PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR

SEDE ESMERALDAS

ESCUELA DE INGENIERÍA DE SISTEMAS Y COMPUTACIÓN

TESIS DE GRADO

ANÁLISIS COMPARATIVO ENTRE LOS ESTÁNDARES ORIENTADO A SERVICIOS WEB SOAP, REST Y GRAPHQL.

PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO DE SISTEMAS Y COMPUTACIÓN

LÍNEA DE INVESTIGACIÓN: PROGRAMACIÓN Y DESARROLLO DE SOFTWARE

AUTOR (A):

CRISTHIAN ANDRÉS RECALDE SOLANO

ASESOR (A):

MGT. EVELIN FLORES

ESMERALDAS – ABRIL 2019

TRIBUNAL DE GRADUACIÓN

Tema:

ANÁLISIS COMPARATIVO ENTRE LOS ESTÁNDARES ORIENTADO A SERVICIOS WEB SOAP, REST Y GRAPHQL.

Autor:

CRISTHIAN ANDRÉS RECALDE SOLANO

Mgt. Evelin Flores García f._____

ASESOR/A

Mgt. Jaime Sayago Heredia f._____

LECTOR 1

Mgt. Kléber Posligua Flores f._____

LECTOR 2

Mgt. Xavier Quiñónez Ku f._____

DIRECTOR DE ESCUELA

Mgt. Maritza Demera Mejía f._____

SECRETARIA GENERAL PUCESE

DECLARACIÓN DE AUTENTICIDAD Y RESPONSABILIDAD

Yo, Cristhian Andrés Recalde Solano, portador de la cédula de ciudadanía No. 0802704171, declaro que los resultados obtenidos en la investigación que presento como informe final, previo a la obtención del título de Ingeniero en Sistemas y Computación, son absolutamente originales, auténticos y personales.

En tal virtud, declaro que el contenido, de las conclusiones, los efectos legales y académicos que se desprenden del trabajo propuesto de Investigación y luego de la redacción de este documento son y serán de mi sola y exclusiva responsabilidad legal y académica.

Cristhian Andrés Recalde Solano

C.I.: 0802704171

CERTIFICACIÓN

Mgt. Evelin Flores, docente investigador de la PUCE-Esmeraldas, certifica que:

La tesis de grado realizada por CRISTHIAN ANDRÉS RECALDE SOLANO bajo el título "ANÁLISIS COMPARATIVO ENTRE LOS ESTÁNDARES ORIENTADO A SERVICIOS WEB SOAP, REST Y GRAPHQL", reúne los requisitos de calidad, originalidad y presentación exigibles a una investigación científica y que han sido incorporadas al documento final las sugerencias realizadas, en consecuencia, está en condiciones de ser sometida a la valoración del tribunal encargado de juzgarla.

Y para que conste a los efectos oportunos, firma la presente en Esmeraldas, 24 de abril del 2019.

Mgt. Evelin Flores

Asesor

DEDICATORIA

Este trabajo de investigación se lo quiero dedicar a mi familia, las personas que más amo en el mundo, a mi madre Martha que a lo largo de mi vida se ha convertido en alguien admirable para mí, la cual me ha brindado un cariño inmenso cada día, a mis abuelos Juan y Laura por ser las personas más maravillosas con las que he compartido y que les guardo un infinito amor y cariño.

Cristhian Recalde Solano

AGRADECIMIENTO

Amo a mi familia que es mi tesoro más preciado, se conforman por las personas que más quiero en el mundo, gracias por todo el amor y apoyo incondicional y por sus enseñanzas que me han hecho quien soy actualmente, y me han orientado hacia sendas correctas, llenas de valores y ética, este logro es de ustedes.

Agradezco a Margaret Selena por ser una persona incondicional y estar en los momentos más amenos y difíciles de mi vida, por hacer de mí cada día una mejor persona, haciendo que me supere y que enfrente con gran valor cada obstáculo que se me presente.

Cristhian Recalde Solano

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE CÓDIGOS

ÍNDICE DE ANEXOS

RESUMEN

La presente investigación tuvo la finalidad de realizar el análisis de tres marcos utilizados para proporcionar servicios web a través del protocolo SOAP, arquitectura REST y el lenguaje de esquemas GraphQL. Como tal se desarrolló un software con las cuatro funciones básicas (CRUD) en tres lenguajes de programación que son C Sharp, Java y PHP con el objetivo de consumir los servicios web y poder analizarlos.

Mediante la técnica del Mapping Study se determinó que las métricas Response Time y Throughput determinan el rendimiento de aplicaciones web y la metodología benchmark nos sirve para obtener información sobre el comportamiento del rendimiento de los servicios web, para posteriormente compararlos.

La herramienta de prueba de carga y estrés Apache JMeter se usa para implementar los tres servicios web y poder calcular los tiempos de respuesta y el rendimiento.

A través de los análisis comparativos se llegó a concluir que, sí llega a impactar el servicio web sobre el rendimiento de ejecución de aplicaciones web, solamente cuando existe una cantidad elevada de usuarios concurrentes, caso contrario el usuario final no notará mucha diferencia en el rendimiento del sistema.

Este documento presenta en detalle el análisis comparativo de los resultados experimentales sobre aspectos de desempeño de los servicios.

Palabras Clave: Servicio Web, SOAP, REST, GraphQL, pruebas, prueba de carga, rendimiento, concurrencia.

ABSTRACT

The present research aims to analyze three frameworks used to provide web services through the SOAP protocol, REST architecture and the GraphQL schema language. As such, software has been developed with the four basic functions (CRUD) in three programming languages that are C Sharp, Java and PHP with the aim of consuming web services and analyzing them.

Using the Mapping Study technique, it was determined that the Response Time and Throughput metrics determine the performance of web applications and the benchmark methodology helps us obtain information on the performance of the web services performance, to later compare them.

The Apache JMeter load and stress test tool is used to implement the three web services and to calculate response times and performance.

Through the comparative analysis it was concluded that, if the web service has an impact on the execution performance of web applications, only when there is a high number of concurrent users, otherwise the end user will not notice much difference in performance of the system.

This document presents in detail the comparative analysis of the experimental results on aspects of service performance.

Keywords: Web Service, SOAP, REST, GraphQL, Test, Load Test, Turnout, Performance.

