a estas herramientas interactivas por medio de la internet con la ayuda de un navegador *web*, para efectuar un sinfín de funciones, actividades, intercambio de datos y muchas otras operaciones. Por lo que está estrechamente relacionado con el almacenamiento en la nube [7]. En general, es un software o agrupación de programas construidos en lenguajes y compilados en tecnologías *web* dedicadas.

2.1.2 Backend

Es la infraestructura que se encarga del correcto funcionamiento de uno o varios sistemas. Procesa aplicaciones “CMF o Back Office” enfocándose en el mantenimiento y la arquitectura de las peticiones y respuestas de los usuarios [8, 9]. En otras palabras, es la parte racional de un sitio que controla, aprueba y restringe la accesibilidad de los datos almacenados por los usuarios de un

software, en su mayoría esta parte lógica es totalmente desapercibida por los mismos.

2.1.3 Frontend

Presenta un enfoque de interfaz gráfica, orientada a la intercomunicación del usuario. Por lo que, parte de sus funcionamientos principales son la diagramación de contenido visual altamente atractivo, funcional y esquemático para mejorar la experiencia del usuario al interactuar con un determinado sistema. Dentro de la codificación del diseño, se pueden utilizar tecnologías como CSS (Hoja de estilo en cascada), HTML (Lenguaje de marcado de hipertexto), JavaScript (Lenguaje de programación), librerías y frameworks como React Angular (Librerías de JavaScript), Bootstrap (Biblioteca CSS para el diseño), entre otros. Asegurando la compatibilidad multiplataforma entre los estilos y los distintos dispositivos. Contrariamente el frontend enlaza y propicia la comunicación con el backend por medio de JavaScript, AJAX (Técnica de sincronización) y JSON (Formato de notación de objetos) [10].

2.1.4 Framework

Un framework es un conjunto de herramientas de desarrollo cuyo objetivo es simplificar la formación de software *web* y móvil. Un framework posee una estructura que se usa como base principal por un desarrollador, para construcción de artefactos de software [11]. Esta estructura incluye archivos de distribución fundamental, elementos anticipadamente desarrollados por los creadores del framework, con el propósito de que sean aprovechadas nuevamente por la comunidad de desarrolladores y puedan construir

dependiendo de las necesidades en sus proyectos [12]. El objetivo del framework es optimizar el tiempo y esfuerzo en el desarrollo *web*. Además, con una serie de clases y funciones ya listas para su uso.

2.1.5 Framework backend

Un framework de desarrollo backend o server-side framework es el que trabaja en el lado del servidor. También están enfocados al desarrollo de sitios, páginas y aplicaciones *web*, pero en este contexto proporcionan herramientas y módulos para trabajar con el cliente-servidor donde se encuentra almacenada la información, seguridad, bases de datos y formularios [13, 14].

2.1.6 Ciclo de vida del desarrollo del software

El ciclo de vida de desarrollo de software también llamado SDLC (*Systems Development Life Cycle*) considera siete etapas imprescindibles que lo rectifican [15]. La conformación de estas fases, se describen de la siguiente manera:

- *Planificación:* se compone de varias actividades, que comprenden el propósito del proyecto, así como, sus objetivos. Se inicia con la recolección de requisitos, se gestiona la cronología de las actividades a realizar y el tiempo que tomará en completarlas. También se incluye la estimación de costos, recursos y capacitaciones. Así como la identificación de los riesgos que podrían afectar el éxito del proyecto.
- *Análisis:* esta etapa implica el establecimiento de requerimientos, comúnmente conocidos como historias de usuario (HU), por el medio del cual se definen y especifican las necesidades para la

creación de un nuevo software. La aprobación de los requerimientos por los miembros interesados (cliente y analista según el caso) verifica y valida la aceptación de las características que contendrá dicho software.

- Diseño: es una fase robusta y compleja. Después del análisis de requisitos se prosigue al diseño (prototipos) de módulos, así como determinar las tecnologías adecuadas e infraestructura requeridas, para construir la arquitectura del software.
- Implementación: esta etapa implica principalmente al personal de desarrollo. De acuerdo al proyecto de software que se construirá los desarrolladores seleccionan los recursos, herramientas y el o los lenguajes de programación que se utilizaran para la construcción del producto de software. Dentro de esta misma fase se elaboran los casos de prueba (análisis y diseño de pruebas) que se utilizarán para probar los productos terminados.
- *Pruebas:* esta es otra de las fases complejas del desarrollo de software. El propósito de las pruebas de software es encontrar todos los defectos posibles de las etapas anteriores para su rectificación. Se considera que una prueba es "victoriosa" si se encuentran errores. La realización de pruebas está conformada por un conjunto de actividades y tareas. Este proceso asegura el control de calidad, persiguiendo que el software sea probado en distintas maneras a lo largo de las fases que conforman el inicio de su formación hasta su disposición con el usuario [15, 16].
- *Instalación o despliegue:* En esta etapa el software es puesto a disposición del cliente para su producción. Antes de esta entrega, se llevan a cabo varias tareas en donde se consideran las dependencias necesarias de los componentes y la planeación de configuración de entorno, así como su instalación.

- *Uso y mantenimiento:* En esta fase se trabaja en las actualizaciones de software y la corrección de errores que pudieron haberse presentado en el sistema.

  La preservación y manteniendo del software abarca varios aspectos:

  - Eliminación de las imperfecciones encontradas a lo largo de su periodo servicial (mantenimiento correctivo)
  - Integración de nuevas características o cambios, así como flexibilidad peticiones (mantenimiento adaptable)
  - Renovación del software al agregarle nuevas funcionalidades (mantenimiento perfeccionado)

  Esta etapa es considerada una de las más cruciales en el desarrollo de software, puesto que un sistema no se desgasta al utilizarlo.

2.1.7 Metodología de desarrollo de software

Es un modelo o marco de trabajo con procedimientos, técnicas y métodos usados para el desarrollo del software. Estos marcos constituyen la estructura del trabajo, para las soluciones metodológicas en las fases de planeación, diseño, implementación, despliegue y mantenimiento [17].

2.1.8 Prueba de software (Software Testing)

Las pruebas de software son los procedimientos para valorar y comprobar que un sistema funciona conforme a lo especificado y requerido para el cliente [18, 19]. También, llegan a ser indagaciones probatorias con la determinación de

prevenir, corregir y aportar información respecto a la calidad del software, antes de su puesta en producción.

2.1.9 Patrón de arquitectura MVC

Se utiliza para catalogar los datos, el control del software (lógica) y la visualización de interfaces que el usuario ve. En este patrón se destaca la partición del trabajo central que administra las acciones del software, la variedad de los modelos que manipulan la información y la vista que expone la respuesta de los datos a los usuarios [20].

- *Modelo (model):* su papel es ser administrador y manejar la lógica ya que se encarga maniobrar, organizar y renovar la información. Cuando se usa en un gestor de datos el modelo es quien efectúa las inserciones, filtros, modificaciones, búsquedas, eliminaciones y consultas.
- *Vista (view):* asignado como el frontend, su trabajo es generar y mostrar la interfaz al usuario, es decir integrar las pantallas, formularios, páginas o cualquier otro componente como resultado visual que integra un sistema. Estas interfaces sirven para mandar peticiones al sistema y podrían ser constituidas por lenguajes CSS, HTML, HTML5, Javascript, etc.
- *Controlador (controller):* Es la parte intermedia entre la vista y el modelo, sus enlaces registran y gestionan las peticiones e instrucciones obtenidas en la vista, para después procesarlas, enviar los resultados correspondientes y mostrar los datos. A su vez los datos pueden ser presentados en un determinado formato, por lo que el controlador también es un transformador de información.

2.1.10 APIs: Información centralizada y sistemas distribuidos

Las interfaces de programación de aplicaciones (API REST) son los procesos protocolarios y definiciones que se ajustan al diseño e integración arquitectónica de microservicios en el software (emplea el uso de transferencia de hipertexto) [21, 22]. Se representa como un acuerdo de comunicación entre los usuarios y el sistema de datos, aquí se constituye la solicitud y respuesta. Si se utilizan diferentes APIs para crear una aplicación, en el fondo, se tiene un sistema distribuido.

2.2 Normas y estándares regulatorios para la calidad del software

2.2.1 Norma ISO/IEC 9126:2001

La norma ISO 9126 es un estándar internacional para la definición del software. Esta norma establece un modelo que enfatiza cuatro temas para asegurar la calidad de los productos de software, los cuales se dividen de la siguiente manera:

- Manifiesta un modelo de calidad de software constituido por características y subcaracterísticas.
- Valora las características externas y propiedades de la calidad adecuando métricas externas.
- Valora las características internas y propiedades de la calidad adecuando métricas internas.
- Especifica parámetros de calidad para la medición de propiedades en su utilización.

Ciertamente, la norma ISO/IEC 9126 fue constituida en el año de 1977 por McCall. Posteriormente, en el siguiente año su colega Boehm, definió un modelo diferente para la calidad que agrupaba las características a evaluar en un software, añadiendo la “utilidad del producto”. Este modelo fue nombrado como ISO 9126-1 [23]. Durante el mismo año se propuso el modelo FURPS (Funcionalidad, Usabilidad, Confiabilidad, Desempeño y Soporte) elaborado por Hewlett Packard y Robert Grady.

En la definición del marco de calidad de la norma ISO/IEC 9126, se declaran seis características tanto internas como externas, a su vez estas características contienen subcaracterísticas que las dividen. Se aplican de forma externa si el software es empleado para complementar un sistema y una secuencia de propiedad interna en el software.

Cuando se valora y se mide el funcionamiento del sistema de acuerdo a las especificaciones indicadas por el usuario, se usa la calidad externa. Mientras que la calidad interna valora todos los atributos que se establecen como necesarios en un software, incluyendo los requisitos para medir sus propiedades inherentes [24]. La Figura 1 presenta el marco referencial del modelo de calidad por la ISO/IEC 9216.

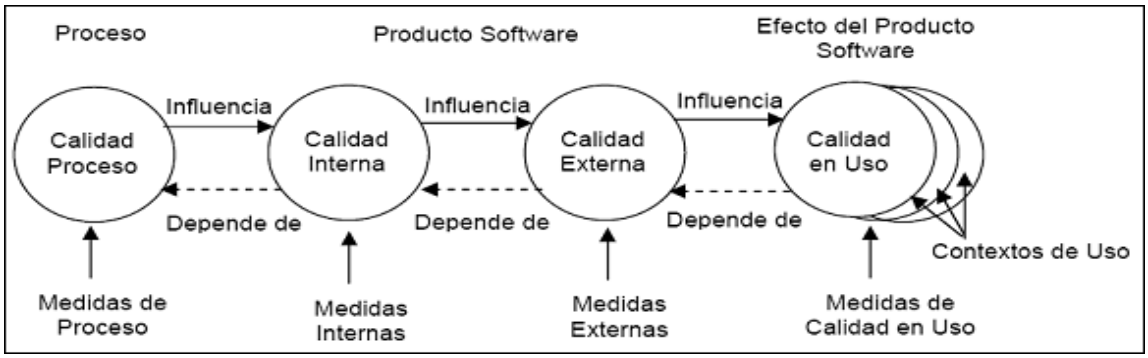


Figura 1. Marco referencial del modelo de calidad por la ISO/IEC 9126 [19].

El conjunto de características conformados según la norma, se organizan de la forma siguiente: Funcional, confiable, usable, eficiente, mantenimiento y portable. Cada propiedad interna y externa con su grupo de susbcaracteristicas, como se muestra en la Figura 2.

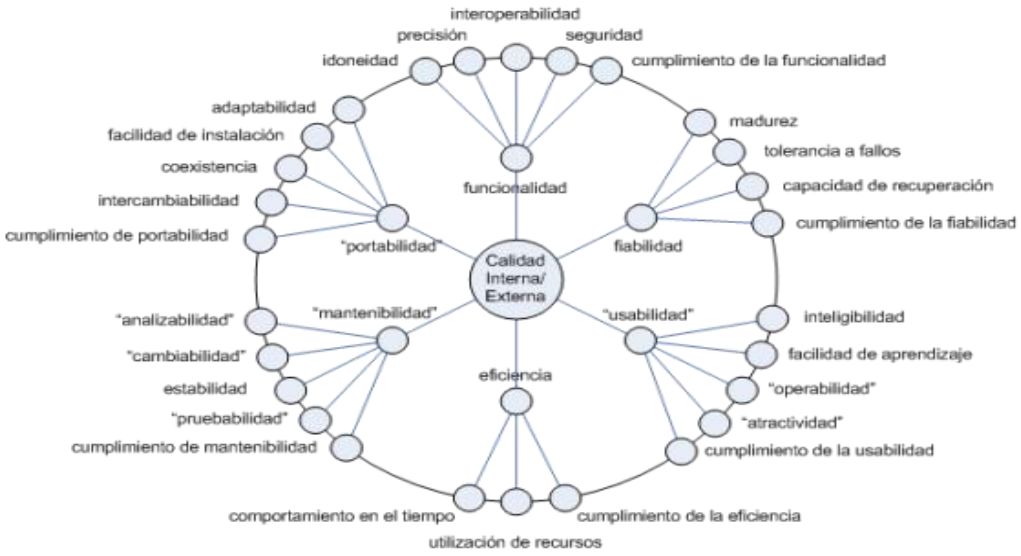


Figura 2. Características y subcaracterísticas de calidad de la ISO 9126 [19].

En los contextos de uso, la manera en que el software cumple su funcionamiento eficientemente, para con los usuarios que lo usan, se le llama calidad en el uso. Las medidas de calidad de uso se centran en probar cada funcionalidad y proceso del software que estará usando el usuario. De la misma forma se asegura de probar atributos alineados a la industria del desarrollo del software [25]. Por tanto, se indican los cuatro principios básicos de la calidad de uso:

- *Eficacia:* se refiere al correcto y exacto desempeño del sistema cuando el usuario interactúa con el software.
- *Productividad:* la producción que se obtiene de un software deriva de los recursos que los usuarios emplean. Entre menor sea el

consumo de recursos y mayor la cantidad de producción, mejor es el software.

- *Seguridad:* apunta a la protección de datos e información de los usuarios o del sistema para impedir el acceso a softwares maliciosos o no autorizados.
- *Satisfacción:* la resolución del usuario al utilizar el software y comprobar si satisface sus expectativas.

2.2.2 Norma ISO/IEC 25000 – SquaRE

Es una agrupación de normas abreviadas como SQuaRE (en inglés *System and Software Quality Requirements and Evaluation*), que comparten la finalidad de constituir un entorno laboral colectivo para la evaluación de los productos de software como condiciones de calidad.

La agrupación ISO/IEC 25000 es el perfeccionamiento como producto del trabajo en descendientes normas, inicia principalmente con la norma ISO/IEC 9126.

Esta norma detalla las propiedades de un marco de calidad en la industria del software y la norma ISO/IEC 14598 que implementó dicha evaluación. Actualmente la agrupación de normas ISO/IEC 25000 se constituye en cinco divisiones ascendentes [26]. La Tabla 1 muestra las divisiones de la norma ISO/IEC 25000.

Tabla 1. Divisiones de la norma ISO/IEC 25000.

ISO/IEC	División
2500n	Gestión de Calidad
2501n	Modelo de Calidad
2502n	Medición de Calidad

ISO/IEC	División
2503n	Requisitos de Calidad
2504n	Evaluación de Calidad

Teniendo presente lo anterior, la división ISO/IEC 2501n referente al modelado de calidad se subdivide en las normas ISO/IEC 25010 y ISO/IEC 25012. De estas, la norma ISO/IEC 25010 también manifiesta la descripción de características y subcaracterísticas como métricas en los productos de software y el uso de los mismos [27]. La Figura 3, muestra las características y subcaracterísticas de la ISO 2510.

Figura 3. Características y subcaracterísticas ISO 2510 [21].

2.2.3 ISO/IEC/IEEE 29119-1:2020 Estándar para pruebas de software

El estándar internacional ISO 29119 tiene una especificación en el desarrollo de pruebas para software, que sirven para administrar, ejecutar y llevar a cabo el proceso de pruebas desde inicio hasta fin. Sus principios claves comprenden la definición que persiguen las pruebas de software en cada fase del proceso genérico, así como los niveles de pruebas. Por lo se apoyan de la descripción de procesos, vocabularios, ejemplos, diagramas informativos y

secuenciales, técnicas, adaptaciones a los modelos para el desarrollo del software, etc. El modelo que ofrece la norma está pensado para el personal o probadores que realiza las pruebas de software “Testers”, así como desarrolladores implicados en la actividad [28]. La Figura 4, muestra la estructura del estándar ISO/IEC 29119.

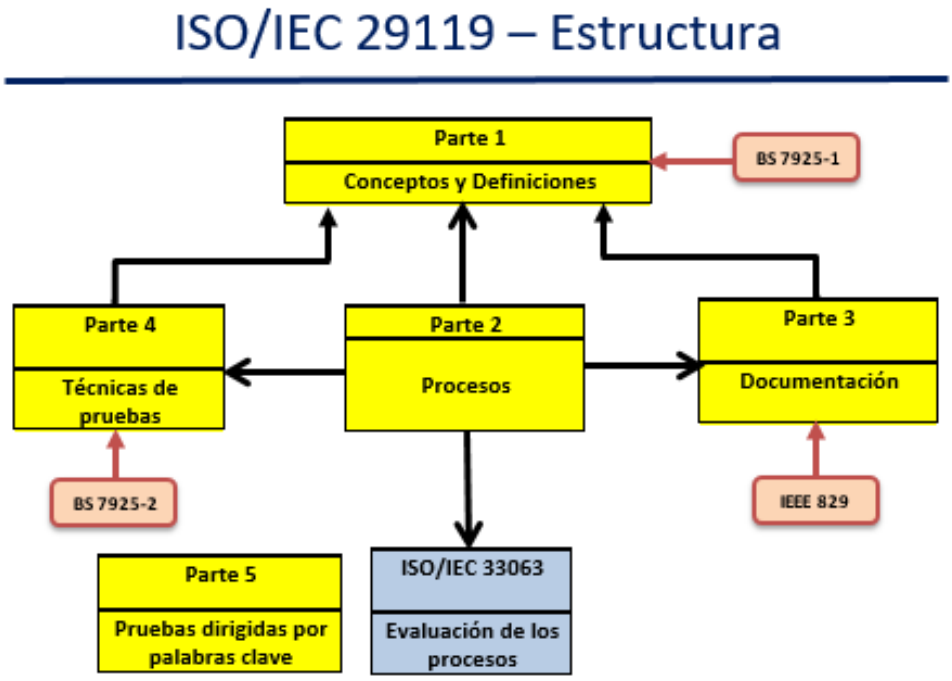


Figura 4. ISO/IEC 29119 – Estructura [29].

2.3 Comité internacional de certificaciones de pruebas de software (ISTQB - International Software Testing Qualifications Board)

La organización internacional ISTQB (en inglés *International Software Testing Qualifications Board*) es una certificadora en la evaluación de pruebas de software. Esta organización cuenta con gran conocimiento y amplia experiencia fundamentada de cientos de profesionales en el área de pruebas de software.

Fue instituida en la capital de Edimburgo (Escocia) en noviembre del año 2002 y legalizada en Bélgica [29]. En otro orden de cosas, el comité ISTQB define prueba o *testing* como un procedimiento que abarca cada una de las etapas del desarrollo de software y los productos de trabajo para encontrar fallos o imperfecciones en un software, asegurando la validación de la calidad en un sistema [30].

2.3.1 Principios y proceso en la gestión de pruebas de software

2.3.1.1 Siete principios de prueba

- *La prueba no demuestra la perfección en un software, si no, la existencia de errores:* Probar no garantiza la erradicación de defectos, pero si disminuye la posibilidad de encontrar y presenciar fallas. Incluso si se encuentran errores, no hay una eliminación total de los mismos.
- *Las pruebas absolutas son irrealizables:* No sería lógico creer que se puede probar en su totalidad la composición y funcionalidad de un software, excepto si este es sumamente simple.
- *La prueba como medida de prevención para el ahorro de recursos:* Resalta lo importante que es probar desde el inicio en el ciclo de desarrollo del software, para prevenir incidencias y retrasos. Comenzar a probar desde los pre requisitos iniciales para coordinar las tareas, actividades y elección de las pruebas con las etapas especificación de requisitos y diseño del software.
- *Defectos en cadena:* Una falla puede generar una cadena de errores. La mitigación de estos defectos puede controlarse y

suprimirse según las funcionalidades del caso y el análisis de riesgos.

- *Reflexión "paradoja del pesticida":* Da a entender que se necesita el constante mejoramiento de las pruebas, buscando implementar casos nuevos y actuales que permitan el descubrimiento de defectos en el software. Reincidir el mismo caso de prueba cotidianamente no mostrará errores existentes.
- *Importancia del contexto que engloba la prueba:* Es uno de los puntos cruciales, la prueba se debe construir partiendo del tipo de software que se esté desarrollando, incluyendo la metodología usada para el desarrollo del proyecto, así como la industria y los usuarios a los que va dirigido, entre otros aspectos.
- *Falacia de la inexistencia de errores:* Es una realidad que solucionar todos los defectos que se vayan encontrando, no respalda el triunfo del software. Puesto que el sistema siempre tendrá fallas que no cumplan con las exigencias de los usuarios [31].

2.3.1.2 Proceso de pruebas

Para realizar pruebas de software es imprescindible llevar a cabo un proceso de pruebas intrínseco y específico basado en sus factores, que garantice la calidad del software y contribuya con el éxito del proyecto. Los siguientes principios detallan el contenido general de este proceso.

2.3.1.2.1 Entorno del proceso de prueba

Generalmente el entorno del proceso de pruebas es el conjunto de elementos o factores que lo constituyen, conocido comúnmente como el

contexto. No existe una limitación de componentes que puedan influir en el proceso de prueba dentro de una corporación, puesto que cada proyecto es único. Así pues, se presentan los factores primordiales:

- Modelo y metodología de desarrollo de software.
- Niveles, tipos y técnicas de prueba.
- Riesgos implicados en el producto y el proyecto.
- Dominio y enfoque del empresarial.
- Limitaciones operacionales, que no se limitan a los tiempos, complejidades, estipulaciones y estipulaciones tanto normativas como procedimentales.
- Directrices, procedimiento y políticas de la corporación.
- Normas y estándares internos y externos, indispensables [31].

2.3.1.2.2 Actividades y tareas de pruebas

Las actividades y tareas de prueba consisten en el siguiente grupo:

- *Planificación:* es donde se elaboran el o los planes de prueba, según sea necesario. Se describen el alcance, objetivos, el enfoque general de la prueba, la calendarización de actividades, se lleva un registro de los riesgos y se realiza un presupuesto de costos y recursos.
- *Monitorización y control*: Se refiere al seguimiento continuo por medio de métricas que comparan los resultados reales en los avances de las pruebas con lo planificado en la planeación. De esto derivan los resúmenes de las pruebas.
- *Análisis:* el análisis se enfoca en los requisitos del software, ya que son tomados como una base para identificar las condiciones de las pruebas, posteriormente definirlas y priorizarlas.

- *Diseño:* una vez completado el análisis y la identificación del entorno, datos, infraestructura y herramientas necesarias para las pruebas. Se diseñan los casos de prueba priorizados.
- *Implementación*: se crea el entorno o script de automatización de la prueba, siguiendo los procedimientos para llevarlos a su ejecución. En esta parte se indican los juegos y calendarios de ejecución de pruebas.
- *Ejecución:* se ejecutan los juegos de pruebas y se realiza documentación sobre el estado de los casos de prueba e informes de defectos encontrados.
- *Compleción:* indica la finalización y los resúmenes de las pruebas para verificar que no se presenten defectos. Se completan todas las actividades y tareas de pruebas para el cierre de las mismas [32].

2.3.1.2.3 Documentación y productos de trabajo de la prueba

Individualmente, cada una de las actividades y tareas mencionadas anteriormente, generan productos y documentación de trabajo, que forman y complementan el proceso de prueba. Algunos de estos productos se mencionan en este apartado [32]:

- Planificación de la prueba
  - ✓ Plan de pruebas (uno o más).
- Monitorización y control de la prueba
  - ✓ Reporte de avance de prueba.
  - ✓ Reporte de resumen de prueba.
- Análisis de la prueba
  - ✓ Documentación de condiciones definidas y priorizadas.
  - ✓ Enunciados de trabajo de prueba.
- Diseño de la prueba

- ✓ Casos y conjuntos de prueba.
- ✓ Documentación de entornos de prueba.

- Implementación de la prueba
  - ✓ Procedimientos y secuencias de prueba (pueden ser automatizadas).
  - ✓ Cronograma de ejecución.
  - ✓ Conjuntos de pruebas.
- Ejecución de la prueba
  - ✓ Documentación del estado individual en los casos de prueba.
  - ✓ Reportes de defectos.
- Compleción de la prueba.
  - ✓ Reportes de prueba.
  - ✓ Documentación de finalización de pruebas y aprendizajes.

2.3.1.2.4 Trazabilidad persistente entre base de prueba y productos del trabajo de prueba

El seguimiento sobre la base y los productos de trabajo de la prueba, es un monitoreo continuo que registra el avance de las pruebas durante todo este proceso. Una adecuada trazabilidad puede implicar, parte de los siguientes puntos:

- Estudiar los efectos y consecuencias de los cambios.
- Viabilidad de auditorías y evaluaciones.
- Realización de los principios en los alineamientos TI.
- Acrecentar el entendimiento en los reportes de avances y resúmenes de prueba.
- Vincular las cuestiones técnicas de las pruebas con las ya involucradas para una buena relación.

- Contribuir con datos y referencias para valorar el progreso del proyecto conforme a los objetivos logrados, la calidad de los productos de trabajo y el desempeño del proceso [25].

2.3.2 Cuatro niveles de prueba de software

En síntesis, hay cuatro niveles de prueba vinculados entre sí, que indican que probar. A continuación, se presentan estos niveles:

- ✓ *Pruebas de unidad o componente:* también denominadas pruebas de módulos o clases. Son comúnmente realizadas por los desarrolladores y se enfocan en ejecutar pruebas funcionales, no funcionales, elementos, métodos y fragmentos de código individuales para verificar su correcto funcionamiento. Esta verificación también puede ser realizada con pruebas automatizadas.
- ✓ *Pruebas de integración:* consisten en validar que las interacciones funcionan de manera correcta entre distintos componentes del software.
- ✓ *Pruebas de sistema:* radica en comprobar que las operaciones de todo el software fueron diseñadas y construidas conforme a los requerimientos funcionales y no funcionales, por lo que, se verifica que el comportamiento sea lo esperado.
- ✓ *Pruebas de aceptación:* se enfoca en probar el sistema en su totalidad para validar que está completamente funcional. En esta prueba pueden participar los usuarios finales para obtener información sobre la aceptación y despliegue del software. Se denomina pruebas Alpha si la prueba de aceptación se ejecuta dentro de las instalaciones de la organización, de lo contrario se le denomina Beta por ser ejecutada en los cimientos del cliente [31].

2.3.3 Tipos de pruebas

A continuación, se presentan los tipos de prueba descritos por la organización ISTQB.

2.3.3.1 Pruebas funcionales

Son las pruebas que evalúan las funcionalidades y comprenden el comportamiento de un sistema, las pruebas se enfocan en el “que” debe hacer el software. Cada funcionalidad del sistema puede estar indicada en las historias de usuario, la especificación de requisitos de software, épicas, etc. Eventualmente los casos de prueba se definen y diseñan partiendo de estas funciones [33]. Como las pruebas funcionales examinan el actuar de un sistema, se emplean técnicas de caja negra. Otras pruebas que se pueden ejecutar en el área de conocimiento, son las siguientes:

- ✓ Idoneidad: Evalúa si el sistema cumple y hace todas las funciones necesarias para satisfacer al usuario.
- ✓ Precisión: Verifica que las funciones se ejecuten conforme a la especificado.
- ✓ Interoperabilidad: Capacidad de un software para interactuar con otros entornos o sistemas, sin presentar errores de compatibilidad.
- ✓ Conformidad: certifica que el sistema cumple con requerimientos de los estándares o normas ISO.
- ✓ Seguridad: se refiere a proveer medidas de protección para la información y los datos del sistema, por medio de privilegios y accesos [34].

2.3.3.2 Pruebas no funcionales

El objetivo de las pruebas no funcionales es probar las características de un software, refiriéndose al cómo debe de funcionar. Por lo tanto, se deben hacer pruebas no funcionales en todos los niveles de prueba, utilizando técnicas de caja negra para las condiciones y casos de prueba. Se puede consultar los estándares de calidad ISO/IEC 9126 [33, 35] e ISO/IEC 25010 para una clasificación de las características de calidad indicadas [31]. Las siguientes características son ejemplos de pruebas no funcionales:

- ✓ Fiabilidad: Es la vialidad que un software presenta para mantener su funcionamiento sin fallos.
- ✓ Usabilidad: Es el entorno e interfaz amigable para el usuario que proporciona el sistema.
- ✓ Eficiencia: Mide la eficiencia del tiempo de respuesta y los recursos del sistema.
- ✓ Mantenibilidad: Es la facilidad que presenta un software para realizar un nuevo cambio, actualización, prueba, análisis, etc.
- ✓ Portabilidad: Cualidad de un sistema para funcionar y adaptarse en diversos entornos [34].

2.3.3.3 Pruebas estructurales o de caja blanca

Las pruebas estructurales, de caja blanca o de cristal se centran en la arquitectura, implementación, flujos y código fuente del software. Dado que el probador comprende la arquitectura del software, su estructura y el código del mismo, se puede medir la prueba con base al porcentaje de código que ha sido probado y es designado como cobertura de código. En esta prueba se utilizan técnicas de caja blanca [30, 34].

2.3.3.4 Pruebas asociadas al cambio

Al presentarse un cambio en el sistema, ya sea por distintos factores como la corrección de un error, agregar una nueva funcionalidad o modificar un elemento. Se realizan este tipo de pruebas para verificar que se han solucionado los defectos o desarrollado una funcionalidad correctamente. También se puede obtener el impacto del cambio, ejecutando pruebas de confirmación que se enfocan en comprobar la solución del error presentado y las pruebas de regresión que buscan efectos colaterales o secundarios provocados por cambios. Estas pruebas también se ejecutan en todos los niveles de prueba y se aplican en todos los componentes afectados o no por el cambio.

2.3.4 Técnicas de pruebas

Por último, se describen las siguientes técnicas de pruebas:

Técnicas de prueba de caja negra, de comportamiento o conductuales: estas técnicas se pueden efectuar tanto en pruebas funcionales como no funcionales. También se enfocan en el comportamiento del software refiriéndose a las bases de prueba, las peticiones y respuestas de los objetos de prueba, pero sin entendimiento de su arquitectura interna.

Técnicas de prueba de caja blanca, cristal o estructurales: son la parte contraria de las técnicas de caja negra. Las técnicas de caja blanca son utilizadas en todos los niveles, pero más a menudo en el nivel de pruebas de componentes. Esto debido a que se dirigen a la arquitectura detallada del software, el código y el diseño detallado de alto nivel. La estructura interna de un sistema es la principal base de pruebas [33].

CAPÍTULO 3. ESTADO DEL ARTE

En este capítulo se exponen datos, ideas y material publicado o inédito relacionado a esta investigación.

3.3 Trabajos relacionados

3.3.1 Análisis de factores que inciden en la selección de un lenguaje y framework de programación para el desarrollo de software web

El trabajo “Análisis de factores que inciden en la selección de un lenguaje y framework de programación para el desarrollo de software” elaborado por Luzuriaga Mendoza Amanda Maribel, trabajo de Investigación por la Universidad Técnica de Machala (UTMACH), Machala (República del Ecuador), indica que su principal enfoque consistió en la recolección y el análisis de variables o aspectos fundamentales que influían en la elección de un framework y un lenguaje de programación [36].

Presentación del caso de estudio

El estudio se orientó en la elaboración de una encuesta, con una colectividad objetiva de veinte personas de corporaciones dedicadas al desarrollo de software y programadores autónomos en la capital bananera, para comprobar los principios más sobresalientes que repercutan en la preferencia de lenguajes de programación y marcos de trabajo.

Procedimiento de estudio empleado

Para este proyecto se identificaron y recopilaron de estudios realizados en artículos científicos los principios y variables consideradas para la elección de lenguajes y frameworks para el desarrollo *web*. Después se realizó una encuesta con dichos factores, en donde estos fueron expuestos a los participantes, para indicar un grado de importancia, basándose en la valoración de escala de Likert. Luego se determinaron las herramientas de programación, para ello se utilizaron los rankings de la IEEE Spectrum y Stack Overflow.

Las valoraciones obtenidas en los factores se analizaron para presentar tres comparaciones entre los lenguajes de programación y los frameworks. La Tabla 2, muestra las categorías y factores de lenguaje para el desarrollo de software.

Tabla 2. Categorías y factores de lenguaje [32].

Categoría	**Factores**
Generales	Rendimiento
	Flexibilidad
	Seguridad
	Curva de aprendizaje
	Tiempo de desarrollo
	Tendencia
Código	Paradigma
	Sintaxis del lenguaje
Plataformas y Herramientas	Sistemas operativos
	IDE / Editores de código
	Conexión a base de datos
	Servidor
	Frameworks y librerías
Soporte	Licencia
	Documentación
	Soporte de comunidad
Recursos	Desarrollador/res con habilidades

Categoría	Factores
	Costo: servidor, base de datos, licencia
Proyecto	Propósito
	Tamaño y alcance

La primera comparación consistió en realizar un análisis con los factores de la tabla; lenguaje para el desarrollo, por valor de relevancia. En donde se obtuvo un valor total por cada categoría. Dentro del análisis, las tres categorías y factores con el valor más alto fueron:

- Generales: destacaron los factores flexibilidad, rendimiento y seguridad.
- Proyecto: destacaron los factores propósito de proyecto, tamaño y alcance; el promedio del valor general y proyecto es de 4.28 correspondiente a la magnitud más alta entre las categorías.
- Soporte: destacaron los factores documentación y soporte; con una puntuación asignada de 4.27.

Seguido de lo anterior, se realizó una comparación entre los lenguajes Python, Java, JavaScript y PHP. Por lo que se indicaron las cualidades que posee cada lenguaje y se determinaron las propiedades en común; uso en distintos entornos, open source, servidor Apache, adaptación con BDR y NoSQL. Estos se encuentran entre los diez más usados y tienen una extensa documentación en la *web*. De forma individual se obtuvo que JavaScript y PHP son utilizados en Backend, Python cuenta con buena seguridad y Java resulta algo confuso por su sintaxis. La Tabla 3, muestra las Categorías y factores de frameworks para el desarrollo de software.

Tabla 3. Categorías y factores de framework [32].

Categoría	Factores
Generales	Rendimiento
	Flexibilidad

Categoría	Factores
	Robustez
	Curva de aprendizaje
	Tendencia
Herramientas	Arquitectura
	Herramientas y librerías
Soporte	Licencia
	Documentación
	Soporte de comunidad
Backend	Seguridad
	ORM y conexión a base de datos
Frontend	Componentes
	Comunicación del cliente-servidor
	Compatibilidad de navegadores
Recursos	Desarrollador/res con habilidades requeridas
	Costo licencia
Proyecto	Propósito
	Tamaño y alcance

La segunda comparación consistió en realizar otro análisis con los factores de la tabla; framework para el desarrollo de software, por valor de relevancia, siguiendo el mismo procedimiento para obtener valores totales por cada categoría. Las tres categorías con el valor más alto fueron:

- Backend: destacaron los factores protección, ORM y enlace a la base de datos; la puntuación obtenida 4.58, que es la magnitud superior.
- Frontend: destacaron la coexistencia con los exploradores y la red cliente/servidor; el promedio obtenido es de 4.50.
- Proyecto: en ambos análisis destacó la proposición y tamaño-alcance, como factores muy importantes; el promedio de esta categoría es de 4.35.

Después se compararon los frameworks a nivel backend respecto al lenguaje de programación. Sobre esta comparativa se encontraron los frameworks:

- Python: Django y Pyramid
- Java: Spring y Struts
- JavaScript: Express.js y Hapi.js
- PHP: Laravel y Symfony

En cada frameworks se destacan las características más óptimas, atractivas y benéficas, así como la arquitectura que manejan, la seguridad, su facilidad de uso, entre otros.

La tercera y última comparación, también consistió en basarse en los factores mencionados para comparar los frameworks de programación Angular, Vue y React, a nivel Frontend y determinar si, los tres frameworks comparados poseen características comunes, además de sus diferencias que son lo que más destacaban cada uno de ellos, por ejemplo: el DOM virtual que tiene React y Vue para un eficiente rendimiento, usa JavaScript, comunicación por Axios y la interfaz de usuario y componentes que se localizan en el mismo archivo, a diferencia de Angular con un DOM real de menor desempeño, usa Typescript, comunicación por HttpClient y la Interfaz de usuario separada en varios archivos. Como resultado de la investigación se desarrolló un software *web* con la metodología kanban.

Conclusiones del trabajo

La conclusión en este trabajo determina, que carece de efecto la existencia de un superior lenguaje y framework para el desarrollo de software, dado que las herramientas pertinentes cubrirán la totalidad de las exigencias del proyecto, sin embargo, el soporte, las librerías, la seguridad, el alcance y el propósito del

proyecto, etc. Son algunos principios que afectan la elección de un framework o lenguajes en la construcción de un backend y frontend de una aplicación *web*.

3.3.2 Análisis comparativo de frameworks open source para el nivel de productividad en los proyectos de desarrollo de software web

El trabajo “Análisis comparativo de frameworks open source para el nivel de productividad en los proyectos de desarrollo de software *web*” elaborado por Norbil Fernando Francia Peralta y realizado en la Universidad Tecnológica de Perú (Chiclayo, Perú), indica que su prioridad fue el análisis de la producción en el desarrollo del software, incluyendo estimar los elementos que inciden en la eficacia de los proyectos y la aportación en la aspectos relevantes en las compañías de desarrollo de software [37].

Estudio del caso expuesto

Este proyecto se realizó en la empresa Siempresoft, dedicada al rubro de software con más de una década en el negocio, ubicada en la localidad de Chiclayo, Perú. Se tomó como población al encargado de software y un proyecto corto como factor de estudio, ambos pertenecientes a dicha empresa.

Procedimiento de estudio empleado

En este análisis, primero se realizaron comparativas, según atributos y características de investigaciones realizadas por otros autores, sobre metodologías de desarrollo, los lenguajes de programación ASP.NET, PHP; y los marcos propios de estos lenguajes. Concluyendo con la metodología Scrum y el framework Laravel como los apropiados en la productividad en la implementación de software por medio de un prototipo. La Tabla 4, muestra el puntaje de los factores establecidos por cada framework.

Tabla 4. Puntaje de los factores establecidos por cada framework [33].

Framework	Laravel	Codeigniter	Fuelphp	Cakephp	Symfony	Zend	Phalcon	Yii	Slim	Phpixie
Validación integrada	3	1	1	1	1	1	1	3	1	1
Sistema de plantillas rápido y flexible	3	1	1	1	1	1	1	2	1	1
Documentación	3	1	2	2	2	1	1	3	1	2
Scaffolding	3	1	2	1	1	1	1	3	1	1

Se determinó una encuesta como el segundo método para la recolección de información. Dicha encuesta se aplicó al jefe de producción, en dos maneras: verbal y no verbal; para obtener datos e información sobre las herramientas, preferencias de tecnologías en los clientes, procesos y la productividad en la producción de proyectos de software. Después se analizó y presentó la información obtenida en la entrevista, por lo que, según la experiencia vivida de la empresa, se obtuvo que:

Los sistemas de escritorio-Cliente servidor es el nivel más alto en la productividad de proyectos de software. La Tabla 5 muestra la proporción asignada al tipo de proyecto.

Tabla 5. Proporción asignada al tipo de proyecto [33].

Sistemas de escritorio - Cliente servidor	Sistemas de entornos *web*	Aplicaciones Móviles
100%	40%	80%

Por lo cual, IBM Informix 4GL, IBM ONE, Nexus y .Net fueron las herramientas más utilizadas dentro de la empresa, sin embargo, PHP es el lenguaje con facilidad de entendimiento por los programadores. En otro aspecto, los equipos de desarrollo usaban los modelos ágiles SCRUM y XP, he indicaba

que una buena productividad depende del monitoreo del cumplimiento en los requerimientos establecidos en el contrato inicial y utilizaban la gestión de cambios, duración real vs duración ideal, los gastos reales vs gastos ideales y la cantidad de devoluciones (control de calidad) como indicadores para medir la productividad.

De igual manera, se consideró que los factores más influyentes, para tener éxito en un proyecto serían la comunicación, la experiencia y las habilidades de desarrollo. Esta empresa, la eficiencia del tiempo invertido, la inversión necesaria de recurso y la perfección de los procesos es una expectativa de los clientes.

Conclusiones del trabajo

Como conclusiones se diagnosticó que debido a que la empresa no hace una previa consulta de un marco de referencia (framework), para establecer las herramientas de desarrollo apropiadas, esto deducía un porcentaje de productividad en los proyectos menor al 50%. El aplazamiento de los proyectos generaba costos más elevados de lo planificado causando inconformidades, su nivel de productividad era medio. Es por ello que los proyectos se aplazan a un tiempo y costo mayor de lo estimado. Ante ello, el investigador plantea que los lenguajes de programación, las capacidades de los desarrolladores, el contraste de las tecnologías para el desarrollo de software y su pertinente elección dependerá del rubro de la empresa.

3.3.3 Análisis comparativo para la evaluación de frameworks usados en el desarrollo de aplicaciones web

La investigación “Análisis comparativo para evaluación de frameworks usados en el desarrollo de aplicaciones *web*” realizado por Raquel Espinosa-

Hurtado y realizada en la Universidad Nacional de Loja, (Loja, Ecuador), indica que su enfoque consistió en designar el más adecuado framework para aplicaciones web usando características y criterios de avaluación de la norma ISO 25000 [38].

Procedimiento de estudio empleado

El objetivo principal del proyecto fue la evaluación comparativa de los frameworks Laravel y Django empleados en el desarrollo de sistemas *web*; con este objetivo, se implementó una aplicación *web* y se determinaron criterios de evaluación en base al estándar de calidad ISO/IEC 25000, con el fin de conocer cuál de estos dos frameworks era el mejor.

Método de análisis empleado

La aplicación *web* se elaboró referente a un catálogo de productos en línea, usando el modelo XP como marco de trabajo. Posterior a ellos, se requirió la colaboración de colegas para el desarrollo del software, por lo que su implementación se constituyó en un servidor local y como gestor de datos MySQL.

Previo a realizar la evaluación de los frameworks, se establecieron criterios de acuerdo a las características de la norma ISO/IEC 25000 (ISO/IEC 25010). Los criterios evaluados fueron: cifrado de datos, conteo de código en líneas, consumo del CPU, consumo de memoria RAM, duración de las peticiones CRUD, documentación/aprendizaje del sistema, desempeño en la duración del montaje, facilidad de montaje, grado del éxito de aprendizaje obtenido, cavidad de acceso, restricciones y autenticaciones. Después se examinaron los valores de los resultados obtenidos para obtener un valor parcial y calcular una ponderación (/10) que diera un valor final, para la comparación.

Conclusiones del trabajo

Dentro de las conclusiones, se determinó que el framework Django fue el mejor y más conveniente entorno en la construcción de aplicaciones *web*, conforme a los principios elegidos en las características de seguridad, portabilidad, rendimiento y usabilidad. Esto dedujo una mejor calificación o porcentaje de Django sobre Laravel.

3.4 Análisis comparativo

En este apartado se incluye una tabla comparativa de los trabajos que tienen mayor relación con el problema a resolver, con la finalidad de justificar el desarrollo que se pretende, como se indica en la Tabla 6.

Tabla 6. Comparativa de trabajos relacionados.

Trabajo relacionado	Caso de estudio	Método o Instrumento de recolección de datos	Lenguaje	Framework	Metodología
Análisis de factores que inciden en la selección de un lenguaje y framework de programación para el desarrollo de software web	Empresas de desarrollo de software Desarrolladores independientes	Encuesta Estudios realizados en artículos científicos	Python Java JavaScript PHP	Django Pyramid Spring Struts Express.js Hapi.js Laravel Symfony	Kanban
Análisis comparativo de frameworks open source para el nivel de productividad en los proyectos de desarrollo de software web	Empresa de desarrollo de software Proyecto de desarrollo web	Entrevista Investigaciones realizadas por otros autores	ASP.NET PHP	Laravel Codeigniter Fuelphp Cakephp Symfony Zend Phalcon Yii	Scrum

Trabajo relacionado	Caso de estudio	Método o Instrumento de recolección de datos	Lenguaje	Framework	Metodología
Análisis comparativo para la evaluación de frameworks usados en el desarrollo de aplicaciones web	Desarrollo de Aplicación web	Norma ISO/IEC 25000	Ninguno	Slim Phpixie Laravel Django	XP (eXtreme Programming)
Implementación de pruebas basadas en ISTQB para el análisis y comparación de frameworks backend en aplicaciones web	Implementación de APIs REST con aplicación web	Pruebas de software basadas en ISTQB (International Software Testing Qualifications Board)	Ninguno	Spring Boot Django Laravel	Desarrollo iterativo e incremental

CAPÍTULO 4. METODOLOGÍA

4.1 Metodología de propuesta de solución

La metodología y el diseño propuesto para resolver el problema de investigación, se divide en 3 fases, a su vez en distintas etapas las cuales consisten en:

4.1.1 Fase 1. Obtención de información

Etapa 1. Análisis y determinación de frameworks backend

En esta etapa se identificaron las tendencias en el uso de frameworks backend y se seleccionó aquellos más utilizados durante el presente año, según revisiones sistemáticas de literatura expuestas en artículos de revistas, sitios de internet e informes tecnológicos.

Etapa 2. Análisis de propiedades de frameworks backend

Se identificaron la arquitectura, los lenguajes de programación y características adicionales que disponían los frameworks backend elegidos.

Etapa 3. Planificación del proceso de pruebas

Se analizó el proceso de pruebas y el conjunto de las principales actividades de pruebas. Así como se indicó la coordinación de las etapas del ciclo de vida en el desarrollo de software y la metodología utilizada en el proyecto.

4.1.2 Fase 2. Desarrollo de APIs REST y pruebas de software

Etapa 1. Componentes del software

Se estableció el enfoque y propósito de la aplicación *web*. Se determinó la construcción y la arquitectura de las APIs REST en la aplicación *web*, desarrolladas de acuerdo a la documentación oficial que se encuentra en los sitios oficiales de los frameworks backend utilizados. Se identificó el tipo de servidor con el que se trabajó, el gestor de base de datos utilizado y otras tecnologías empleadas en el desarrollo de la misma.

Etapa 2. Análisis y diseño de las pruebas

Se analizó la información referente a las pruebas seleccionadas, para priorizar las condiciones y diseñar los casos de prueba. Identificando los datos necesarios, la configuración del entorno, la infraestructura y las herramientas necesarias.

Etapa 3. Implementación y ejecución de las pruebas

Se desarrollaron y priorizaron los procedimientos de prueba y se configuraron scripts de automatización de pruebas en las herramientas de rendimiento indicadas. Se ejecutaron los casos de prueba y se registraron las valoraciones en la ejecución.

4.1.3 Fase 3. Cierre de pruebas y análisis de resultados

Etapa 1. Cierre de pruebas y resumen

Se presentaron los reportes o resúmenes de los casos y procedimientos de prueba.

Etapa 2. Análisis y comparación de resultados en las pruebas de software

Se graficaron los resultados obtenidos en las pruebas de software, para su comparación.

CAPÍTULO 5. RESULTADOS

5.1 Obtención de la información

5.1.1 Análisis y determinación de frameworks backend

Para la determinación de los frameworks backend, se realizaron desde revisiones de literatura en artículos, informes tecnológicos y sitios de internet. Por lo que se precisó utilizar los datos estadísticos de tecnología de la plataforma en línea Statista [1, 2], los resultados anuales de las encuestas para los desarrolladores en Stack OverFlow [3], la asignación de calificaciones en GitHub por desarrolladores hacia los frameworks en Acropolim [4] y la información proporcionada por los apartados de Radix [5] y Wearedevelopers [6]. Estos son sitios *web* que proveen datos e información sobre diferentes temas de contemporaneidad y poseen referencias a través de opiniones, encuestas y estudios del mercado actual en la industria del software.

La Tabla 7 presenta la lista de los frameworks de backend más populares mencionados en cada sitio.

Tabla 7. Lista de frameworks backend más populares en cada sitio.

Statista	Stack OverFlow	Acropolim	Radixweb	Wearedevelopers
Laravel	Express.js	Laravel	ASP.NET Core	ASP.NET Core
Django	ASP.NET Core	Django	Django	Spring Boot
Spring Boot	Vue.js	ABP	Laravel	Express.js
Flask	ASP.NET	Ruby on Rails	Ruby on Rails	Django
Express.js	Spring Boot	Spring Boot	Express.js	Ruby on Rails
Nest.js	Django		CakePHP	Laravel
Ruby on Rails	Laravel		Flask	FastAPI

Statista	Stack OverFlow	Acropolim	Radixweb	Wearedevelopers
Strapi	Ruby on Rails		Spring Boot	Vue.js

De acuerdo con la concurrencia de los datos presentados en dichas páginas *web*, se definió usar para esta investigación los siguientes frameworks backend:

- Spring Boot
- Django
- Laravel

5.1.2 Análisis de propiedades de frameworks backend

5.1.2.1 Framework Spring Boot

El framework Spring Boot se enfoca en el desarrollo de aplicaciones, páginas *web* y principios de diseño de código en el que el framework manipula la formación de los objetos e instancias (loC-contenedor de inversión de control), de código abierto de alto rendimiento, basado en Java y popular en los desarrolladores back-end. Fue inicialmente escrito por Rod Johnson y lanzado en el año 2003 durante el mes de junio y conforme a la licencia de Apache 2.0. Anteriormente a esto, en el año 2002 Johnson, había publicado el libro Diseño y desarrollo exporto de J2Ee uno a uno. Ya que Spring Boot da mantenimiento a varios frameworks Java, se puede interpretar es que el promotor de algunos ellos, como lo son Tapestry, EJB, Hibernate, JSF, Struts, etc [39, 40].

Características

Spring Framework consta de funciones organizadas en 20 módulos que proveen un rango de servicios. Estos módulos se agrupan en un Contenedor

central, Acceso/integración de datos, *Web*, AOP (Programación orientada a aspectos), Instrumentación y Prueba [40]:

- Maneja Stand-alone por lo que no necesita softwares terceros para su funcionalidad y tiene configuración automática.
- Soporte en abstracciones para JDBC, DAO, ORM, Marshalling XML y transacciones en la entrada a los datos. Así como la gestión remota por la configuración local o remota en Java por JMX.
- Puede desarrollar y hacer peticiones en los lenguajes Groovy y Kotlin.
- Presenta un software de testing con soporte para JUnit, WebTestClient, TestNG y TestContext. Al igual que mecanismos en el simulacro de objetos.
- Arquitectura MVC (Modelo, Vista y Controlador) como API REST en Spring WebFlux.
- Brinda el soporte enfocado a la programación orientada a aspectos (AOP), para las rutinas transversales y su funcionamiento.
- Basado en Core container y la inyección de dependencias (ID): ayuda a la inversión de control, el mantenimiento y la unión de clases. Gestiona el procesamiento de datos por lotes.
- Cuenta con servicios y tecnologías de correo electrónico, caché, tareas, eventos, automatizadas, relación remota, recursos, desarrollo, inyección de dependencias y JMS.
- Alta seguridad con credenciales, OAuth, LDAP, JWT, etc.
- Gestión de transacciones y acceso a base de datos por medio del ORM.
- Es un marco ligero por el uso, la reutilización y la legibilidad de POJO (Plain Old Java Object).

5.1.2.2 Framework Django

Django es una herramienta de código abierto y programado en Python, cuyo lanzamiento inicial se origina en el año 2005 después de su liberación bajo la licencia de BSD. La fundación del software Django quedó como responsable de la evolución de Django, esto en el año 2008. Parte de su objetivo es proporcionar y facilitar la producción en el desarrollo en software complejo y robusto. El framework añade énfasis a la reutilización de módulos, su eficiente compatibilidad con distintos gestores de base de datos, esto debido al ORM (Mapeo de objetos relacional) que permite y establece una conexión directa [41].

Características

Al usar Django, se adecua e incorpora una aplicación para la organización y administrar de todos los componentes que se encuentran alojados y pueden ser incluidos dentro de cualquier página *web* desarrollada en Django, a su vez permite gestionar, crear, actualizar, eliminar elementos propios del contenido e inspeccionar cada una de las acciones que se efectúan sobre estos elementos en distintas páginas partiendo de la misma instalación. También incluye el manejo de la autenticación y administración de los usuarios (con su respectivo señalamiento de permisos)

Así mismo el repartimiento de Django adhiere aplicaciones que contribuyen con un sistema de observaciones, instrumentos para vincular el contenido en las páginas *web* dinámicas y estáticas por medio de RSS y Atom, permitiendo la organización en la estructura de los elementos en una página, sin tener que recurrir a codificar vistas y controladores para dichas páginas [42]. Más particularidades de Django son:

- Entorno virtual (*Virtualenv*) para aislar la configuración de Python/Django.
- ORM para la comunicación de base de datos.

- API seguras, lógicas y robustas.
- Sistema incorporado a la vista que indican la lógica de las funciones, para gestionar solicitudes HTTP y enviar las respuestas.
- Mapeo de URLs cimentado en expresiones regulares y normalizado.
- Clases de software intermedio (Solicitudes, Vistas y Respuestas).
- Plantillas (Formularios HTML) con herencias y apoyado en etiquetas para estructurar y renderizar los datos.
- Autenticación, soporte de sesiones, interfaz de administración y permisos de los usuarios.
- Proporciona cacheo dinámico y flexible.
- Sistema que incluye serialización de datos como XML, RSS feeds, JSON, etc.
- Protección CSRF para inserción de tokens.
- Incluye documentación por medio de la aplicación administrativa.

El framework Django simula seguir la arquitectura Modelo Vista Controlador, esto se refiere a que su propio patrón es manejado por MTV (modelo, template, vista), es decir, lo que comúnmente se denomina controlador en otros frameworks en Django se llama vista y por ende las plantillas (templates) muestran los datos presentados. Los desarrolladores pueden enfocarse a construir los objetos entidades y la lógica de sus proyectos, esto debido a las capas de datos mediadora y fundación que posee Django.

5.1.2.3 Framework Laravel

Laravel es un framework libre multiplataforma que usa el lenguaje scripting para desarrollar proyectos *web* completos es decir full-stack (Backend y Frontend) con versiones soportadas en PHP 5, PHP 7 y PHP 8. Actualmente se puede trabajar con Laravel 9 que permite escribir sintaxis (código) de forma

elegante y simple de entender. Fue creado por Taylor Otwel en el año 2011 y viene integrado con herramientas referentes a los frameworks Codeigniter, Ruby on Rails, Yii, Sinatra, entre otros.

Características

Laravel tiene una destacada organización de directorios, que lo hace ideal, para realizar proyectos pequeños o grandes de alta eficiencia, ya que aporta estructura y rendimiento durante el desarrollo de los mismos, para servir como un estándar o guía consistente, que agiliza todos los procesos y necesidades en diferentes proyectos. Este framework también ofrece la autenticación y autorización, la interfaz de comandos, la biblioteca orientada a objetos, entre muchas más [43]. Algunas otras características importantes que incluye, son:

- Queues para trabajos en segundo plano.
- Motor de plantillas PHP y Blade.
- Herramienta Vite para la compilación y actualización.
- Soporte integrado para sistemas de caché.
- Patrón arquitectónico MVC.
- Mapeador Objeto-Relacional Eloquent (ORM).
- Middleware para filtrar peticiones.
- Paquetes Collective.
- WebSockets para los usuarios.
- Base de datos y las migraciones.
- Utiliza Composer y funcionalidades de Symfony.
- Aplica y establece las PSR e PSR.

El patrón arquitectónico de software MVC, es implementado por Laravel, base a esto facilita la independencia que incluye la lógica de negocio y entrada, así como la relación entre los componentes y la presentación de la interfaz de usuario (GUI) en una aplicación *web* [38].

5.1.3 Planificación del proceso de pruebas

El proceso de pruebas de software es conformado por actividades y tareas. Para la correcta realización de las mismas se desarrolla la planificación y organización de un plan de pruebas main que funciona como ejemplar de pruebas maestro para gestionar dicho proceso. Por lo cual, se consulta el programa de estudio de nivel básico v2018 de la organización ISTQB [31] e información complementaria de la norma ISO 29119-2:2021 [44] y otro sitio ilustrativo [45].

5.1.3.1 Plan de pruebas maestro

El propósito del plan de pruebas tiene como función llevar a cabo la gestión de las estrategias que se utilizarán para proporcionar la información y el marco requerido en la planificación y desarrollo de las actividades del proceso de pruebas. El alcance de esta planificación se enfoca en ejecutar pruebas de rendimiento y fiabilidad (no funcionales) en los frameworks backend Spring Boot, Django y Laravel.

Objetivo

El objetivo general del plan de pruebas es establecer la cronología y condiciones de prueba en los backend (API Rest) Spring Boot, Django y Laravel. Con el propósito de analizar y realizar una comparación de los resultados obtenidos en dichas pruebas que se emplean en esta investigación.

Integración y coordinación del proceso de pruebas en el ciclo de vida de desarrollo de software

Para el proceso de desarrollo de software (SDLC), es decir los sistemas que se emplearán como entorno de prueba en cada framework backend establecido; se tomaron en cuenta el objetivo específico, objetivos generales y

las características primordiales que figura esta investigación. Por consiguiente, se inició con una revisión bibliográfica, en donde se reconocieron diferentes modelos, como lo son: Spiral, Kanban, Scrum, Cascada, XP (Extreme Programming), por mencionar algunos. No obstante, conforme a los criterios de este proyecto, predominaron el modelo iterativo-incremental y cascada, por lo que, se recurrió a apoyarse en una comparativa. La Tabla 8, muestra los pros y contras de los modelos en cascada e iterativo-incremental.

Tabla 8. Pros y contras de los modelos en cascada e iterativo-incremental [44].

Modelo	Características a favor	Características en contra
Cascada	• Simple de entender e implementar. • La administración del proyecto es más sencilla, porque cada etapa es claramente definida y es rígida. • Comúnmente utilizado y más antiguo. • Las etapas de desarrollo son ejecutadas una a una de manera secuencial. • Útil para pequeños proyectos o de mediano tamaño con requerimientos bien definidos y sin ambigüedad. • Permite la clasificación y priorización de tareas. • Fácil identificación de los puntos clave del ciclo de desarrollo.	• El software solo estará disponible al final de la última etapa del ciclo. • Los stakeholders no pueden ver el avance del producto a lo largo de su desarrollo. • Los riesgos son altos e inciertos. • Necesidad de documentación permanente. • No se recomienda para proyectos a largo plazo, no permite flexibilización. • Dificultad para corregir errores de peso en un corto tiempo. • Se debe contar con todos los requerimientos de antemano. • El progreso de cada etapa es difícil de medir mientras se está en el proceso de desarrollo.
	• El proyecto puede arrancar con un conjunto de requerimientos	• Requiere mayores recursos en el proceso de cascada.

Modelo	Características a favor	Características en contra
Modelo iterativo e incremental	bien definidos, no exige la totalidad de los requerimientos. • El proceso es repetitivo por lo que permite crear nuevas versiones del producto en cada iteración (con una duración de 2 a 6 semanas). • Cada iteración puede incluir el desarrollo de componentes independientes que pueden ser integrados al producto de desarrollo. • Mejor manejo de riesgos. • Problemas o nuevos riesgos identificados en una iteración pueden ser manejados en la siguiente iteración. • Desarrollo en paralelo. • El progreso del proceso es medible.	• No se recomienda en proyectos pequeños. • Exige una constante administración del proyecto. • Pueden surgir inconvenientes con la arquitectura o el diseño pues no se prevén todos los requerimientos en la etapa de planeación. • Surgen nuevos riesgos en el plan de gestión del proyecto por exceso de cambios. • Los riesgos requieren una participación permanente por parte del equipo multidisciplinar especificado.

Dadas las características a favor y en contra de los modelos y la finalidad del proyecto de investigación, se optó por emplear el modelo Iterativo e Incremental.

Asimismo, la integración del proceso de pruebas de software (actividades y tareas) dentro de las fases del desarrollo del software y el modelo iterativo e incremental en su respectiva construcción se ilustran en la Figura 5, que interpreta la coordinación del modelo iterativo-incremental en las etapas de la creación de software y el grupo de actividades del proceso de pruebas.

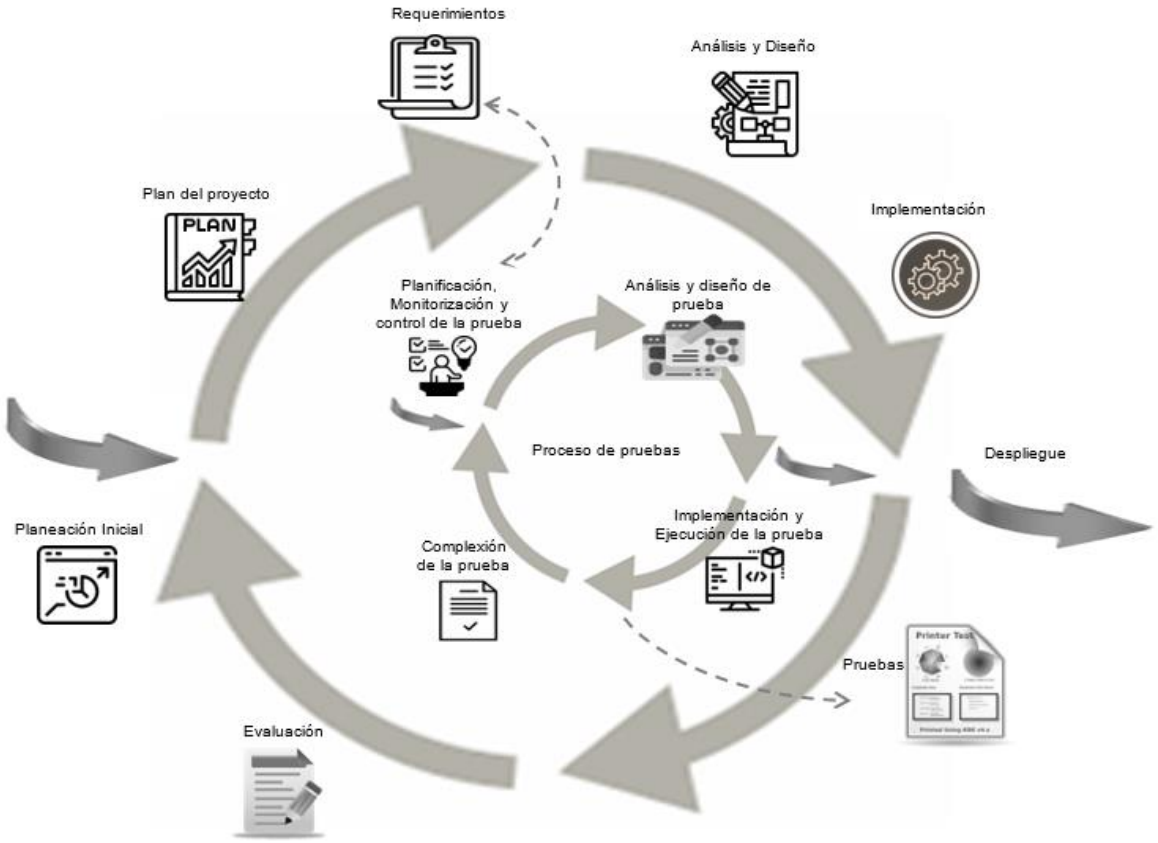


Figura 5. Coordinación del desarrollo de software y el proceso de pruebas.

El funcionamiento del modelo iterativo e incremental consta en la planificación de “N” iteraciones. Cada iteración reitera un proceso de trabajo semejante y fragmentado, que se va incrementando añadiendo nuevos requerimientos que van mejorando el producto para proporcionar un resultado completo hasta aumentar la perfección del producto final.

Ambiente de trabajo

El sistema es una aplicación *web* de tipo red social, que simula parte del entorno y comportamiento básico de las redes sociales más concurrentes en

actualidad. Se ejecuta en un servidor local y maneja una arquitectura en capas. Para el diseño y desarrollo de la capa del frontend se utiliza el framework React JS para construir las interfaces de usuario y el marco Tailwind CSS. Mientras que, para la capa del backend se utilizan los frameworks Spring Boot, Django y Laravel, que funcionan individualmente como un API REST y su comunicación se basa en ser un puente entre la interfaz de usuario (UI) y una base de datos relacional gestionada por MongoDB.

Estrategia general de la prueba y subprocesos de prueba

Puesto que esta investigación está enfocada en el análisis y comparación de frameworks backend se hace referencia a la fundación ISTQB. Esta organización a su vez se apoya y hace mención en sus distintas versiones, a los estándares ISO/IEC/IEEE 29119, ISO/IEC/25010, ISO/IEC/IEEE 20246, entre otros, para la correcta guía y especificación de las pruebas, procesos de prueba, técnicas, planes, etc.

Las pruebas se constituirán conforme a estrategias basadas en modelos con técnicas de caja negra ya que se enfoca principalmente en características y requerimientos no funcionales. En la Tabla 9, se indican las principales pruebas que se implementaran en los backend (API REST) previamente desarrollados, dichas pruebas son alusión a las características y subcaracterística descritas en el esquema de aprendizaje y formación en su nivel avanzado para el estudio de pruebas técnicas V4.0 de ISTQB [46]. A su vez, también se encuentran descritas en el documento especificación de requisitos como requerimientos no funcionales.

Tabla 9. Pruebas a implementar.

Categoría de la Prueba	**Prueba**	**Definición**
Rendimiento	*Carga*	Evaluación de un sistema bajo volúmenes de cargas de trabajo determinadas, para conocer y

Categoría de la Prueba	**Prueba**	**Definición**
		analizar el comportamiento que puede ser soportado por el mismo.
	Comportamiento Temporal	Es una valoración que se puede expresar en medidas de tiempo, es decir la duración en que un sistema responde y procesa las tareas o funciones en condiciones previamente definidas.
	Utilización de recurso	Hace referencia a la utilización, cantidad y tipo de recurso que se está empleando mientras se ejecutan funciones específicas.
	Estrés	Procedente de la prueba carga que llevan a un sistema, bajo límites máximos para ver el comportamiento de la presión y el punto de ruptura de un software.
Fiabilidad	*Capacidad de recuperación*	Se refiere a la medición de la suficiencia en un sistema para continuar y recuperar su integridad después de un presentar un fallo, puede ser en términos de software o hardware y respecto a varios factores, como lo son: la caída de red, el tiempo de reinicio, la cantidad de datos perdidos y la restauración de los mismos, etc.

Como muchas otras pruebas no funcionales, las pruebas de rendimiento y fiabilidad, son comúnmente aplicadas a servidores, sitios *web*, aplicaciones *web*, bases de datos, API, etc.

Conjuntos de planes y documentación

Es común y recomendable que la preparación del tipo de pruebas especificadas anteriormente, incluya su propio plan de pruebas. La Tabla 10

muestra los conjuntos de planes de prueba y la documentación relacionada con el plan de pruebas maestro, estos planes se incluyen posteriormente.

Tabla 10. Conjuntos de planes de prueba y la documentación.

Etapa	Nombre
Desarrollo de software	Especificación de requisitos
Pruebas (Plan de pruebas maestro)	Plan de carga
	Plan de comportamiento Temporal
	Plan de utilización de recursos
	Plan de estrés
	Plan de capacidad de recuperación

Personas y recursos requeridos

La Tabla 11 muestra los recursos requeridos que se estarán utilizando para la realización de esta investigación.

Tabla 11. Recursos requeridos.

	Cantidad	Nombre	Descripción	
Humanos	1	Tester	Analizador, planificador y probador de las pruebas de software.	
Software	1	Apache JMeter v5.6	Herramienta de código abierto y gratuita, con gran variedad de servicios para la evaluación de pruebas en software.	
	1	OpenJDK17	Implementación de kit o plataforma Java de desarrollo libre (herramientas para crear aplicaciones).	
			Sistema Operativo	Windows 10 Enterprise 64 bits v22H2
			Disco Duro	930.27 GB NTFS

	Cantidad	Nombre	Descripción	
Hardware	1	Computador	Procesador	AMD A4-9125 Radeon R3, 4 Compute Cores 2C+2G 2.30 GHz
			RAM	8.00 GB (8192MB)

Cronograma de actividades

Con relación a la fluidez, cronometría de las tareas y el programa del proyecto, se establece un calendario detallado que indica los procesos, actividades y plazos asignados en cada iteración. Así pues, se implican tanto el ciclo de desarrollo de software (no se incluyen las fases Instalación/Despliegue y Uso/Mantenimiento ya que no es un entregable dentro del proyecto), el proceso de pruebas y la comparación de los resultados concluidos en las pruebas.

Concretando un total de seis iteraciones en el proyecto, los propósitos de las mismas se describen de la siguiente manera:

- Iteración 1: Gestión del proyecto
- Iteración 2: Implementación del software
- Iteración 3: Implementación y ejecución de pruebas del framework backend Spring Boot.
- Iteración 4: Implementación y ejecución de pruebas del framework backend Django.
- Iteración 5: Implementación y ejecución de pruebas del framework backend Laravel.
- Iteración 6: Compleción de pruebas, análisis y comparación de resultados.

La Tabla 12 muestra el cronograma de actividades que se empleó a lo largo del proyecto.

Tabla 12. Cronograma de actividades.

	Junio – Diciembre 2023	Enero – Junio 2024
	Iteración 1	
Gestión del proyecto	Junio 15 - Junio 20 **Planificación inicial (Enunciado de trabajo)** Junio 20 - Julio 10 **Plan del proyecto** Julio 11 - Julio 12 **Plan revisado** Julio 13 - Agosto 20 **Especificación de requisitos** Agosto 21 - Septiembre 20 **Diseño de software**	
Implementación de software	**Iteración 2**	
	Septiembre 21 - Octubre 20 **Implementación de software framework Spring Boot** Octubre 21 - Noviembre 20 **Implementación de software framework Django** Noviembre 21 - Diciembre 20 **Implementación de software framework Laravel** Diciembre 21 - Enero 20 **Planificación, Monitorización y control de la prueba (Conjuntos de planes de pruebas)** Enero 21 - Febrero 01 **Análisis y diseño de pruebas**	
Proceso de pruebas	**Iteración 3**	
	Febrero 02- Marzo 10 **Implementación y ejecución de la prueba framewor Spring Boot**	

	Junio – Diciembre 2023	Enero – Junio 2024
	Iteración 4	
	Marzo 11 - Abril 20 **Implementación y ejecución de la prueba framewor Django**	
	Iteración 5	
	Abril 21 - Mayo 31 **Implementación y ejecución de la prueba framewor Laravel**	
Análisis y comparación de resultados	**Iteración 6**	
	Junio 01 - Junio 5 **Compleción de la prueba frameworks Spring Boot, Django y Laravel**	
	Febrero 02 - Mayo 31 **Análisis y comparación de resultados**	

Riesgos

Para llevar un buen progreso del proyecto, se registran los eventos que son considerados como riesgos, ya sea que se presentan a futuro o en el transcurso de la investigación, afectando el avance y los objetivos de la misma. En particular, se hace relación a la tabla de riesgos que emplea PMI (*Managing project and analyzing risks*), donde se analiza la probabilidad y las consecuencias en función de los riesgos por niveles, la Tabla 13 indica la matriz de riesgos.

Tabla 13. Matriz de riesgos [46].

	Consecuencias				
Probabilidad	Insignificante 1	Menor 2	Moderada 3	Mayor 4	Catastrófica 5
Raro 1	Bajo	Bajo	Moderado	Alto	Alto
Improbable 2	Bajo	Bajo	Moderado	Alto	Extremo
Posible 3	Bajo	Moderado	Alto	Extremo	Extremo
Probable 4	Moderado	Alto	Alto	Extremo	Extremo
Casi seguro 5	Alto	Alto	Extremo	Extremo	Extremo

Extremo:	Los riesgos extremos deben ponerse en conocimiento de los Directores y ser objeto de seguimiento permanente.
Alto:	Los riesgos altos requieren la atención del Presidente / Director General / Director Ejecutivo.
Moderado:	Los riesgos moderados deben ser objeto de seguimiento adecuado por parte de los niveles medios de Dirección.
Bajo:	Los riesgos bajos deben ser objeto de seguimiento por parte de los supervisores.

La Tabla 14 muestra los riesgos identificados en el proyecto.

Tabla 14. Riesgos.

No	Riesgos	Estado	Probabilidad	Consecuencia	Severidad	Actividad de prevención	Actividad de erradicación
1	Recursos de hardware y software insuficientes para la implementación de las pruebas	Latente	Probable	Moderada	Alto	Optimizar los recursos del computador antes de la ejecución de las pruebas	Buscar alternativas de hardware y software que soporten los requerimientos de las pruebas
2	Retrasos en la implementación de las funcionalidades.	Latente	Improbable	Menor	Bajo	Llevar un seguimiento disciplinario y constante en la implementación del software	Evaluar el avance del desarrollo de las funcionalidades y re-planificar acorde al avance de ser necesario
3	Prolongación en la ejecución de alguna prueba por contrariedades en el software	Latente	Improbable	Menor	Bajo	Ejecución previa de scripts para pruebas APIs automatizadas en Postman.	Tomar cursos o asesorías que ayude a la correcta realización de las pruebas

Suposiciones, restricciones, datos y referencias de estudio

- Solo se implementarán las pruebas especificadas anteriormente y en los backend (API REST) desarrollados en los frameworks Spring Boot, Django y Laravel.
- Para estos casos de estudio no se cuenta con datos reales y referentes al ambiente de producción que podría tener la aplicación *web*. Debido a esta razón, se pretende usar datos relativos y referentes a medios sociales, por lo cual se realizó una revisión de literatura sobre el tráfico *web* y estadísticas globales de redes sociales.

Entregables de prueba

- Plan de prueba
- Casos de pruebas
- Informe de resúmenes de las pruebas (reporte)

Criterios de finalización de las pruebas

Deben ejecutarse en cada uno de los frameworks seleccionados todas las pruebas mencionadas anteriormente, sin presentar falla alguna (error en los escuchadores Apache JMeter).

5.2 Desarrollo de APIs REST y pruebas de software

5.2.1 Componentes del software

El contexto experimental radicó en la construcción de una aplicación *web* de tipo red social básica, que se componía e imitaba algunos aspectos más comunes y distintivos de diversas redes sociales. Esto debido a que los programas de redes sociales son sumamente populares y usadas por los usuarios. Según Digital 2023: Informe General Global, hoy en día, se contempla un aproximado de 4.760 millones de consumidores de plataformas sociales en todo el mundo, esto corresponde a una diferencia menor del 60%, del poblamiento total del mundo [47].

Como gestor de base de datos se utilizó MongoDB, debido a que es una base de datos NoSQL, que almacena datos como documentos y es adecuado para manejar grandes cantidades de información. El modelado de la base de datos se conformó por colecciones con características sencillas y referentes a variadas redes sociales, como lo son Facebook [48], Instagram [49, 50], Pinterest [51], entre otros. La Figura 6, muestra el diseño y las colecciones de la base de datos, constituida por las entidades de los usuarios, eventos, intereses, post; comentarios y likes que se conceden a los posts.

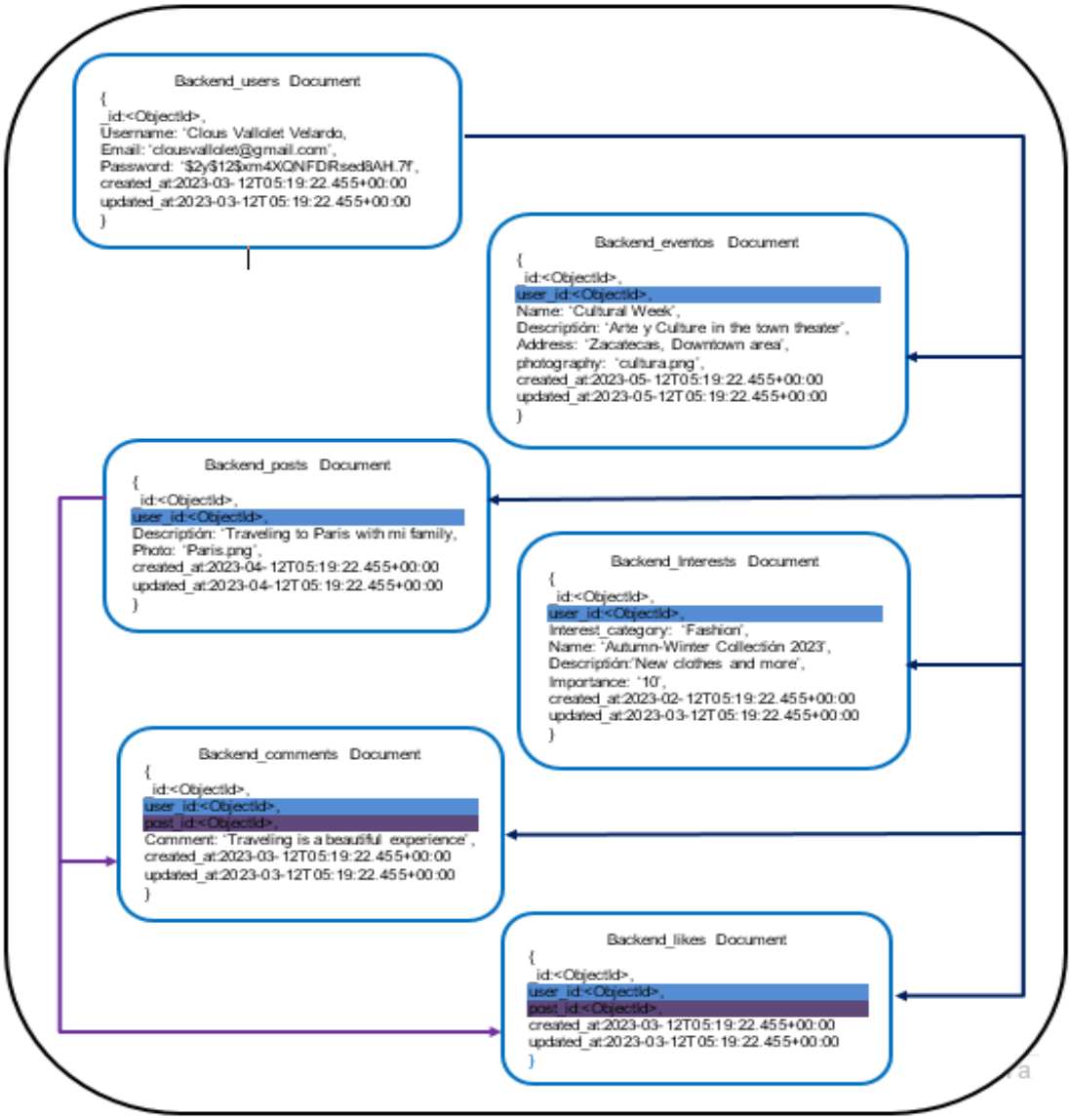


Figura 6. Diseño y colecciones de la base de datos.

El flujo de trabajo en la aplicación *web* consistió en las siguientes capas:

- *Capa Frontend:* se empleó ReactJS para el desarrollo del lado del cliente y para dar forma a la parte frontal de las aplicaciones. Se utilizará el mismo frontend para los tres distintos frameworks backend.

ReactJS es una biblioteca de JavaScript y de código abierto, diseñada para el desarrollo de interfaces de usuarios en sitios, plataformas, aplicaciones *web* y móviles. Fundada por Meta, ReactJS integra una estructura de elementos y fragmentos robustos en código JavaScript XML que es lógico, auto contenido, reutilizables y usados para describir la construcción de interfaces (UI) interactivas y flexibles, que se basan en componentes con propiedades enfocados en el diseño de dichas interfaces [52].

React Js tiene varios objetivos, como lo son crear páginas *web* con mejor rendimiento y reducir los fallos que suceden cuando los programadores crean interfaces de usuario con el propósito de ahorrar trabajo y tiempo [53].

- *Capa Backend*: (API REST individuales) se desarrollará en Spring Boot, Django y Laravel, para procesar toda la información que alimentara al frontend.

Posteriormente, se muestra el marco de la arquitectura de software que se empleó en el sistema. La Figura 7, muestra la descripción de la arquitectura en la aplicación *web*. Se compone de varios niveles jerárquicos que indican la construcción del sistema:

- El primer nivel denota los usuarios que interactúan con la aplicación y sus acciones.
- En el segundo nivel se constituyen los componentes y el contenido estático JSX, así como el proceso de la información que se envía desde las peticiones a la vista.
- El tercer nivel marca la comunicación de las operaciones (HTTP) como cliente a cada uno de los controladores (MVC) de las API REST (Spring Boot, Django y Laravel) por medio de la librería Axios.
- El último nivel señala la comunicación y conexión de cada controlador, para acceder a la base de datos MongoDB. Cada API REST (backend) se comunica a la misma base de datos y usan el puerto 8000, pero se ejecutan en distintos tiempos.

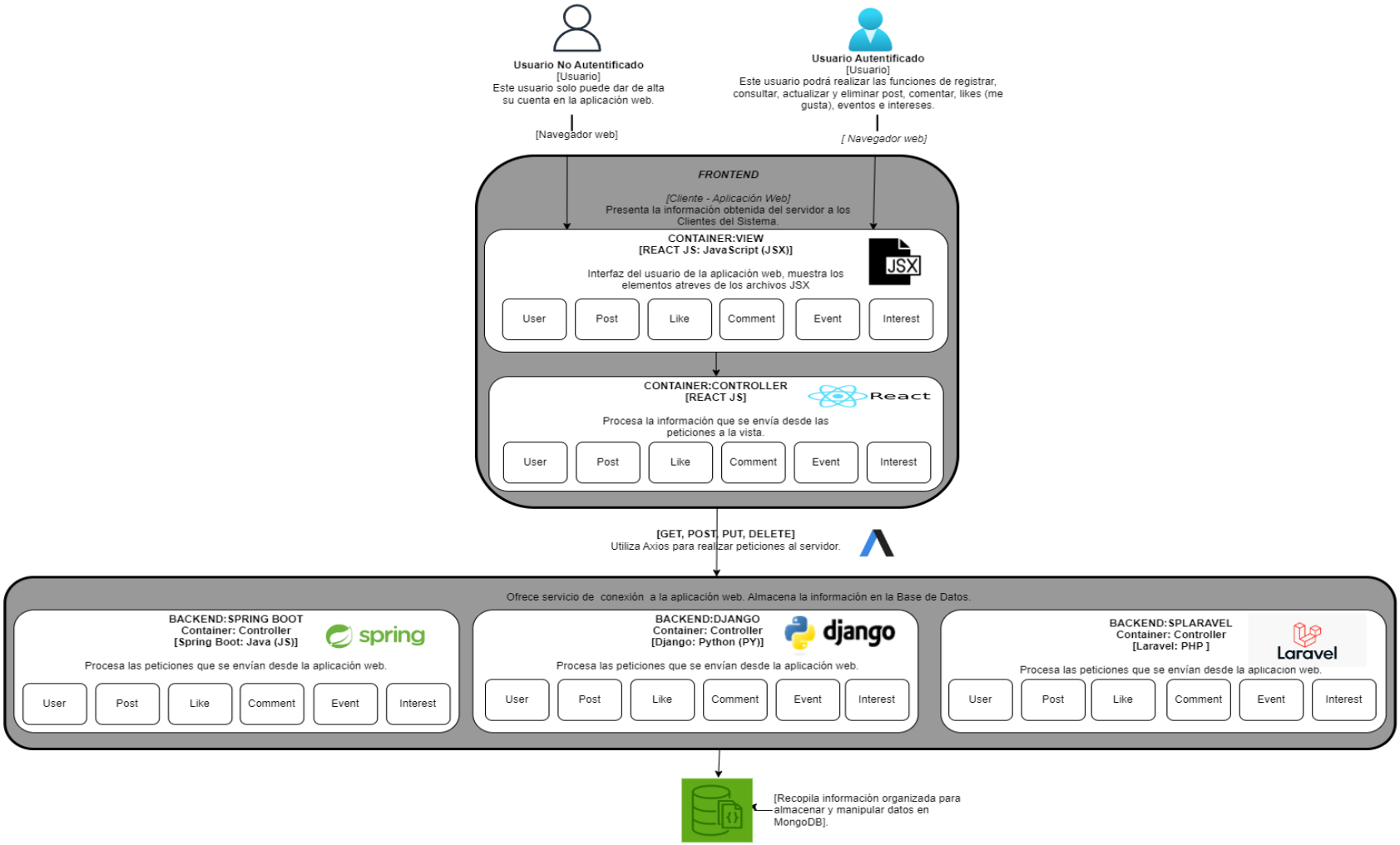


Figura 7. Descripción de la arquitectura en la aplicación *web*

Respecto a las versiones de las tecnologías, en la Tabla 15 se muestran las tecnologías empleadas para la construcción del software, es decir los backend (API REST) y el frontend en la aplicación web.

Tabla 15. Tecnologías empleadas para la construcción del software.