1. INTRODUCCIÓN

1.1. Presentación del Tema

El análisis comparativo entre los servicios web SOAP, REST y GraphQL tiene como finalidad evaluar la efectividad de sus capacidades de transferencia de datos. Decidir el estilo de interacción de servicio es una elección muy importante para desarrolladores y diseñadores ya que influye de manera significativa en los requisitos subyacentes para implementar soluciones de servicios web.

Los servicios web basados en SOAP incluyen varios estándares como WSDL, WSSecurity, WS–Transaction entre otros desarrollados por W3C y potenciado por empresas como Microsoft e IBM que brindan soporte de herramientas para desarrollar servicios web basados en SOAP, pero REST no se queda atrás ya que existe un fuerte número de defensores de este enfoque en que las soluciones de servicios web se pueden desarrollar simplemente representando y exponiendo los recursos del sistema, y transfiriendo datos a través de HTTP. En el año 2012 una de las compañías de medios sociales llamada Facebook comenzó a desarrollar una especificación de servicio web basado en un lenguaje de queries, en la que permite al cliente decidir qué datos pedir al servidor y de qué forma le conviene recibirlos sin modificar el backend, y así reducir la sobrecarga de la transferencia, en término tanto en la cantidad de datos transferidos innecesariamente, como del número de consultas separadas requeridas para hacerlo.

Comparar estos estándares no es una tarea trivial, aunque estos paradigmas son formas de crear servicios web, difieren en la forma en que se procesan los datos y en los servicios ofrecidos. SOAP es un protocolo de intercambio de mensajes basado en XML, mientras que REST es un principio de diseño que se adhiere estrechamente a la arquitectura cliente-servidor y defiende el uso de mínimos métodos HTTP, y GraphQL se basa en queries en la que proporciona una interfaz unificada entre el cliente y servidor para la captación y manipulación de datos.

1.2. Planteamiento del Problema

La sociedad ha brindado nuevas herramientas tecnológicas, tal como los dispositivos móviles que no tienen restricción de acceso a diversos programas y aplicaciones con las que fueron diseñados en un inicio, sin embargo, los programas han ido presentando mejoras en su capacidad de almacenamiento y ejecución de aplicaciones. Dichas aplicaciones actualmente pueden ejecutarse desde un ordenador o un Smartphone, a pesar de ello en ningún caso los servicios dejarían de ser funcionales, rápidos, seguros y dinámicos.

Se tiene en cuenta que los estándares de servicio web que más han impactado en la comunidad de desarrolladores ha sido SOAP, REST y GraphQL, razón por la que surge un debate acerca qué estándar sería más adecuado para la elaboración o desarrollo de aplicativos webs basándonos en el rendimiento y el tiempo de respuesta.

El tiempo de respuesta según Bixby [1], calcula la latencia de solicitud, que es el tiempo que toma el servidor para responder una petición especifica.

Esta investigación surge por la necesidad de elegir un estándar óptimo para la realización de aplicativos webs ya que como mencionan Echeverría y Paumier [2], es primordial la calidad del software para el éxito de los negocios, la enseñanza y la seguridad de las personas.

1.2.1. Formulación del Problema

¿Qué impacto tiene el estándar empleado en el servicio web sobre el rendimiento de ejecución en aplicaciones de hipermedia?

1.2.1.1. Pregunta de Investigación

¿Cuáles son las métricas de evaluación del rendimiento de aplicaciones web desarrolladas bajo los estándares SOAP, REST y GraphQL?

1.3. Justificación

Han surgido numerosos cambios en el desarrollo tecnológico, evidencia de ello son las considerables tecnologías de aplicación móvil u ordenadores que se ofrecen en el mercado. El desarrollo de la globalización que ocasionan solicitudes veloces de innovación tecnológica requeridas por los usuarios para solucionar todo tipo de necesidades desde solicitud y ofrecimiento de información, mejores oportunidades de investigación, comunicaciones en tiempo real hasta la generación de grandes transacciones comerciales.

De los servicios web SOAP, REST, GraphQL se determinará cuál de ellos es mejor en rendimiento, lo cual servirá y beneficiará como apoyo a los desarrolladores en sus próximas implementaciones.

Distintas entidades manejan distintos marcos de trabajo en las cuales no tienen el conocimiento apropiado de cómo implementar los servicios web y otras instituciones que no cuentan con los estándares y patrones de desarrollo de estas arquitecturas, es por tal motivo que se desarrolla este proyecto el cual brindará información adecuada sobre los estándares de servicios web, rendimiento y tiempos de respuesta, para lograr la aprobación del cliente y mejores resultados para las entidades de desarrollo.

1.4. Objetivos

1.4.1. Objetivo General

Analizar los estándares orientados a servicios web SOAP, REST y GraphQL utilizando métricas para determinar el impacto en el rendimiento de aplicativos webs.

1.4.2. Objetivos Específicos

- Construir aplicaciones que consuman los servicios webs SOAP, REST y GraphQL.
- Determinar parámetros comparativos de los servicios web para establecer el rendimiento de cada estándar.
- Diseñar un método de pruebas de rendimiento, mediante criterios de evaluación para SOAP, REST y GraphQL.
- Comparar el rendimiento de aplicaciones webs con los estándares SOAP, REST y GraphQL, según los resultados del plan de pruebas benchmark.

CAPÍTULO I: MARCO TEÓRICO

1.5. Antecedentes

En esta sección se presentan varios trabajos de investigación relacionados rigurosamente al tema del análisis comparativo entre los estándares de servicios web SOAP, REST y GraphQL, mismos que darán un aporte significativo a la investigación.

En la investigación de Hidalgo y Jiménez [3], en la cual mencionan el desarrollo de una aplicación móvil aplicada a la gestión de información geolocalizada del turismo de la provincia de Chimborazo, esta aplicación se desarrolló en dos estándares orientado a servicios web SOAP y REST, bajo el concepto y metodología SCRUM para implementar mejores prácticas, todo esto con el objetivo de analizar el comportamiento del rendimiento de dicha aplicación y así brindar y proporcionar una aplicación rápida, de calidad y promover el turismo en la provincia de Chimborazo.

El estudio del rendimiento determinó que REST presenta un mejor desempeño para sistemas distribuidos en dispositivos con escasos recursos como PDAs o teléfonos móviles.

En el artículo de Bora y Bezboruah [4] mencionan que su principal objetivo es comparar los aspectos y descubrir los factores que impactan en el rendimiento de los servicios web SOAP y REST, para ello han desarrollado, implementado y probado dos prototipos de servicio web, uno basado en SOAP y otro en REST, considerando métricas analizadas como el rendimiento, escalabilidad, carga y estabilidad del sistema.

Para implementar el servicio web, ambos estándares se basan en la tecnología JAVA implementada con el servidor apache Tomcat y MySQL como servidor de base de datos.

Se enfocaron en los indicadores:

- Tiempo de respuesta que es el tiempo de espera para recibir la respuesta del servicio cuando el usuario suministra datos o envía solicitudes.
- Throughput se refiere al volumen de información que maneja el sistema dentro de una unidad de tiempo.

En esta investigación los autores concluyen en su análisis que los valores de los atributos de rendimiento como el Throughput y el tiempo de respuesta basado en la arquitectura REST es mucho menor que los atributos de rendimiento del servicio web basado en SOAP, este último es estable hasta los 800VU sin errores, pero al llegar a los 1500VU el servicio muestra un bajo rendimiento con un rechazo de conexión del 61%.

Pavan, Sanjay y Zomitza en su artículo [5], describen que crearon servicios web basados en SOAP y REST que realizan operaciones *crear, leer, actualizar y eliminar* (CRUD) ambos servicios realizan series de operaciones de intercambio de datos en un servidor de base de datos con clientes cableados e inalámbricos.

En este análisis comparativo utilizaron métricas de rendimiento para comparar los servicios web mediante los indicadores:

- Medición de tiempo de respuesta, en donde el escenario general para medir los tiempos de respuesta del servicio implica una aplicación cliente que invoca una funcionalidad apropiada proporcionada por el servicio CustomerInfo junto con los datos relevantes en una instancia de tiempo (milisegundos).
- Medición Throughput, se define como los datos procesados por segundo, se mide la cantidad de bytes de aplicaciones por segundo y la cantidad de clientes por segundo.

Los autores determinaron que el servicio web basado en REST se desempeñó mejor que el servicio desarrollado en SOAP con clientes conectados y el rendimiento en KB por segundo con clientes inalámbricos.

En el estudio [6] tratan de mejorar la interoperabilidad del ecosistema del Observatorio de Empleo y Empleabilidad (también conocido como OEEU), ya que necesitan gestionar estos flujos de datos porque los ecosistemas necesitan altos niveles de interoperabilidad, y que permita la colaboración y la independencia de los componentes internos y externos, para ello realizan implementaciones tanto en API’s REST como en GraphQL, y así analizar cuantitativamente el tiempo de respuesta de cada una y determinan la implementación más óptima.

La investigación de Čechák [7] se trata de un minucioso estudio de la tecnología GraphQL como una posible alternativa al Delivery API de Kentico Cloud. Y para ellos la investigación se divide en cuatro partes, en la cual la primera parte trata de una explicación del estado actual y la estructuración de datos de la API de Kentico Cloud, en la segunda parte se comparan los conceptos REST y GraphQL para la implementación, en la tercera parte se describe la implementación de la solución de prueba de concepto, y por último a través del parámetro de tiempo de respuesta se realiza una medición de las dos tecnologías (REST y GraphQL) y se presentan los resultados.

1.6. Bases Teórico – Científicas

1.6.1. Servicios Web

Un servicio web es un componente de software que se comunica con otras aplicaciones codificando los mensajes en XML y enviando estos mensajes a través de protocolos estándares de Internet tales como el Hipertexto Transfer Protocol (HTTP)[8].

Básicamente un web Service son elementos que permiten a las aplicaciones comunicarse entre sí, sin importar el lenguaje o plataforma en que se desarrollen, en la figura 1 se detalla el proceso de un servicio web.

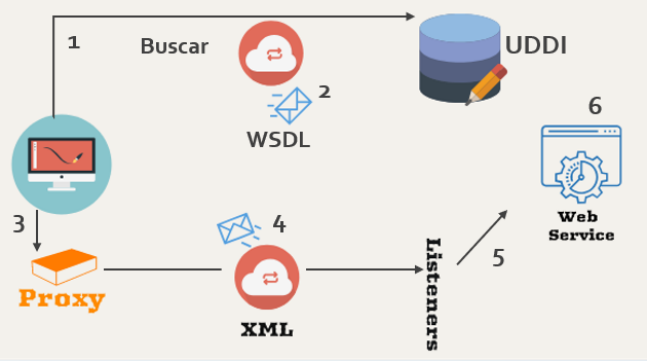


Figura 1. *El flujo de proceso de un servicio web*

1.6.1.1. Ventajas de los Servicios Web

Es importante destacar las ventajas y desventajas de los servicios web[9]:

- Garantizan la interoperabilidad entre los servicios sin importar su plataforma.
- Minimizan los problemas de firewall por usar el protocolo de comunicación HTTP.
- Cuenta con herramientas de fácil implementación, optimizando recursos y presupuesto.

1.6.1.2. Desventajas de los Servicios Web

- Existe poca información de implementación de servicios web para algunos lenguajes de programación.
- Al estar apoyado en el protocolo HTTP, pueden esquivar medidas de seguridad basadas en firewall cuyas reglas tratan de bloquear la comunicación entre programas.

1.6.2. Arquitecturas Web

Las arquitecturas web han pasado por muchos enfoques y soluciones sobre cómo construir una aplicación web y cada cierto tiempo llegan ideas nuevas e innovadoras que hacen al sector evolucionar, a continuación, en la figura 2 se detallan los enfoques que se han producido en estos últimos años.

Spaguetti	• Destaca su sencillez a nivel de arquitectura y como desventaja su poca flexibilidad.
Modelo 1	• Promueve la modularización y el uso de componentes a traves de la programación.
Modelo 2 MVC	• Separación de responsabilidades entre: Vista, Controlador y Modelo.
Modelo 2 FrontController/Enrutador	• Existe únicamente un controlador principal y se gestiona a través de acciones.
Web & Ajax	• Surge para mejorar el rendimiento entre cliente y servidor.
SPA	• Responsabilidad al navegador que se encargue de cargar las vistas y datos usando Ajax.

Figura 2. *Evolución de las arquitecturas web.*

1.6.3. Servicios web SOAP

SOAP es un formato de mensaje XML utilizado en interacciones de servicios web, habitualmente los mensajes se envían sobre HTTP o JMS, pero pueden utilizar otros protocolos, en un servicio web determinado se describe mediante la definición WSDL[10].