Software	Versión
Django	v4.2.5
Python	v3.11.5
Laravel	v8.2
PHP	v8.2.11
Spring Boot	v3.1.0
OpenJDK	v17.0.7
ReactJs	v 18.1
MongoDB	v6.0.2

5.2.2 Análisis y diseño de las pruebas

Teniendo presente las características que engloban la investigación se determinar principalmente el fundamento de porque se eligieron las pruebas a realizar, con forme a los siguientes criterios:

a) La investigación tiene como prioridad el análisis y comparación de frameworks backend para evaluar su desempeño y guiar al programador en la elección del más adecuado.
b) Los principales servicios *web* desarrollados son APIs REST. Por lo cual, se implementan pruebas automatizadas que se aplican del lado del servidor.
c) Las pruebas de carga, comportamiento temporal, utilización de recursos, estrés y capacidad de recuperación apuntan directamente a los servicios e infraestructura que proporciona el backend.

d) Las pruebas seleccionadas tienen como finalidad la obtención de datos e información sobre las peticiones al backend directo al servidor y la base de datos.

Para el análisis y diseño de las pruebas se revisaron y analizaron varios datos, documentos, necesidades y otros aspectos implicados en el proyecto.

Por consiguiente, se pretende que los distintos frameworks backend sean probados bajo las mismas condiciones, dado esto, se realiza el acondicionamiento para los planes por tipo de prueba individuales, que se mencionan y se incluyen como conjuntos de planes de prueba en el plan de pruebas maestro.

Conforme a los siguientes planes se identificaron y definieron las condiciones de prueba para diseñar los casos de prueba priorizados en secciones de descripciones generales ya que se aplicarán los mismos términos y procedimientos por tipo de prueba. Además, se incluyen los requerimientos no funcionales elaborados en la plataforma Trello, para cada una de los distintos casos de prueba.

Inicio de la prueba de carga

La Figura 8 muestra el requerimiento no funcional RNF_01 de carga.

Figura 8. Requerimiento no funcional RNF_01 Carga.

5.2.2.1 Prueba de carga

Planificación de la prueba

La finalidad de la prueba de carga es identificar y observar el comportamiento durante cargas elevadas que puede manejar una aplicación *web* y el tiempo de respuesta que se obtiene.

Objetivo

- Verificar el comportamiento al realizar pruebas de carga sobre los frameworks backend: Spring Boot, Django y Laravel. Para analizar las respuestas y el rendimiento de cada uno de los entornos durante la ejecución de solicitudes pesadas.

Escenarios de prueba

Para este caso de prueba, se propuso predecir una cantidad de usuarios activos proporcional al volumen en producción que podría tener en una red social básica. Como primer parámetro se fija un “Baseline” es decir, el menor número posible de participantes, que podrían oscilar entre 1 y 5 usuarios, mientras los usuarios concurrentes podrían ir incrementando de 10 en 10. Para el manejo del tiempo, según las buenas prácticas del monitoreo de aplicaciones *web* y estadísticas de los datos y uso de redes sociales, es concurrente que la frecuencia de monitoreo sea cada minuto [54, 55]. Por lo cual, la duración irá conforme al rango y la unidad de tiempo estará basada en segundos.

El contexto de la prueba, se basa en ser una prueba de carga clásica que irá incrementando proporcionalmente el tiempo de periodo de subida, fija durante el periodo estable y disminuyendo a razón constante con el periodo de decremento, utilizará el componente Grupo de subprocesos definitivo, que iniciará con un recuento de hilos de 1, 10 y 20 usuarios por fila. Cada hilo tendrá un retraso inicial de 0, 2 y 5 segundos, con un tiempo uniforme tanto de inicio como de apagado de 10, 30 segundos consecutivamente y un periodo estable de 40, 30 y 20 segundos. Es decir, la prueba durará 1 minuto y 32 segundos con periodos de 30 segundos. La Tabla 16 muestra el script de automatización de la prueba de carga.

Tabla 16. Script de automatización de la prueba de carga.

No.	Descripción
1	Iniciar un plan de pruebas en JMeter
2	Añadir un Grupo de hilos definitivo para indicar el número de peticiones que estarán interactuando entre JMeter y el servidor. Con el nombre: Prueba Carga “Nombre del Framework” a probar.
3	Indicar el Inicio de Recuento de hilos, Retraso inicial, Tiempo de inicio, Carga estable y el Tiempo de apagado según lo especificado en los datos del caso de prueba.
4	Añadir un Http Request (Escenario 1- Login) y configurar los siguientes campos: Protocolo: http, IP del servidor: 127.0.0.1 o localhost y el número de puerto: 8000 Seleccionar el método a usar: POST y el Path:/api/signin/ En donde el módulo para la ejecución de este caso de prueba será Backend_users (inicio de sesión).
5	Añadir un Http Request (Escenario 2- Crear Interés) y configurar los siguientes campos: Protocolo: http, IP del servidor: 127.0.0.1 o localhost y el Numero de Puerto: 8000 Seleccionar el método a usar: POST y el Path:/api/interests/ En donde el módulo para la ejecución de este caso de prueba será Backend_interests
6	Agregar un HTTP Header Manager con el campo: - Content-Type: application/json
7	Incorporar un elemento de configuración de tipo CSV Data Set Config para que sea alimente del archivo “dataload_csv.txt” que contiene información en la colección usuarios registrados en la base de datos, correspondientes a los campos (email, password y user_id) en la colección de backend_users y backend_interests.
8	Para los parámetros de la colección backend_users, se agregaron los siguientes valores: - email ${email} - password ${password}
9	Para los parámetros de la colección backend_interests, se agregaron los siguientes valores: - user_id ${user_id} - interest_category ${__RandomString(6, abcdefghijklmnopqrstuvxwz)} - name ${__RandomString(8, abcdefghijklmnopqrstuvxwz)} - description ${__RandomString(15, abcdefghijklmnopqrstuvxwz)}

No.	Descripción
	- importance ${__RandomString(1,1234567890)}
10	Seleccionar en el Controlador objetivo el grupo de hilos con el nombre de la prueba para añadir los escuchadores o receptores: o RTG- Gráfica de tiempo de respuesta o TPS-Transacciones por segundo.
11	Las indicaciones para las pruebas de carga por framework backend son semejantes.

Sección de descripción general del diseño que correspondió a los casos de prueba CP01, CP02 y CP03 (carga), como se muestra en la Tabla 17.

Tabla 17. Diseño de los casos de prueba CP01, CP02 y CP03.

Nombre: Rendimiento – Carga	
Fecha de Revisión:	**Prioridad:** Alta **Revisor:** Berenice Ortiz Torres
Información sobre el caso de prueba	
Descripción: Esta prueba consta del lanzamiento de cargas concurrentes de hilos virtuales, que van aumentando, bajando y estableciéndose en distintos periodos, para examinar el proceder del sistema bajo determinadas condiciones.	**Prerrequisitos:** Seguir las instrucciones descritas en el script de automatización de la prueba de carga.
Datos de entrada	
**Componente de Apache JMeter	
Procedimiento: A través de la herramienta Apache JMeter se indica un plan de prueba de carga que incluye grupo de hilos definitivo, para la configuración del recuento de hilos, el lapso inicial, la vertiente de subida, el tiempo de carga estable y el periodo de bajada. Esto para enviar las peticiones al servidor y sean procesadas por el controlador para la obtención o consulta de objetos con los parámetros correctos y probar la carga que soporta el sistema.	**Resultado esperado:** Se espera obtener la capacidad del sistema para tratar con diferentes cargas máximas, sobre funcionamientos en específico y la correcta ejecución de las mismas.

Inicio de la prueba de comportamiento temporal

La Figura 9 muestra el requerimiento no funcional RNF_02 Comportamiento temporal.

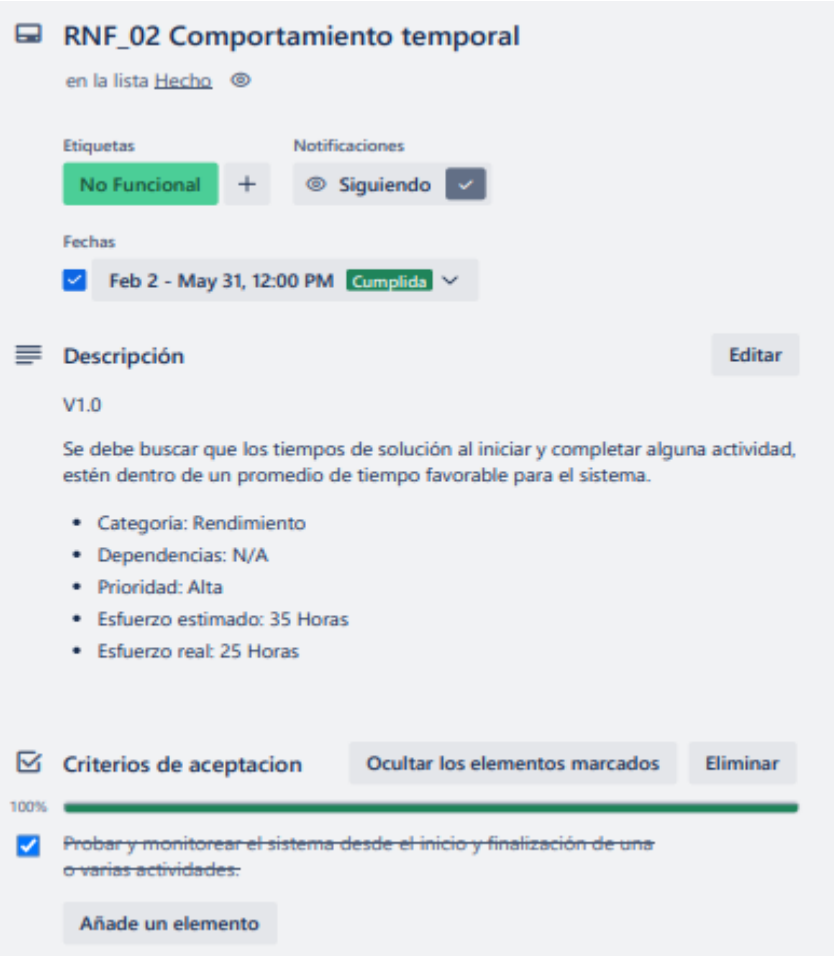


Figura 9. Requerimiento no funcional RNF_02 Comportamiento temporal.

5.2.2.2 Prueba de comportamiento temporal

Planificación de la prueba

Cuando se realizan pruebas de carga es una buena práctica realizar pruebas de comportamiento temporal y la utilización de recursos [46]. Para medir estos parámetros se realiza otra prueba de carga, derivada de la anterior, pero se utilizará un grupo de subprocesos de concurrencia como una alternativa diferente.

Escenarios de prueba

Como parámetros se tiene 20 hilos, un tiempo de aceleración de 60 segundos, con un recuento de 10 pasos de aceleración para aumentar los subprocesos y 30 segundos reteniendo el ritmo objetivo de la tasa. Al alcanzar los 20 subprocesos todos los hilos continuarán efectuándose y realizando peticiones al servidor para acceder juntos durante 30 segundos.

La Tabla 18 muestra la descripción del script de automatización para la prueba comportamiento temporal en Apache JMeter.

Tabla 18. Script de automatización para la prueba de comportamiento temporal.

No.	Descripción
1	Iniciar un plan de pruebas en JMeter
2	Añadir un Grupo de subprocesos de concurrencia para indicar el número de peticiones que estarán interactuando entre JMeter y el servidor. Con el nombre: Prueba Comportamiento Temporal “Nombre del Framework” a probar.
3	Seleccionar la unidad de tiempo en segundos e indicar los valores de los parámetros: Simultaneidad objetivo, Tiempo de aceleración, Pasos de aceleración y Tiempo de tasa objetivo según lo especificado en los datos del caso de prueba.
4	Marcar la opción “Detener la prueba ahora”, para interrumpir cualquier muestreador si se presenta un error y agregar la ruta de un archivo de registro para guardar los estados de los hilos.
5	Añadir dos Http Request (Escenario 1- Obtener Events) y (Escenario 2- Obtener Intereses) para configurar los siguientes campos: Protocolo: http, IP del servidor: 127.0.0.1 o localhost y el número de Puerto: 8000 Seleccionar el método a usar: GET, el Path:/api/interests/ y Path:/api/events/ según sea el escenario. En donde los módulo para la ejecución de este caso de prueba será Backend_Interests y Backend_Events.
6	Seleccionar en el Controlador objetivo el grupo de hilos con el nombre de la prueba para añadir los escuchadores o receptores: o ATOT - Hilos activos. o RTOT - Tiempo de Respuesta. o SR - Informe resumido.
7	Para medir la utilización de recursos incorporar el receptor:

No.	Descripción
	○ PMC - Recopilador de métricas.
8	CPU, Memoria y el Disco. Con las configuraciones por defecto. Las configuraciones para las pruebas de comportamiento temporal en cada uno de los frameworks backend son semejantes.

Sección de descripción general del diseño en los casos de prueba CP04, CP05 y CP06 (Comportamiento temporal), como se muestra en la Tabla 19.

Tabla 19. Diseño de los casos de prueba CP04, CP05 y CP06.

Nombre: Rendimiento – Comportamiento Temporal	
Fecha de Revisión:	**Prioridad:** Alta **Revisor:** Berenice Ortiz Torres
Información sobre el caso de prueba	
Descripción: Procedente de la prueba de carga, esta prueba consta en la monitorear el comportamiento del sistema desde que se inicia una transacción o actividad, hasta el momento en que se termina o completa, durante la ejecución de cargas elevadas de usuarios virtuales.	**Prerrequisitos:** Seguir las instrucciones descritas para la prueba de comportamiento temporal en el script de automatización.
Datos de entrada	
**Componente Apache JMeter	
Procedimiento: A través de la herramienta Apache JMeter se incluyen los complementos: Hilos activos y un informe resumido, para obtener el tiempo de respuesta y de procesamiento.	**Resultado esperado:** Se espera la correcta ejecución de las actividades en condiciones de funcionamiento específicas para obtener el tiempo del comportamiento en los diferentes procesos.

Inicio de la prueba de utilización de recursos

La Figura 10 muestra el requerimiento no funcional RNF_03 Utilización de recursos.

Figura 10. Requerimiento no funcional RNF_03 Utilización de recursos.

5.2.2.3 Prueba de utilización de recursos

Planificación de la prueba

La prueba de utilización de recursos es una prueba dependiente, por lo su procedimiento de ejecución se indica en el inicio en la script de automatización de la prueba de comportamiento temporal y continua la configuración en este script propio.

La Tabla 20 muestra la descripción del script de automatización para la prueba utilización de recursos en Apache JMeter.

Tabla 20. Script de automatización para la prueba de utilización de recursos.

No.	Descripción
1	Dentro del plan de pruebas de comportamiento en JMeter
2	Incorporar un recopilador o receptor de métricas para medir la utilización de recursos: CPU, Memoria y el Disco. o PMC-Recopilador de métricas
3	Agregar la dirección HOST/IP del telnet que indica el computador en cada una de las métricas
4	Descargar el complemento ServerAgent para escuchas por TCP y UDCP. Después iniciar el agente servidor Shell Script ServerAgent
5	Se inicializará la terminal mostrando el puerto
6	Al comenzar la prueba se registraran los resultados monitoreados
7	Las configuraciones para las pruebas de utilización de recursos en cada uno de los frameworks backend son semejantes.

Sección de descripción general del diseño en los casos de prueba CP07, CP08 y CP09 (Utilización de recursos), como se muestra en la Tabla 21.

Tabla 21. Diseño de los casos de prueba CP07, CP08 y CP09.

Nombre: Rendimiento – Utilización de recursos	
Fecha de Revisión:	**Prioridad:** Alta **Revisor:** Berenice Ortiz Torres
Información sobre el caso de prueba	
Descripción: Procedente de la prueba de comportamiento temporal, esta prueba consiste en medir la utilización de la CPU, el uso de memoria y el disco.	**Prerrequisitos:** Seguir las instrucciones descritas para la prueba de utilización de recursos en el script de automatización de la prueba de comportamiento temporal.
Datos de entrada	
** Dependencia (ID de los casos de prueba)	
Procedimiento: A través de la herramienta Apache JMeter se incluye el complemento: Recopilador de métricas para obtener los valores de uso, de los recursos del computador.	**Resultado esperado:** Se espera obtener el tiempo de utilización de recursos del sistema.

Inicio de la prueba de estrés

La Figura 11 muestra el requerimiento no funcional RNF_04 Estrés.

Figura 11. Requerimiento no funcional RNF_04 Estrés.

5.2.2.4 Prueba de estrés

Planificación de la prueba

La determinación de la planificación de las pruebas de estrés es describir y realizar la implementación de dichas pruebas en los diversos frameworks backend, con el fin de proporcionar y gestionar un registro del avance de las pruebas y los resultados obtenidos.

Objetivo

- Valorar el rendimiento de los frameworks backend Spring, Django y Laravel, para examinar cómo se comporta cada uno de los sistemas, bajo cargas intensas de trabajo.

Crear scripts de automatización de la prueba

Para este caso se tomará como referencias ejemplares de pruebas de estrés [56] y datos de tráfico que podría tener una *web* para ser rentable.

Dado que no se proporciona un número exacto de visitas a un determinado sitio *web*, varía según la industria, las estrategias de posición, marketing y el nicho, etc. Sin embargo, el número general está entre un rango de 1000 a 50,000 visitantes para empezar a obtener retribuciones [57, 58, 59].

De igual manera, pese a que cada red social presenta un tráfico distinto, se considera las horas que un usuario clásico pasa activamente usando distintas redes sociales, por lo que se estima un promedio de 2 horas y 23 minutos por día [54]. Según distintos blogs de información, se interpreta de manera general y consistente que los mejores días para publicar o donde hay más usuarios ingresando a una red social son los días lunes, miércoles y jueves [60, 61, 62].

Finalmente, para estimar qué cantidad de usuarios que se podrían conectar por día, se obtendrá el número intermedio entre 1000 y 50000 usuarios.

$$\text{Núm. } Intermedio \ \ o \ Promedio = \frac{1{,}000 + 50{,}000 \ (Usuarios)}{2} \ = \ 25{,}500 \ Usuarios$$

Entonces, durante los tres días donde hay más usuarios ingresando a una red, hay un total de 7 horas y 9 minutos. Por lo que se interpreta, que serían: 25,500 usuarios conectados en un lapso de 429 minutos aproximadamente.

Teniendo en cuenta el dato anterior, se utiliza la proporcionalidad directa para saber cuántos usuarios habría por minuto:

$$x = \frac{25{,}500 \ (\text{Visitantes})}{429 \ (\text{Minutos})} = \frac{59.44055}{1.0000} \ en \ donde \ x = \frac{59 \ (Usuarios)}{1 \ (Minuto)}$$

En las pruebas de estrés se pueden contemplar tres sub-pruebas incrementales (Primaria, Intermedia y Elevada) [63, 64], simulando los usuarios o hilos conectados durante un lapso de tiempo. De acuerdo con los cálculos anteriores, se obtiene la subprueba primaria.

$$Subprueba - Primaria = \frac{59\ (Hilos)}{60\ Seg}$$

Para obtener el número de hilos en las siguientes pruebas, se considera sumar el porcentaje respectivamente al número de usuarios.

Sumar el porcentaje de 50% al valor para obtener la subprueba intermedia:

$$SPV = 59.44055 + \left(\left(\frac{50}{100}\right) * 59.44055\right) = 89.16083\ en\ donde\ SPV$$

$$= 89\ (Usuarios)$$

$$Subprueba - Intermedia = \frac{89\ (Hilos)}{60\ Seg}$$

Sumar el porcentaje de 100% al valor para obtener la subprueba elevada:

$$SPV = 59.44055 + \left(\left(\frac{100}{100}\right) * 59.44055\right) = 118.88811\ en\ donde\ SPV$$

$$= 119\ (Usuarios)$$

$$Subprueba - Elevada = \frac{119\ (Hilos)}{60\ Seg}$$

La Tabla 22 muestra la descripción del script de automatización en Apache JMeter para la prueba de estrés.

Tabla 22. Script de automatización para la prueba de estrés.

No.	Descripción
1	Iniciar un plan de pruebas en JMeter
2	Añadir un Grupo de Hilos para almacenar todas las peticiones e información que esté viajando entre JMeter y el servidor. Con el nombre: Prueba Estrés “Nombre del Framework” a probar.
3	Indicar el número de hilos, el Periodo de subida y el contador de bucle, según lo especificado los datos del caso de prueba
4	Añadir un Http Request y configurar los siguientes campos: Protocolo: http, IP del servidor: 127.0.0.1 o localhost y el número de Puerto: 8000. En donde el módulo para la ejecución de este caso de prueba será Backend_posts.
5	Seleccionar el método a usar: POST y el Path:/api/posts/ Agregar un HTTP Header Manager y se agregara: - Accept: application/json - Content-Type: application/json - Content-Encoding: utf-8
6	Incorporar un elemento de configuración de tipo CSV Data Set Config para que sea alimente del archivo “data_csv.txt” que contiene datos dinámicos, correspondientes a los campos (user_id, description) en la colección de backend_posts y marcar la casilla Use Multipart/form-data para incluir una imagen que corresponde al campo (photo) en la misma colección e indicar los parámetros de los campos con estructura de variable.
7	Seleccionar en el Controlador objetivo el grupo de hilos con el nombre de la prueba para añadir los receptores: o VRT-Árbol de resultados o AR-Informe agregado
8	Agregar un Temporizador aleatorio uniforme a nivel de las peticiones, para indicar un retraso en milisegundos por cada petición al servidor con diferentes tiempos de espera.
9	Para el valor de Máximo Retardo Aleatorio se indicará un retardo de 5000 segundos con un Desplazamiento de Retraso Constante de 0 segundos. Por defecto se utilizará la fórmula proporcionada por JMeter. En donde X puede ser entre 0 y 9 $TAU = 0.X *$ Temporizador $aleatorio\ uniforme$ $+ Desplazamiento\ de\ Retraso\ Constante$ $TAU = 0.5 *\ 5000 + 0 = 5000\ ms\ (Valor\ de\ ocurrencia\ aleatoria)$

No.	Descripción
10	Al configurar el grupo de hilos se representa como cada hilo un usuario conectado y la cantidad de procesos en un intervalo de tiempo, por lo que se consideran tres pruebas en incremento: Primaria, Intermedia y Elevada.
11	El contador de bucles es la cantidad de veces que se repitiera el proceso, este será de 143 veces para simular dos horas y veintitrés minutos por día.
12	Las configuraciones para las sub-pruebas intermedia y elevada son semejantes.

Sección de descripción general del diseño en los casos de prueba CP10, CP11 y CP12 (Estrés), como se muestra en la Tabla 23.

Tabla 23. Diseño de los casos de prueba CP10, CP11 y CP12.

Nombre: Rendimiento – Estrés	
Fecha de Revisión:	**Prioridad:** Alta **Revisor:** Berenice Ortiz Torres
Información sobre el caso de prueba	
Descripción: Esta prueba consta del lanzamiento de tres pruebas de estrés, con distintas cantidades de usuarios virtuales simultáneos que van en aumento, para examinar el proceder del sistema bajo determinadas condiciones.	**Prerrequisitos:** Seguir las instrucciones descritas en el script de automatización de la prueba de estrés.
Datos de entrada	
**Componente Apache JMeter	
Procedimiento: A través de la herramienta Apache JMeter se indica un plan de prueba que incluye un grupo de hilos virtuales, para la configuración del número de hilos, el periodo de subida y un bucle. Esto para enviar las peticiones al servidor y sean procesadas por el controlador para la creación de inserciones de carga de nuevos objetos con los parámetros correctos y probar el estrés que soporta el sistema.	**Resultado esperado:** Se espera al correr la prueba se ejecute sin errores y obtener la medida de tiempo de respuesta durante la incrementación de cargas en funciones específicas.

Inicio de la prueba de capacidad de recuperación

La Figura 12 muestra el requerimiento no funcional RNF_05 Capacidad de recuperación.

Figura 12. Requerimiento no funcional RNF_05 Capacidad de recuperación.

5.2.2.5 Prueba de capacidad de recuperación

Planificación de la prueba

Dado que la prueba de capacidad de recuperación determina las funciones y capacidad del sistema para operar y recuperarse después de una deficiencia inesperada [46]. La perspectiva de esta prueba se enfoca en registrar el tiempo que se invierte en realizar las tareas: copia de seguridad (Backup) y la restauración de los datos.

Dentro de los términos que conforman el escenario de la prueba: la base de datos cuenta con un tamaño de almacenamiento de 3.71 MiB, en donde cada documento va en un rango de 10,000 documentos en las distintas colecciones.

Escenarios de prueba

Las funciones para la copia y restauración de la base de datos fueron codificadas basadas en las operaciones básicas que MongoDB proporciona en su página oficial, así como subprocesos, intérprete y líneas de comando, para tratar de hacer los procesos lo más similares posibles, por cada framework.

La Tabla 24 muestra la descripción del script de automatización para la prueba capacidad de recuperación en Apache JMeter.

Tabla 24. Script de automatización para la prueba de capacidad de recuperación.

No.	Descripción
1	Iniciar un plan de pruebas en JMeter
2	Añadir un Grupo de Hilos para almacenar todas las peticiones e información que este viajando entre JMeter y el servidor. Con el nombre: Prueba Comportamiento temporal “Nombre del Framework” a probar.
3	Dejar en default sus propiedades: número de hilos (1), el periodo de subida (1) y el contador de bucle (0)
4	Agregar un árbol de resultados para observar la información recibida al momento de ejecutar peticiones. o VRT-Árbol de resultados
5	Añadir un primer Http Request (Backup) y configurar los siguientes campos: Protocolo: http, IP del servidor: 127.0.0.1 o localhost y el número de puerto: 8000. Seleccionar el método a usar: GET y el Path:/api/backup/ para la copia de seguridad
6	Iniciar la prueba para obtener el tiempo de carga al realizar la copia de seguridad
7	Al finalizar la ejecución, inhabilitar el primer Http Request (Backup) y aplicar limpiar todo
8	Añadir un segundo Http Request (Restoration) y configurar los siguientes campos: Protocolo: http, IP del servidor: 127.0.0.1 o localhost y el número de puerto: 8000. Seleccionar el método a usar: GET y el Path:/api/restoration/ para el respaldo de la información

No.	Descripción
9	Iniciar la prueba para obtener el tiempo de carga al realizar la restauración de los datos
10	Las configuraciones para las pruebas de capacidad de recuperación en cada uno de los frameworks backend son semejantes.

Sección de descripción general del diseño en los casos de prueba CP13, CP14 y CP15 (Capacidad de recuperación), como se muestra en la Tabla 25.

Tabla 25. Diseño de los casos de prueba CP13, CP14 y CP15.

Nombre: Fiabilidad – Capacidad de recuperación	
Fecha de Revisión:	**Prioridad:** Alta **Revisor:** Berenice Ortiz Torres
Información sobre el caso de prueba	
Descripción: Esta prueba consta en hacer la copia de seguridad (BackUp) y restauración de la información en la base, para medir el tiempo de carga (Load Time).	**Prerrequisitos:** Ejecutar los procedimientos indicados en el script de automatización de la prueba de capacidad de recuperación.
Datos de entrada	
**Funciones en los frameworks backend	
Procedimiento: A través del módulo y controlador se indican los procedimientos para la copia de seguridad y restauración completa de los datos.	**Resultado esperado:** Se espera el correcto respaldo y restauración de los datos para obtener el tiempo de carga, sin que se vean afectadas las colecciones.

5.2.2.6 Diseño del contexto de prueba

El contexto de las pruebas se indica como el ámbito o espacio en el que se estarán ejecutando las distintas pruebas. La Figura 13 muestra el entorno de pruebas general, para probar individualmente cada framework backend, según la planificación y en distintos periodos.

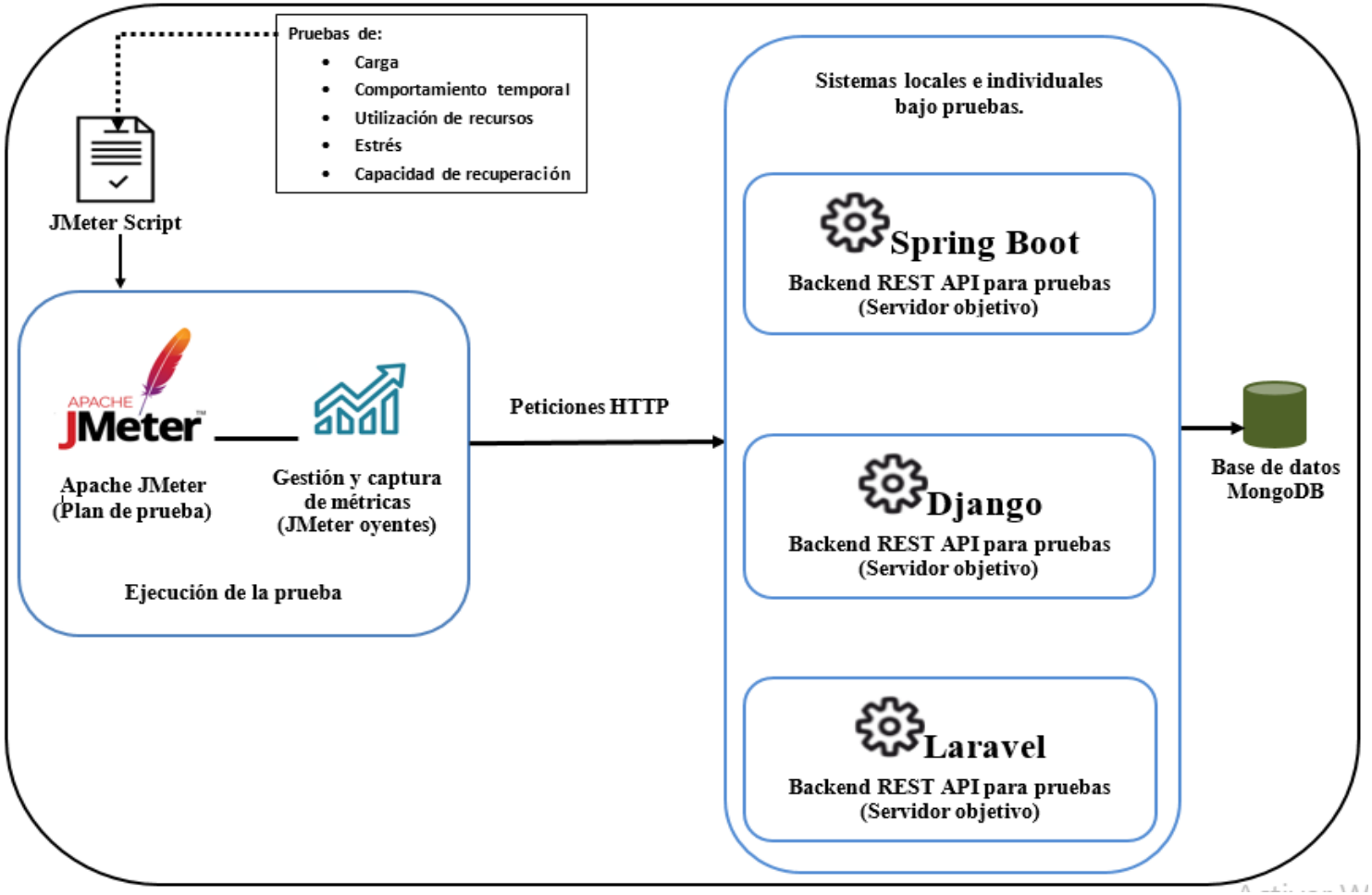


Figura 13. Entorno de prueba general.

5.2.3 Implementación y ejecución de las pruebas

Para comenzar, se detalla una cronometría de los procedimientos y juegos de prueba establecidos. En la Tabla 26 se muestra el cronograma de ejecución de las pruebas que indica la organización de los casos de prueba, el orden en el cual serán ejecutados y sus dependencias.

Tabla 26. Cronograma de ejecución de las pruebas.

Categoría de prueba	Prueba	ID Caso de prueba	Framework Backend	Prioridad	Dependencia	Fecha de Inicio-Fin
Rendimiento	Carga	CP01 CP02 CP03	Spring Boot Django Laravel	Alta	N/A	02/02/2024 al 31/05/24
	Comportamiento Temporal	CP04 CP05 CP06	Spring Boot Django Laravel	Alta	N/A	02/02/2024 al 31/05/24
	Utilización de recurso	CP07 CP08 CP09	Spring Boot Django Laravel	Alta	CP04 CP05 CP06	02/02/2024 al 31/05/24
	Estrés	CP10 CP11 CP12	Spring Boot Django Laravel	Alta	N/A	02/02/2024 al 31/05/24
Fiabilidad	Capacidad de recuperación	CP13 CP14 CP15	Spring Boot Django Laravel	Alta	N/A	02/02/2024 al 31/05/24

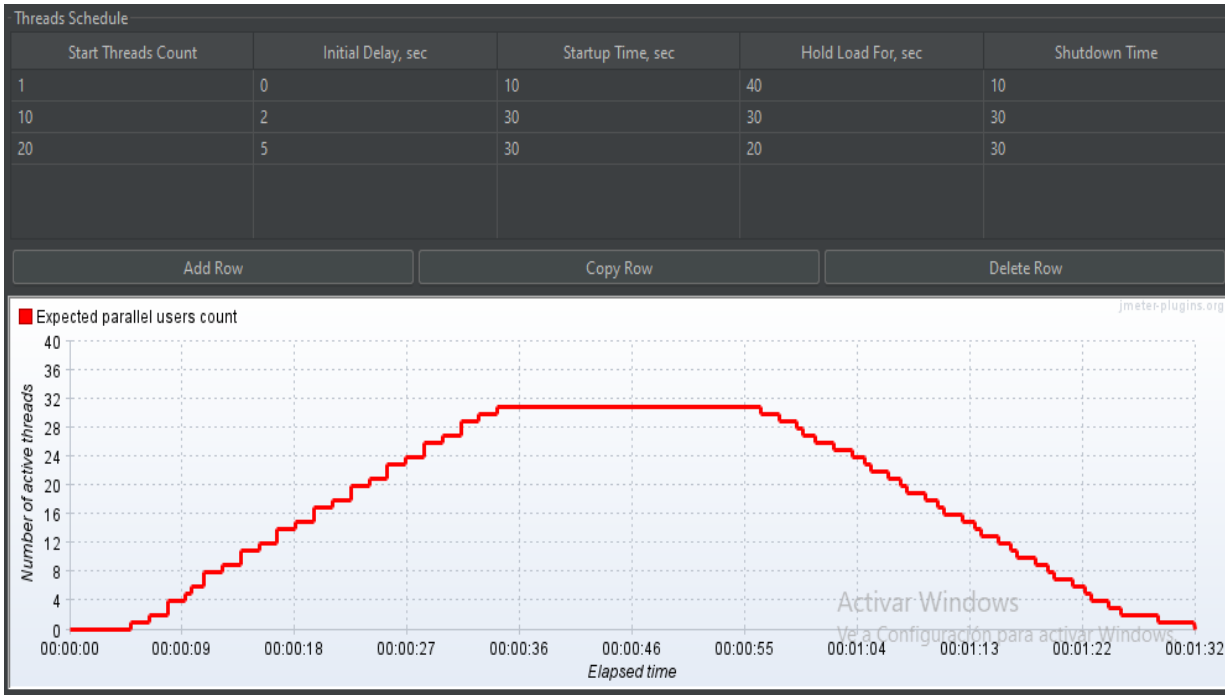
Eventualmente y conforme a los scripts de automatización, se realizó la correcta configuración de los entornos de prueba en la herramienta Apache JMeter. Por ende, se procedió a su ejecución.

5.2.3.1 Ejecución y seguimiento de las pruebas de carga

La Tabla 27 muestra la sección de descripción general para la ejecución de las pruebas de carga.

Tabla 27. Sección de descripción general para la ejecución de las pruebas de carga.

Nombre: Rendimiento – Carga	
Fecha de Revisión: 02/02/2024 al 31/05/24 **Prioridad:** Alta **Revisor:** Berenice Ortiz Torres	
Información sobre el caso de prueba	
Descripción: Esta prueba consta del lanzamiento de cargas concurrentes de hilos virtuales, que van aumentando, bajando y estableciéndose en distintos periodos, para examinar el proceder del sistema bajo determinadas condiciones.	**Prerrequisitos:** Seguir las instrucciones descritas en el script de automatización de la prueba de carga.
Datos de entrada	



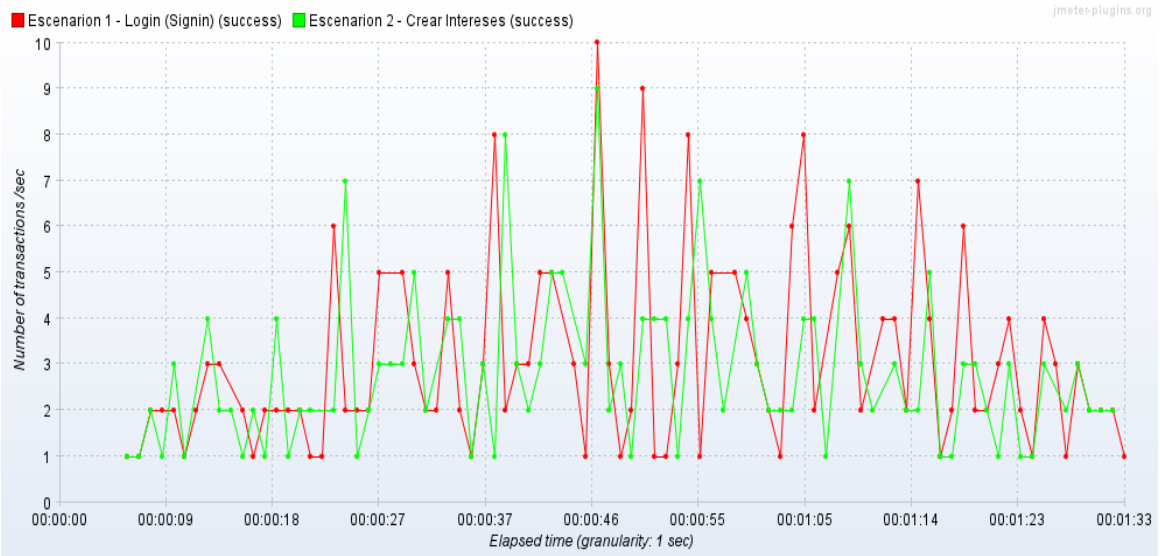
Gráfica 1. Grupo de subprocesos definitivo para la prueba de carga.

Procedimiento: A través de la herramienta Apache JMeter se indica un plan de prueba de carga que incluye grupo de hilos definitivo, para la configuración del recuento de hilos, el lapso	**Resultado esperado:** Se espera obtener la capacidad del sistema para tratar con diferentes cargas máximas, sobre funcionamientos en específico y la correcta ejecución de las mismas.

inicial, la vertiente de subida, el tiempo de carga estable y el periodo de bajada. Esto para enviar las peticiones al servidor y sean procesadas por el controlador para la obtención o consulta de objetos con los parámetros correctos y probar la carga que soporta el sistema.

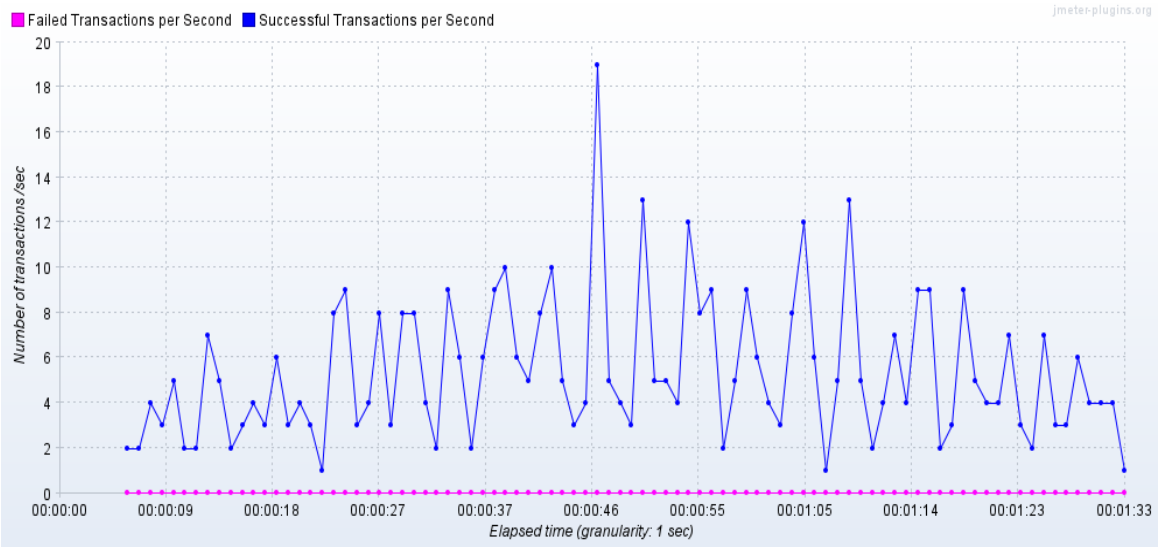
Resultados de la ejecución del caso de prueba CP01 correspondiente al framework backend Spring Boot.

La Gráfica 2 muestra el número de transacciones exitosas por segundo registradas en Spring Boot, por cada muestra (escenarios 1 y 2).



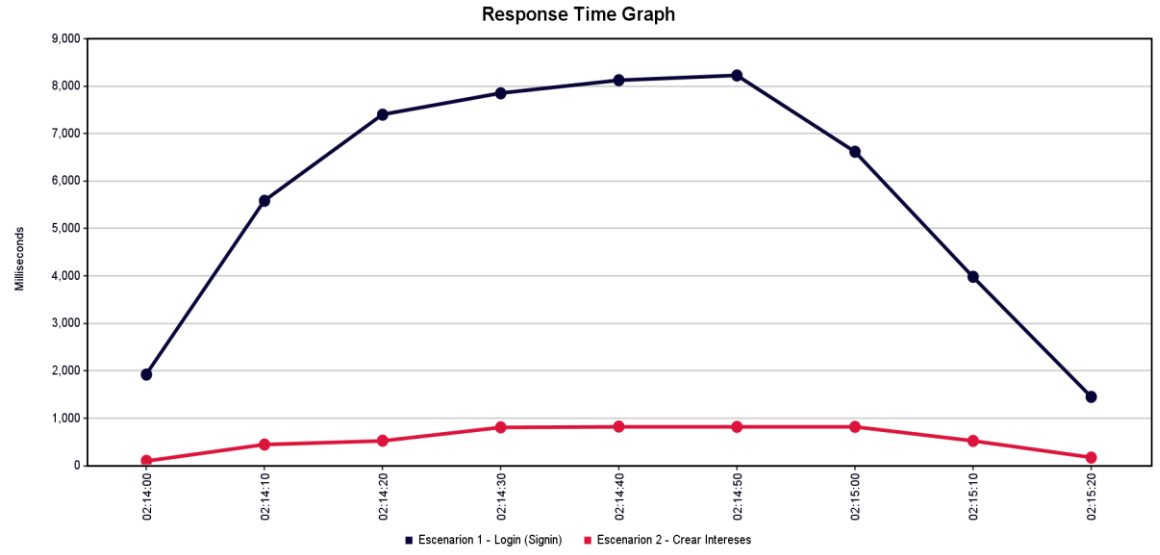
Gráfica 2. Transacciones por segundo en Spring Boot.