Derivado de las siglas Simple Object Access Protocol es un formato de mensaje XML. Este protocolo deriva de un protocolo creado por David Winer, XML-RPC en el año 1998. Con este protocolo se pedían realizar RPC o remote procedure calls.

SOAP es considerado como un formato a través del cual se define el intercambio de datos XML entre dos usuarios, sin considerar la plataforma de lenguaje o programación que se emplee, su modelo de empaquetado de datos modular y la serie de mecanismos de codificación de datos se encuentra estructurado por varias extensiones, derivadas en comunicación, formatos de envío, mensaje, enrutamiento, entre otras.

SOAP implementa tecnologías de fácil adaptación a un dispositivo móvil, ya que se conoce su limitación y consumo de recursos, además del formato en el cual se remiten los mensajes que generan un mayor consumo de datos y por ende demanda memoria más amplia para tener un funcionamiento óptimo.

1.6.3.1. Objetivos de SOAP

Los objetivos que proporcionan el cumplimiento de los servicios SOAP se basan en un modelo estándar y se toman en consideración las siguientes:

- Establecer un protocolo que esté basado en HTTP para la transmisión y XML para la codificación de datos [11].
- Independencia de plataforma hardware, lenguaje de programación e implementación del servicio Web [8].

En contexto de ello, la usabilidad de SOAP se encuentra basada en el uso de protocolos ligeros y estándares que permitan una óptima conexión para la transferencia de datos.

1.6.3.2. Estructura de un mensaje

El formato de mensajes SOAP define una estructura de sobre o envoltura que se compone de un encabezado o cuerpo del mensaje, los datos con los que se define el encabezado aumentan de la funcionalidad del mensaje tales como direccionamiento, seguridad y mensajería confiable. El cuerpo contiene datos a ser transmitidos y soporta bloques de elementos XML, texto u otro contenido como se muestra la figura 3 y se representa mediante etiquetas cómo se observa en código 1.

Figura 3. *Estructura del mensaje SOAP* [9].

Encabezado del mensaje

Es opcional, por su flexibilidad es de gran ayuda y dicha información puede ser modificada a lo largo de su vida útil.

Cuerpo del Mensaje

Dentro de esta se trasfieren los datos en XML simples, texto o estructuras complejas que no sean binarias o multimedia, permitiendo que el servicio potencialice su funcionalidad.

Código especial adjunto

“Tratar de transmitir contenido misceláneo, como documentos multimedia y binario es muy costoso en términos de procesamiento y tamaño de datos, para ello SOAP crea una estructura compuesta, que separa al contenido especial y al mensaje en dos secciones bien definidas”[9].

Código 1. *Ejemplo de solicitud de servicio usando la estructura de un mensaje SOAP.*

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:sch="http://com.pe.apps/schema">
   <soapenv:Header/>
   <soapenv:Body>
      <sch:employeeRequestElement>
         <sch:action>1</sch:action>
         <!--Optional:-->
         <sch:empname>Cristhian Recalde</sch:empname>
         <!--Optional:-->
         <sch:empalias>Andrescr</sch:empalias>
         <!--Optional:-->
         <sch:emppassword>024896</sch:emppassword>
         <!--Optional:-->
         <sch:empdni>0878963541</sch:empdni>
         <!--Optional:-->
         <sch:empemail>azw1021@gmail.com</sch:empemail>
      </sch:employeeRequestElement>
   </soapenv:Body>
</soapenv:Envelope>
```

1.6.3.3. Lenguaje de Marcado Extensible (XML)

Los servicios SOAP emplean XML para la estructuración de información base para el descubrimiento, descripción y mensajes, destacando que al ser una plataforma independiente no permite su uso en SOAP al igual que en REST. Para que un documento XML sea bien definido debe cumplir correctamente con la sintáctica de XML, y para que sea válido debe cumplir con un esquema según la naturaleza del mensaje que se esté estructurando[12].

Las partes con las que debe contar un documento XML son:

- Declaración
- Elementos
- Atributos
- Espacios de nombres
- Comentarios

1.6.3.4. Lenguaje de definición de servicios web WSDL

El WSDL permite realizar una descripción extensa sobre los servicios web, especificando una interfaz abstracta mediante la cual el usuario accede a los servicios y las características de su uso[13].

Elementos del WSDL

Se encuentra compuesto por el Types que contiene las definiciones de los datos empleados por el sistema. De igual forma dispone del Message, el cual es una definición de los datos usados para la comunicación. Así mismo se encuentra la Operation que es una acción admitida por el servicio, el Port Type es el conjunto de operaciones, el Binding que especifica el protocolo de uso y sus características, el Port se entiende como un único punto final definido como una combinación de un enlace y una dirección de red y por último el Service es una colección de puntos finales relacionados.

En consideración de ello, en la figura 4 se exponen los elementos del WSDL:

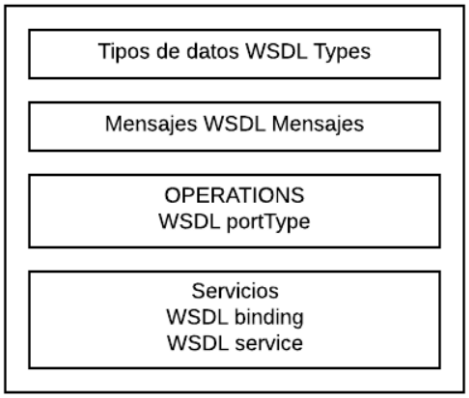


Figura 4. *Elementos del mensaje WSDL [9].*

Se establece que las herramientas de desarrollo construyen el documento WSDL de forma automática, por ende, es necesario que se defina la interfaz de los servicios de forma que el usuario pueda interactuar de forma dinámica.

1.6.4. Servicios web REST

Estos son considerados como un estilo de arquitectura para los sistemas distribuidos de hipermedia generando de tal forma un avance en torno al desarrollo de las aplicaciones, considerando que se toma también como un modelo de Transferencia de Estado Representacional o RESTful (implementación de REST), reconocido por la optimización de recursos y el mejoramiento del trabajo en torno a las aplicaciones y sitios.

Estilo arquitectónico que consiste en clientes y servidores y que expone funcionalidades mediante recursos identificados por URI, los clientes interactúan con los recursos mediante métodos generando solicitudes a los servidores, estos la procesan, generando una respuesta conveniente[3].

En torno a ello la construcción de solicitudes y respuestas se realizan alrededor de la transferencia de representaciones de los recursos, siendo uno de los más importantes la representación del recurso capturado en estado actual. A diferencia del Protocolo SOAP, el consumo de datos es menor ya que no emplea el formato XML, además de que no requiere de unas cabeceras.

En cuestión de ello se establece que el servicio web Restfull es un diseño que se encuentra basado en la arquitectura REST, la cual se encuentra enfocada en construir aplicaciones que empleen el protocolo de manera explícita, y con relación a ello optimicen recursos y mejoren el desarrollo del trabajo.

1.6.4.1. Principios

RESTful sigue 4 principios fundamentales [14]:

- Interfaz uniforme para la identificación de recursos.
- Utilización de métodos estandarizados de HTTP.
- Comunicación sin mantener estados.
- Recursos con múltiples representaciones.

Utilización de métodos estándares de HTTP

HTTP expone los métodos estandarizados: GET, PUT POST, DELETE, que son utilizados por desarrolladores estableciendo relaciones con las operaciones (CRUD) de Leer, Actualizar, Crear y Borrar de la siguiente manera:

- GET: Permite obtener la información del servidor.
- PUT: Permite cambiar o actualizar un recurso.
- POST: Enviar información al servidor para que actualice o agregue información.
- DELETE: Permite eliminar información del servidor.

1.6.4.2. Características

Se pueden identificar las principales características de los servicios web RESTFULL, en el Código 2 se refleja un endpoint para listar clientes desde el lenguaje de programación Java.

- Se orientan recursos descritos por un id único.
- Transfiere datos en formatos XML y JSON.
- Son asertivos en ambientes con un bajo ancho de banda, al no poseer estados optimiza el almacenamiento en memoria cache.
- Para la identificación de cada recurso se utiliza un URL en donde los clientes lo invocan directamente.
- Permite reutilización de mecanismos de autorización, cifrado y autenticación por HTTP.

Código 2. *Endpoint REST para obtener todos los clientes.*