La Gráfica 3 muestra el número de transacciones fallidas y exitosas por segundo registradas en Spring Boot.



Gráfica 3. Número de transacciones fallidas y exitosas por segundo en Spring Boot.

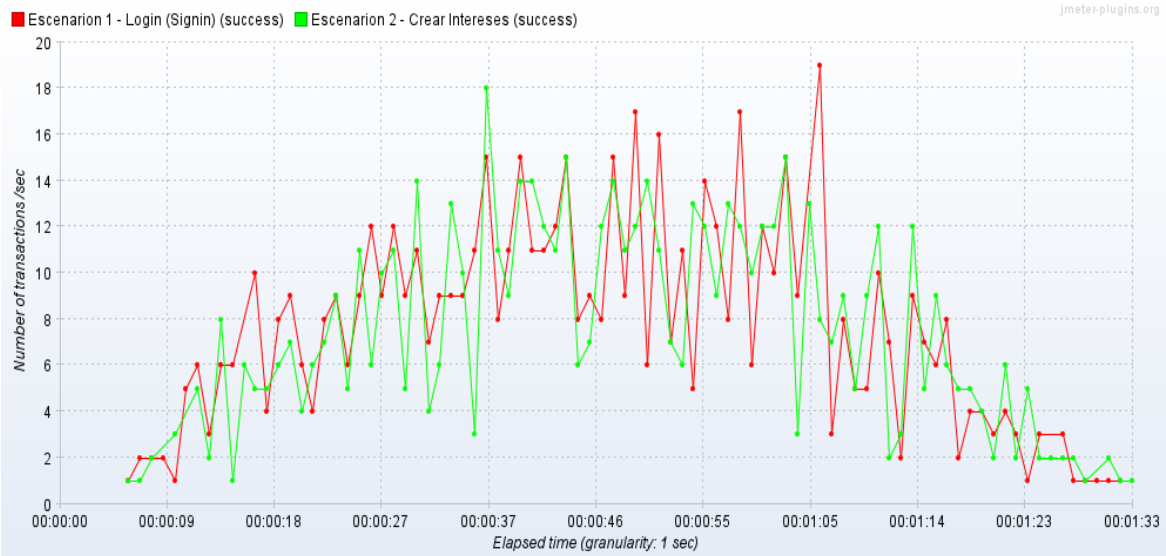
La Gráfica 4 muestra el tiempo de respuesta en milisegundos registrado en Spring Boot, para los escenarios 1 y 2.



Gráfica 4. Tiempo de respuesta en Spring Boot.

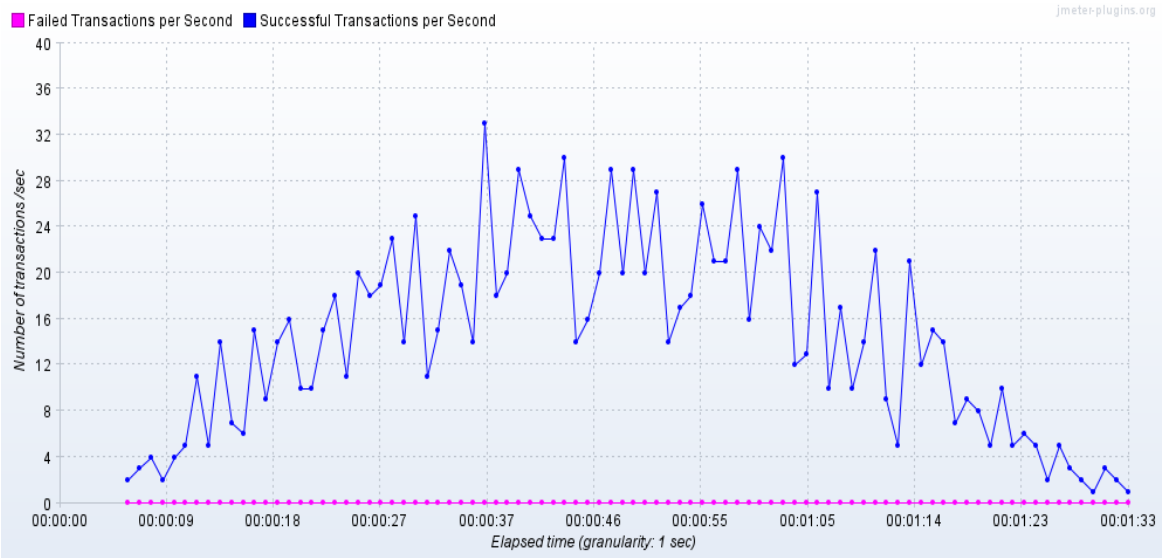
Resultados de la ejecución del caso de prueba CP02 correspondiente al framework backend Django.

La Gráfica 5 muestra el número de transacciones exitosas por segundo registradas en Django, por cada muestra (escenarios 1 y 2).



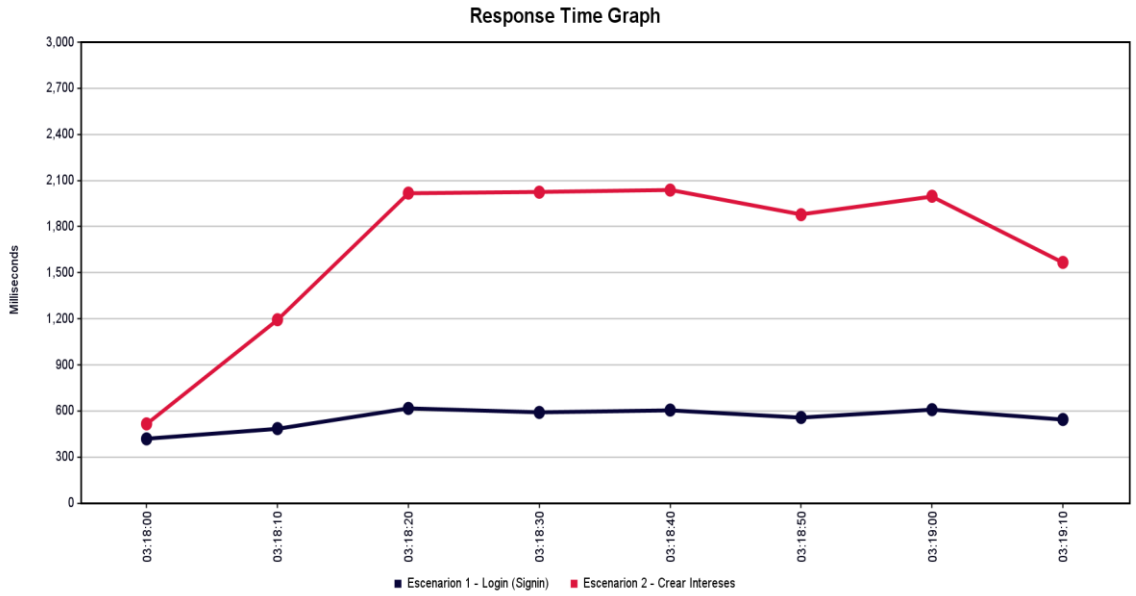
Gráfica 5. Transacciones por segundo en Django.

La Gráfica 6 muestra el número de transacciones fallidas y exitosas por segundo registradas en Django.



Gráfica 6. Número de transacciones fallidas y exitosas por segundo en Django.

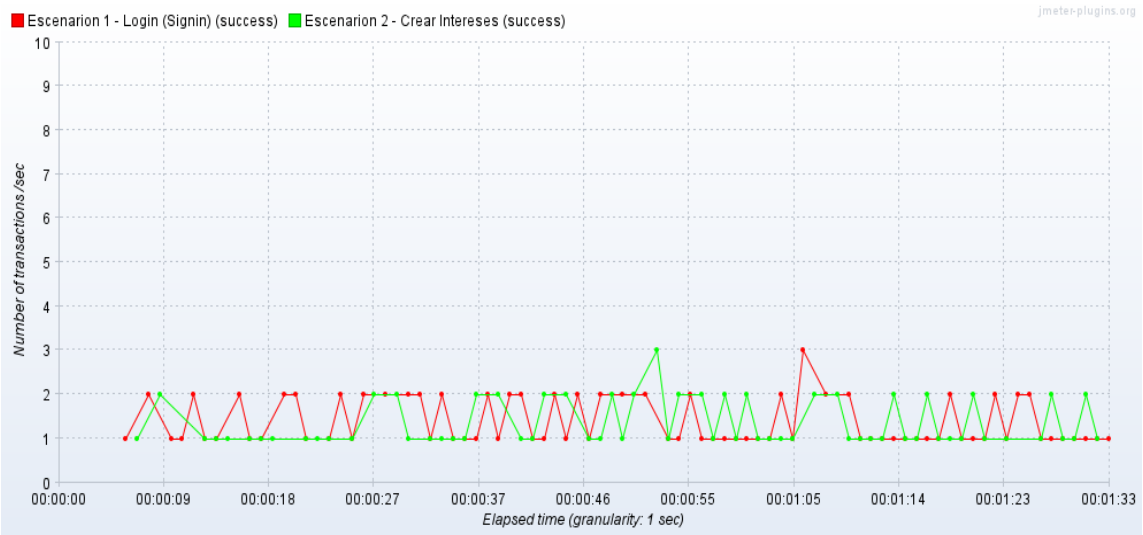
La Gráfica 7 muestra el tiempo de respuesta en milisegundos registrado en Django, para los escenarios 1 y 2.



Gráfica 7. Tiempo de respuesta en Django.

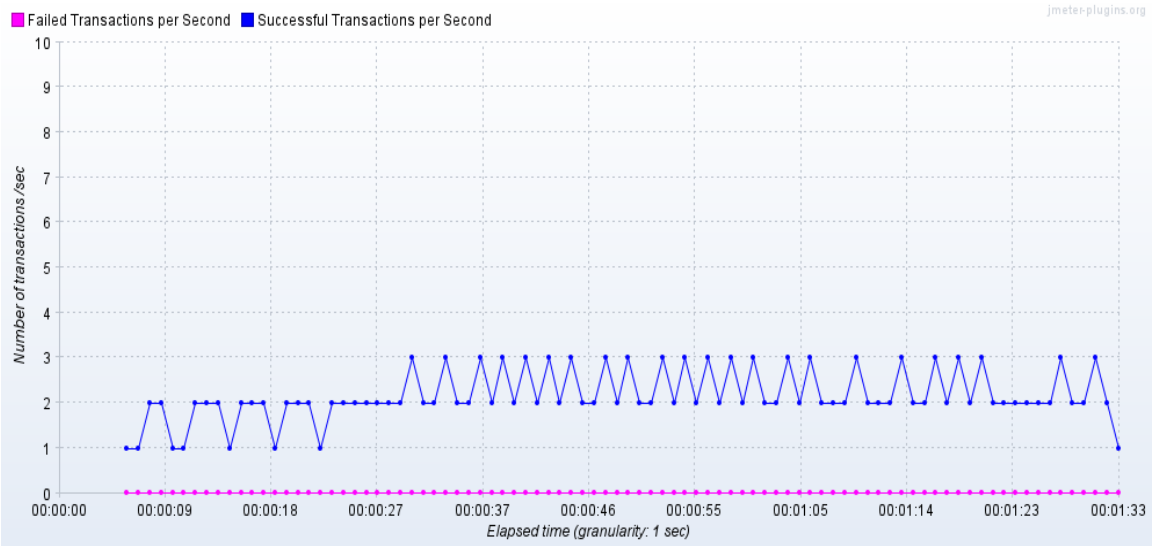
Resultados de la ejecución del caso de prueba CP03 correspondiente al framework backend Laravel.

La Gráfica 8 muestra el número de transacciones exitosas por segundo registradas en Laravel, por cada muestra (escenarios 1 y 2).



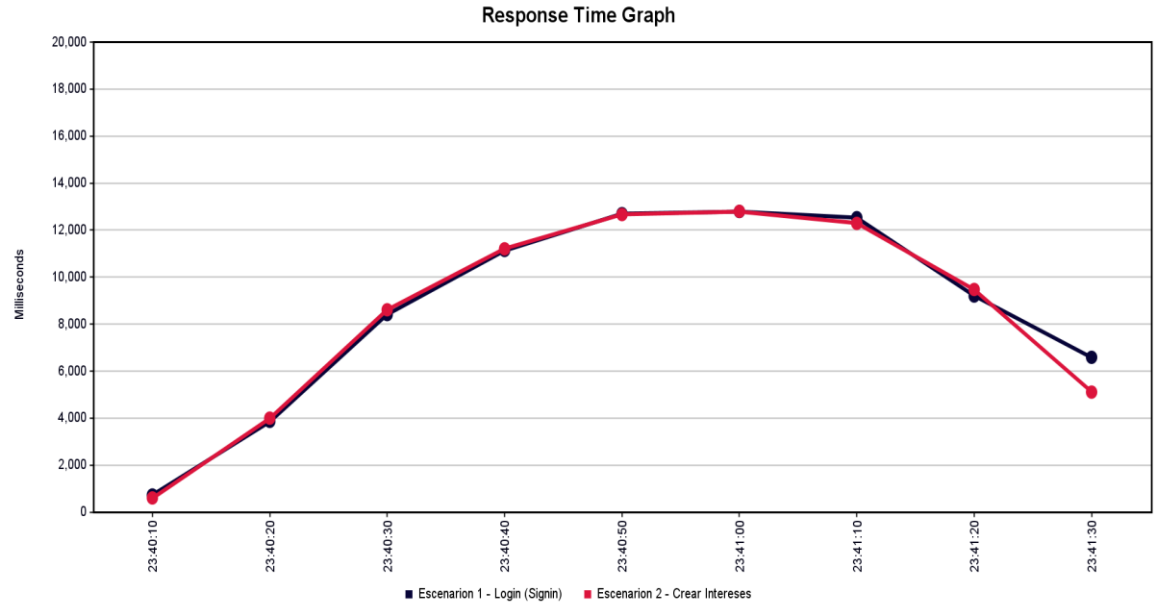
Gráfica 8. Transacciones por segundo en Laravel.

La Gráfica 9 muestra el número de transacciones fallidas y exitosas por segundo registradas en Laravel.



Gráfica 9. Número de transacciones fallidas y exitosas por segundo en Laravel.

La Gráfica 10 muestra el tiempo de respuesta en milisegundos registrado en Laravel, para los escenarios 1 y 2.



Gráfica 10. Tiempo de respuesta en Laravel.

La Tabla 28 muestra la sección de descripción general para el informe de defectos (Estatus) para los casos de prueba de carga, así como detalles, errores, soluciones, entre otras características.

Tabla 28. Informe de defectos (Estatus) para los casos de prueba de carga.

<table>
<tr><td></td><td></td><td colspan="3">No. Caso de Prueba</td></tr>
<tr><td></td><td></td><td>CP01</td><td>CP02</td><td>CP03</td></tr>
<tr><td></td><td></td><td colspan="3">Estatus</td></tr>
<tr><td rowspan="2">Resumen de resultados obtenidos: La ejecución de la prueba de carga CP01, CP02 y CP03 en los framework Spring Boot, Django y Laravel se realizó de manera exitosa, ya que no se obtuvo ningún error.</td><td>Aprobado</td><td>✓</td><td>✓</td><td>✓</td></tr>
<tr><td>No aprobado</td><td colspan="3"></td></tr>
<tr><td rowspan="6">Observaciones: sin observaciones</td><td colspan="4">Especificar razones de no aprobación:</td></tr>
<tr><td colspan="4">Grado de error</td></tr>
<tr><td colspan="4">Medio
Grave
Severo</td></tr>
<tr><td colspan="4">Detalles de la solución de la falla</td></tr>
<tr><td colspan="4">Estado de la falla: Sin Falla
Solución de la falla:
Fecha de la última revisión: 31/05/24
Fecha de la solución:</td></tr>
</table>

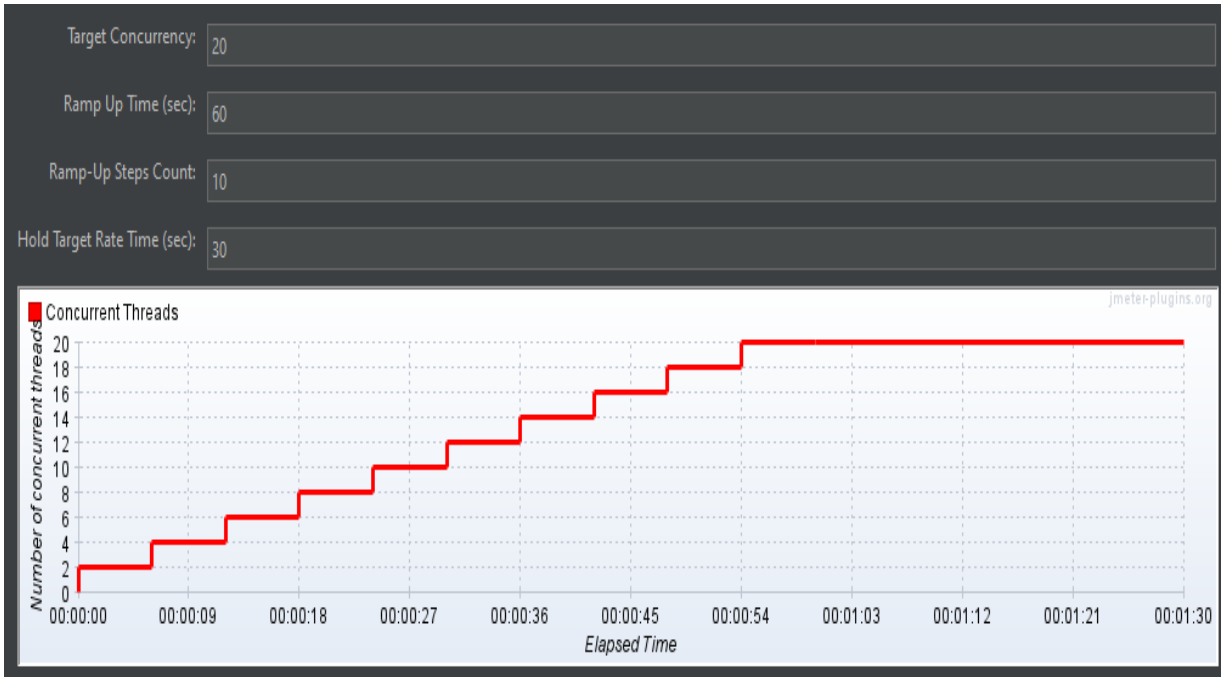
5.2.3.2 Ejecución y seguimiento de las pruebas de comportamiento temporal

La Tabla 29 muestra la sección de descripción general para la ejecución de las pruebas de comportamiento temporal.

Tabla 29. Sección de descripción general para la ejecución de las pruebas de comportamiento temporal.

Nombre: Rendimiento – Comportamiento Temporal			
Fecha de Revisión: 02/02/2024 al 31/05/24	**Prioridad:** Alta	**Revisor:** Berenice Ortiz Torres	
Información sobre el caso de prueba			

Descripción: Procedente de la prueba de carga, esta prueba consta en la monitorear el comportamiento del sistema desde que se inicia una transacción o actividad, hasta el momento en que se termina o completa, durante la ejecución de cargas elevadas de usuarios virtuales.	**Prerrequisitos:** Seguir las instrucciones descritas para la prueba de comportamiento temporal en el script de automatización.
Datos de entrada	

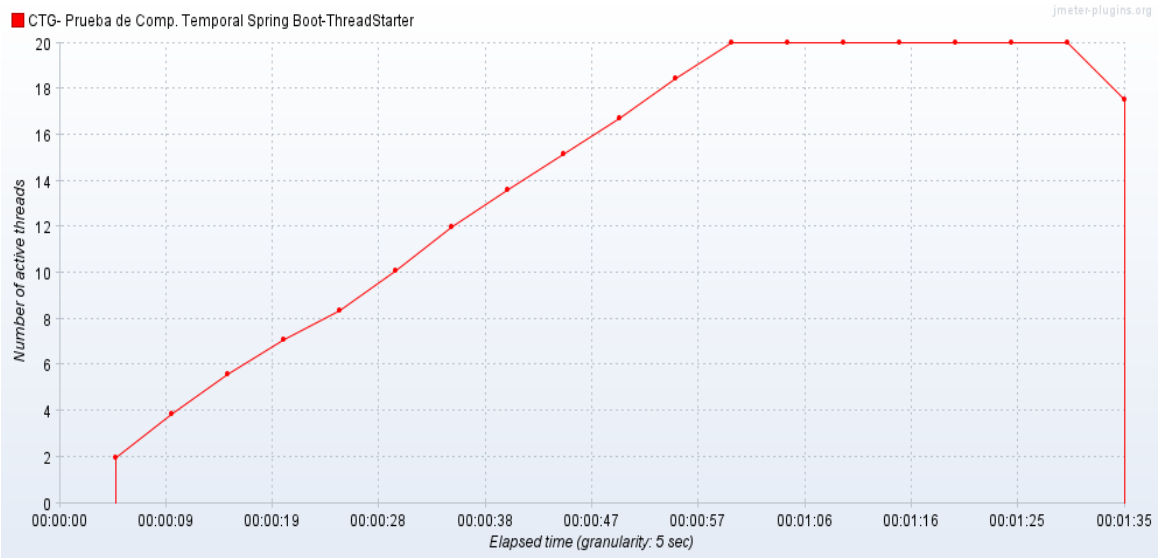


Gráfica 11. Grupo de subprocesos de concurrencia.

Procedimiento: A través de la herramienta Apache JMeter se incluyen los complementos: Hilos activos y un informe resumido, para obtener el tiempo de respuesta y de procesamiento.	**Resultado esperado:** Se espera la correcta ejecución de las actividades en condiciones de funcionamiento específicas para obtener el tiempo del comportamiento en los diferentes procesos.

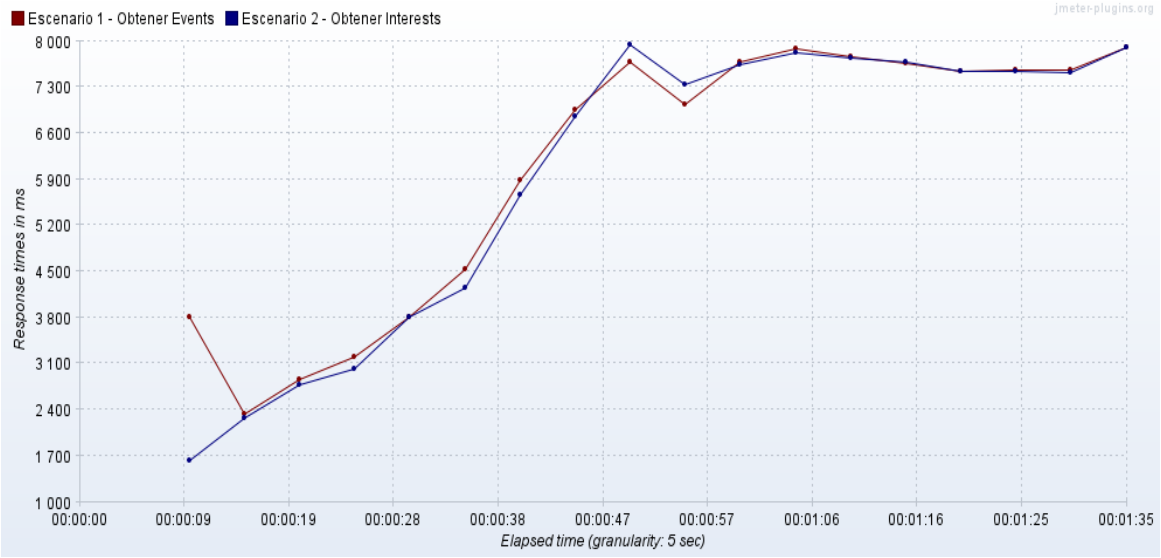
Resultados de la ejecución del caso de prueba CP04 correspondiente al framework backend Spring Boot.

La Gráfica 12 muestra el número de hilos activos en Spring Boot, sobre un valor de línea de tiempo en grupo de 5 segundos.



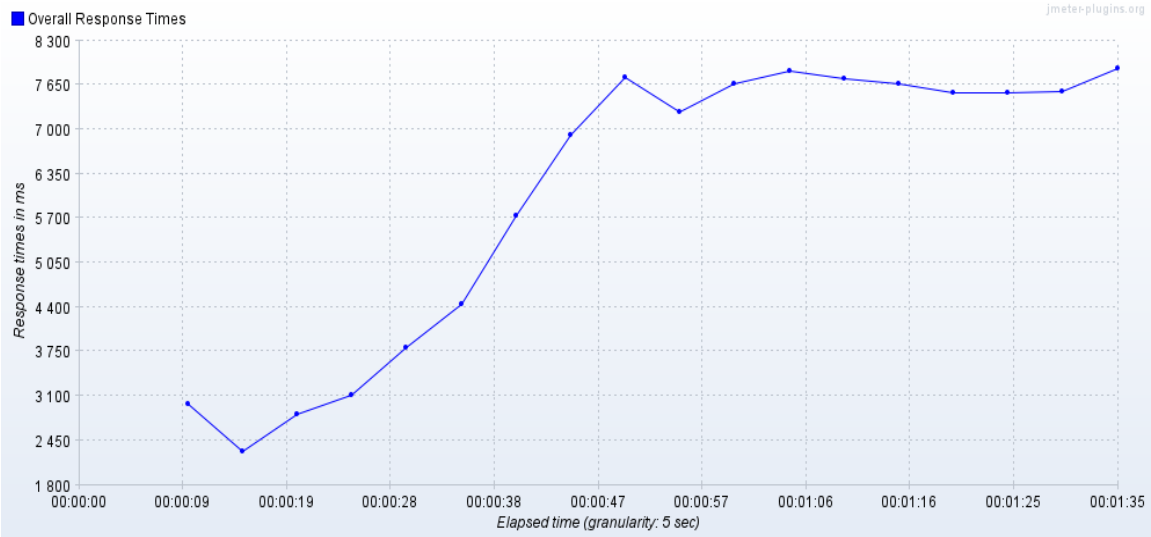
Gráfica 12. Hilos activos en Spring Boot.

La Gráfica 13 muestra el tiempo de respuesta en milisegundos, registrado en Spring Boot.



Gráfica 13. Tiempo de respuesta en Spring Boot.

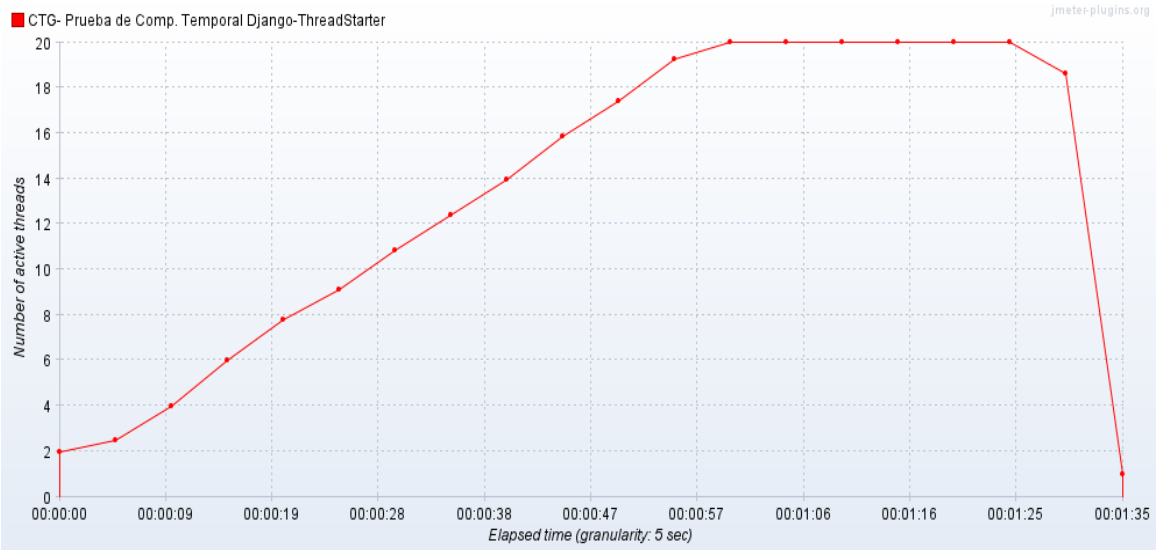
La Gráfica 14 muestra el tiempo de respuesta en milisegundos general en Spring Boot.



Gráfica 14. Tiempo de respuesta general en Spring Boot.

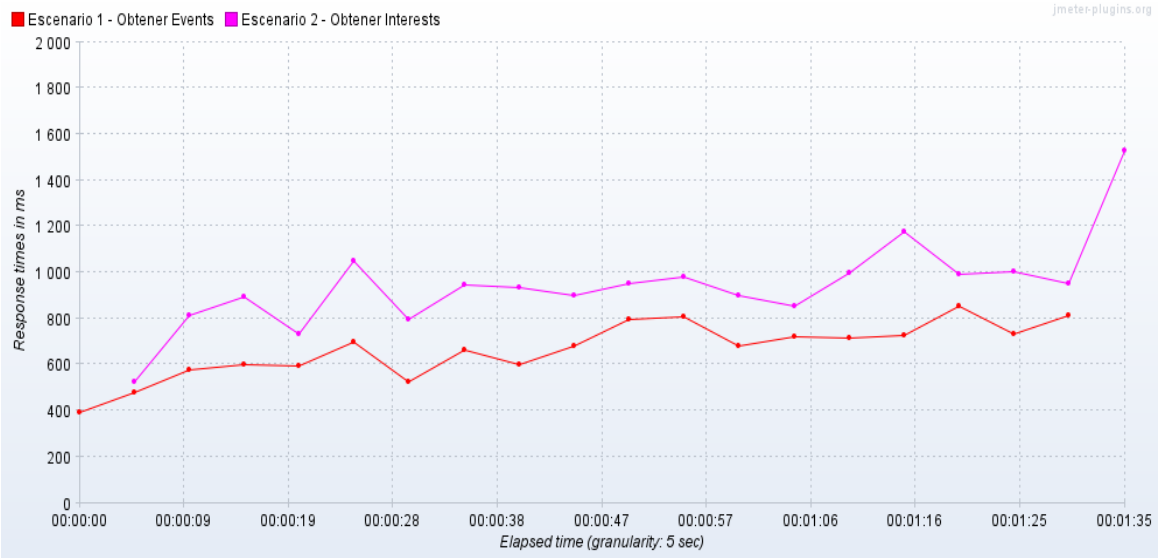
Resultados de la ejecución del caso de prueba CP05 correspondiente al framework backend Django.

La Gráfica 15 muestra el número de hilos activos a lo largo del tiempo en Django, sobre un valor de línea de tiempo en grupo de 5 segundos.



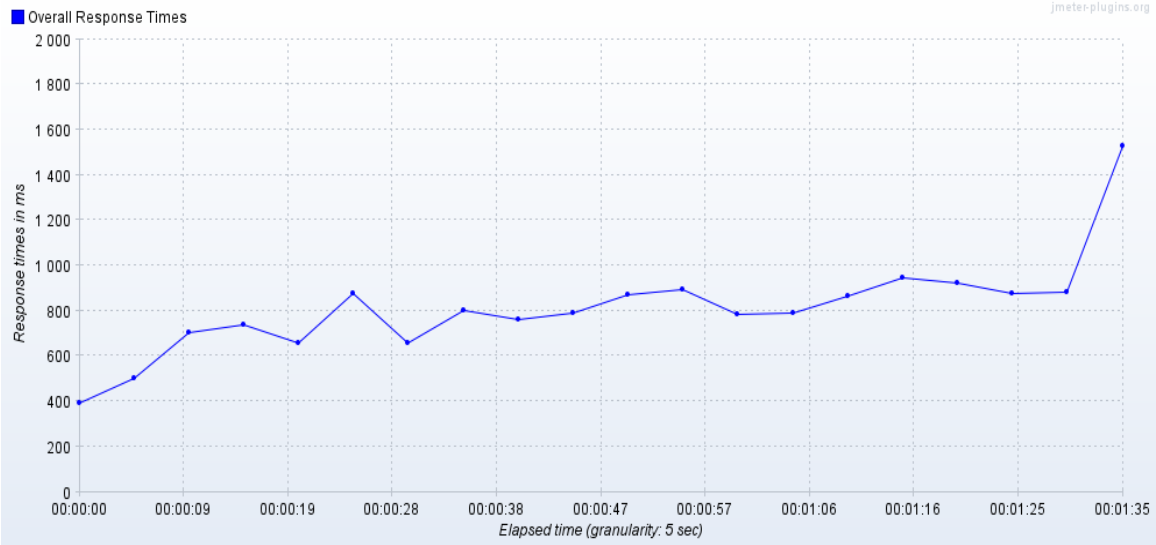
Gráfica 15. Hilos activos en Django.

La Gráfica 16 muestra el tiempo de respuesta en milisegundos, registrado en Django.



Gráfica 16. Tiempo de respuesta en Django.

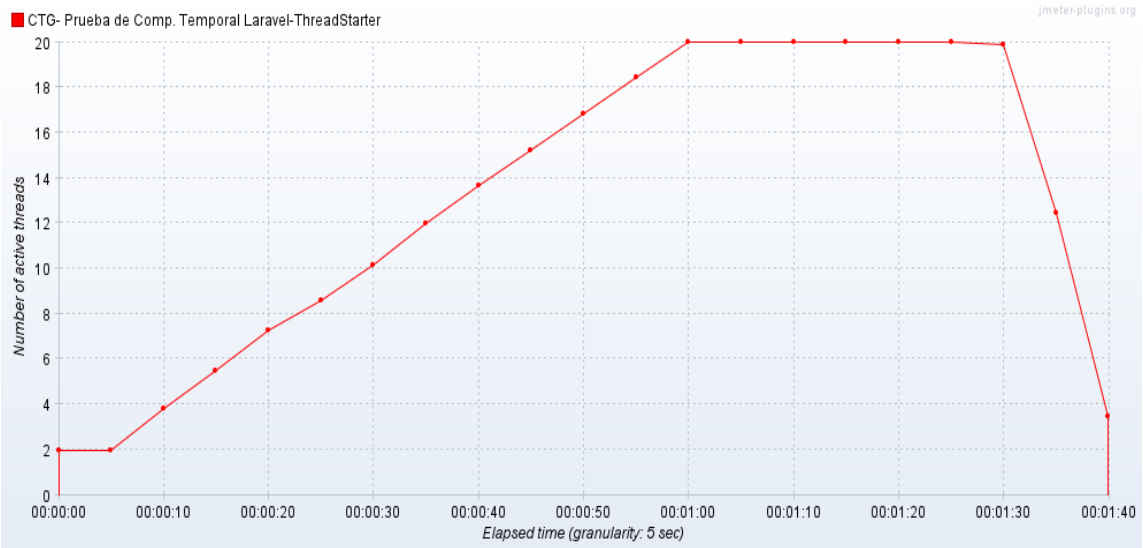
La Gráfica 17 muestra el tiempo de respuesta en milisegundos general en Django.



Gráfica 17. Tiempo de respuesta general en Django.

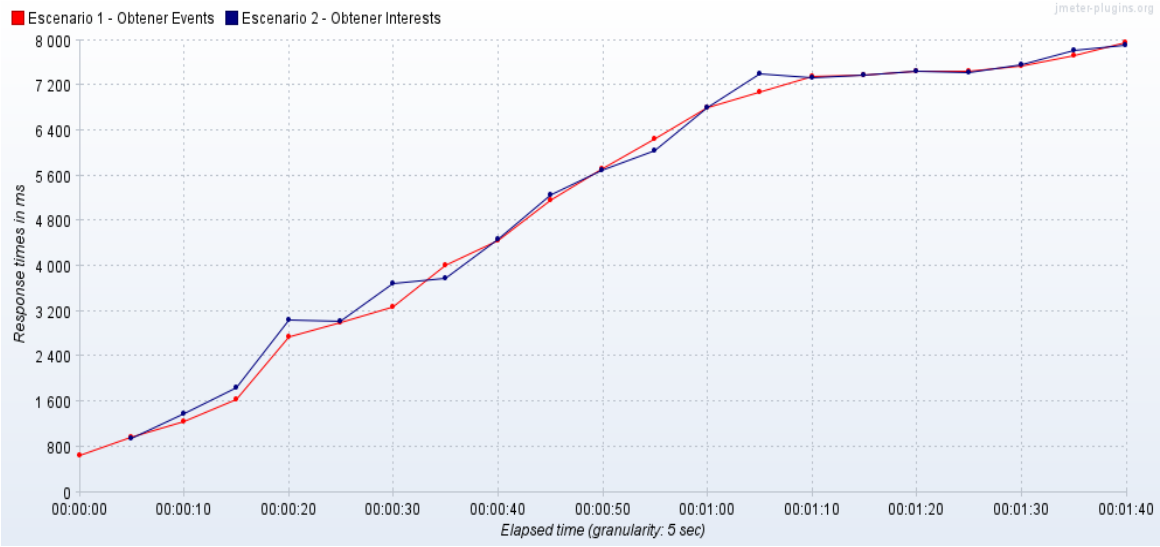
Resultados de la ejecución del caso de prueba CP06 correspondiente al framework backend Laravel.

La Gráfica 18 muestra el número de hilos activos registrados en Laravel, sobre un valor de línea de tiempo en grupo de 5 segundos.



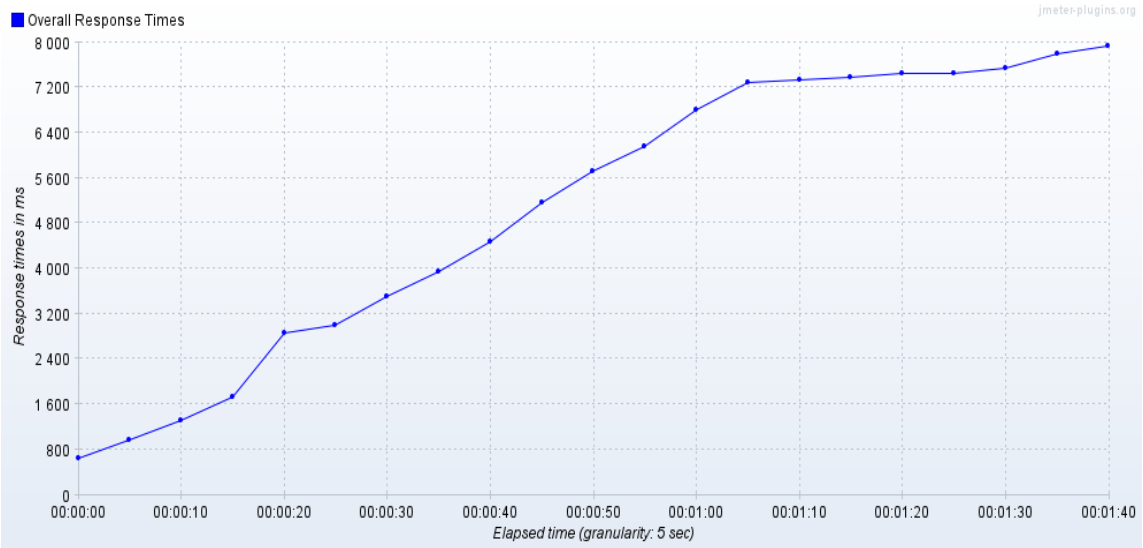
Gráfica 18. Hilos actívos en Laravel.

La Gráfica 19 muestra el tiempo de respuesta a lo largo del tiempo en milisegundos, registrado en Laravel.



Gráfica 19. Tiempo de respuesta en Laravel.

La Gráfica 20 muestra el tiempo de respuesta en milisegundos general en Laravel.



Gráfica 20. Tiempo de respuesta general en Laravel.

La Tabla 30 muestra la sección de descripción general para el informe de defectos (Estatus) para los casos de prueba de comportamiento temporal, así como detalles, errores, soluciones, entre otras características.

Tabla 30. Informe de defectos (Estatus) para los casos de prueba de comportamiento temporal.

<table>
<tr><td colspan="2" rowspan="2"></td><td colspan="3">No. Caso de Prueba</td></tr>
<tr><td>CP04</td><td>CP05</td><td>CP06</td></tr>
<tr><td colspan="2" rowspan="2">Resumen de resultados obtenidos:
La ejecución de la prueba de comportamiento temporal en los framework Spring Boot, Django y Laravel se realizó de manera exitosa, ya que no se obtuvo ningún error.</td><td colspan="3">Estatus</td></tr>
<tr><td>Aprobado

No aprobado</td><td>✓</td><td>✓</td><td>✓</td></tr>
<tr><td colspan="2" rowspan="5">Observaciones: sin observaciones</td><td colspan="3">Especificar razones de no aprobación:</td></tr>
<tr><td colspan="3">Grado de error</td></tr>
<tr><td colspan="3">Medio
Grave
Severo</td></tr>
<tr><td colspan="3">Detalles de la solución de la falla</td></tr>
<tr><td colspan="3">Estado de la falla: Sin Falla
Solución de la falla:
Fecha de la última revisión: 31/05/24</td></tr>
</table>

	No. Caso de Prueba		
	CP04	CP05	CP06
Fecha de la solución:			

5.2.3.3 Ejecución y seguimiento de las pruebas de utilización de recursos

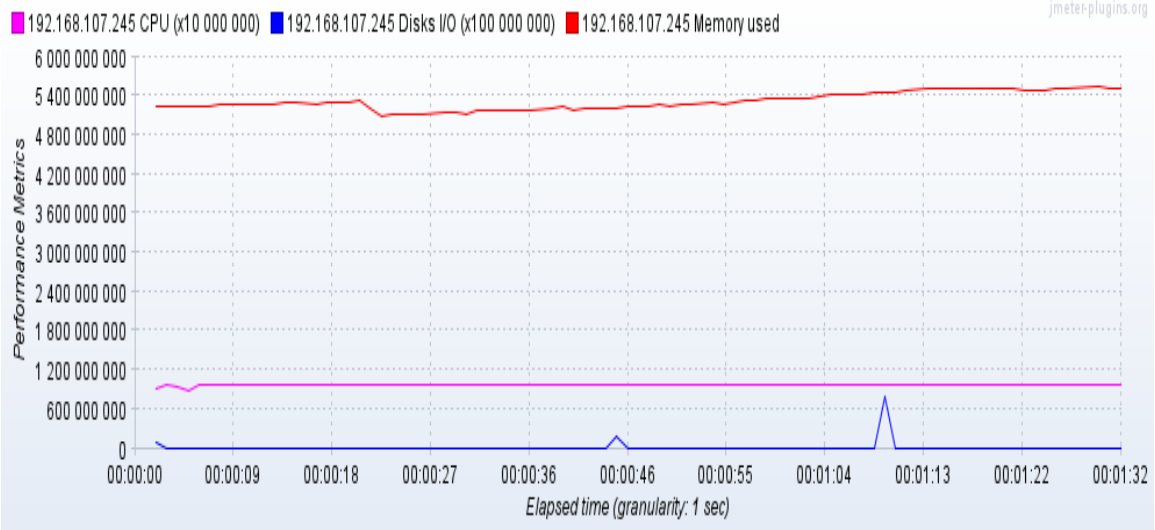
La Tabla 31 muestra la sección de descripción general para la ejecución de las pruebas de utilización de recursos.

Tabla 31. Sección de descripción general para la ejecución de las pruebas de utilización de recursos.

Nombre: Rendimiento – Utilización de recursos	
Fecha de Revisión: 02/02/2024 al 31/05/24 **Prioridad:** Alta **Revisor:** Berenice Ortiz Torres	
Información sobre el caso de prueba	
Descripción: Procedente de la prueba de comportamiento temporal, esta prueba consiste en medir la utilización de la CPU, el uso de memoria y el disco.	**Prerrequisitos:** Seguir las instrucciones descritas para la prueba de utilización de recursos en el script de automatización de la prueba de comportamiento temporal.
Datos de entrada	
Prueba dependiente a los casos de prueba CP04, CP05, CP06, según sea el framework backend correspondiente. - **Dependencia:** Rendimiento – Comportamiento temporal	
Procedimiento: A través de la herramienta Apache JMeter se incluye el complemento: Recopilador de métricas para obtener los valores de uso, de los recursos del computador.	**Resultado esperado:** Se espera obtener el tiempo de utilización de recursos del sistema.

Resultados de la ejecución del caso de prueba CP07 correspondiente al framework backend Spring Boot.

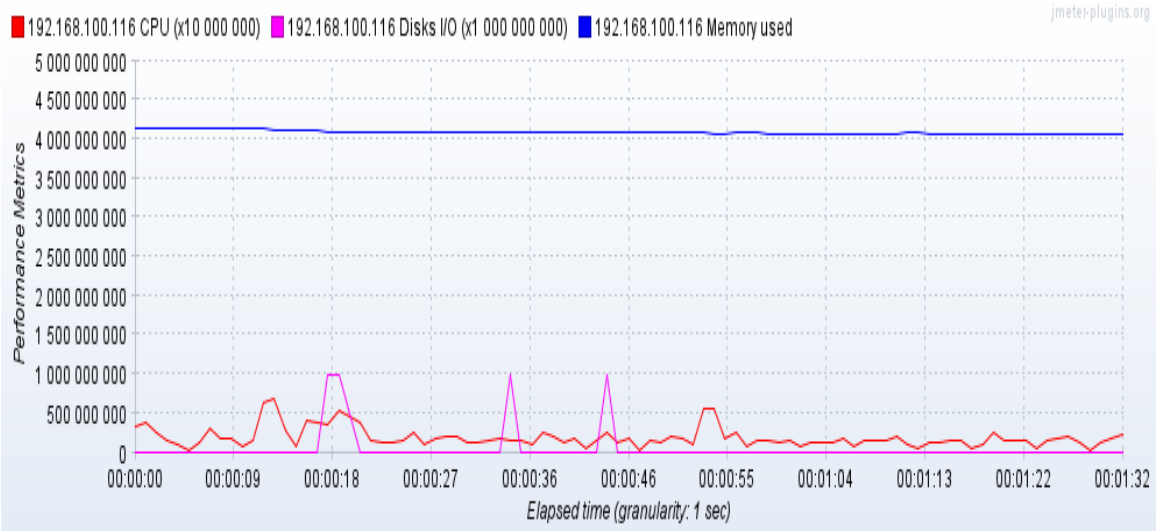
La Gráfica 21 muestra el monitor del rendimiento en Bytes/s con las métricas: CPU, Disco I/O y la utilización de memoria, durante la prueba CP04 en Spring Boot.



Gráfica 21. Colección de métricas de rendimiento en Spring Boot.

Resultados de la ejecución del caso de prueba CP08 correspondiente al framework backend Django.

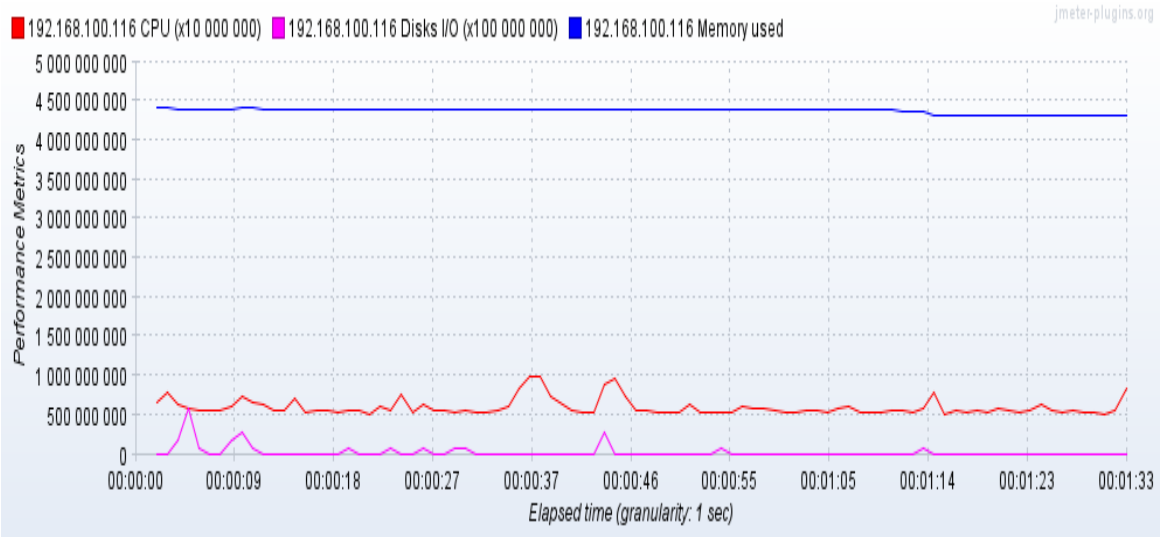
La Gráfica 22 muestra el monitor del rendimiento en Bytes/s con las métricas: CPU, Disco I/O y la utilización de memoria, durante la prueba CP05 en Django.



Gráfica 22. Colección de métricas de rendimiento en Django.

Resultados de la ejecución del caso de prueba CP09 correspondiente al framework backend Laravel.

La Gráfica 23 muestra el monitor del rendimiento en Bytes/s con las métricas: CPU, Disco I/O y la utilización de memoria, durante la prueba CP06 en Laravel.



Gráfica 23. Colección de métricas de rendimiento en Laravel.

La Tabla 32 muestra la sección de descripción general para el informe de defectos (Estatus) para los casos de prueba de utilización de recursos, así como detalles, errores, soluciones, entre otras características.

Tabla 32. Informe de defectos (Estatus) para los casos de prueba de utilización de recursos.

		No. Caso de Prueba		
		CP07	CP08	CP09
		Estatus		
Resumen de resultados obtenidos: La ejecución de la prueba de utilización de recursos en los framework Spring Boot, Django y Laravel se realizó de manera	Aprobado	✓	✓	✓
	No aprobado			

		No. Caso de Prueba		
		CP07	CP08	CP09
exitosa, ya que no se obtuvo ningún error.				
	Especificar razones de no aprobación:			
	Grado de error			
Observaciones: sin observaciones	Medio Grave Severo			
	Detalles de la solución de la falla			
	Estado de la falla: Sin Falla Solución de la falla: Fecha de la última revisión: 31/05/24 Fecha de la solución:			

5.2.3.4 Ejecución y seguimiento de las pruebas de estrés

La Tabla 33 muestra la sección de descripción general para la ejecución de las pruebas de estrés.

Tabla 33. Sección de descripción general para la ejecución de las pruebas de estrés.

Nombre: Rendimiento – Estrés	
Fecha de Revisión: 02/02/2024 al 31/05/24 **Prioridad:** Alta **Revisor:** Berenice Ortiz Torres	
Información sobre el caso de prueba	
Descripción: Esta prueba consta del lanzamiento de tres pruebas de estrés, con distintas cantidades de usuarios virtuales simultáneos que van en aumento, para examinar el proceder del sistema bajo determinadas condiciones.	**Prerrequisitos:** Seguir las instrucciones descritas en el script de automatización de la prueba de estrés.
Datos de entrada	

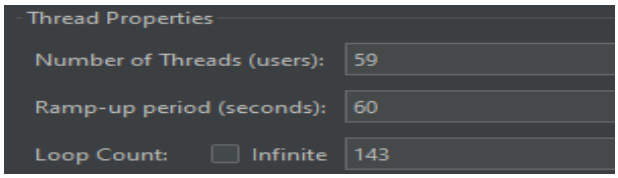


Figura 14. Grupo de 59 Hilos.

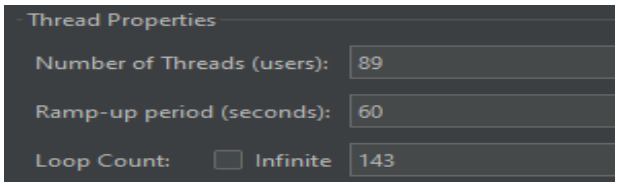


Figura 15. Grupo de 89 Hilos.

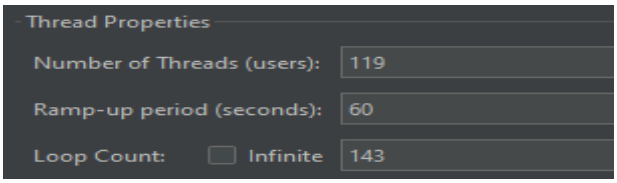


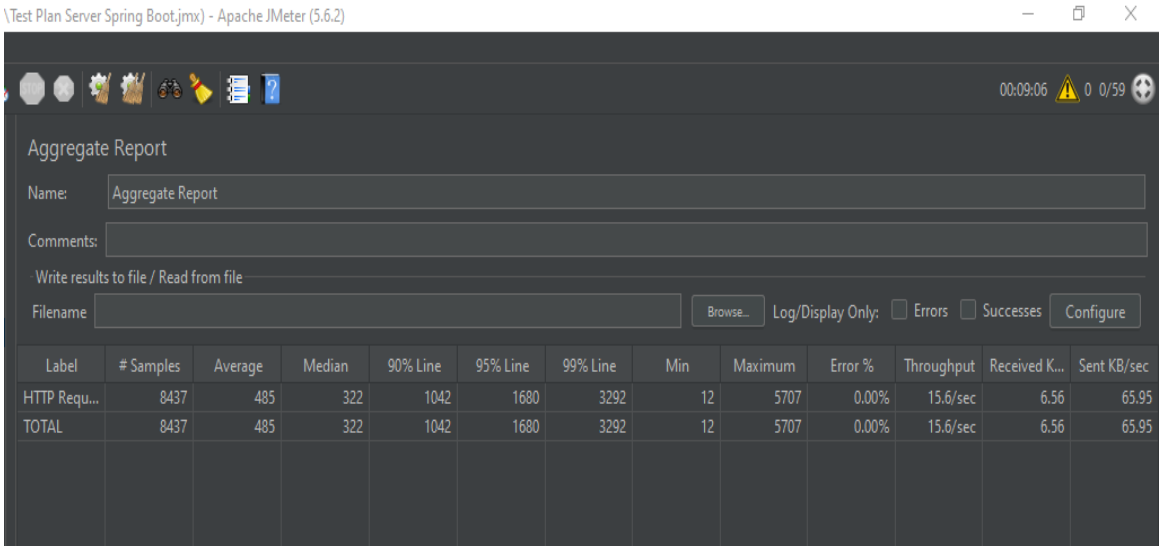
Figura 16. Grupo de 119 Hilos.

Procedimiento: A través de la herramienta Apache JMeter se indica un plan de prueba que incluye un grupo de hilos virtuales, para la configuración del número de hilos, el periodo de subida y un bucle. Esto para enviar las peticiones al servidor y sean procesadas por el controlador para la creación de inserciones de carga de nuevos objetos con los parámetros correctos y probar el estrés que soporta el sistema.	**Resultado esperado:** Se espera al correr la prueba se ejecute sin errores y obtener la medida de tiempo de respuesta durante la incrementación de cargas en funciones específicas.

Resultados de la ejecución del caso de prueba CP10 correspondiente al framework backend Spring Boot.

La Tabla 34 muestra el informe agregado de la sub-prueba primaria en Spring Boot con 8437 muestras.

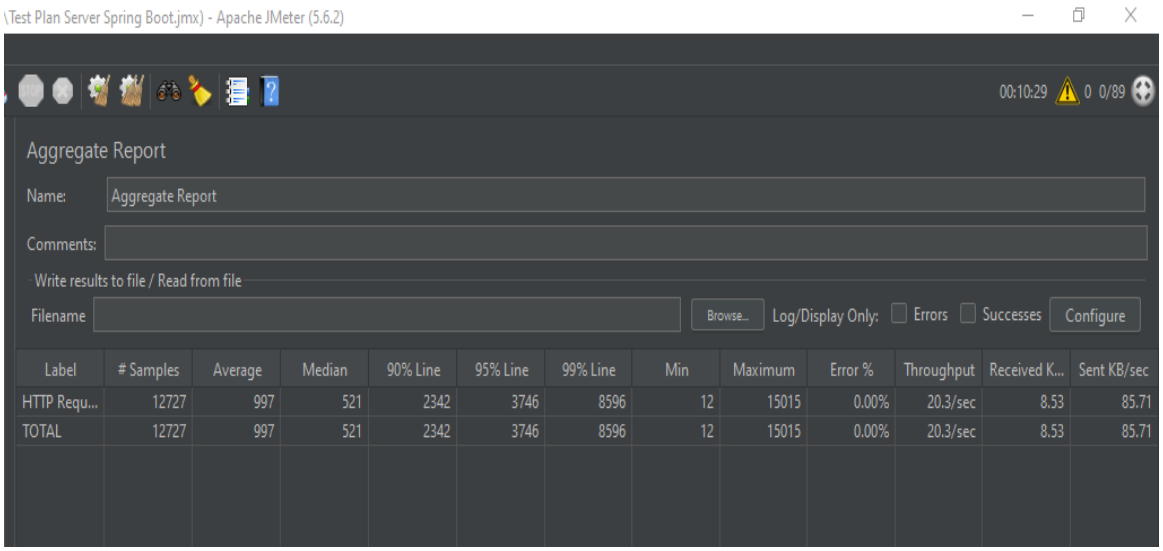
Tabla 34. Informe agregado de la subprueba primaria en Spring Boot.



Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Maximum	Error %	Throughput	Received K...	Sent KB/sec
HTTP Requ...	8437	485	322	1042	1680	3292	12	5707	0.00%	15.6/sec	6.56	65.95
TOTAL	8437	485	322	1042	1680	3292	12	5707	0.00%	15.6/sec	6.56	65.95

La Tabla 35 muestra el informe agregado de la subprueba intermedia en Spring Boot con 12727 muestras.

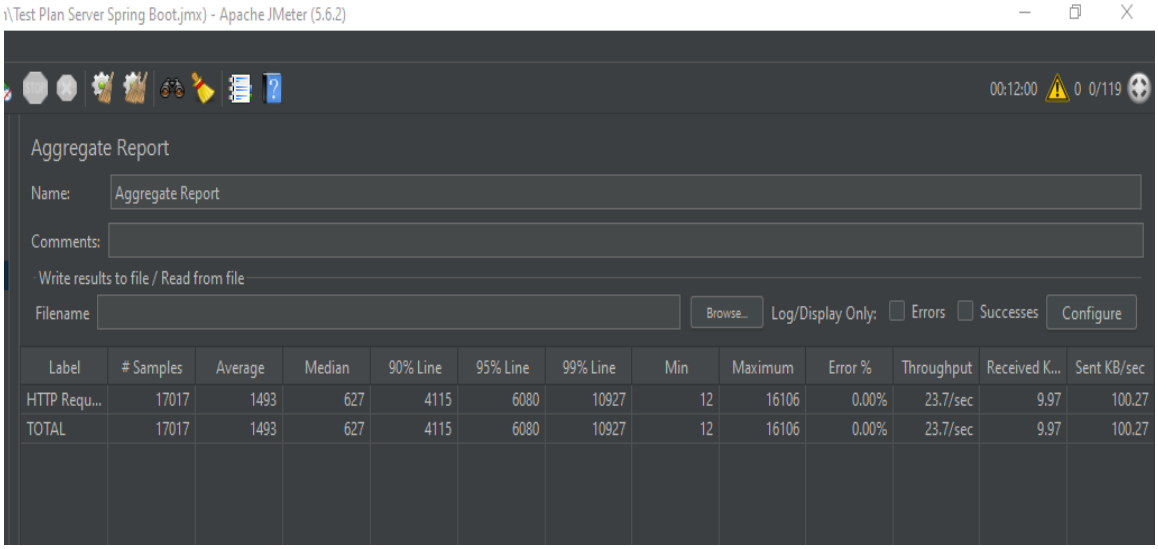
Tabla 35. Informe agregado de la subprueba intermedia en Spring Boot.



Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Maximum	Error %	Throughput	Received K...	Sent KB/sec
HTTP Requ...	12727	997	521	2342	3746	8596	12	15015	0.00%	20.3/sec	8.53	85.71
TOTAL	12727	997	521	2342	3746	8596	12	15015	0.00%	20.3/sec	8.53	85.71

La Tabla 36 muestra el informe agregado en la subprueba elevada en Spring Boot con 17017 muestras.

Tabla 36. Informe agregado de la subprueba elevada en Spring Boot.

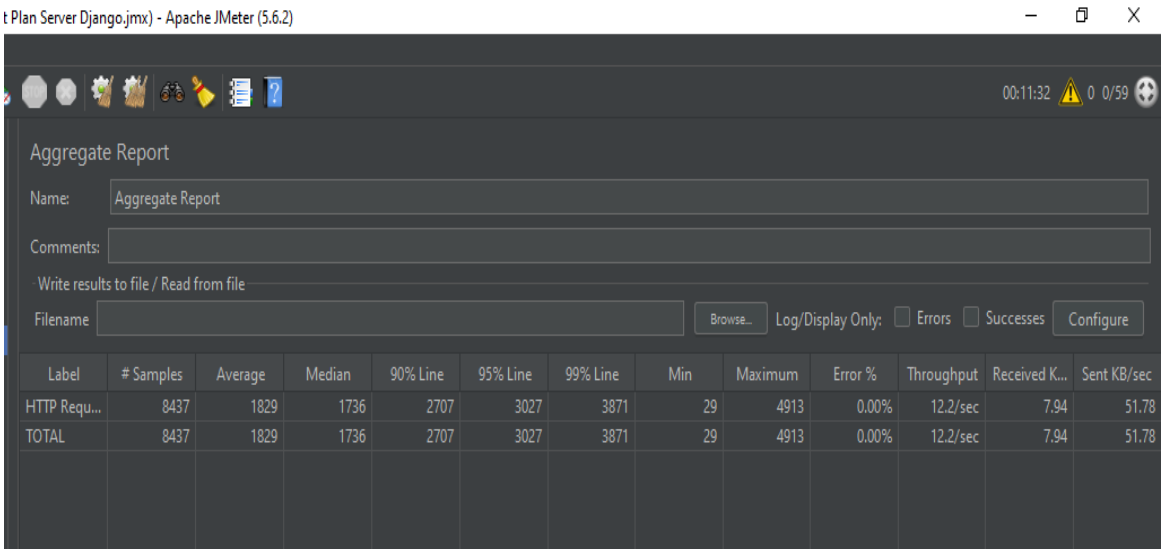


Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Maximum	Error %	Throughput	Received K...	Sent KB/sec
HTTP Requ...	17017	1493	627	4115	6080	10927	12	16106	0.00%	23.7/sec	9.97	100.27
TOTAL	17017	1493	627	4115	6080	10927	12	16106	0.00%	23.7/sec	9.97	100.27

Resultados de la ejecución del caso de prueba CP11 correspondiente al framework backend Django.

La Tabla 37 muestra el informe agregado en la subprueba primaria en Django con 8437 muestras.

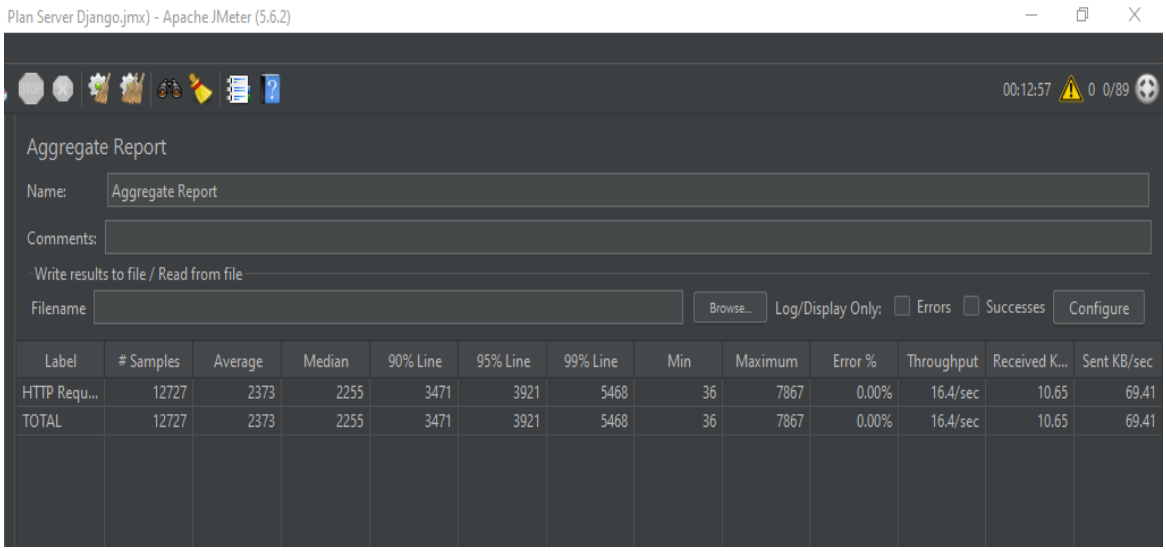
Tabla 37. Informe agregado de la subprueba primaria en Django.



Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Maximum	Error %	Throughput	Received K...	Sent KB/sec
HTTP Requ...	8437	1829	1736	2707	3027	3871	29	4913	0.00%	12.2/sec	7.94	51.78
TOTAL	8437	1829	1736	2707	3027	3871	29	4913	0.00%	12.2/sec	7.94	51.78

La Tabla 38 muestra el informe agregado en la subprueba intermedia en Django con 12727 muestras.

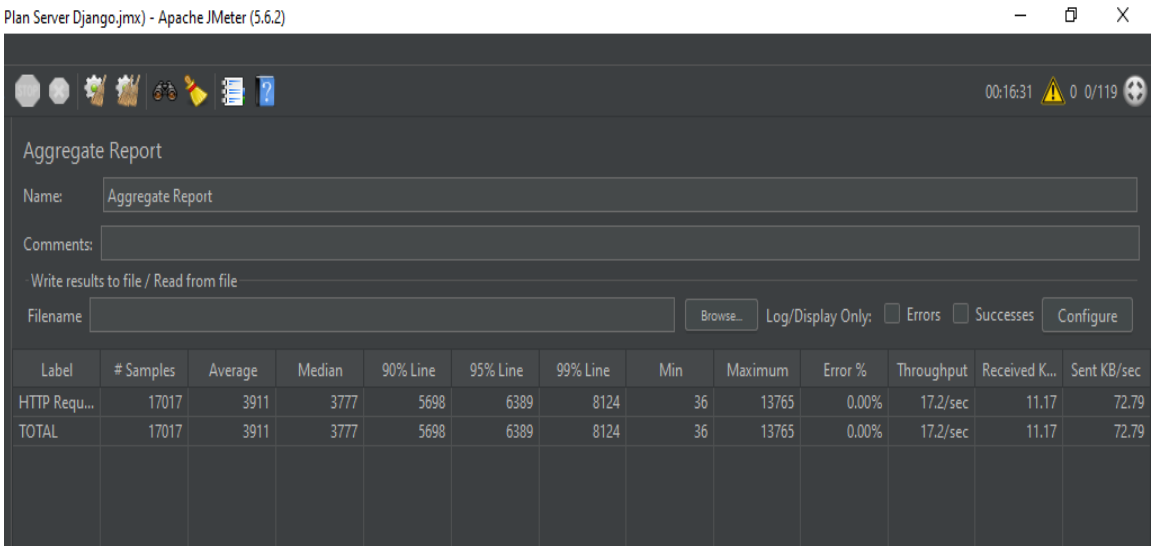
Tabla 38. Informe agregado de la subprueba intermedia en Django.



Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Maximum	Error %	Throughput	Received K...	Sent KB/sec
HTTP Requ...	12727	2373	2255	3471	3921	5468	36	7867	0.00%	16.4/sec	10.65	69.41
TOTAL	12727	2373	2255	3471	3921	5468	36	7867	0.00%	16.4/sec	10.65	69.41

La Tabla 39 muestra el informe agregado en la subprueba elevada en Django con 17017 muestras.

Tabla 39. Informe agregado de la subprueba elevada en Django.

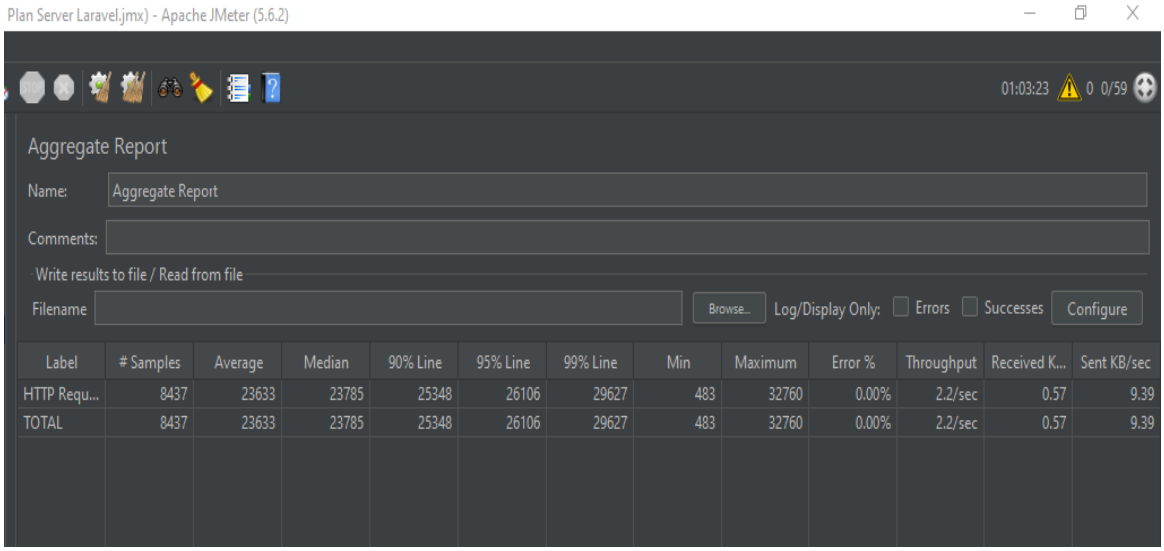


Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Maximum	Error %	Throughput	Received K...	Sent KB/sec
HTTP Requ...	17017	3911	3777	5698	6389	8124	36	13765	0.00%	17.2/sec	11.17	72.79
TOTAL	17017	3911	3777	5698	6389	8124	36	13765	0.00%	17.2/sec	11.17	72.79

Resultados de la ejecución del caso de prueba CP12 correspondiente al framework backend Laravel.

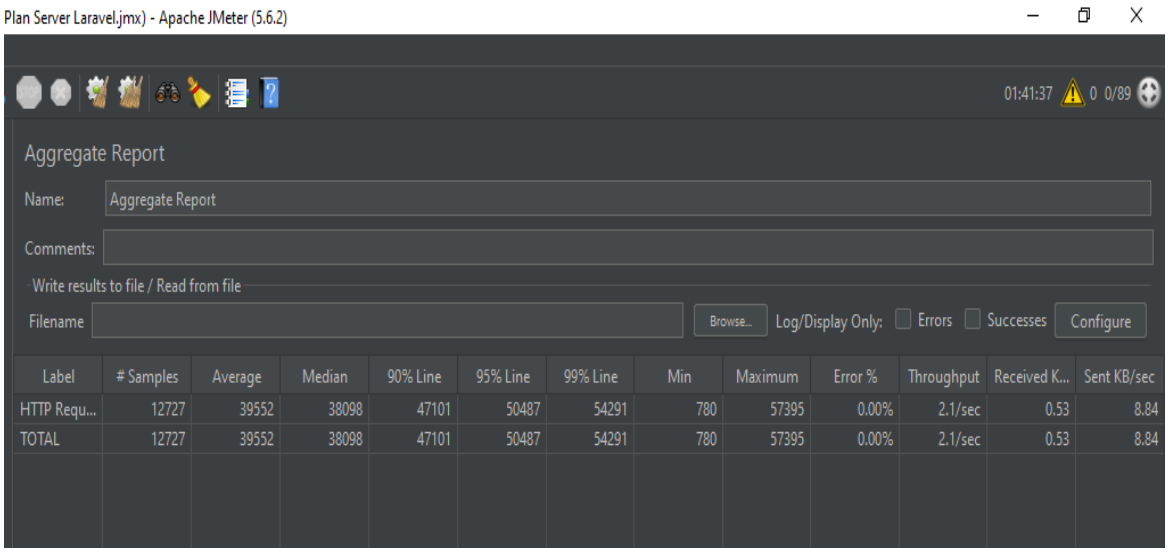
La Tabla 40 muestra el informe agregado en la subprueba primaria en Laravel con 8437 muestras.

Tabla 40. Informe agregado de la subprueba primaria en Laravel.



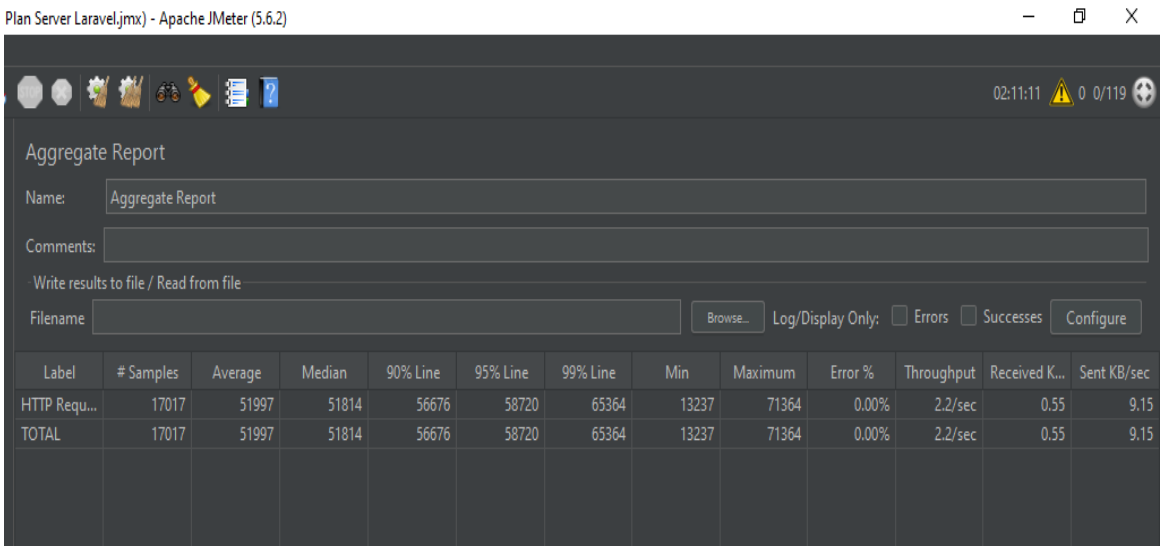
La Tabla 41 muestra el informe agregado en la subprueba intermedia en Laravel con 12727 muestras.

Tabla 41. Informe agregado de la subprueba intermedia en Laravel.



La Tabla 42 muestra el informe agregado en subprueba elevada en Laravel con 17017 muestras.

Tabla 42. Informe agregado de la subprueba elevada en Laravel.



La Tabla 43 muestra la sección de descripción general para el informe de defectos (Estatus) para los casos de prueba de estrés, así como detalles, errores, soluciones, entre otras características.

Tabla 43. Informe de defectos (Estatus) para los casos de prueba de estrés.

<table>
<tr><td colspan="2"></td><td colspan="3">No. Caso de Prueba</td></tr>
<tr><td colspan="2"></td><td>CP10</td><td>CP11</td><td>CP12</td></tr>
<tr><td colspan="2"></td><td colspan="3">Estatus</td></tr>
<tr><td colspan="2">Resumen de resultados obtenidos:
La ejecución de la prueba de estrés en los framework Spring Boot, Django y Laravel se realizó de manera exitosa, ya que no se obtuvo ningún error.</td><td>Aprobado

No aprobado</td><td>✓</td><td>✓</td><td>✓</td></tr>
<tr><td rowspan="5">Observaciones: sin observaciones</td><td colspan="4">Especificar razones de no aprobación:</td></tr>
<tr><td colspan="4">Grado de error</td></tr>
<tr><td colspan="4">Medio
Grave
Severo</td></tr>
<tr><td colspan="4">Detalles de la solución de la falla</td></tr>
<tr><td colspan="4">Estado de la falla: Sin Falla
Solución de la falla:
Fecha de la última revisión:31/05/24
Fecha de la solución:</td></tr>
</table>

5.2.3.5 Ejecución y seguimiento de las pruebas de capacidad de recuperación

La Tabla 44 muestra la sección de descripción general para la ejecución de las pruebas de capacidad de recuperación.

Tabla 44. Sección de descripción general para la ejecución de las pruebas de capacidad de recuperación.

Nombre: Fiabilidad – Capacidad de recuperación	
Fecha de Revisión: 02/02/2024 al 31/05/24 **Prioridad:** Alta **Revisor:** Berenice Ortiz Torres	
Información sobre el caso de prueba	
Descripción: Esta prueba consta en hacer la copia de seguridad (BackUp) y restauración de información en la base, para medir el tiempo de carga (Load Time)	**Prerrequisitos:** Ejecutar los procedimientos indicados en el script de automatización de la prueba capacidad de recuperación.
Datos de entrada	

```java
public static long backup()
        throws IOException, InterruptedException {
            Process process = Runtime.getRuntime().exec(command:"mongodump --db socialnetworks");
            int processComplete = process.waitFor();
            if(processComplete == 0){
                System.out.println(x:"BackUp");
                return 0;
            }else{
                return 1;
            }
        }
```

Figura 17.Función de respaldo de base de datos en Spring Boot.

```java
public static long restoration()
throws IOException, InterruptedException {
    Process process = Runtime.getRuntime().exec(command:"mongorestore --db socialnetworks dump/socialnetworks");
    int processComplete = process.waitFor();
    if(processComplete == 0){
        System.out.println(x:"RestorationDB");
        return 0;
    }else{
        return 1;
        }
    }
```

Figura 18. Función de restauración de base de datos en Spring Boot.

```python
def backup(request):
    try:
        import subprocess
        p = subprocess.run('mongodump --db socialnetworks',
                           shell=True, check=True, capture_output=True)
        print('BackUp')
        return HttpResponse(status=200)
    except IntegrityError:
        return HttpResponse(status=400)
```

Figura 19. Función de respaldo de base de datos en Django.

```python
def restorationDB(request):
    try:
        import subprocess
        p = subprocess.run('mongorestore --db socialnetworks dump/socialnetworks',
                           shell=True, check=True, capture_output=True)
        print('RestorationDB')
        return HttpResponse(status=200)
    except IntegrityError:
        return JsonResponse(status=400)
```

Figura 20. Función de restauración de base de datos en Django.

```php
public function backup()
{
    try {
        $cmd="mongodump --db socialnetworks";
        exec($cmd);
        echo "BackUp";
    } catch (\Throwable $th) {
        return $th;
    }
}
```

Figura 21. Función de respaldo de base de datos en Laravel.

```php
public function restorationDB()
{
    try {
        $cmd="mongorestore --db socialnetworks dump/socialnetworks";
        exec($cmd);
        echo "RestorationDB";
    } catch (\Throwable $th) {
        return $th;
    }
}
```

Figura 22. **Función de restauración de base de datos en Laravel.**

Procedimiento: A través del módulo y controlador se indican los procedimientos para la copia de seguridad y restauración completa de los datos.	**Resultado esperado:** Se espera el correcto respaldo y restauración de los datos para obtener el tiempo de carga, sin que se veas afectadas las colecciones.

Resultados de la ejecución del caso de prueba CP13 correspondiente al framework backend Spring Boot

La Figura 23, indica el tiempo de carga al realizar la copia de seguridad en Spring Boot.

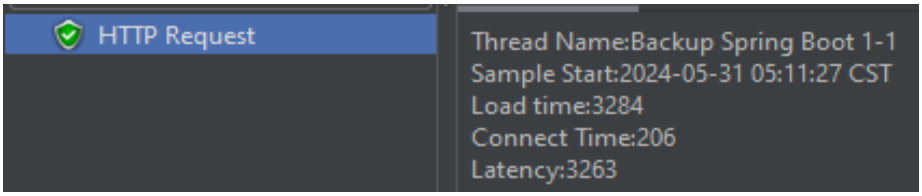


Figura 23. Tiempo de carga al realizar copia de seguridad en Spring Boot.

La Figura 24, indica el tiempo de carga al realizar la restauración de la base de datos en Spring Boot.

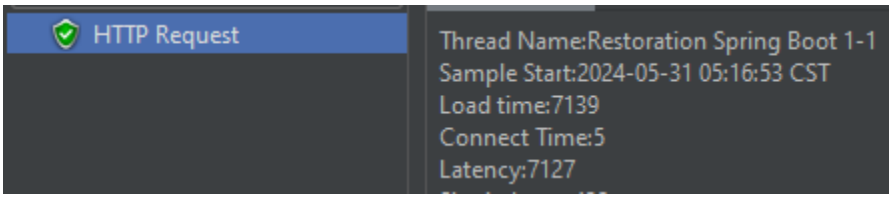


Figura 24. Tiempo de carga al realizar la restauración de la DB en Spring Boot.

Resultados de la ejecución del caso de prueba CP14 correspondiente al framework backend Django.

La Figura 25, indica el tiempo de carga al realizar la copia de seguridad en Django.

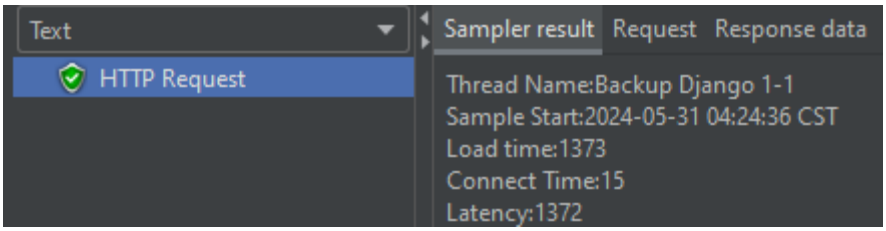


Figura 25. Tiempo de carga al realizar copia de seguridad en Django.

La Figura 26, indica el tiempo de carga al realizar la restauración de la base de datos en Django.

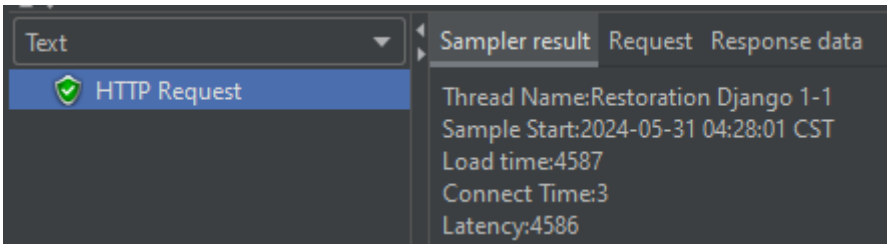


Figura 26. Tiempo de carga al realizar la restauración de la DB en Django.

Resultados de la ejecución del caso de prueba CP15 correspondiente al framework backend Laravel.

La Figura 27 indica el tiempo de carga al realizar la copia de seguridad en Laravel.

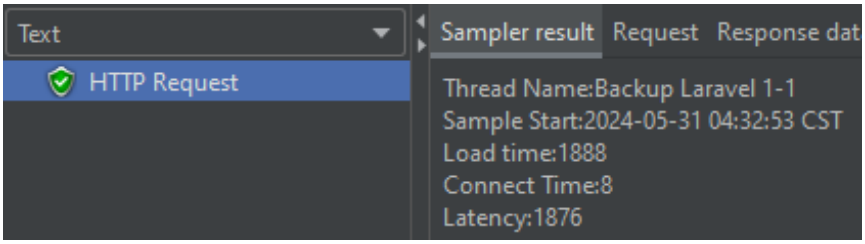


Figura 27. Tiempo de carga al realizar copia de seguridad en Laravel.

La Figura 28, indica el tiempo de carga al realizar la restauración de la base de datos en Laravel.

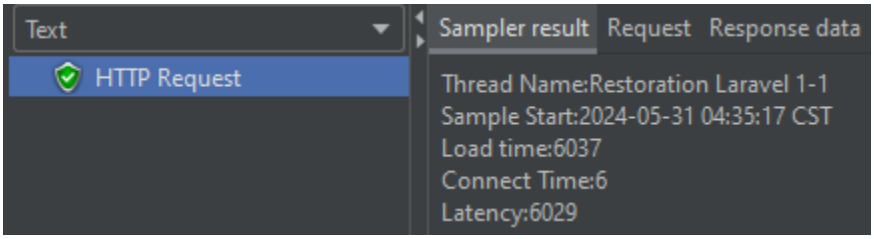


Figura 28. Tiempo de carga al realizar la restauración de la DB en Laravel.

La Tabla 45 muestra la sección de descripción general para el informe de defectos (Estatus) por cada uno de los casos de prueba de capacidad de recuperación, así como detalles, errores, soluciones, entre otras características.

Tabla 45. Informe de defectos (Estatus) para los casos de prueba de capacidad de recuperación.

		No. Caso de Prueba		
		CP13	CP14	CP15
		Estatus		
Resumen de resultados obtenidos: La ejecución de la prueba de capacidad de recuperación en los framework Spring Boot, Django y Laravel se realizó de manera	Aprobado	✓	✓	✓
	No aprobado			

		No. Caso de Prueba		
		CP13	CP14	CP15
exitosa, ya que no se obtuvo ningún error.				
	Especificar razones de no aprobación:			
	Grado de error			
	Medio Grave Severo			
Observaciones: sin observaciones	Detalles de la solución de la falla			
	Estado de la falla: Sin Falla Solución de la falla: Fecha de la última revisión: 31/05/24 Fecha de la solución:			

5.3 Cierre de pruebas y análisis de los resultados

En esta sección se describe la conclusión y cierre de las pruebas para dar informe y un resumen de los resultados de las pruebas.