```
@RequestMapping(value = "/listar", method = RequestMethod.GET)
public String listar(@RequestParam(name = "page", defaultValue = "0") int page, Model model) {

    Pageable pageRequest = PageRequest.of(page, 5);
    Page<Cliente> clientes = clienteService.findAll(pageRequest);
    PageRender<Cliente> pageRender = new PageRender<>("/listar", clientes);
    model.addAttribute("titulo", "Listado de Clientes");
    model.addAttribute("clientes", clientes);
    model.addAttribute("page", pageRender);
    return "listar";

}
```

1.6.4.3. Web Application Description Language (WADL)

El Lenguaje de descripción de aplicaciones web (WADL) es un formato de archivo basado en XML que proporciona una descripción legible por máquina de aplicaciones web basadas en HTTP.

Debido al incremento e interés por parte de la comunidad de desarrolladores, los servicios REST cuentan con un lenguaje de descripción que se ajusta a sus requisitos, el cual se denomina WADL un lenguaje de descripción muy similar al WSDL.

1.6.5. Lenguaje de consulta GraphQL

Es un lenguaje de consulta fuertemente tipado desarrollado por la compañía de medios sociales llamada Facebook en el año 2012, proporciona una sintaxis flexible para describir los requisitos de datos y las interacciones para crear aplicativos[15], y a partir del año 2015 se anuncia públicamente el lanzamiento del servicio como un borrador de especificación de idioma. [16] El lenguaje fue concebido por varios objetivos generales:

- Reducir la sobrecarga de la transparencia de datos en relación con los modelos de servicio web similares a REST, en términos tanto de la cantidad de datos transferidos innecesariamente, como del número de consultas separadas requeridas para hacerlo.
- Reducir los errores causados por consultas no válidas por parte del cliente.
- Admitir un modelo de datos en evolución sin la necesidad de versiones API.

1.6.5.1. Schema

Describe los tipos disponibles y sus relaciones, así como los puntos de entrada para los clientes; consultas y mutaciones. En el código 3 se muestra un Schema de ejemplo.

Código 3. *Definición de esquema GraphQL.*

```
schema    {
     query: Query
     mutation: Mutation
}

type Usuario   {
     id: ID!
     Nombre: String
     posts: [Post]!
     Seguidores: [Usuario]!
     Siguiendo: [Usuario]!
}

type Post {
     id: ID!
     autor: Usuario
     contenido: String
     createdAt: String
     replies: [Post]!
     replyTo: Post
}

type Query {
     timeline(of: String): [Post]!
     Usuario(Nombre: String): Usuario
     usuarios: [Usuario]!
}

type Mutation {
     escribirPost (Nombre: String, contenido: String, replyTo: String = null): Post
     NuevoUsuario (Nombre: String): Usuario
     SeguidoresUsuario (me: String, other: String): Usuario
}
```

El código 3 muestra un ejemplo del Schema de GraphQL. En la raíz se define el elemento de esquema que además se divide en consultas (queries) y Mutaciones (Mutation). Un tipo concreto, como el Usuario, tiene campos a rellenar, en la que un signo de admiración indica que el campo no puede ser nulo. Los corchetes alrededor de la definición indican una matriz del tipo cerrado. Los parámetros como en las consultas y mutaciones se nombran, por lo tanto, se pueden especificar en cualquier orden.

1.6.5.2. Queries

El tipo de consulta es el punto de entrada para cada solicitud del lado del cliente. Dentro de este tipo todas las posibilidades para consultar datos se definen mediante la inclusión de otros tipos de objetos, quiere decir que, para recuperar datos, se le envía una consulta, El código 4 muestra la consulta y en el código 5 se visualiza el resultado.

***Código 4**. Consulta en GraphQL basado en el código 3.*

```
query {
    timeline(of: "Cristhian") {
    ...basicPostFields
    replies {
    ...basicPostFields
        }
    }
}

fragment basicPostFields on Post {
    content
    autor{
    Nombre
    }
}
```

***Código 5.** Resultado de la consulta del código 4.*

```
{
   "data": {
      "timeline": [
      {
         "contenido": "Mi primer blog!",
         "autor": {
         "Nombre": "CR_Andres"
       },
         "replies": [
         {
            "contenido": "Muy interesante!",
            "autor": {
            "Nombre": "Cristhian"
                  }
               }
            ]
         }
      ]
   }
}
```

1.6.5.3. Mutaciones

Las mutaciones permiten modificar datos en el servidor GraphQL, las mutaciones devuelven valores igual que las consultas, por lo tanto, un cliente puede consultar datos en función del valor de retorno de una mutación.

***Código 6**. Mutación en GraphQL, agregando una nueva publicación*