5.3.1 Cierre de pruebas y resumen

Se han realizado las pruebas indicadas en los frameworks backend Spring Boot, Django y Laravel, dichas pruebas se han ejecutado de manera exitosa y no se presentaron anomalías o errores en ellas. La Tabla 46 muestra el informe en resumen de las pruebas, que contiene información de los resultados generales y finales obtenidos en el transcurso de la ejecución en las mismas y conforme al plan y subplanes de pruebas.

Tabla 46. Informe de resúmenes de las pruebas.

No funcionales					
Rendimiento – Fiabilidad					
Descripción del informe					
ID Caso de prueba	**Prueba**	**Fecha de Inicio-Fin**	**Descripción del Problema/Anomalía**	**Método de solución de la anomalía**	**Estado**
CP01 CP02 CP03	Carga	02/02/2024 al 31/05/24	No se presentó error en cada grupo de hilos definitivos durante las pruebas de carga	Sin configuración o método, dado que no existió error.	Aprobados
CP04 CP05 CP06	Comportamiento temporal	02/02/2024 al 31/05/24	No se presentó error en cada uno de los grupos de hilos de concurrencia durante las pruebas de comportamiento temporal	Sin configuración o método, dado que no existió error.	Aprobados
CP07 CP08 CP09	Utilización de recursos	02/02/2024 al 31/05/24	Al ser pruebas dependientes, no se presentaron errores en los grupos de hilos de concurrencia (referentes a las pruebas de comportamiento temporal) que detuvieran el registro del recolector de métricas en las pruebas de utilización de recursos	Sin configuración o método, dado que no existió error	Aprobados
CP10 CP11 CP12	Estrés	02/02/2024 al 31/05/24	No se presentó error cada grupo de hilo durante las pruebas de estrés	Sin configuración o método, dado que no existió error	Aprobados

No funcionales					
Rendimiento – Fiabilidad					
Descripción del informe					
ID Caso de prueba	**Prueba**	**Fecha de Inicio-Fin**	**Descripción del Problema/Anomalía**	**Método de solución de la anomalía**	**Estado**
CP13 CP14 CP15	Capacidad de recuperación	02/02/2024 al 31/05/24	No se presentó error cada árbol de resultados durante las pruebas de capacidad de recuperación	Sin configuración o método, dado que no existió error	Aprobados

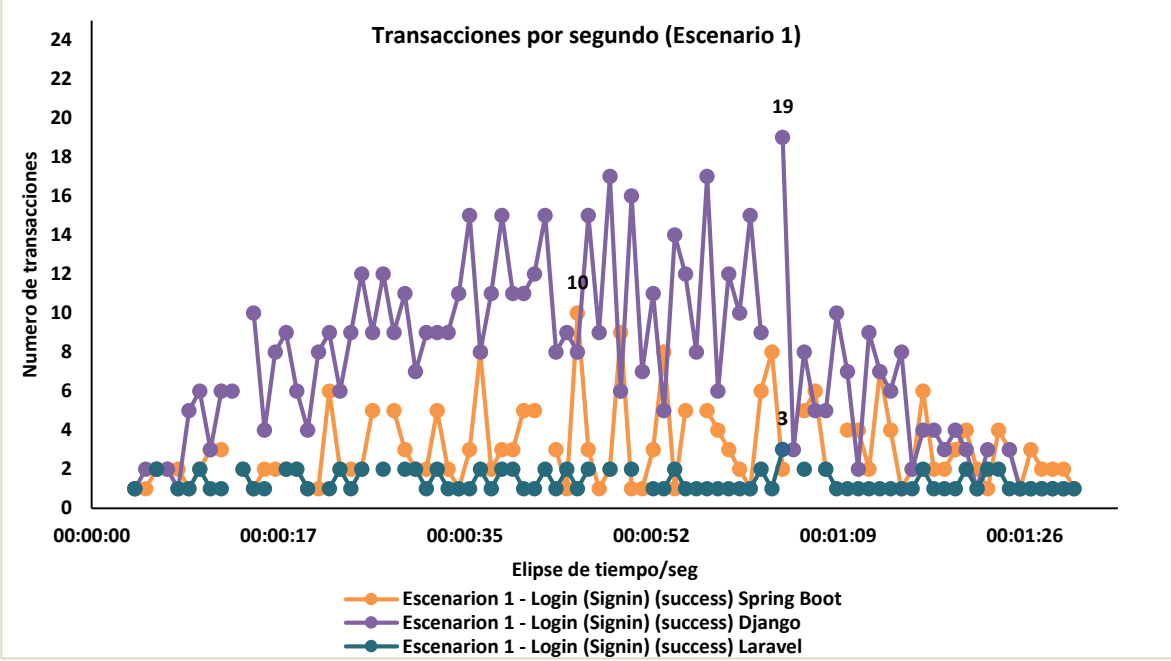
Casos de prueba	**Porcentaje**
Pendientes	0%
Ejecutados	100%
Aprobados	100%
Error	0%

5.3.2 Análisis y comparación de los resultados en las pruebas de software

5.3.2.1 Análisis y comparación de las pruebas de carga

A continuación, la siguiente comparativa corresponde, a los datos en los casos de prueba CP01, CP02 y CP03 (pruebas de carga).

Para esta primera comparativa se analizó la cantidad de transacciones procesadas por segundo que los diferentes entornos pudieron procesar durante un determinado lapso de tiempo. La Gráfica 24 muestra tres series de datos indicado el número de transacciones por segundo (TPS), para petición HTTP Request (escenario 1).

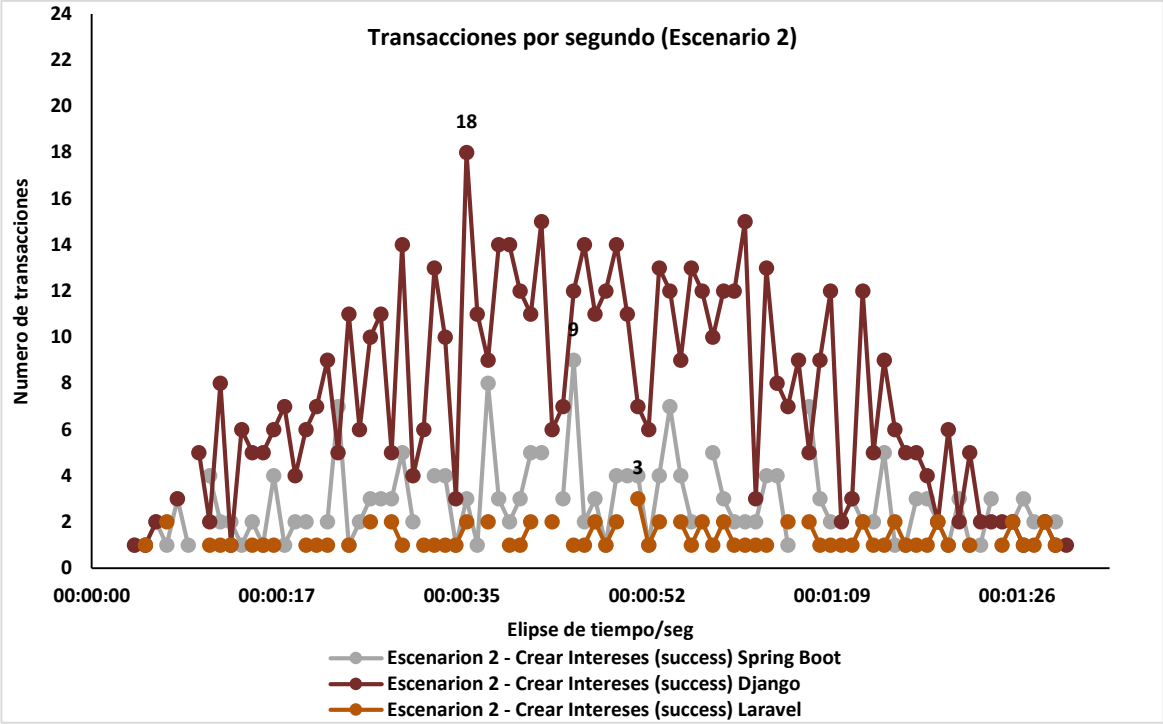


Gráfica 24. Transacciones por segundo Escenario 1.

De acuerdo a los resultados, se observa en la Gráfica 24, una diferencia amplia entre cada framework backend. Se interpreta que la rapidez y gestión de las transacciones, es procesada de diferente manera por cada uno de ellos, por lo que se tiene que, durante 90 segundos: el framework Laravel permanecía mayormente estable durante casi todo el periodo de tiempo, manejando un total de 3 transacciones como su cifra máxima. Para el framework Spring Boot que tuvo un rendimiento más elevado de 10 transacciones, lo que indica que la velocidad para procesar las peticiones es más ágil. Por último, el framework Django obtuvo una diferencia de transacciones muy considerable, manejando una capacidad para responder a la frecuencia de las peticiones más amplia durante toda la duración de la prueba, tanto en cantidad como en velocidad, obteniendo un total de 19 transacciones por segundo.

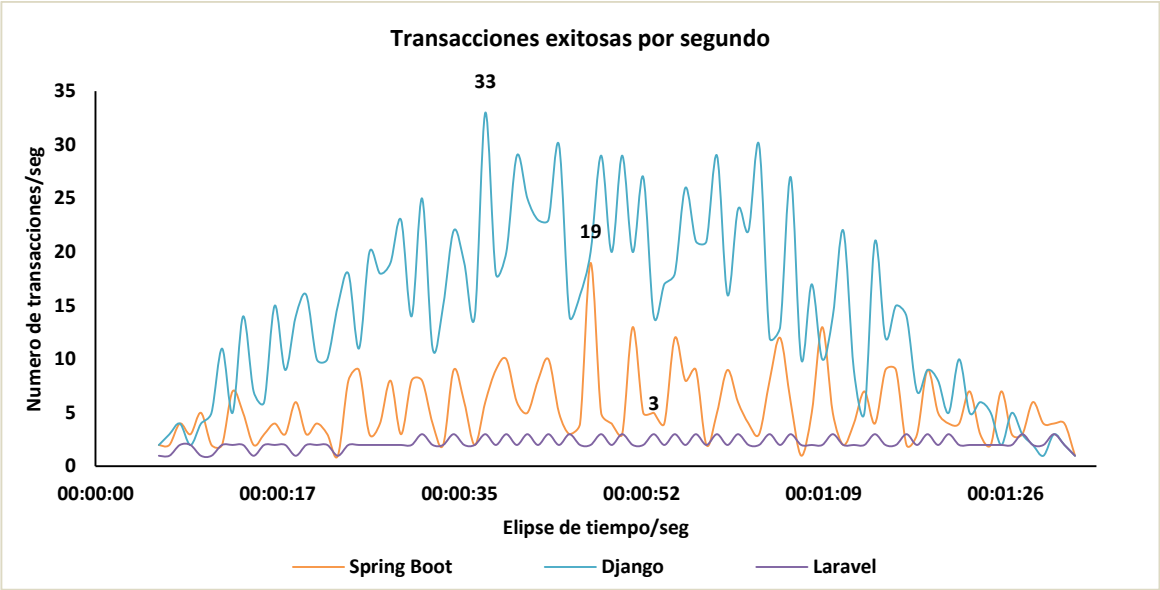
En la segunda petición HTTP Request (escenario 2), se tuvo una similitud en número de transacciones procesadas por los frameworks Django y Spring Boot, con tan solo una solicitud menos, dejándolos en 18 y 9 respectivamente.

Mientras que Laravel se mantuvo con 3 transacciones por segundo, como se muestra en la Gráfica 25.



Gráfica 25. Transacciones por segundo Escenario 2.

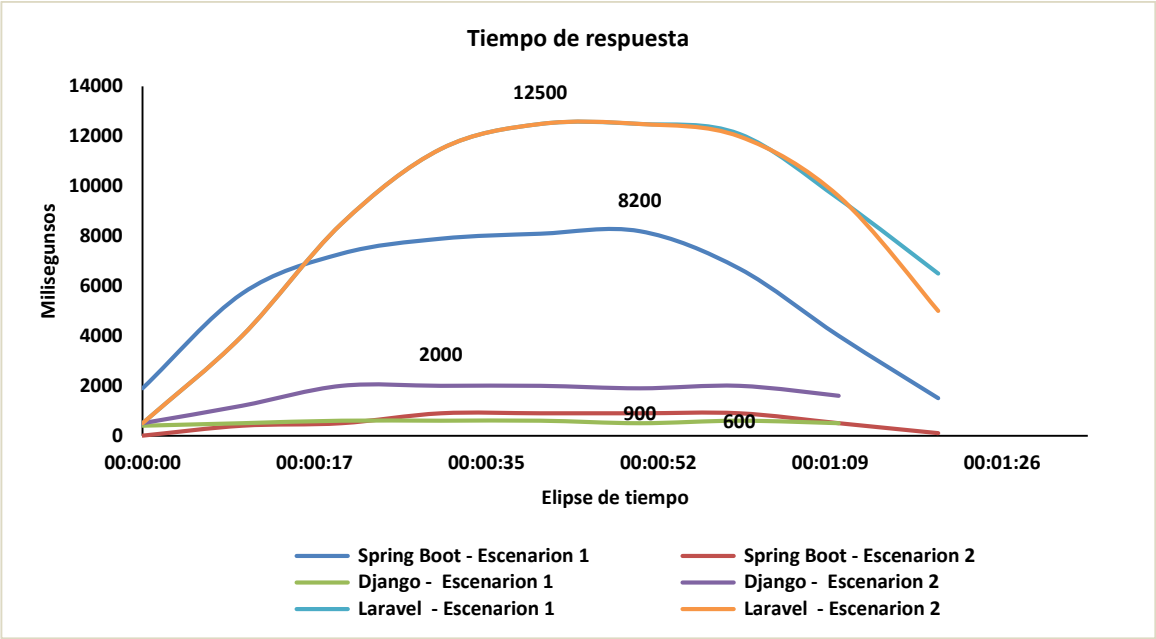
La Gráfica 26 muestra el total de transacciones exitosas por segundo. Por lo que se obtuvo que durante el periodo estable de la prueba de carga el framework Django alcanzó las 33 transacciones siendo el más alto, Spring Boot con 19 y Laravel se mantuvo una línea de tendencia fija de 3 transacciones procesadas por segundo durante el tiempo de duración de la prueba.



Gráfica 26. Transacciones exitosas por segundo.

Asimismo, para la comparación de las gráficas de tiempo de respuesta, se obtuvo el tiempo de reacción en milisegundos registrados en la primera petición http (escenario 1) situándose de la siguiente manera: Spring Boot registro un tiempo de 8200ms, mientras que Django obtuvo un tiempo de 600ms y Laravel adquirió un tiempo de 12500ms. Para la segunda petición http (escenario 2), se posicionan de acuerdo a lo siguiente: Spring Boot recabo un tiempo de 900ms, Django alcanzó un tiempo de 2000ms y Laravel se mantuvo dentro del mismo intervalo de tiempo, que en el escenario 1.

Los anteriores datos, indican que Laravel tuvo un valor no variante, si no que fue uniforme desde que iniciaron las peticiones hasta que fueron respondidas. Para Spring Boot y Django si hubo cambios en el tiempo de respuesta totalmente distintos, en ambos frameworks una de las solicitudes se elevaba más que la otra sucesivamente y siendo escenarios diferentes. Quedando así: Laravel presentó una demora de tiempo de respuesta obtenido en la prueba de carga, seguido de Spring Boot y por último Django con el mínimo tiempo, como se muestra en la Gráfica 27.



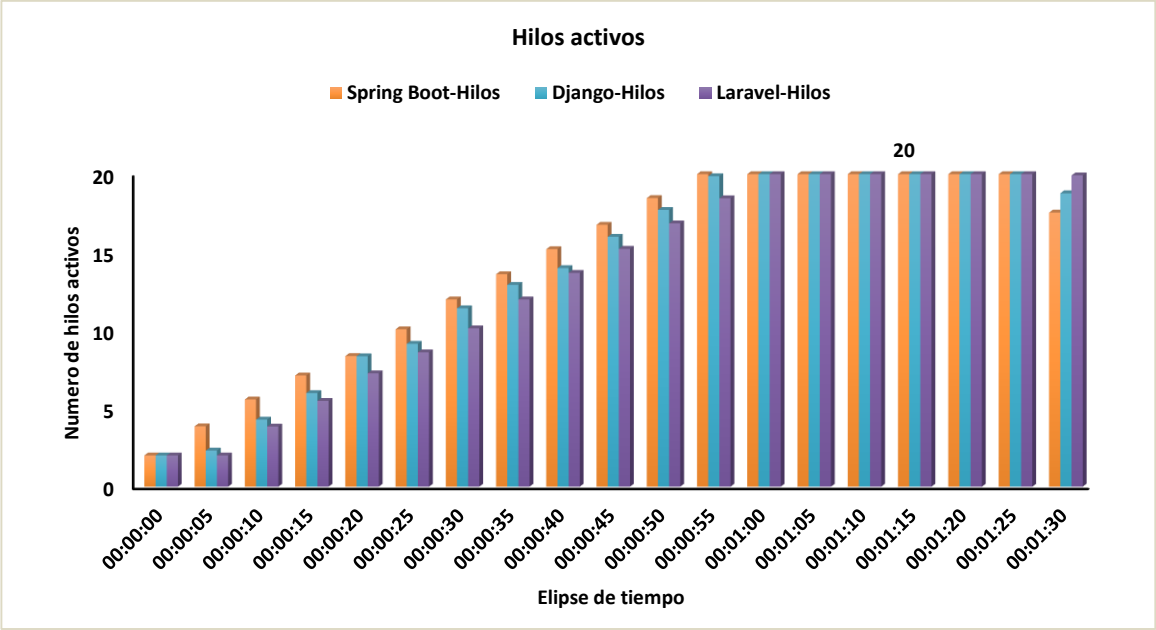
Gráfica 27. Tiempo de respuesta.

5.3.2.2 Análisis y comparación de las pruebas de comportamiento temporal

Comparativa de los casos de prueba CP04, CP05 y CP06 (pruebas de comportamiento temporal).

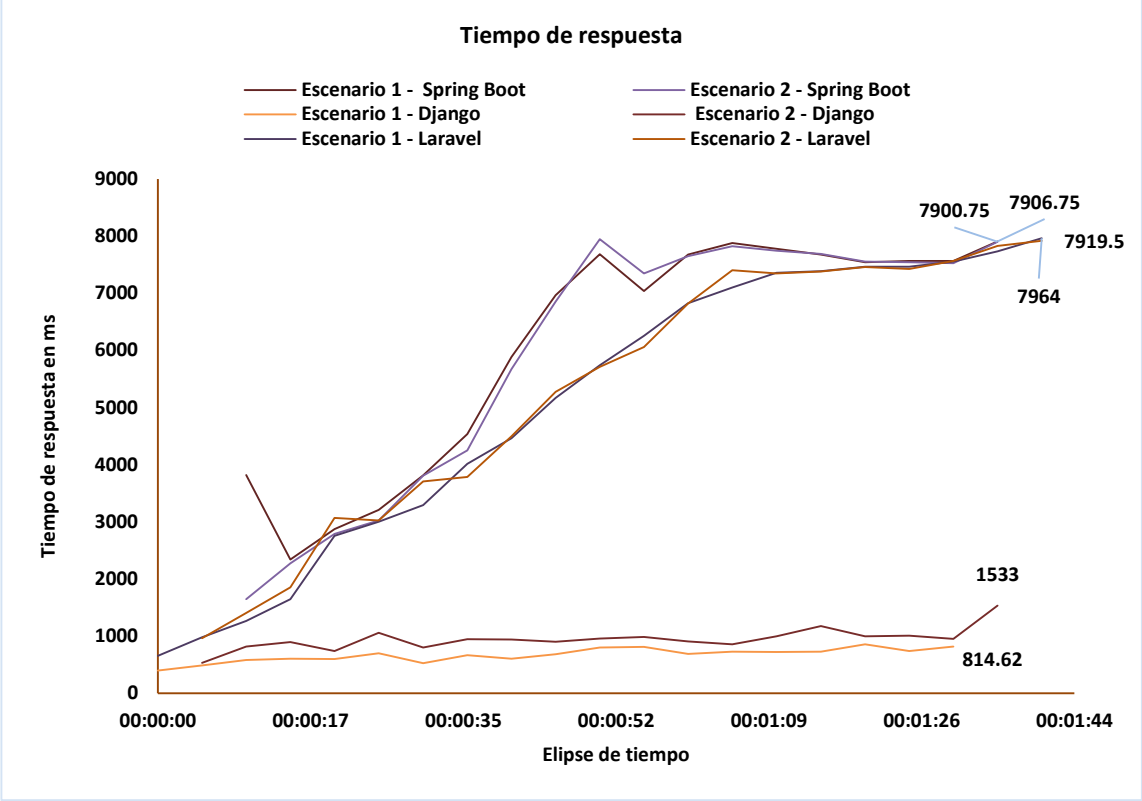
Al examinar los subprocesos activos por grupo, se observa en la Gráfica 28 en formato de columnas que los tres frameworks iniciaron la prueba con el mismo número de hilos. No obstante, al transcurrir el tiempo se disciernen, cada uno de ellos manejo entre uno y dos hilos más de diferencia, que el anterior framework, quedando de la siguiente manera por mayor valor: Spring, Django y Laravel. Por consiguiente, al llegar al recuento de pasos de la aceleración, todos tomaron el mismo valor de concurrencia en 20 hilos. Por otra parte, en el último intervalo de tiempo se tuvo un desplazamiento, es decir, Laravel se mantuvo ejecutando la misma cantidad de tareas de forma simultánea, en tanto Spring

Boot bajando a la tercera posición y Django se mantuvo en el mismo sitio con los hilos activos como se puede observar en la Grafica 28.



Gráfica 28. Hilos activos.

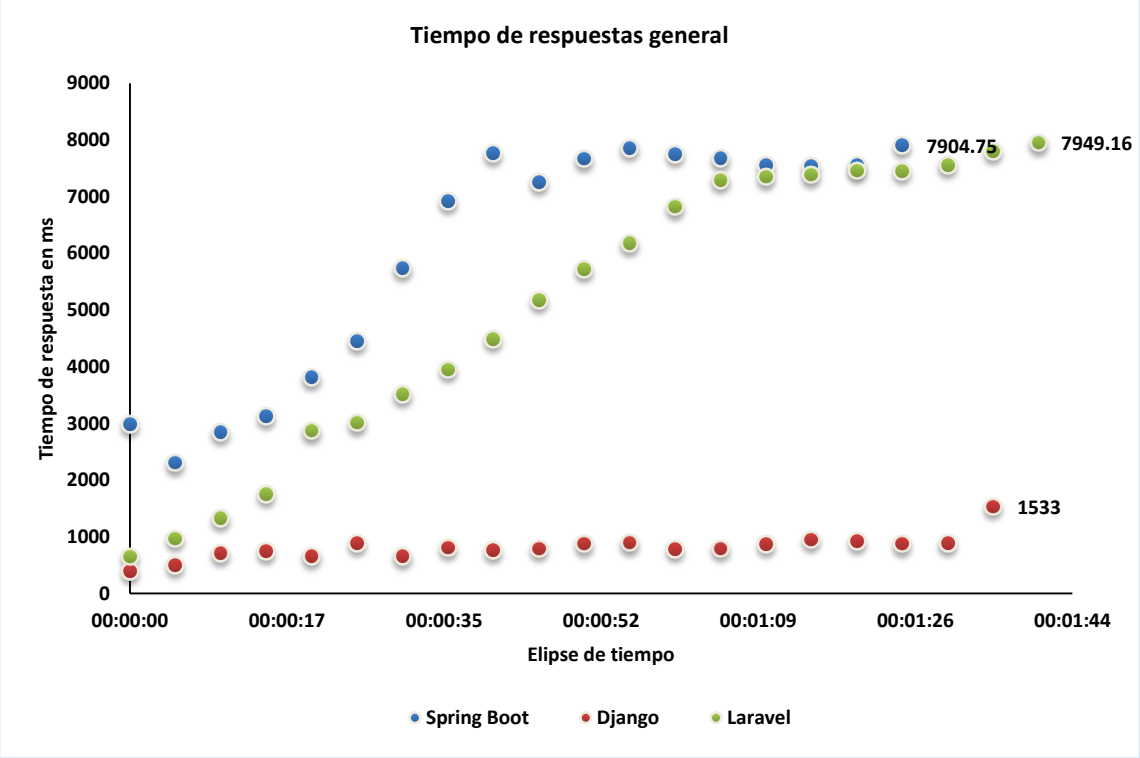
Por consiguiente, la Gráfica 29 muestra el tiempo de respuesta promedio en milisegundos para los casos de prueba de comportamiento temporal, si ordenamos de menor a mayor el tiempo de cada muestreador, podemos visualizar que en la petición 1 (escenario 1): Django presentó un tiempo de 814.62ms, Spring Boot de 7906.75ms y Laravel de7964ms. En la petición 2 (escenario 2): Django indico un tiempo de 1533ms, Spring Boot de 7900.75ms y Laravel indico un tiempo de 7919.5ms. Además, se puede observar que Laravel y Spring Boot adquirieron cantidades superiores y aproximadamente similares dentro del margen de los 7900ms, a comparación de Django que indico valores bajos que no sobrepasaron el límite de 2000ms.



Gráfica 29. Tiempo de respuesta de ambos escenarios.

La Gráfica 30 indica el tiempo de respuesta general y las muestras combinadas, es decir el promedio de las solicitudes de las peticiones (escenarios 1 y 2), por lo que procede de la información presentada en la Gráfica 29. El promedio de los valores en los datos nos da una descripción en conjunto de la duración en milisegundos por cada framework backend, los cuales organizados de forma descendente, se colocan de la siguiente forma: Laravel 7904.75ms, Spring Boot 7949.16ms, ambos relativamente parecidos, a diferencia de Django que continuó en 1533ms.

Aunque la información recolectada no varía mucho de la primera (Gráfica 29), sirve para calcular una tendencia central en el tiempo de respuesta en la prueba de comportamiento temporal.



Gráfica 30. Tiempo de respuesta general.

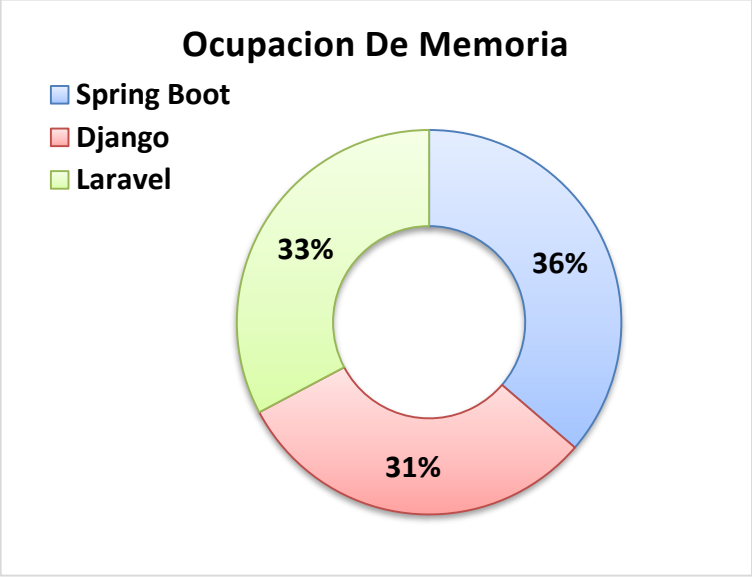
5.3.2.3 Análisis y comparación de las pruebas de utilización de recursos

Comparativa de los casos de prueba CP07, CP08 y CP09 (pruebas utilización de recursos).

Las siguientes Graficas 31, 32 y 33 expresan los porcentajes y el total de números de operaciones que describen que tan usado está el recurso: CPU, Memoria y Disco E/S, durante la ejecución de las pruebas de comportamiento temporal, ya que la prueba utilización de recursos es una prueba dependiente a las mismas.

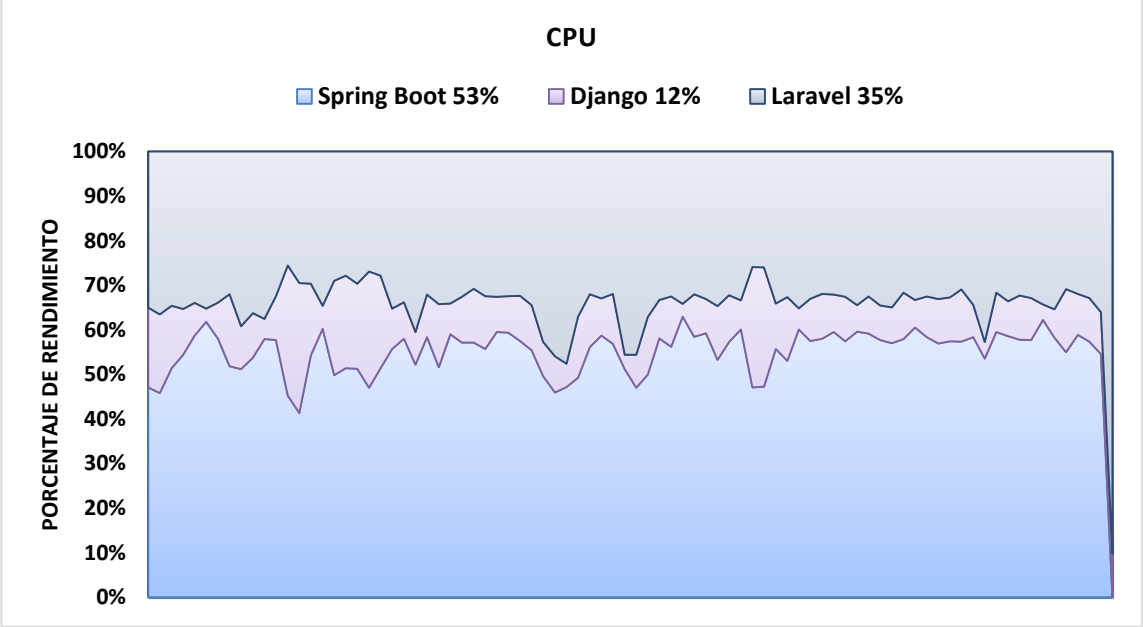
De acuerdo al rendimiento proporcionado en los resultados, la unidad métrica para el uso de la memoria fue en Bytes (b), por lo cual la Gráfica 31

representa el valor de dichos datos en porcentaje. Los valores por sector, indican que la memoria usada por Spring Boot fue una proporción del 36%, siendo el que más recursos de memoria utilizo de los tres frameworks backend, entre tanto Laravel uso 33% quedándose como intermedio y el que menos recurso empleo fue Django con el 31% de uso.



Gráfica 31. Porcentaje de ocupación en memoria.

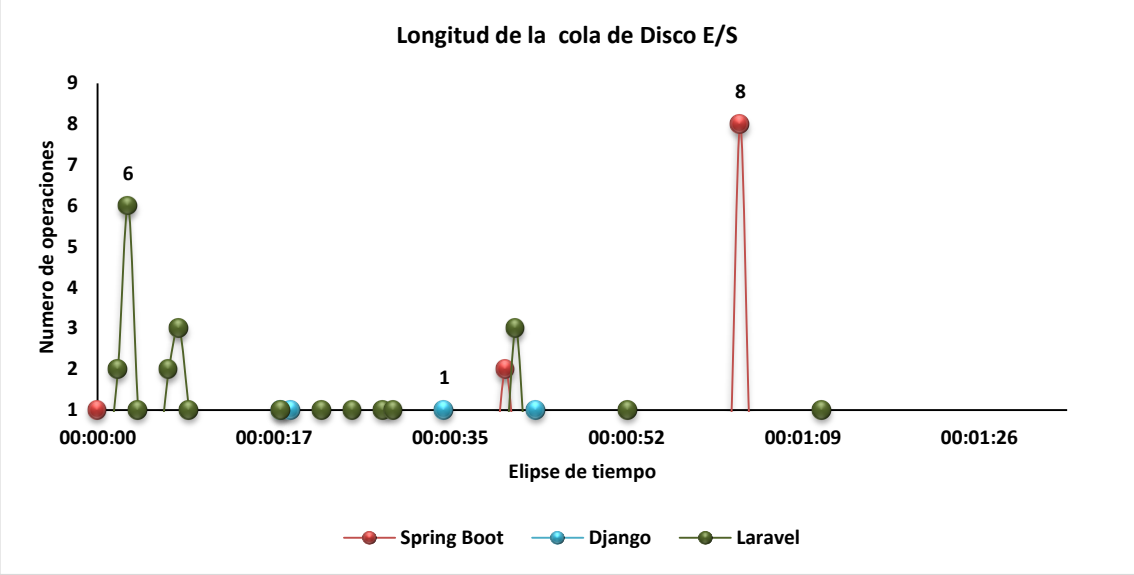
De la misma forma, el rendimiento recopilado en las métricas para la utilización del CPU, se indican en la Gráfica 32 Uso del CPU en porcentaje. Las relaciones apiladas en el gráfico permiten observar que la tendencia en los datos denota a Spring Boot con un consumo de 53%, esto refleja que el CPU estuvo mayormente ocupado en el transcurso de la prueba para ejecutar instrucciones y procesar datos. Posteriormente está Django que obtuvo un porcentaje del 12% de uso, su gestión en la realización de tareas y secuencias fue de menor ocupación. El restante le compete a Laravel, el porcentaje que el CPU uso para efectuar procesos en el momento actual fue del 35% de ocupación.



Gráfica 32. Porcentaje de CPU usado.

Entre tanto, el último recurso monitorizado fue el Disco E/S. La Gráfica 33 presenta la Longitud de la cola de Disco E/S con valores mayores a 0, en el intervalo de tiempo de la prueba.

Conforme a la métrica longitud de cola (queue), el punto más alto en la gráfica compete a Spring Boot, que alcanzó una cantidad máxima de 8 operaciones pendientes. El punto medio le sigue a Laravel, con un número máximo de 6 solicitudes, aunque con una mayor recopilación de picos presentes en cola mayores o iguales a 1. Por último, el mínimo de operación pendientes en Django fue de 1, con poca frecuencia de registros de picos, igualitarios a su mínimo valor. Aunque los frameworks Spring Boot y Django obtuvieron sólo tres registros de picos como similitud, hay una gran diferencia entre los datos obtenidos.



Gráfica 33. Número de operaciones en el Disco E/S.

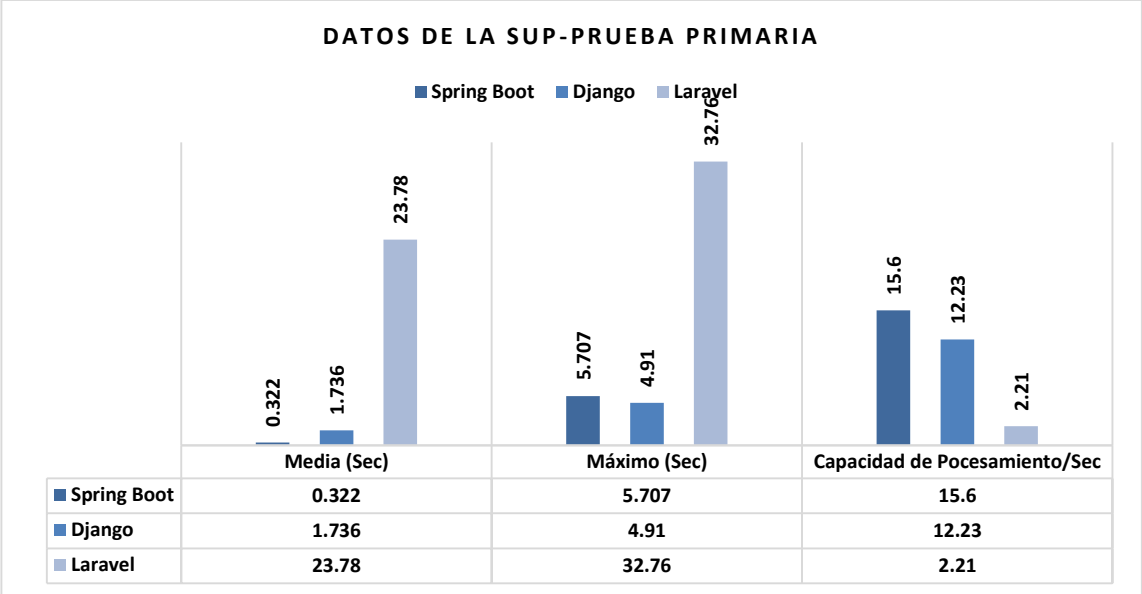
5.3.2.4 Análisis y comparación de las pruebas de estrés

Comparativa de los casos de prueba CP10, CP11 y CP12 (pruebas de estrés), para esto se convirtieron a segundos y analizaron los datos: Media, Máximo y Cantidad de procesamiento, transcritos de las tablas de Reporte agregado en los resultados en las pruebas de estrés.

La Gráfica 34 presenta datos de la subprueba primaria en las pruebas de estrés. Esta categoría tuvo una muestra de 8437 veces en que se efectuó la petición, por lo cual los valores obtenidos indican los siguientes resultados en los datos.

- Media (Subprueba Primaria): el menor valor de tiempo medio, del conjunto de muestras logrado lo obtuvo Spring Boot con 0.322s, posterior a Django que adquirió 1.736s y finalmente Laravel alcanzando un tiempo de 23.78s, lo cual indica un mayor tiempo de respuesta.

- Máxima (Subprueba Primaria): el máximo punto donde la aplicación se demoró en hacer la petición correspondió a Laravel que presentó 32.76s como tiempo de respuesta. Después, con una variada diferencia de segundos respecto a Laravel, fue Sprint Boot sobre un tiempo de 5.707s y Django, tomando el menor tiempo registrado con 4.91s.
- Cantidad de procesamiento (Subprueba Primaria): pasado el periodo de tiempo definido, la mejor cantidad trabajo completado fue registrado por Spring Boot consiguiendo 15.60/s, continuo de Django, que produjo 12.23/s y por último Laravel con una cantidad menor de 2.21/s.

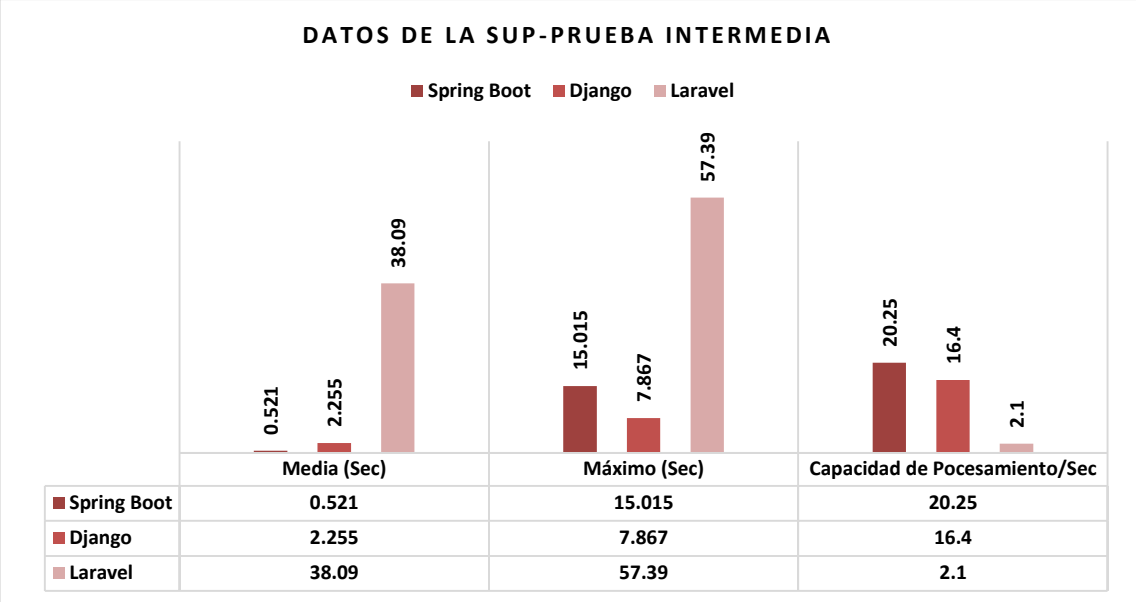


	Media (Sec)	Máximo (Sec)	Capacidad de Pocesamiento/Sec
Spring Boot	0.322	5.707	15.6
Django	1.736	4.91	12.23
Laravel	23.78	32.76	2.21

Gráfica 34. Datos de la subprueba primaria (Estrés).

Las muestras para la subprueba intermedia y la subprueba elevada, tuvieron un aumento entre el 50% y 100% respectivamente, según la planificación de la prueba de estrés en la Fase 2: Etapa 3 - Implementación y ejecución de las pruebas.

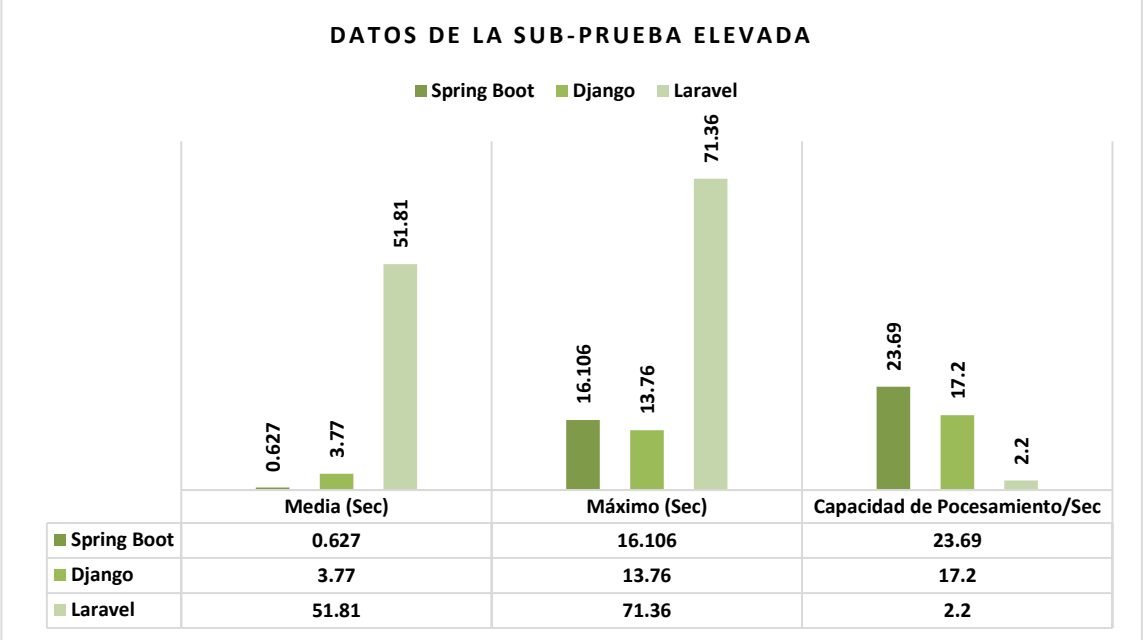
La Gráfica 35 muestra datos de la subprueba intermedia, referente al receptor AR (Tablas 35, 38 y 41) en las pruebas de estrés. Esta subprueba tuvo una muestra de 12727 veces en que se efectuó la petición.



	Media (Sec)	Máximo (Sec)	Capacidad de Pocesamiento/Sec
Spring Boot	0.521	15.015	20.25
Django	2.255	7.867	16.4
Laravel	38.09	57.39	2.1

Gráfica 35. Datos de la subprueba intermedia (Estrés).

La Gráfica 36 indica datos de la subprueba elevada, referente al receptor AR (Tablas 36, 39 y 42) en las pruebas de estrés. Esta subprueba tuvo una muestra de 17017 veces en que se efectuó la petición.



	Media (Sec)	Máximo (Sec)	Capacidad de Pocesamiento/Sec
Spring Boot	0.627	16.106	23.69
Django	3.77	13.76	17.2
Laravel	51.81	71.36	2.2

Gráfica 36. Datos de la Subprueba elevada.

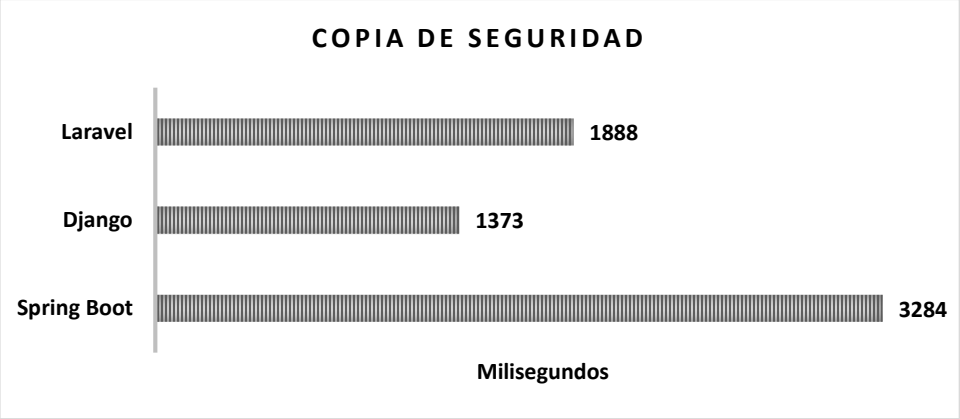
Como se observó en las diferentes subpruebas incrementales de la prueba de estrés, el tiempo en las distintas métricas: Media, Máximo y la Cantidades de procesamiento, fueron incrementando correlativamente hasta llegar a un punto en donde empieza a disminuir las transacciones por la concurrencia, determinando una aproximación de cuando el sistema empieza a bajar su nivel de respuestas.

De igual importancia, el orden dado en los resultados de la comparación de la subprueba primaria, se conservó. Es decir, por las tres subpruebas: Sprint Boot obtuvo mejores resultados en la media, la máxima y la Cantidad de procesamiento, posterior a este fue Django y luego Laravel.

5.3.2.5 Análisis y comparación de las pruebas de capacidad de recuperación

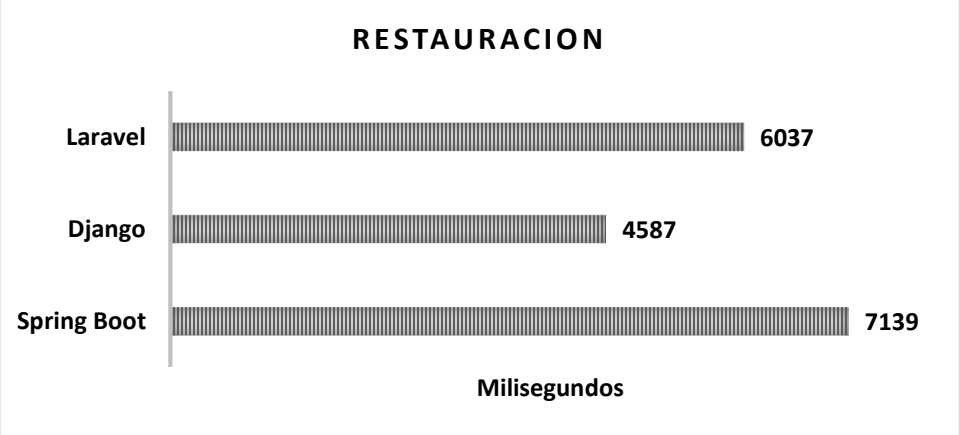
Comparativa de los casos de prueba CP13, CP14 y CP15 (prueba de capacidad de recuperación).

La Gráfica 37 representa, el tiempo de carga que tardó cada framework backend en realizar la copia de seguridad (completa). Como se muestra el menor tiempo lo registró Django con 1373ms, seguido de Laravel con 1888ms y por último Spring Boot con tiempo de 3284ms.



Gráfica 37. Copia de seguridad.

De la misma manera, el menor tiempo que demoró cada framework backend en realizar la restauración de la información (completa) lo obtiene Django con 4587ms, después con 6037ms a Laravel y por consiguiente Spring Boot con 7139ms. La Gráfica 38 muestra, el tiempo de carga, en la restauración de la DB.



Gráfica 38. Restauración.

5.3.2.6 Discusión de los resultados obtenidos

En este apartado, se presenta la discusión de los resultados obtenidos en los casos de pruebas y en la comparación de las gráficas.

Para valorar el desempeño de los frameworks backend se utilizaron las métricas de los receptores o escuchadores, que recopilan la información obtenida por JMeter en el transcurso de la ejecución de cada una de las distintas pruebas y se empleó la evaluación cualitativa y cuantitativa, como instrumentos de medición sobre escalas a evaluar y valores sumarios en puntajes. La Tabla 47 indica la escala valorativa y puntaje.

Tabla 47. Escala valorativa y puntaje.

Escala valorativa	Puntaje
Bien	1
Muy Bien	2
Excelente	3

La Tabla 48 muestra los puntajes asignados a los frameworks backend en las pruebas de carga, por lo que se observa que el mayor puntaje lo obtiene

Django con 9 puntos, lo que indica que tuvo mayor eficiencia al procesar las transacciones y mejor tiempo de respuesta, posteriormente le sigue Spring Boot con 6 puntos y Laravel con 3 puntos.

Tabla 48. Puntuación asignada a los frameworks backend en las pruebas de carga.

	Framework Backend		
	Spring Boot	Django	Laravel
Criterio de evaluación			
Mayor número de transacciones procesadas sobre segundo	2	3	1
Mayor totalidad de transacciones exitosas sobre segundo	2	3	1
Menor tiempo de respuesta en milisegundos	2	3	1
Total	**6**	**9**	**3**

La Tabla 49 muestra los puntajes asignados a los frameworks backend en las pruebas de comportamiento temporal. En esta evaluación, el mayor número de hilos, fue obtenido por Spring Boot, sin embargo, el menor tiempo de respuesta, fueron manejados por Django, sumando 8 puntos en total y siendo el mejor. Después fue Spring Boot con 7 puntos y Laravel con 3 puntos.

Tabla 49. Puntuación asignada a los frameworks backend en las pruebas de comportamiento temporal.

	Framework Backend		
	Spring Boot	Django	Laravel
Criterio de evaluación			
Mayor número de hilos activos	3	2	1
Menor tiempo de respuesta en milisegundos	2	3	1
Menor tiempo de respuesta general en milisegundos	2	3	1
Total	**7**	**8**	**3**

Para los tres criterios de evaluación el framework backend Django obtuvo excelentes puntajes, por consiguiente, fue el más alto con 9 puntos. Luego de

esto, el segundo framework que utilizo menos recursos del computador fue Laravel con 5 puntos y por último con 4 puntos, Spring Boot como se muestra en la Tabla 50 que indica los puntajes asignados a los frameworks backend en las pruebas de utilización de recurso.

Tabla 50. Puntuación asignada a los frameworks backend en las pruebas de utilización de recursos.

	Framework Backend		
Criterio de evaluación	Spring Boot	Django	Laravel
Menor uso de memoria	1	3	2
Menor consumo del CPU	1	3	2
Menor cantidad de operaciones pendientes en el Disco E/S	2	3	1
Total	**4**	**9**	**5**

En las pruebas de estrés las métricas: mínimo valor obtenido en la media y mayor cantidad de procesamientos fueron obtenidas por el framework Spring Boot, el cual tuvo un puntaje total de 8, seguido de Django con 7 puntos y finalmente Laravel con 3 puntos, como se muestra en la Tabla 51.

Tabla 51. Puntuación asignada a los frameworks backend en las pruebas de estrés.

	Framework Backend		
Criterio de evaluación	Spring Boot	Django	Laravel
Mínimo valor obtenido en la Media	3	2	1
Mínimo valor como punto de demora obtenido en los Máximos	2	3	1
Mayor cantidad de procesamientos (Throughput)	3	2	1
Total	**8**	**7**	**3**

El menor tiempo de carga que transcurrió desde el momento en que se envió la petición hasta el momento de recibir en su totalidad la última parte de la respuesta por parte del servidor, lo registró Django con un total de 6 puntos. El

segundo menor tiempo lo asentó Laravel, su total fue de 4 puntos y el tercero fue Spring Boot con 2 puntos. La Tabla 52 muestra la puntuación asignada a los frameworks backend en las pruebas de capacidad de recuperación.

Tabla 52. Puntuación asignada a los frameworks backend en las pruebas de capacidad de recuperación.

	Framework Backend		
Criterio de evaluación	Spring Boot	Django	Laravel
Menor tiempo de carga que transcurrió en realizar la copia de seguridad	1	3	2
Menor tiempo de carga que transcurrió en hacer la restauración de la DB	1	3	2
Total	**2**	**6**	**4**

En este caso los tres frameworks backend obtuvieron resultados finales satisfactorios y aceptables. Como se puede ver Django sacó una ventaja de 39 puntos sobre Spring Boot y Laravel, tanto en las pruebas de rendimiento como en la prueba de fiabilidad. De forma similar Spring Boot obtuvo ventaja sobre Laravel, con una diferencia de 27 y 18 puntos respectivamente en ambas pruebas. La Tabla 53 muestra las puntuaciones totales de los frameworks backend. En general, Django obtuvo el primer lugar, Spring Boot el segundo lugar y Laravel el tercer lugar.

Tabla 53. Puntuaciones totales de los frameworks backend.

		Framework Backend		
		Spring Boot	Django	Laravel
Características de prueba	**Subcaracterísticas de prueba**	**Puntajes**		
Rendimiento	Carga	6	9	3
	Comportamiento temporal	7	8	3
	Utilización de recursos	4	9	5
	Estrés	8	7	3

		Framework Backend		
		Spring Boot	Django	Laravel
Características de prueba	**Subcaracterísticas de prueba**	**Puntajes**		
Fiabilidad	Capacidad de recuperación	2	6	4
	Puntuación Total/Final	**27**	**39**	**18**

CAPÍTULO 6. CONCLUSIONES Y RECOMENDACIONES

6.1 Conclusiones

Al trabajar conforme a los procesos, actividades y directrices de estándares y normas de calidad, se fomenta la elaboración de productos sustentados en buenas prácticas y guías. De ahí la importancia, que en esta investigación se basará en la organización ISTQB (International Software Testing Qualifications Board) conforme al proceso de prueba, tipos y técnicas de prueba. Así también, se indago en las normas ISO 29119 y la norma ISO 25010 para la retroalimentación de este estudio.

Primordialmente, para cumplir con los objetivos de esta investigación, se ejecutaron cinco pruebas no funcionales, que se categorizaron en las pruebas de rendimiento (Carga, Comportamiento temporal, Utilización de recursos, Estrés) y pruebas de fiabilidad (Capacidad de recuperación). Así pues, en base a las métricas establecidas en la ejecución de las pruebas, se analizaron y compararon los frameworks backend Spring Boot, Django y Laravel que actual y frecuentemente son utilizados en la industria del software. Los resultados obtenidos en las pruebas, se concluyen de la siguiente manera:

En la prueba de carga el framework Django obtuvo una mejor totalidad de transacciones exitosas y procesadas manejando un lapso de respuestas sobre el tiempo mínimo. Secundariamente, el framework Spring Boot presentó cantidades de transacciones más bajas con periodos de tiempo un tanto más prolongados. Al lado de Django y Spring Boot, el framework Laravel manejó la menor cantidad de transacciones procesadas con un tiempo de respuesta más amplio.

En la prueba de comportamiento temporal, de acuerdo a la medición de los hilos activos el framework Spring Boot mantuvo la mejor cifra. Sin embargo, fue superado por el framework Django. Es decir, tanto el framework Spring Boot como Laravel presentaron tiempos de respuesta generales con más retraso que Django, estos tiempos se encontraban aproximadamente en el mismo rango de valor, pero Spring Boot un poco menos que Laravel respectivamente.

En la prueba de utilización de recursos, el menor consumo que se registró al emplear los medios del computador, referentes a las métricas del uso de la memoria, del CPU y las operaciones pendientes de escritura y lectura en el disco, se obtuvo por el framework Django, lo que lo hace favorable. De forma similar, el framework Laravel también manejó buenos resultados en el consumo de los mismos, excepto que experimentó más números de acciones pendientes y continuas en el disco. Por el contrario, el framework Spring Boot tuvo el mayor consumo de los tres frameworks, tanto en ejecuciones, memoria y CPU.

En la prueba de estrés, el framework que mantuvo mejores resultados de las tres subpruebas incrementales, fue Spring Boot, consiguiendo una mayor cantidad de procesamientos y valores mínimos en las métricas media y máximo. Después, persistió el framework Django manteniendo un mínimo valor de demora relativo a la solitud de respuesta en los máximos. El framework Laravel marcó y preservó un número menor de pedidos procesados con un tiempo de respuesta más extenso correspondiente a los conjuntos de muestra en la media y en la máxima.

Para terminar, en la prueba de capacidad de recuperación, se verificó el menor tiempo de carga que transcurrió al realizar la copia de seguridad y la restauración de la base de datos. Debido a lo cual, el framework Django logró la mínima duración avanzada de inicio hasta fin de la solicitud. Continuó a este, siguió el framework Laravel y por último el framework Spring Boot.

El desempeño de los frameworks backend comparados fue productivo y confortable. No obstante, en función y según la información adquirida se puede

concluir que Django obtuvo excelentes resultados y puntuaciones en las pruebas de rendimiento y fiabilidad, manejando una gran velocidad para procesar y responder las transacciones. Mientras que Laravel ocupó el tercer lugar en cada una de las pruebas, mantienen una prominente gestión y estabilidad que no presenta errores al momento de recibir las peticiones, si no que va tratándolas con moderación. Finalmente, Spring Boot es la parte intermedia entre los frameworks Django y Laravel, ya que consumió más recursos del computador y su velocidad no fue tan rápida como Django, pero manejaba una muy buena organización al responder las peticiones.

Por estas cuestiones, se concluye, que cada uno de ellos está hecho y pensado para resolver distintas necesidades y funcionalidades en específico. En virtud y consecuencia de lo cual, es responsabilidad tanto del arquitecto de software como del equipo de trabajo la elección para trabajar con uno o más framework backend, esto dependerá del tipo y constitución global del proyecto, dado que, serán factores claves que incidirán en este resultado.

6.2 Recomendaciones

Conforme a los objetivos cumplidos y el progreso logrado en el análisis y comparación de frameworks backend por medio de implementación de pruebas de software, se sugieren las siguientes recomendaciones:

- Mejorar el entorno de prueba: Implementar pruebas en equipos con una arquitectura escalable tanto vertical como horizontal, más amplia y robustas.
- Explorar otras pruebas: Según las funcionalidades del sistema, buscar la ejecución de otro tipo de pruebas de software que retribuyan más información.

6.3 Trabajo futuro

El trabajo futuro que se proponen para mejorar y dar continuidad a la implementación de pruebas para el análisis y comparación de frameworks backend, son las siguientes:

- Adición de frameworks backend: Agregar nuevos propuesta de frameworks backend a la lista comparativa con el propósito de analizarlos y generar más referencias de los mismos.
- Adición de pruebas y métricas: incluir pruebas de capacidad, escalabilidad e inestabilidad, entre otras pruebas, según lo óptimo. Así, como distintos componentes o escuchadores de métricas que atribuyan y generen más datos comparativos.
- Aumento de hilos virtuales: planear y ejecutar pruebas con mayores cantidades de usuarios virtuales y tiempos más prolongados.
- Servidor dedicado: emplear un espacio virtual dedicado y exclusivo para realizar pruebas de software con recursos más escalables.
- Software en ambiente de producción: hacer pruebas con aplicaciones o sistemas en entornos.

Referencias

[1] Statista, «Statista,» 2023. Disponible en: https://es.statista.com/.

[2] S. a. Data, «Framework backend mas popular – 2012/2023,» 2023. Disponible en: https://statisticsanddata.org/data/most-popular-backend-frameworks-2012-2023/.

[3] S. Overflow, «2023 Encuesta para desarrolladores » 13 Junio 2023. Disponible en: https://survey.stackoverflow.co/2023/.

[4] Acropolium, «Framework backend mas popular 2024 [Pros and contras, que elegir],» 30 Agosto 2023. Disponible en: http://acropolium.com/blog/most-popular-backend-frameworks-in-2021-2022-pros-and-cons-what-to-choose/.

[5] M. Gadhavi, «Framework backend mas popular para el desarrollo web 2024,» 16 Junio 2024. Disponible en: https://radixweb.com/blog/best-backend-frameworks.

[6] L. Minvielle, «Los 7 frameworks backend mas populares para desarroladores,» 17 Marzo 2024. Disponible en: https://www.wearedevelopers.com/magazine/most-popular-backend-frameworks.

[7] ICTEA, «¿Qué es una aplicación web?,» 2023. Disponible en: www.ictea.com/cs/index.php?rp=/knowledgebase/4205/iQue-es-una-aplicacion-web.html.

[8] Arimetrics, «Qué es backend,» 2022. Disponible en: https://www.arimetrics.com/glosario-digital/backend.

[9] Nicole, «Qué es frontend y backend: diferencias y características - Platzi,» 2019. Disponible en: https://platzi.com/blog/que-es-frontend-y-backend/.

[10] L. M. A. MARIBEL, «Análisis de factores que inciden en la selección de un lenguaje y framework de programación para desarrollo de software web,» 2020.

[11] N. S. Samper, «Análisis de frameworks para desarrollo de aplicaciones móviles y web,» 2019.

[12] E. BELLO, «Framework: Qué es, para qué sirve y por qué deberías usarlo,» 27 12 2021. Disponible en: https://www.iebschool.com/blog/framework-que-es-agile-scrum/. [Último acceso: 23 2 2023].

[13] V. Morelli, «¿Qué es un framework de desarrollo backend?,» 25 Febrero 2019. Disponible en: https://wheelhub.es/blog/que-es-un-framework-de-desarrollo-backend/.

[14] O. SL, «¿Qué es un framework y qué tipos hay?,» 2022. Disponible en: https://open-bootcamp.com/aprender-programar/que-es-un-framework.

[15] Intelequia, «Ciclo de vida del software: Todo lo que necesitas saber,» 28 Noviembre 2020. Disponible en: https://intelequia.com/blog/post/ciclo-de-vida-del-software-todo-lo-que-necesitas-saber.

[16] J. Turrado, «Qué son las pruebas de software,» 10 Marzo 2020. Disponible en: https://www.campusmvp.es/recursos/post/que-son-las-pruebas-de-software.aspx.

[17] S. Solera, «Las mejores metodologías para un correcto desarrollo de software,» 27 04 2022. Disponible en: https://www.occamagenciadigital.com/blog/las-mejores-metodologias-para-un-correcto-desarrollo-de-software.

[18] «Pruebas de software,» 10 03 2023. Disponible en: https://es.wikipedia.org/wiki/Pruebas_de_software.

[19] I. Corporation, «¿Qué es la prueba de software?,» 10 03 2023. Disponible en: https://www.ibm.com/mx-es/topics/software-testing.

[20] M. García, «MVC (Modelo-Vista-Controlador): ¿Qué es y para qué sirve?,» 5 Octubre 2017. Disponible en: https://codingornot.com/mvc-modelo-vista-controlador-que-es-y-para-que-sirve.

[21] J. A. C. PACHECO, «Diseño de un backend escalable de recolecion y analisis de datos georeferenciados obtenidos via crowdsourcing,» 2017.

[22] R. Hat, «Interefaz de programacion de aplicaciones,» 31 Julio 2023. Disponible en: https://www.redhat.com/es/topics/api/what-is-a-rest-api.

[23] J. L. M. M. Z. O. M. L. S. E. Molina Ríos, «Evaluación de los frameworks en el desarrollo de aplicaciones web con python,» 2016.

[24] R. F. D. ZÁRATE, «Aplicación de métricas de calidad en uso utilizando la ISO 9126 para determinar el grado de satisfacción del sistema único de matrícula,» 2016.

[25] «Calidad en los sistemas de informacion,» 26 7 2017. Disponible en: http://calidadsi17.blogspot.com/2017/07/42-la-norma-isoiec-9126.html.

[26] N. I. 25000, «Normas ISO 25000,» 2022. Disponible en: https://iso25000.com/index.php/normas-iso-25000.

[27] N. I. 25000, «ISO/IEC 25010,» 2022. Disponible en: https://iso25000.com/index.php/normas-iso-25000/iso-25010.

[28] INTECO, «INTE/ISO/IEC/IEEE 29119-2:2020,» 31 7 2020. Disponible en: https://www.inteco.org/shop/inte-iso-iec-ieee-29119-2-2020-ingenieria-de-software-y-de-sistemas-pruebas-de-software-parte-2-procesos-de-prueba-7654#attr=.

[29] Testeando software, «ISTQB. ¿Qué es? ¿Cuales son los niveles de certificación?,» 2022. Disponible en: https://testeandosoftware.com/istqb-que-es-cuales-son-los-niveles-de-certificacion/.

[30] J. M. S. Peño, «Pruebas de software. Fundamentos y tecnicas.,» 2015.

[31] ISTQB, «Probador certificado del ISTQB,» 2018

[32] «Fundamentos del Proceso de Prueba – ISTQB (1/6),» 2022. Disponible en: https://metodoneritas.com/fundamentos-del-proceso-de-prueba-istqb-1-6/#4-proceso-de-prueba.

[33] R. M. Velasquez, «Implementacion de un modelo de gestion de pruebas de software segun ISTQB para mejorar el proceso del area de certificacion en tecnologias web de una identidad financiera,» 2020.

[34] N. G. Rodriguez, «Las Pruebas de Integración como Proceso de la Calidad del Software en el Ámbito de las Telecomunicaciones,» 2015.

[35] «Tipos de pruebas de software definidos por el ISTQB,» 2018. Disponible en: http://www.pmoinformatica.com/2014/01/tipos-de-pruebas-de-software-istqb.html.

[36] L. M. A. MARIBEL, «Análisis de factores que inciden en la selección de un lenguaje y framework de programación para desarrollo de software web,» 2020.

[37] N. F. F. Peralta, «Análisis comparativo de frameworks open source para el nivel de productividad en los proyectos de desarrollo de software web,» 2019.

[38] R. Espinosa-Hurtado, «Análisis comparativo para la evaluación de frameworks usados en el desarrollo de aplicaciones web,» 2021.

[39] «Spring Framework,» 21 Marzo 2023. Disponible en: https://es.wikipedia.org/wiki/Spring_Framework.

[40] Y. Muradas, «Qué es spring framework y por qué usarlo,» 5 Junio 2018. Disponible en: https://openwebinars.net/blog/conoce-que-es-spring-framework-y-por-que-usarlo/.

[41] D. V. Pomata, «Arquitectura de desarrollo web con Django y Apps con Flutter,» 2020.

[42] «Django (framework),» 1 12 2022. Disponible en: https://es.wikipedia.org/wiki/Django_(framework).

[43] «Laravel,» 16 2 2023. Disponible en: https://es.wikipedia.org/wiki/Laravel.

[44] I.-. I. O. F. Standardization, «ISO/IEC/IEEE 29119-2:2021 Ingenieria de software y sitemas - Parte 2: Prrocesos de prueba,» 2021. Disponible en: https://www.iso.org/obp/ui/en/#iso:std:79428:en.

[45] QAwerk, «¿Qué es el ciclo de vida de las pruebas de software (STLC)? Guía paso a paso,» 12 03 2019. Disponible en: https://qawerk.es/blog/que-es-el-ciclo-de-vida-de-las-pruebas-de-software/.

[46] I. S. T. Q. Board, «Analista de pruebas técnicas V4.0,» 30 de junio del 2021.

[47] S. K. Datareportal, «digital 2023: Informe general global, » 26 Enero 2023. Disponible en: https://datareportal.com/reports/digital-2023-global-overview-report.

[48] D. Santos, «Facebook en 2023: que es, ventajes y como iniciar en esta red,» 05 Junio 2023. Disponible en: https://blog.hubspot.es/marketing/guia-marqueting-facebook.

[49] Brandon, «Creando instagram (version web) con laravel #2:Diseño de base de datos,» 2017 Noviembre 2017. Disponible en: http://medium.com/laravel-chile/creando-instagram-versi%B3n-web-com-laravel-1-358249ac2107.

[50] Meta, «Mas detalles sobre como funciona Instagram,» 8 Junio 2021. Disponible en: https://about.instagram.com/es-la/blog/announcements/shedding-more-light-on-how-instagram-works.

[51] I. Matters, «¿Que es pinterest? - Lo que los padres tienen que saber,» 06 Octubre 2022. Disponible en: https://www.internetmatters.org/es/hub/news-blogs/what-is-pinterest-what-parents-need-to-know/.

[52] D. A., «Qué es React: definición, características y funcionamiento,» 29 Junio 2023. Disponible en: https://www.hostinger.es/tutoriales/que-es-react#:~:text=ReactJS%20es%20una%20de%20las,usuario%20(UI)%20llamadas%20componentes..

[53] MDN, «Primeros pasos en React,» 02 Agosto 2023. Disponible en: https://developer.mozilla.org/es/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/React_getting_started.

[54] DataReportal, digital en todo el mundo, 01 2024. Disponible en: https://datareportal.com/global-digital-overview.

[55] I. dotcom-monitor, «Las 10 mejores practicas de monitoreo de aplicaciones web | Descripcion general del servicio de supervision d sitios web,» 2024. Disponible en: https://www.dotcom-monitor.com/blog/es/sitio-web-monitoreo-servicio-mejores-practicas/.

[56] A. M. E. Michelle, «Analisis de vulnerabilidades y pruebas de estres en el sistema academico de la escuela superior politenica de Chimborazo: Una evaluacion integral,» 2023.

[57] P. M. &. Tecnología, «¿Cuántas visitas debería tener mi página web?,» 2024. Disponible en: https://www.pgrmt.com/blog/cuantas-visitas-deberia-tener-mi-pagina-web.

[58] R. Sospedra, «Cuánto paga Google adsense y cómo pasar el filtro para ser aceptado,» 12 10 2023. Disponible en: https://www.rafasospedra.com/google-adsense-cuanto-paga/.

[59] ADSTERRA, «¿Cómo monetizar un blog? 12 métodos rentables en 2023,» 08 02 2024. Disponible en: https://adsterra.com/blog/es/como-monetizar-un-blog/.

[60] HubSpot, «Los mejores días y horas para publicar en redes sociales (+ infografía),» 01 08 2023. [Disponible en: https://blog.hubspot.es/marketing/mejor-hora-para-publicar-en-redes-sociales.

[61] Hootsuite, «El mejor momento para publicar en redes sociales en 2024 [todas las redes],» 11 10 2023. Disponible en: https://blog.hootsuite.com/es/mejor-momento-para-publicar-en-redes-sociales/.

[62] M. Keutelian, «Los mejores horarios para publicar en las redes sociales durante 2023,» 23 06 2023. Disponible en: https://sproutsocial.com/es/insights/mejores-momentos-publicar-en-redes-sociales/.

[63] GeeksforGeeks, «¿Que son las pruebas de estres en pruebas de software?,» 24 Abril 2024. Disponible en: https://www.geeksforgeeks.org/stress-testing-software-testing/.

[64] Zaptest, «Pruebas de estres en pruebas de software:¿Que son, tipos, procesos, enfoques, herramientas y mas?,» 3 Julio 2023. Disponible en: https://www.zaptest.com/stress-testing-in-software-testing-what-is-it-types-processes-approaches-tools-more.

[65] B. D. y. R. J. C., «Metodologia XP,» 2014.

[66] C. Gomez, «Niveles de prueba,» 27 Octubre 2022. Disponible en: https://www.diariodeqa.com/post/niveles-de-prueba#:~:text=Los%20niveles%20de%20prueba%20se,que%20se%20encuentre%20el%20producto..

[67] D. Yang, «Los 9 mejores lenguajes de programacion para aprender 2023,» 27 Julio 2023. Disponible en: https://www.fullstackacademy.com/blog/nine-best-programming-languages-to-learn.

[68] «Especificacion de requisitos.,» Disponible en: http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/456#:~:text=Una%20plantilla%20de%20requisitos%20puede,obligatorios%20y%20otros%20son%20opcionales..

[69] A. Mendoza, «¿Has escuchado de Tailwind CSS? - un framework de CSS distinto,» 2020. Disponible en: https://ed.team/comunidad/has-escuchado-de-tailwind-css-un-framework-de-css-distinto.

[70] E. Rodriguez, «Estos serán los lenguajes de programación con más salida en 2023. Puedes empezar a aprenderlos gratis,» 01 Enero 2023. Disponible en: https://www.genbeta.com/actualidad/estos-seran-lenguajes-programacion-salida-2023-puedes-empezar-a-aprenderlos-gratis.

[71] i. I. S. T. Q. Board, «Comité internacional de certificaciones de pruebas de software, » 2015.

[72] «IEEE Spectrum / Principales lenguajes de programacion 2022,» Disponible en: https://spectrum.ieee.org/top-programming-languages-2022.

[73] kimagure44, «Tensorflow.js y otros servicios API de inteligencia artificial,» 28 Diciembre 2019. Disponible en: https://tecnops.es/tensorflow-js-y-otros-servicios-api-de-inteligencia-artificial/.

[74] «Aspectos anuales destacados de JetBrains 2022,» Disponible en: https://www.jetbrains.com/lp/annualreport-2022/.

[75] J. Clark, «Los 10 principales Frameworks backend 2023,» 2023. Disponible en: https://blog.back4app.com/backend-frameworks/.

[76] I. s. l. c. d. software, « ISTQB sobre la calidad del software,» 2015.

[77] «Lenguajes de programacion mas populares para aprender 2022,» Disponible en: https://www.simplilearn.com/best-programming-languages-start-learning-today-article#:~:text=JavaScript%20and%20Python%2C%20two%20of,languages%20to%20learn%20for%20beginners..

[78] «MongoDB Atlas,» Disponible en: https://cloud.mongodb.com/v2/636bf71a3e873e7eb5288f3f#metrics/replicaSet/636bf783b38fe565ee49f725/explorer/test/users/find.

[79] «Mongo Atlas,» Disponible en: https://cloud.mongodb.com/v2/636bd7ec053113501f0ad209#metrics/replicaSet/636bd8766d66d562229b310e/explorer/test/users/find.

[80] M. d. usuario, «,Plantillas» Disponible en: https://ejamplos/plantillas/de/manual/de/suario.com

[81] L. dev, «Los 8 lenguajes de programacion mas demandados 2023,» 21 Junio 2023. Disponible en: https://www.devjobsscanner.com/blog/top-8-most-demanded-programming-languages/.

[82] P. L. A. y. S. I. Mariño, «Los estándares internacionales y su importancia para la industria del software,» 15 02 2013. Disponible en: http://www.cyta.com.ar/ta1202/v12n2a3.htm.

[83] M. Corporation’s, «Introducción a Express/Node,» 2023. Disponible en: https://developer.mozilla.org/es/docs/Learn/Server-side/Express_Nodejs/Introduction.

[84] «Pruebas unitarias y de integracion,» Disponible en: https://pruebas_unitarias_Junit_pruebas_de_integracion_selenium.

[85] S. P. Alonso, «¿Qué es API First?,» 19 Enero 2023. Disponible en: https://es.linkedin.com/pulse/qu%C3%A9-es-api-first-sergio-pantoja-alonso.

[86] S. S. Alianza de inovacion en pruebas de software, «ISO/IEC/IEEE 29119 El nuevo estándar internacional para pruebas de software,» 2015.

[87] «Los mejores lenguajes de programacion para aprender 2021, » Disponible en: https://codeburst.io/best-programming-languages-to-learn-in-2021-for-job-future-9adf0a7d9b6d.

[88] E. Solano-Fernández, «El modelo iterativo e incremental para el desarrollo de la aplicación de realidad aumentada Amón_RA,» 2020.

[89] Jesus, «¿Qué es MongoDB? Todo sobre la popular base de datos de código abierto,» 14 Febrero 2023. Disponible en: https://www.dongee.com/tutoriales/que-es-mongodb/.

[90] D. Guide, «¿Qué es MySQL?,» 18 Enero 2023. Disponible en: https://www.ionos.mx/digitalguide/servidores/know-how/que-es-mysql/.

[91] P. Levante, «Riesgos,» Diciembre 2021. Disponible en: https://pmi-levante.org/pmief-riesgos/.

[92] «Apache Jmeter: Todo lo que necesitas saber,» 12 Octubre 2022. Disponible en: geekflare.com/es/apache-jmeter-guide/.

[93] R. I. G. Benalcázar, «Estudio comparativo de los frameworks Ruby on rails y Django para la implementacion de un sistema informatico de control y administracion,» 2016.

[94] «¿Por qué aprender a programar?,» Disponible en: www.edix.com/es/instituto/por-que-aprender-

programar/#:~:text=Aprender%20a%20programar%20supone%20aprender,del%20mundo%20que%20te%20rodea..

ANEXOS

Anexo 1: Automatización de pruebas APIs REST

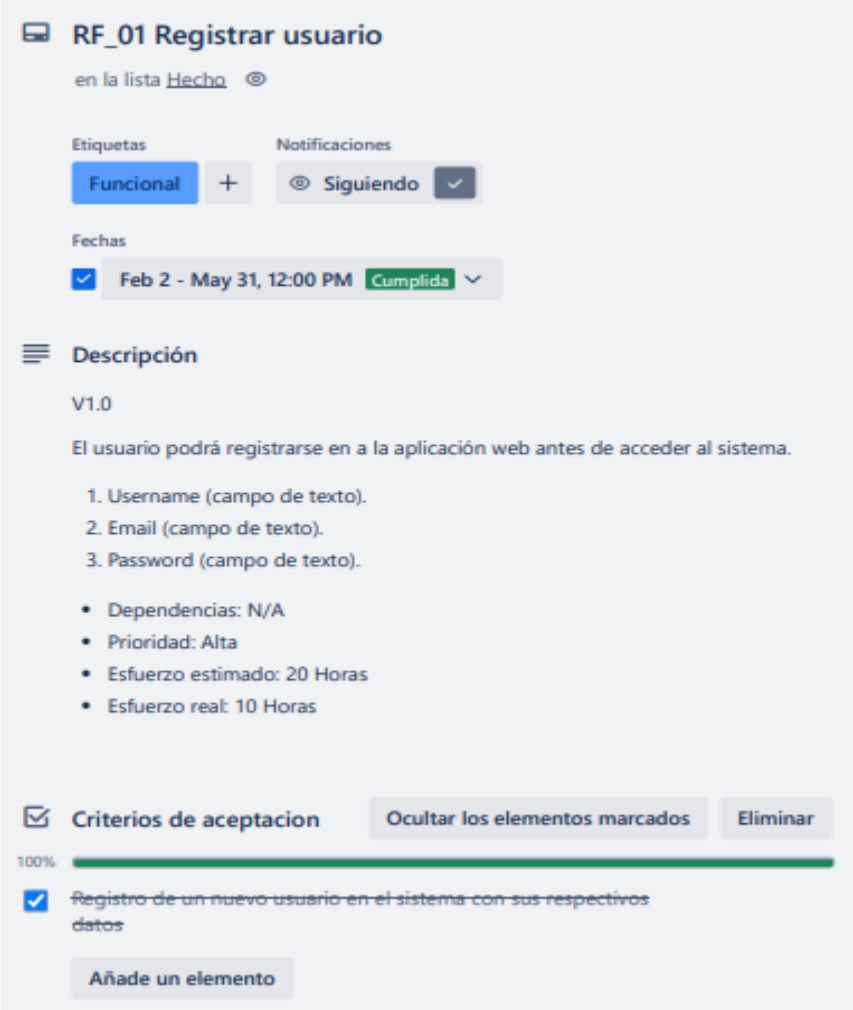
Las pruebas de software API REST automatizadas se realizaron a nivel backend (Spring Boot, Django y Laravel), mediante la herramienta Postman, para comprobar las funcionalidades y métodos establecidos en los requerimientos funcionales de la aplicación *web* que fungía como entorno experimental.

RF_##: Acrónimo para identificar el requerimiento funcional, continuo por el nombre del módulo perteneciente.

CPF##: Acrónimo para identificar el caso de prueba funcional, continuo por un número de ordenamiento natural.

Módulo usuarios

RF_01 Registro de usuario.



Caso de prueba general CPF01, CPF02 y CPF03.

Nombre: Registrar usuario		
Fecha de Revisión: 02/02/2024 al 31/05/24	**Prioridad:** Alta	**Revisor:** Berenice Ortiz Torres
Modulo: Usuarios	**Autorización sobre el módulo:** Creación de usuario en la base de datos	
Información sobre el caso de prueba		
Descripción: Esta prueba consta de tres casos de uso, el primero para corroborar que el código de estado sea exitoso, el segundo para verificar que el _id del nuevo usuario no	**Prerrequisitos:** Completar los campos de la colección en la base de datos.	

sea nulo y el tercer caso, verifica que no esté vacía la información del documento insertado en la colección del usuario.

Datos de entrada

```
pm.test("Status code is 201", function () {
    pm.response.to.have.status(201);
});
pm.test("Verifying User _id", function () {
    var jsonData = pm.response.json();
    var user_id= jsonData._id;
    pm.environment.set("User_id",user_id);
    pm.expect(user_id).to.not.be.null;
});
pm.test("Verfiying User information", function () {
    var jsonData = pm.response.json();
    var email= jsonData.email;
    var username= jsonData.username;
    var password= jsonData.password;

    pm.environment.set("Email",email);
    pm.environment.set("Username",username);
    pm.environment.set("Password",password);
    pm.expect(email, username, password).to.not.be.empty;
});
```

Procedimiento: Por medio del módulo y controlador, se crea un nuevo elemento con los campos indicados para realizar una inserción a la colección usuario en la base de datos.

Resultado esperado: Para la primera prueba se espera un estado de respuesta 201. Para la segunda y tercera se espera una correcta adición de los datos.

Resumen de resultados obtenidos	Aprobado	No. Caso de Prueba
La ejecución de las pruebas para el registro de usuarios en los framework Spring Boot, Django y Laravel se realizó de manera exitosa.	✓	**CPF01 CPF02 CPF03**

RF_02 Iniciar sesión.

Caso de prueba general CPF04, CPF05 y CPF06.

Nombre: Iniciar sesión	
Fecha de Revisión: 02/02/2024 al 31/05/24 **Prioridad:** Alta **Revisor:** Berenice Ortiz Torres	
Modulo: Usuarios	**Autorización sobre el módulo:** Autenticación de usuario
Información sobre el caso de prueba	
Descripción: Esta prueba consta de dos casos de uso, el primero para corroborar que el código de estado sea exitoso, el segundo para verificar que el _id del usuario previamente registrado no sea nulo. El entorno corrobora la existencia del usuario, si son correctos sus datos.	**Prerrequisitos:** Completar los campos de la colección en la base de datos.
Datos de entrada	

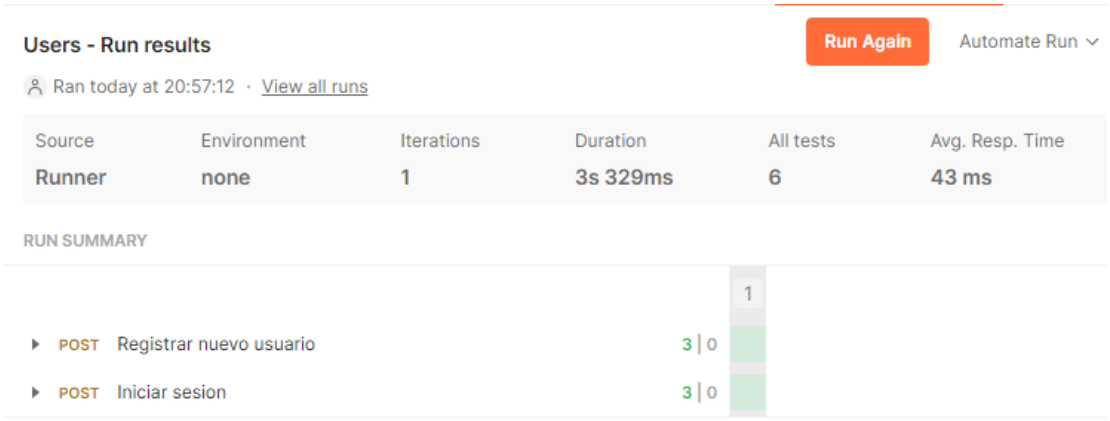
```
pm.test("Status code is 201", function () {
    pm.response.to.have.status(201);
});
pm.test("Verifying Signin User _id", function () {
    var jsonData = pm.response.json();
    var user_id= jsonData._id;
    pm.environment.set("User_id",user_id);
    pm.expect(user_id).to.not.be.null;
});
```

Procedimiento: Por medio del módulo y controlador, se verifica los datos recibidos (credenciales) y se comprueba que existan.

Resultado esperado: Para la primera prueba se espera un estado de respuesta 201. Para la segunda y tercera se espera la verificación de los datos ingresados.

Resumen de resultados obtenidos	Aprobado	No. Caso de Prueba
La ejecución de las pruebas para el inicio sesión en los framework Spring Boot, Django y Laravel se realizó de manera exitosa.	✓	**CPF05 CPF4 CPF06**

Resumen de ejecución y resultados en la colección usuarios.



Módulo post

RF_03 Agregar post.

Caso de prueba general CPF07, CPF08 y CPF09.