```
mutation {
    writePost (Nombre: "Cristhian", content: "Mis primeras vacaciones") {
        autor{
        posts{
        contenido
            }
        }
    }
}
```

1.6.5.4. Arquitectura GraphQL

GraphQL requiere una interfaz, que es proporcionada por un servidor, este debe procesar consultas y devolver un conjunto de datos especificado por el cliente, en la figura 5 se muestra que GraphQL tiene como posibilidad usar una puerta de enlace API en la que se puede encapsular el acceso a los sistemas externos[17].

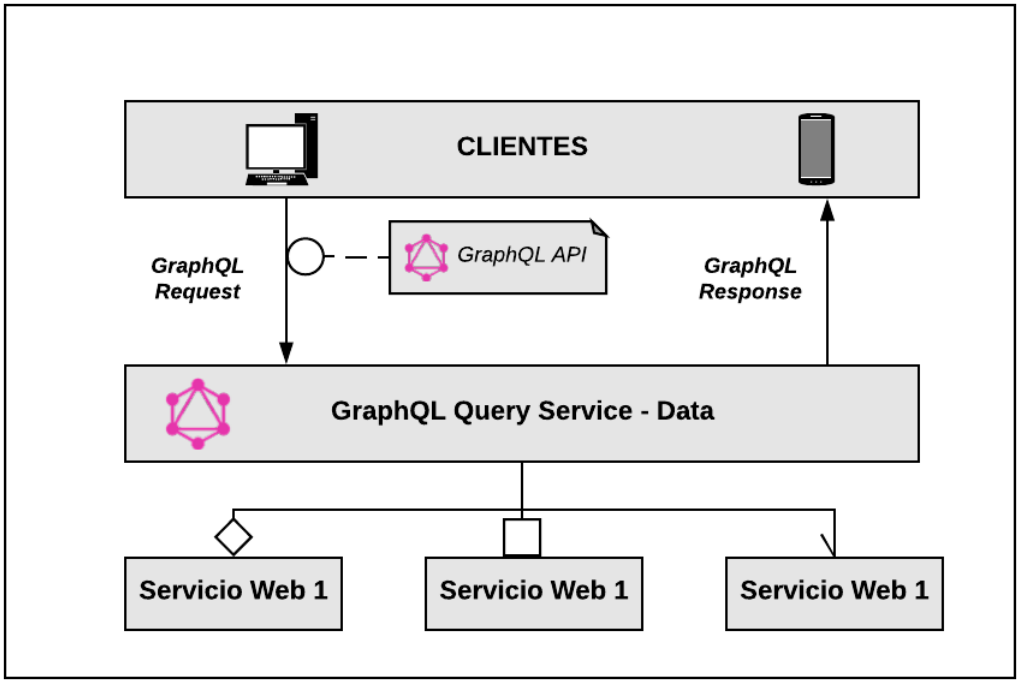


Figura 5. *Descripción arquitectónica de una aplicación web que utiliza el servicio de consulta GraphQL.*

El servicio de consultas es una instancia que se ejecuta por separado del cliente y puede considerarse como un servicio web. El servidor GraphQL permite que el cliente envíe consultas más potentes de lo que sería posible con otro estándar como SOAP o RESTful.

1.6.6. Tablas comparativas entre SOAP, REST y GraphQL

La tabla 1 muestra un cuadro comparativo de la investigación [18] de servicios web basados en SOAP y REST.

Tabla 1. *Comparación de servicios web basados en SOAP Y RESTful [12].*

SOAP	REST
Tecnología tradicional bien conocida	Tecnología nueva con respecto a SOAP.
Empresarialmente y en escenarios B2B, SOAP es muy atractivo.	Esto no quiere decir que REST no está listo para la empresa. De hecho, se conocen implementaciones exitosas en aplicaciones de misión crítica como la banca
La integración cliente-servidor es estrictamente acoplada	La interacción cliente-servidor es ligeramente acoplada
En caso de implementación, SOAP supera a REST ya que existen kits de desarrollo establecidos	REST tiene flexibilidad de interfaz
Es un formato más pesado, tanto en tamaño como en procesamiento.	REST es definitivamente liviano ya que está diseñado para una transferencia de datos liviana a través de una interfaz más comúnmente conocida, URI.
Los servicios web SOAP siempre devuelven datos XML	REST proporciona flexibilidad con respecto al tipo de datos devueltos.
Consume más ancho de banda porque una respuesta SOAP podría requerir 10 veces más bytes en comparación con REST.	Consume menos ancho de banda porque su respuesta es liviana.
Mas difícil de desarrollar, requiere herramientas.	Mucho más fácil para desarrollar servicios web.

La diferencia más notoria entre estos dos servicios es el enfoque para manejar la carga útil de la aplicación. SOAP con su formato XML es muy detallado tanto en el lado de la solicitud como en el de la respuesta, la cantidad de datos de SOAP es enorme, requiriendo más recursos y ralentizando la comunicación. REST con su flexibilidad de interfaz rápidamente se acopló mucho a la comunidad de desarrolladores.

La tabla 2 se muestra una comparación entre los servicios web SOAP y GraphQL.

Tabla 2. *Tabla comparativa entre los servicios web SOAP y GraphQL.*

SOAP	GraphQL
Tecnología tradicional, publicada en 1998	Tecnología nueva, publicada en 2015
Fuerte tipificación de datos	Fuerte tipificación de datos
Es un formato más pesado, tanto en tamaño como en procesamiento	La definición de datos del lado del cliente permite obtener exactamente los datos solicitados. Reduce la carga útil y aumenta el rendimiento de la aplicación
Único endpoint para buscar o modificar datos	Único endpoint para buscar o modificar datos
Sin mecanismo de almacenamiento en caché	Sin mecanismo de almacenamiento en caché
Posee versionado	Falta de soporte de versiones, sin versiones los desarrolladores no pueden saber cuándo necesitan actualizar su aplicación.
Los servicios web SOAP siempre devuelven datos XML	Los datos se definen mediante la inclusión de otros tipos de objetos, devolviendo datos en JSON

GraphQL al ser una innovación combina varias características sólidas de los estándares SOAP y REST, esta comparativa entre GraphQL y SOAP se puede notar que ambos utilizan URLs de endpoints para la búsqueda y modificación de datos, GraphQL es mucho más ligero, lo que reduce la carga útil de la red.

A continuación en la tabla 3 se muestra comparativa entre los servicios web REST y GraphQL dictaminada por Nogatz & Seipel en la investigación [19].

Tabla 3. *Tabla comparativa entre los servicios web REST y GraphQL*

Característica	REST	GraphQL
API Endpoints	Múltiple	Single
Formato de mensaje para consultas	String-based	String
Formato de mensaje para mutaciones	Cualquiera	String
Formato de mensaje para respuesta	Cualquiera, especialmente hipermedia	JSON
Tipo de sistema	Débilmente tipado, sin metadatos legibles por máquinas	Fuerte tipado, metadatos para la introspección, validación de consulta, introspección
Funciones incorporadas	Cacheable	Sin mecanismo de almacenamiento cache

REST fue un avance tecnológico en el área de las arquitecturas orientadas a API, pero esto dejó queriendo más a los desarrolladores y es por eso que Facebook comenzó a innovar para obtener los datos de una forma diferente, ya que en SOAP y REST una solicitud de ciertos datos devolvía todas las propiedades asociadas, incluso aquellas que el usuario no necesitaba.