Nombre: Agregar post		
Fecha de Revisión: 02/02/2024 al 31/05/24	**Prioridad:** Alta	**Revisor:** Berenice Ortiz Torres
Módulo: Post	**Autorización sobre el módulo:** Inserción de post	
Información sobre el caso de prueba		
Descripción: Esta prueba consta de tres casos de uso, el primero para corroborar que el código de estado sea exitoso, el segundo para verificar que el _id del nuevo post no sea nulo y el tercer caso, verifica que no estén vacíos los datos agregados.	**Prerrequisitos:** Completar los campos de la colección en la base de datos.	
Datos de entrada		

```
pm.test("Status code is 201", function () {
    pm.response.to.have.status(201);
});
pm.test("Verifying Post _id", function () {
    var jsonData = pm.response.json();
    var post_id= jsonData._id;
    pm.environment.set("Post_id",post_id);
    pm.expect(post_id).to.not.be.null;
});
pm.test("Verfiying Post information", function () {
    var jsonData = pm.response.json();
    var user_id= jsonData.user_id;
    var description= jsonData.description;
    var photo= jsonData.photo;

    pm.environment.set("User_id",user_id);
    pm.environment.set("Description",description);
    pm.environment.set("Photo",photo);
    pm.expect(user_id, description, photo).to.not.be.empty;
});
```

Procedimiento: Por medio del módulo y controlador, se crea un nuevo elemento con los campos indicados para realizar una inserción a la colección post en la base de datos.

Resultado esperado: Para la primera prueba se espera un estado de respuesta 201. Para la segunda y tercera se espera una correcta adición de los datos.

Resumen de resultados obtenidos	Aprobado	No. Caso de Prueba
La ejecución de las pruebas para registrar post en los framework Spring Boot, Django y Laravel se realizó de manera exitosa.	✓	**CPF07 CPF08 CPF09**

RF_04 Acciones post.

Caso de prueba general CPF10, CPF11 y CPF12.

Nombre: Acciones post	
Fecha de Revisión: 02/02/2024 al 31/05/24 **Prioridad:** Alta **Revisor:** Berenice Ortiz Torres	
Módulo: Post	**Autorización sobre el módulo:** Acciones de post
Información sobre el caso de prueba	
Descripción: Esta prueba consta de tres acciones: - Obtener: constan de la comprobación del estatus éxitos, la correcta verificación de las respuestas esperadas en los campos descripción y foto. - Actualizar: se insertan datos aleatorios por medio del random proporcionado por Postman y se verifica que los datos actualizados no estén vacíos.	**Prerrequisitos:** Completar los campos de la colección en la base de datos.

- Eliminar: se valida la eliminación exitosa por medio del código de estado y se ejecuta la búsqueda del documento eliminado para comprobar que no exista en la base de datos.

Datos de entrada

**Obtener

```
pm.test("Status code is 201", function () {
    pm.response.to.have.status(201);
});
pm.test("Verifying _id post", function () {
    var jsonData = pm.response.json();
    var has_id=Object.keys(jsonData).includes('_id');
    pm.expect(true).to.eql(has_id);
});
```

**Actualizar

```
pm.test("Status code is 204", function () {
    pm.response.to.have.status(204);
});

const postSchema ={
    _id: "",
    user_id: "",
    description: "",
    photo: "",
}
pm.test('Schema composition'),()=>{
    var body= JSON.parse(pm.request.body.raw);
    var sameSchema = JSON.stringify(
    Object.keys(eventSchema) == JSON.stringify(Object.keys(body)));
    pm.expect(true).to.be.eql(sameSchema);
}
pm.test("Post information update verfiying", function () {
    const res= pm.response.json();
    pm.expect(res.description).to.eql('The new language');
});
```

**Eliminar

```
pm.test("Status code is 204", function () {
    pm.response.to.have.status(204);
});
```

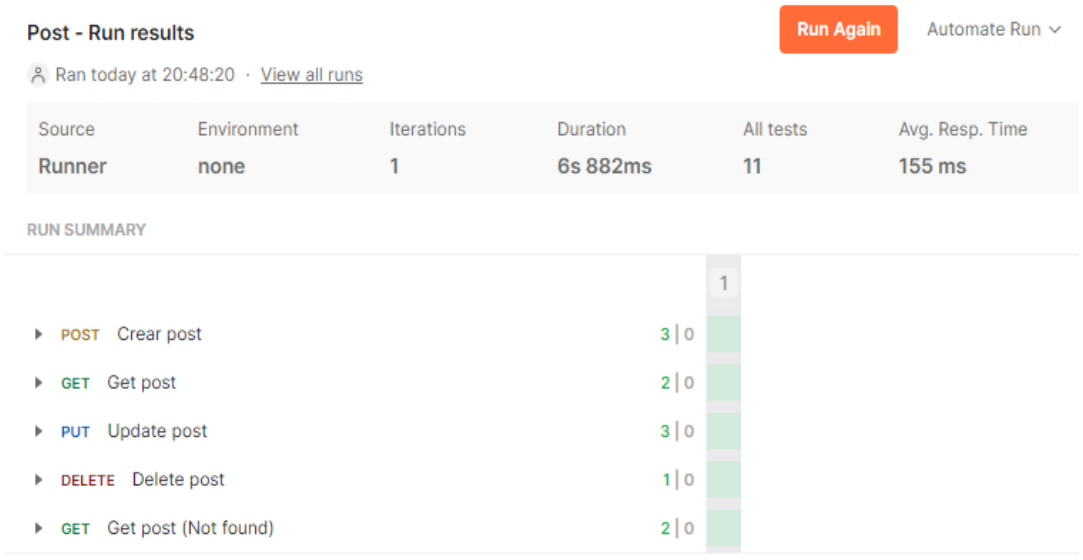
```
pm.test("Status code is 404", function () {
    pm.response.to.have.status(404);
});
pm.test("Body matches string. Module Post", function () {
    pm.expect(pm.response.text()).to.include("Not found.");
});
```

Procedimiento: Por medio del módulo y controlador:
- Para obtener: se verifica que los datos esperados sean los correctos.
- Para actualizar: con los datos aleatorios se envía la petición, se realiza la actualización en los campos y se comprueba que se hayan actualizado.
- Para eliminar: se verifica el estatus correcto de la acción y se procede a la búsqueda del documento eliminado para su verificación.

Resultado esperado: Obtención, actualización y eliminación correcta de los documentos indicados.

Resumen de resultados obtenidos	Aprobado	No. Caso de Prueba
La ejecución de las pruebas para las acciones de los posters en los framework Spring Boot, Django y Laravel se realizó de manera exitosa.	✓	**CPF10 CPF11 CPF12**

Resumen de ejecución y resultados en la colección post.



Módulo like

RF_05 Agregar like

Caso de prueba general CPF13, CPF14 y CPF15.

Nombre: Agregar like	
Fecha de Revisión: 02/02/2024 al 31/05/24 **Prioridad:** Alta **Revisor:** Berenice Ortiz Torres	
Módulo: Like	**Autorización sobre el módulo:** Inserción de like
Información sobre el caso de prueba	
Descripción: Esta prueba consta de tres casos de uso, el primero para corroborar que el código de estado sea exitoso, el segundo para verificar que el _id del nuevo like no sea nulo y el tercer caso, verifica que no estén vacíos los datos agregados.	**Prerrequisitos:** Completar los campos de la colección en la base de datos.
Datos de entrada	

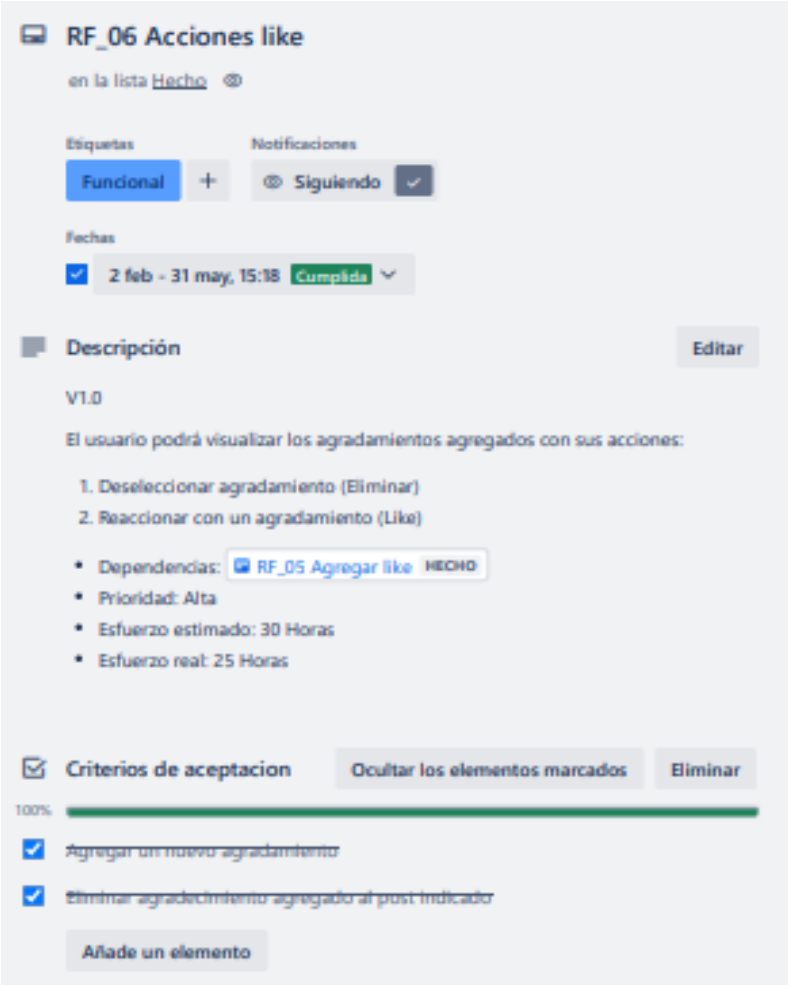
```
pm.test("Status code is 201", function () {
    pm.response.to.have.status(201);
});
pm.test("Verifying Like _id", function () {
    var jsonData = pm.response.json();
    var like_id= jsonData._id;
    pm.environment.set("Like_id",like_id);
    pm.expect(like_id).to.not.be.null;
});
```

Procedimiento: Por medio del módulo y controlador, se crea un nuevo elemento con los campos indicados para realizar una inserción a la colección like en la base de datos.

Resultado esperado: Para la primera prueba se espera un estado de respuesta 201. Para la segunda y tercera se espera una correcta adición de los datos.

Resumen de resultados obtenidos	Aprobado	No. Caso de Prueba
La ejecución de las pruebas para agregar like en los framework Spring Boot, Django y Laravel se realizó de manera exitosa.	✓	**CPF13 CPF14 CPF15**

RF_06 Acciones like



Caso de prueba general CPF16, CPF17 y CPF18.

Nombre: Acción like		
Fecha de Revisión: 02/02/2024 al 31/05/24	**Prioridad:** Alta	**Revisor:** Berenice Ortiz Torres
Módulo: Like	**Autorización sobre el módulo:** Acción like	
Información sobre el caso de prueba		
Descripción: Esta prueba consta de tres acciones: - Obtener: constan de la comprobación del estatus éxitos y la correcta verificación del _id. - Eliminar: se valida la eliminación exitosa por medio del código de estado y se ejecuta la búsqueda del documento eliminado para comprobar que no exista en la base de datos.	**Prerrequisitos:** Completar los campos de la colección en la base de datos.	
Datos de entrada		

**Obtener todos

```
pm.test("Status code is 201", function () {
    pm.response.to.have.status(201);
});
```

**Eliminar

```
pm.test("Status code is 201", function () {
    pm.response.to.have.status(201);
});
```

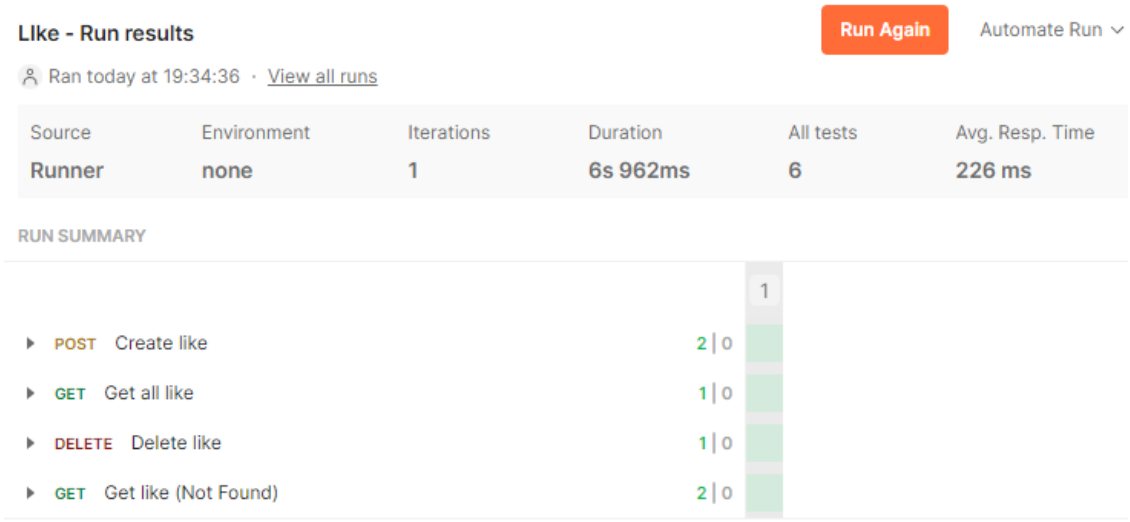
```
pm.test("Status code is 404", function () {
    pm.response.to.have.status(404);
});
pm.test("Body matches string. Module Like", function () {
    pm.expect(pm.response.text()).to.include("Not found.");
});
```

Procedimiento: Por medio del módulo y controlador:
- Para obtener: se verifica que los datos esperados sean los correctos.
- Para eliminar: se verifica el estatus correcto de la acción y se procede a la búsqueda del documento eliminado para su verificación.

Resultado esperado: Obtención y eliminación correcta de los documentos indicados.

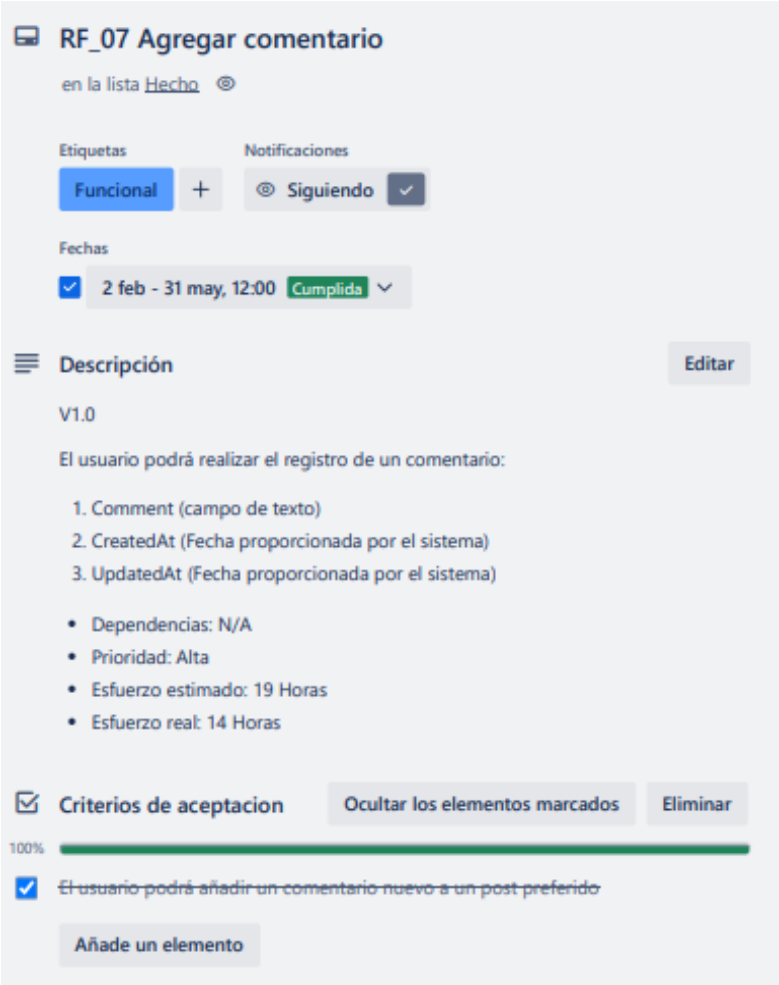
Resumen de resultados obtenidos	Aprobado	No. Caso de Prueba
La ejecución de las pruebas para las acciones de los like en los framework Spring Boot, Django y Laravel se realizó de manera exitosa.	✓	**CPF16 CPF17 CPF18**

Resumen de ejecución y resultados en la colección like.



Módulo comentarios

RF_07 Agregar comentario.



Caso de prueba general CPF19, CPF20 y CPF21.

Nombre: Agregar comentario		
Fecha de Revisión: 02/02/2024 al 31/05/24	**Prioridad:** Alta	**Revisor:** Berenice Ortiz Torres
Módulo: Comentario	**Autorización sobre el módulo:** Inserción de comentario	
Información sobre el caso de prueba		
Descripción: Esta prueba consta de tres casos de uso, el primero para corroborar que el código de estado sea exitoso, el segundo para verificar que el _id del nuevo comentario no sea nulo y el tercer caso, verifica que no esté vacíos los datos agregados.	**Prerrequisitos:** Completar los campos de la colección en la base de datos.	
Datos de entrada		

```
pm.test("Status code is 201", function () {
    pm.response.to.have.status(201);
});
pm.test("Verifying Comment _id", function () {
    var jsonData = pm.response.json();
    var comment_id= jsonData._id;
    pm.environment.set("Comment_id",comment_id);
    pm.expect(comment_id).to.not.be.null;
});
pm.test("Verfiying Comment information", function () {
    var jsonData = pm.response.json();
    var user_id= jsonData.user_id;
    var post_id= jsonData.post_id;
    var comment= jsonData.comment;

    pm.environment.set("User_id",user_id);
    pm.environment.set("Post_id",post_id);
    pm.environment.set("Comment",comment);
    pm.expect(user_id, post_id, comment).to.not.be.empty;
});
```

Procedimiento: Por medio del módulo y controlador, se crea un nuevo elemento con los campos indicados para realizar una inserción a la colección comentarios en la base de datos.

Resultado esperado: Para la primera prueba se espera un estado de respuesta 201. Para la segunda y tercera se espera una correcta adición de los datos.

Resumen de resultados obtenidos	Aprobada	No. Caso de Prueba
La ejecución de las pruebas para agregar comentarios en los framework Spring Boot, Django y Laravel se realizó de manera exitosa.	✓	**CPF19 CPF20 CPF21**

RF_08 Acciones comentarios.

Caso de prueba general CPF22, CPF23 y CPF24.

Nombre: Acciones de comentario		
Fecha de Revisión: 02/02/2024 al 31/05/24	**Prioridad:** Alta	**Revisor:** Berenice Ortiz Torres
Módulo: Comentarios	**Autorización sobre el módulo:** Acciones de comentarios	
Información sobre el caso de prueba		
Descripción: Esta prueba consta de tres acciones: - Obtener: constan de la comprobación del estatus éxitos, la correcta verificación de las respuestas esperadas en el campo comentario. - Actualizar: se insertó un dato aleatorio por medio del random proporcionado por Postman y se verifica que el dato actualizado no esten vacío.	**Prerrequisitos:** Completar los campos de la colección en la base de datos.	

- Eliminar: se valida la eliminación exitosa por medio del código de estado y se ejecuta la búsqueda del documento eliminado para comprobar que no exista en la base de datos.

Datos de entrada

**Obtener

```
pm.test("Status code is 201", function () {
    pm.response.to.have.status(201);
});
pm.test("Verifying _id comment", function () {
    var jsonData = pm.response.json();
    var has_id=Object.keys(jsonData).includes('_id');
    pm.expect(true).to.eql(has_id);
});
```

**Actualizar

```
pm.test("Status code is 204", function () {
    pm.response.to.have.status(204);
});

const commentSchema ={
    _id: "",
    user_id: "",
    post_id: "",
    comment: "",
}
pm.test('Schema composition'),()=>{
    var body= JSON.parse(pm.request.body.raw);
    var sameSchema = JSON.stringify(
    Object.keys(commentSchema) == JSON.stringify(Object.keys(body)));
    pm.expect(true).to.be.eql(sameSchema);
}
pm.test("Comment information update verfiying", function () {
    const res= pm.response.json();
    pm.expect(res.comment).to.eql('utilize digital invoice');
});
```

**Eliminar

```
pm.test("Status code is 204", function () {
    pm.response.to.have.status(204);
});
```

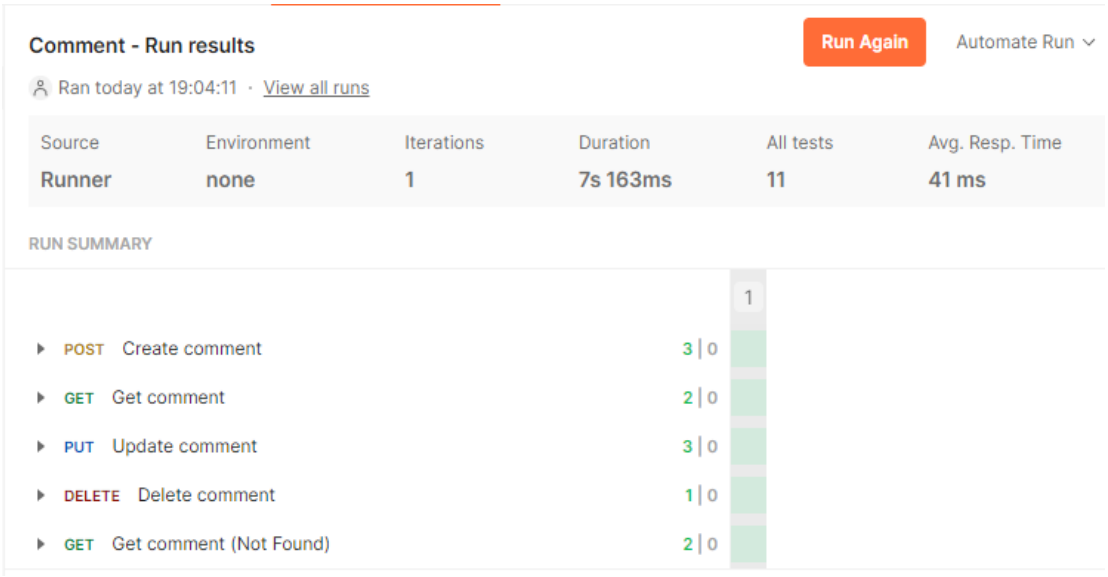
```
pm.test("Status code is 404", function () {
    pm.response.to.have.status(404);
});
pm.test("Body matches string. Module Comment", function () {
    pm.expect(pm.response.text()).to.include("Not found.");
});
```

Procedimiento: Por medio del módulo y controlador:
- Para obtener: se verifica que los datos esperados sean los correctos.
- Para actualizar: con los datos aleatorios se envía la petición, se realiza la actualización en los campos y se comprueba que se hayan actualizado.
- Para eliminar: se verifica el estatus correcto de la acción y se procede a la búsqueda del documento eliminado para su verificación.

Resultado esperado: Obtención, actualización y eliminación correcta de los documentos indicados.

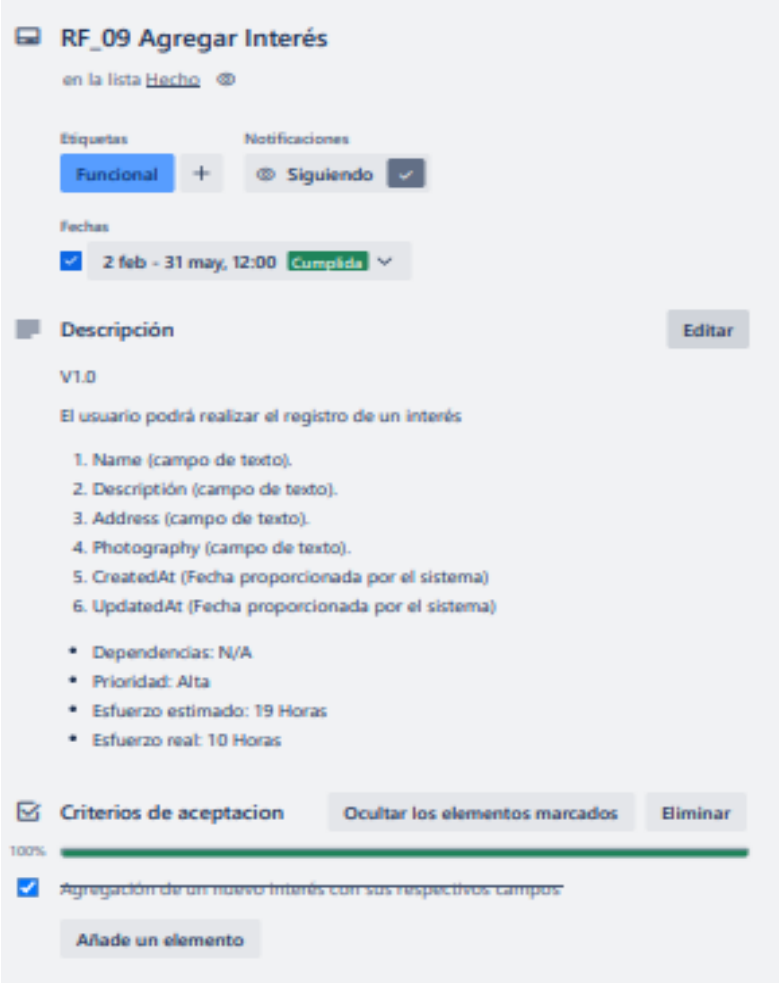
Resumen de resultados obtenidos	Aprobado	No. Caso de Prueba
La ejecución de las pruebas para las acciones de los comentarios en los framework Spring Boot, Django y Laravel se realizó de manera exitosa.	✓	**CPF22 CPF23 CPF24**

Resumen de ejecución y resultados en la colección comentarios.



Módulo intereses

RF_09 Agregar interés.



Caso de prueba general CPF25, CPF26 y CPF27.

Nombre: Agregar interés		
Fecha de Revisión: 02/02/2024 al 31/05/24	**Prioridad:** Alta	**Revisor:** Berenice Ortiz Torres
Módulo: Interés	**Autorización sobre el módulo:** Inserción de interés	
Información sobre el caso de prueba		
Descripción: Esta prueba consta de tres casos de uso, el primero para corroborar que el código de estado sea exitoso, el segundo para verificar que el _id del nuevo interés no sea nulo y el tercer caso, verifica que no estén vacíos los datos agregados.	**Prerrequisitos:** Completar los campos de la colección en la base de datos.	

Datos de entrada

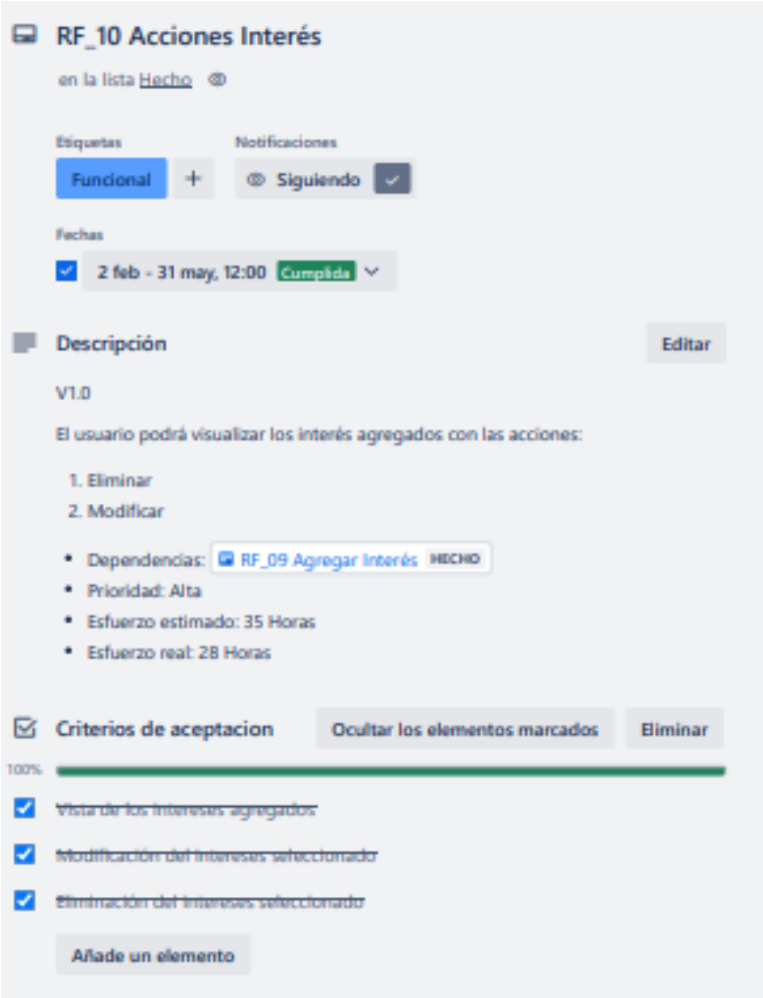
```
pm.test("Status code is 201", function () {
    pm.response.to.have.status(201);
});
pm.test("Verifying Interest _id", function () {
    var jsonData = pm.response.json();
    var interest_id= jsonData._id;
    pm.environment.set("Interest_id",interest_id);
    pm.expect(interest_id).to.not.be.null;
});
pm.test("Verfiying Interest information", function () {
    var jsonData = pm.response.json();
    var user_id= jsonData.user_id;
    var interest_category= jsonData.interest_category;
    var name= jsonData.name;
    var description= jsonData.description;

    pm.environment.set("User_id",user_id);
    pm.environment.set("Interest_category",interest_category);
    pm.environment.set("Name",name);
    pm.environment.set("Description",description);
    pm.expect(interest_category, user_id, name, description).to.not.be.
});
```

Procedimiento: Por medio del módulo y controlador, se crea un nuevo elemento con los campos indicados para realizar una inserción a la colección interés en la base de datos.

Resultado esperado: Para la primera prueba se espera un estado de respuesta 201. Para la segunda y tercera se espera una correcta adición de los datos.

Resumen de resultados obtenidos	Aprobado	No. Caso de Prueba
La ejecución de las pruebas para agregar intereses en los framework Spring Boot, Django y Laravel se realizó de manera exitosa.	✓	**CPF25 CPF26 CPF27**

RF_10 Acciones interés.



Caso de prueba general CPF28, CPF29 y CPF30.

Nombre: Acciones interés	
Fecha de Revisión: 02/02/2024 al 31/05/24 **Prioridad:** Alta **Revisor:** Berenice Ortiz Torres	
Módulo: Posters	**Autorización sobre el módulo:** Acciones de posters
Información sobre el caso de prueba	
Descripción: Esta prueba consta de tres acciones: - Obtener: constan de la comprobación del estatus éxitos, la correcta verificación de las respuestas esperadas en los campos categoría, nombre, descripción e importancia. - Actualizar: se insertan datos aleatorios por medio del random proporcionado por Postman y se verifica que los datos actualizados no estén vacíos	**Prerrequisitos:** Completar los campos de la colección en la base de datos.

- Eliminar: se valida la eliminación exitosa por medio del código de estado y se ejecuta la búsqueda del documento eliminado para comprobar que no exista en la base de datos.

Datos de entrada

**Obtener

```
pm.test("Status code is 201", function () {
    pm.response.to.have.status(201);
});
pm.test("Verifying _id interest", function () {
    var jsonData = pm.response.json();
    var has_id=Object.keys(jsonData).includes('_id');
    pm.expect(true).to.eql(has_id);
});
```

**Actualizar

```
pm.test("Status code is 204", function () {
    pm.response.to.have.status(204);
});

const interestSchema ={
    _id: "",
    user_id: "",
    interest_category: "",
    name: "",
    description: "",
    importance: "",
}
pm.test('Schema composition'),()=>{
    var body= JSON.parse(pm.request.body.raw);
    var sameSchema = JSON.stringify(
    Object.keys(interestSchema) == JSON.stringify(Object.keys(body)));
    pm.expect(true).to.be.eql(sameSchema);
}
pm.test("Interest information update verfiying", function () {
    const res= pm.response.json();
    pm.expect(res.name).to.eql('Gorgeous');
});
```

**Eliminar

```
pm.test("Status code is 204", function () {
    pm.response.to.have.status(204);
});
```

```
pm.test("Status code is 404", function () {
    pm.response.to.have.status(404);
});
pm.test("Body matches string. Module of Interest", function () {
    pm.expect(pm.response.text()).to.include("Not found.");
});
```

Procedimiento: Por medio del módulo y controlador:
- Para obtener: se verifica que los datos esperados sean los correctos.
- Para actualizar: con los datos aleatorios se envía la petición, se realiza la actualización en el campo importancia con un valor numérico agregado para comprobar que sean los mismo.
- Para eliminar: se verifica el estatus correcto de la acción y se procede a la búsqueda del documento eliminado para su verificación.

Resultado esperado: Obtención, actualización y eliminación correcta de los documentos indicados.

Resumen de resultados obtenidos	Aprobada	No. Caso de Prueba
La ejecución de las pruebas para las acciones de los intereses en los framework Spring Boot, Django y Laravel se realizó de manera exitosa.	✓	**CPF28 CPF29 CPF30**

Resumen de ejecución y resultados en la colección intereses.

Módulo eventos

RF_11 Agregar evento.

Caso de prueba general CPF31, CPF32 y CPF33.

Nombre: Agregar evento		
Fecha de Revisión: 02/02/2024 al 31/05/24	**Prioridad:** Alta	**Revisor:** Berenice Ortiz Torres
Módulo: Evento	**Autorización sobre el módulo:** Inserción de evento	
Información sobre el caso de prueba		
Descripción: Esta prueba consta de tres casos de uso, el primero para corroborar que el código de estado sea exitoso, el segundo para verificar que el _id del nuevo evento no sea nulo y el tercer caso, verifica que no esté vacíos los datos agregados.	**Prerrequisitos:** Completar los campos de la colección en la base de datos.	

Datos de entrada

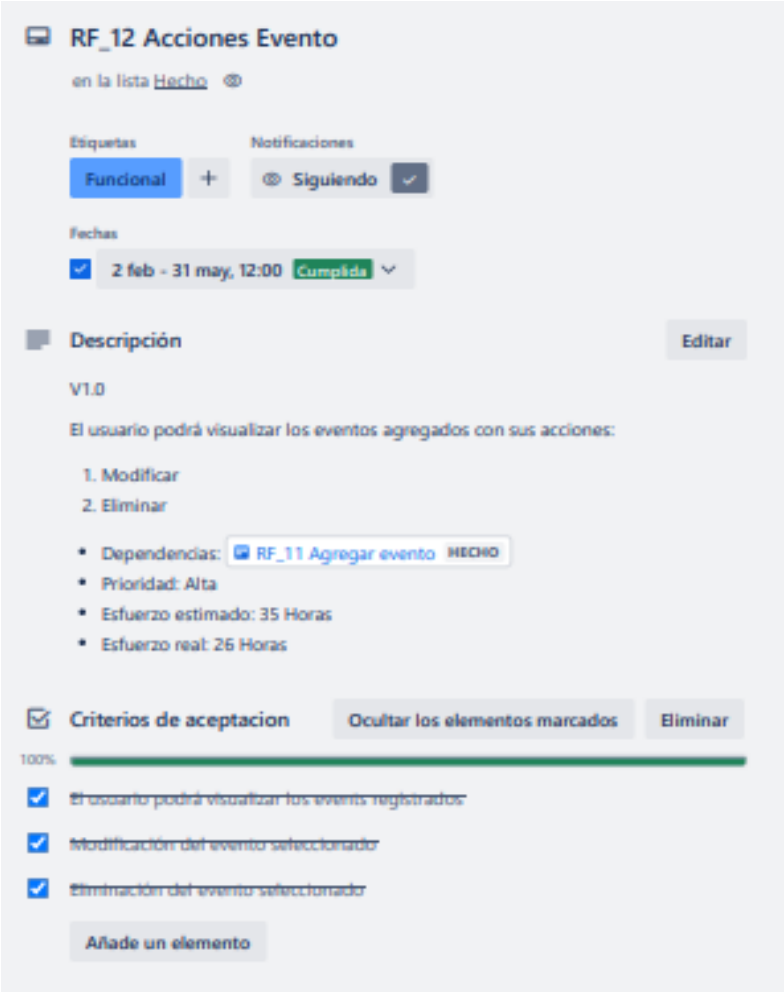
```
pm.test("Status code is 201", function () {
    pm.response.to.have.status(201);
});
pm.test("Verifying Event _id", function () {
    var jsonData = pm.response.json();
    var event_id= jsonData._id;
    pm.environment.set("Event_id",event_id);
    pm.expect(event_id).to.not.be.null;
});
pm.test("Verfiying Event information", function () {
    var jsonData = pm.response.json();
    var user_id= jsonData.user_id;
    var name= jsonData.name;
    var description= jsonData.description;
    var address= jsonData.address;
    var photography= jsonData.photography;

    pm.environment.set("User_id",user_id);
    pm.environment.set("Name",name);
    pm.environment.set("Description",description);
    pm.environment.set("Address",address);
    pm.environment.set("Phothography",photography);
    pm.expect(user_id, name, description, address, photography).to.not.be.empty;
});
```

Procedimiento: Por medio del módulo y controlador, se crea un nuevo elemento con los campos indicados para realizar una inserción a la colección evento en la base de datos.

Resultado esperado: Para la primera prueba se espera un estado de respuesta 201. Para la segunda y tercera se espera una correcta adición de los datos.

Resumen de resultados obtenidos	Aprobado	No. Caso de Prueba
La ejecución de las pruebas para agregar eventos en los framework Spring Boot, Django y Laravel se realizó de manera exitosa.	✓	**CPF31 CPF32 CPF33**

RF_04 Acciones evento.



Caso de prueba general CPF34, CPF35 y CPF36.

Nombre: Acciones evento	
Fecha de Revisión: 02/02/2024 al 31/05/24 **Prioridad:** Alta **Revisor:** Berenice Ortiz Torres	
Módulo: Evento	**Autorización sobre el módulo:** Acciones de evento
Información sobre el caso de prueba	
Descripción: Esta prueba consta de tres acciones: - Obtener: constan de la comprobación del estatus éxitos, la correcta verificación de las respuestas esperadas en los campos descripción y foto. - Actualizar: se insertan datos aleatorios por medio del random proporcionado por Postman y se verifica que los datos actualizados no estén vacíos	**Prerrequisitos:** Completar los campos de la colección en la base de datos.

- Eliminar: se valida la eliminación exitosa por medio del código de estado y se ejecuta la búsqueda del documento eliminado para comprobar que no exista en la base de datos.

Datos de entrada

**Obtener

```
pm.test("Status code is 201", function () {
    pm.response.to.have.status(201);
});
pm.test("Verifying _id event", function () {
    var jsonData = pm.response.json();
    var has_id=Object.keys(jsonData).includes('_id');
    pm.expect(true).to.eql(has_id);
});
```

**Actualizar

```
pm.test("Status code is 204", function () {
    pm.response.to.have.status(204);
});

const eventSchema ={
    _id: "",
    user_id: "",
    name: "",
    description: "",
    address: "",
    photography: "",
}
pm.test('Schema composition'),()=>{
    var body= JSON.parse(pm.request.body.raw);
    var sameSchema = JSON.stringify(
    Object.keys(eventSchema) == JSON.stringify(Object.keys(body)));
    pm.expect(true).to.be.eql(sameSchema);
}
pm.test("Event information update verfiying", function () {
    const res= pm.response.json();
    pm.expect(res.address).to.eql('Linconln Kennedy 243');
});
```

**Eliminar

```
pm.test("Status code is 204", function () {
    pm.response.to.have.status(204);
});
```

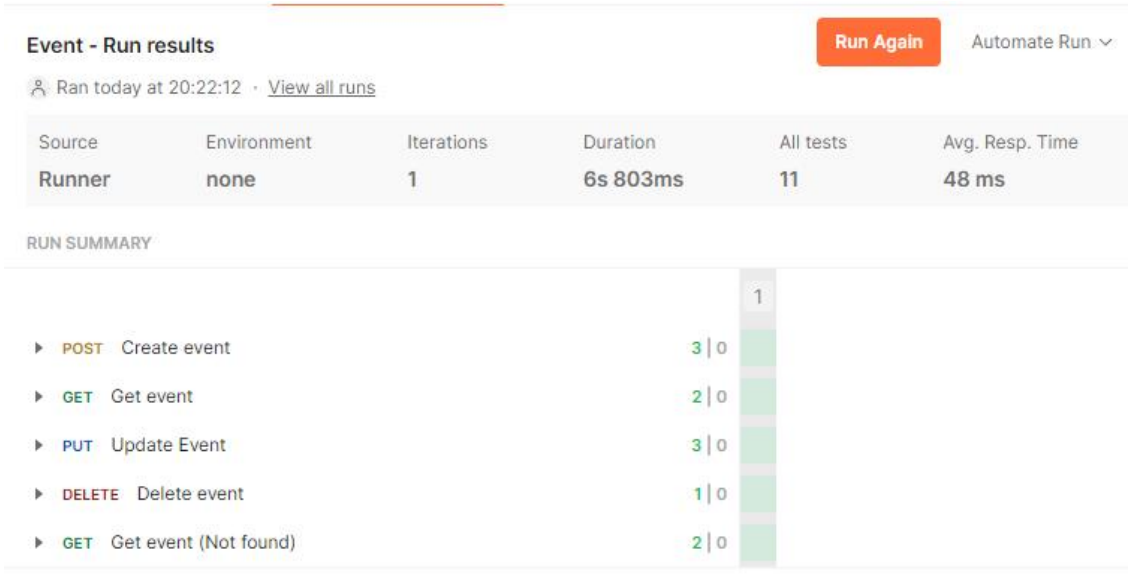
```
pm.test("Status code is 404", function () {
    pm.response.to.have.status(404);
});
pm.test("Body matches string. Module Event", function () {
    pm.expect(pm.response.text()).to.include("Not found.");
});
```

Procedimiento: Por medio del módulo y controlador:
- Para obtener: se verifica que los datos esperados sean los correctos.
- Para actualizar: con los datos aleatorios se envía la petición, se realiza la actualización en los campos y se comprueba que se hayan actualizado.
- Para eliminar: se verifica el estatus correcto de la acción y se procede a la búsqueda del documento eliminado para su verificación.

Resultado esperado: Obtención, actualización y eliminación correcta de los documentos indicados.

Resumen de resultados obtenidos	Aprobado	No. Caso de Prueba
La ejecución de las pruebas para las acciones de los eventos en los framework Spring Boot, Django y Laravel se realizó de manera exitosa.	✓	**CPF34 CPF35 CPF36**

Resumen de ejecución y resultados en la colección eventos.



Esta tesis se terminó de imprimir el 17 de octubre de 2024