GraphQL fue desarrollado para resolver ese problema, en el servidor GraphQL solo declara los datos disponibles y el cliente es quien especifica lo que debe devolver, además, puede obtener datos de diferentes bases de datos con solo una consulta.

1.6.7. Pruebas de Rendimiento

Según el libro “Introducción a la ingeniera del Software” afirman que las pruebas tienen como propósito evaluar el tiempo de respuesta del sistema o memoria que ocupa[20], de acuerdo a esta información el libro “Técnicas cuantitativas para la gestión en la ingeniería del software” del año 2007 también determina si los tiempos de respuesta del sistema tanto en condiciones normales como en condiciones especiales, se encuentra dentro de los límites predefinidos[21].

Esta prueba es llevada a cabo por un grupo de pruebas independientes para asegurar que el rendimiento del sistema está dentro de los parámetros definidos. Se puede requerir herramientas para simular clientes y cargas pesadas, y se realizan las mediciones precisas de desempeño.

El objetivo de la prueba de rendimiento es verificar el rendimiento del sistema especificado (el tiempo de respuesta y disponibilidad del servicio). Donde se simulan cientos o más accesos de usuarios simultáneos durante un intervalo de tiempo definido. Para las aplicaciones web, el rendimiento del sistema es un problema crítico porque a los usuarios de la web no les gusta esperar demasiado para responder a sus solicitudes, también esperan que los servicios estén siempre disponibles [22].

1.6.7.1. Prueba Benchmark

En términos informáticos, un Benchmark es un método destinado a medir el rendimiento de un sistema, para ello, se pone a prueba al sistema con tareas muy exigentes y variadas con la intención de medir el rendimiento[23].

El objetivo de Benchmark es someter a un sistema bajo carga para poder medir su comportamiento y el tiempo de respuesta, de esta forma se puede estimar bajo qué tareas un determinado sistema se comporta de manera fiable y efectiva o por el contrario se muestra ineficiente.

Es un programa o conjunto de programas que evalúan las prestaciones de un sistema informático reproduciendo una carga de trabajo genérica en dicho sistema informático[13].

En muchos casos las personas omiten que, al instalarse Apache, no solo descargan el servidor sino también viene un complemento muy importante para medir el comportamiento en base al rendimiento que se llama ApacheBench.

1.6.8. Pruebas de Carga (Load testing)

Para la satisfacción del cliente los productos de software gestionan altos estándares de calidad y para la construcción de aplicaciones Web se utilizan herramientas de pruebas de carga para la eficiencia y fiabilidad de los productos de software.

Estas herramientas simulan conexiones simultaneas de usuarios virtuales y permiten encontrar los puntos de quiebre de los aplicativos, revelando problemas de arquitectura o configuración[24].

Una herramienta de carga opera enviando continuamente peticiones a un sitio Web y parando por periodos de tiempos programables, para comenzar de nuevo con el envío de peticiones continuas, concurrentes y escalables, tanto como el sistema y el software de prueba lo permitan. Cabe destacar que en cada ingreso al aplicativo web se designa un usuario virtual como se refleja en la figura 6, y este permite[25]:

- Obtener resultados similares con usuarios reales conectados en forma concurrente.
- Obtener respuestas negativas por ingresos no concluidos, abandonos de sesión y rechazo de peticiones por tiempo excesivo de respuesta.

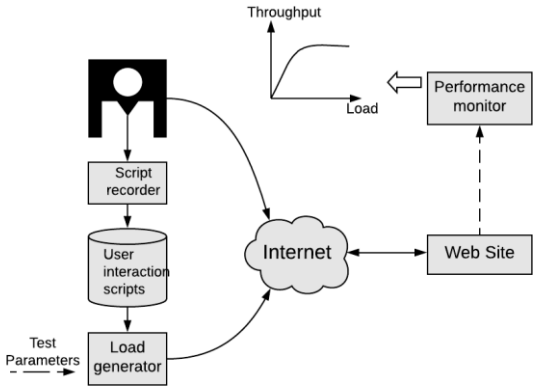


Figura 6. *Proceso de las pruebas de carga [25].*

1.6.8.1. Relación entre las pruebas de carga y el rendimiento del sistema.

Para predecir el rendimiento de una aplicación web, las pruebas de carga son muy útiles, y sus factores básicos de análisis son[25]:

- N_{VU}: Significa los números de usuarios virtuales concurrentes.
- Z: Tiempo entre las peticiones realizadas al servicio o sistema probado.
- R: Es el tiempo promedio de respuesta por cada petición del servicio o sistema probado.
- X_o: Tiempo promedio de peticiones por segundo realizadas al servicio o sistema probado.

La ley del tiempo de respuesta se representa[26], [27]:

$$R = \frac{N_{VU}}{X_O} - Z$$

Para el estudio de la variable R se relaciona la carga con el rendimiento de la aplicación adjuntando el gráfico de tiempos de respuesta frente a usuarios virtuales concurrentes como muestra la figura 7.

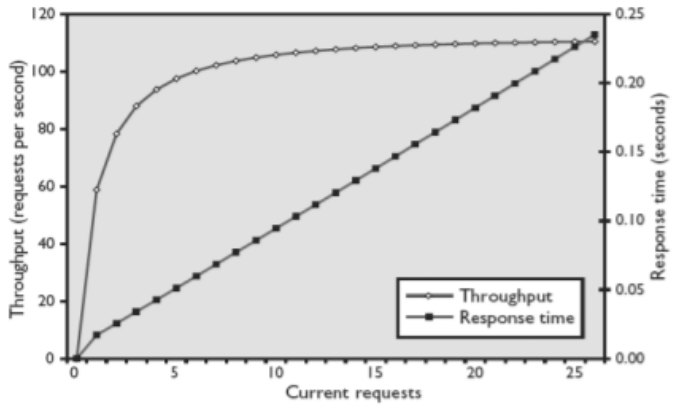


Figura 7*. Carga vs. Rendimiento [25].*

En el ámbito de pruebas de carga existen otros conceptos, que son notables al realizar un análisis de resultados y se relacionan directamente con las características de las herramientas a medir:

- Hits: Solicitud que hace un aplicativo o service web a un servidor.
- Punto de quiebre: Es la cantidad máxima de usuarios virtuales concurrentes que la aplicación o el servicio puede soportar.
- UVC: Usuarios virtuales que se configuran para generar concurrencia.
- Sesión: Tiempo en que el usuario virtual ejecuta la acción programada.
- MTR: Es el máximo tiempo de respuesta que nos señala algún objeto o elemento probado.
- Throughput: Respuesta del servidor de aplicaciones a la petición que envía el generador de carga[24].
- Response time: Es el tiempo transcurrido entre la petición y la respuesta de un servicio cliente-servidor[28].

1.6.9. Mapping Study

Como menciona Cooper [29] el Mapping Study busca seguir el flujo a través de publicaciones y se basan en el concepto de que los artículos publicados no solo representan los hallazgos, sino que, indirectamente, representan la actividad relacionada con el hallazgo.

Se ha aplicado Mapping Study porque nos permitirá responder la siguiente interrogante de la investigación:

¿Cuáles son las métricas de evaluación del rendimiento de aplicaciones web desarrolladas bajo los estándares SOAP, REST y GraphQL?

Petticrew & Roberts [30], sugieren que tal estudio "implica una búsqueda de la literatura para determinar qué tipo de estudios abordan la cuestión de la revisión sistemática se han llevado a cabo, dónde se publican, en qué bases de datos se han indexado, qué tipo de resultados se han evaluado y en qué poblaciones ".

Se ha llegado a aplicar las primeras etapas del Mapping Study, a fin de abordar adecuadamente el tema de investigación y su alcance, aunque probablemente sea mucho más amplia de lo pensado; estas etapas son:

1) Búsqueda
2) Inclusión/Exclusión
3) Sesgo/Validez

El objetivo de esta técnica permite identificar los criterios de evaluación del rendimiento en aplicativos webs con los estándares SOAP, REST y GraphQL, por eso se definió las siguientes cadenas de búsqueda:

[“Performance testing metrics” AND [“SOAP” AND/OR “REST” AND/OR “GRAPHQL”]].

[“Comparative” AND [“SOAP” AND/OR “REST” AND/OR “GRAPHQL”]]

[“Métricas de pruebas de rendimiento” AND (“SOAP” AND/OR “REST” AND/OR “GRAPHQL”]].

- La estrategia de análisis se basó en los buscadores tales como:
    - Wiley Online Library
    - IEEE Digital Library
    - Elsevier
- Estudios considerados
    - Papers, artículos científicos, tesis.
- Periodo de publicación: Año 2005 y 2012

Se considera el año 2005 puesto que en ese periodo SOAP y REST comenzaron a ser tendencia, y el año 2012 ya resaltaban las especificaciones de GraphQL.

1.6.9.1. Selección de estudios

Los criterios de inclusión para los estudios primarios son:

- Estudios descriptivos sobre métricas de evaluación del rendimiento en aplicativos webs.
- Estudios descriptivos de implementaciones de pruebas de rendimiento en estándares SOAP, REST y GraphQL.

Se excluyeron los siguientes estudios:

- Estudios que no aportan rigurosamente con el tema de investigación.
- Estudios que no responden a la interrogante de la investigación
- Estudios que no aportan ninguna noción o propuesta relevante hacia las pruebas de rendimiento en estándares SOAP, REST y GraphQL.

1.7. Marco Legal

La presente investigación se centra en la generación de tecnología que aporte al crecimiento tecnológico del país. Para simplificar lo expuesto se presenta el siguiente análisis:

- El Plan Nacional de Gobierno electrónico 2016 – 2017 versión 2.0 en base a la Carta Iberoamericana de Gobierno electrónico (2017), formula 12 principios que precautelan el derecho de los ciudadanos a relacionarse con el Estado electrónicamente. Como el:
    - "Principio de adecuación tecnológica: Garantiza que las administraciones elegirán las tecnologías más adecuadas para satisfacer sus necesidades, por lo que se recomienda el uso de estándares abiertos y de software libre debido a la seguridad, sostenibilidad a largo plazo y la socialización del conocimiento."

- La constitución de la República del Ecuador (2008) que garantiza la soberanía nacional, y define los sectores estratégicos entre los cuales están las tecnologías como hardware y software:
    - "Art. 322. Se reconoce la propiedad intelectual de acuerdo con las condiciones que señale la ley. Se prohíbe toda forma de apropiación de conocimientos colectivos, en el ámbito de las ciencias, tecnologías y saberes ancestrales…"
    - "Art. 385. El sistema nacional de ciencia, tecnología, innovación y saberes ancestrales, en el marco del respeto al ambiente, la naturaleza, la vida, las culturas y la soberanía, tendrá como finalidad desarrollar tecnologías e innovaciones que impulsen la producción nacional, eleven la eficiencia y productividad, mejoren la calidad de vida y contribuyan a la realización del buen vivir.

- Norma técnica ecuatoriana NTE INEN-ISO/IEC 29363: Tecnología de la información – Interoperabilidad de servicios web; mediante su promulgación en el Registro oficial, promueve un justo equilibrio de intereses entre proveedores y consumidores.

CAPÍTULO II: METODOLOGÍA

2.1. Diseño de la investigación

El diseño de la investigación es comparativa, como mencionan Piovani [31] y Arias [32] que este tipo de investigación tiene como finalidad identificar las diferencias y semejanzas en torno a un evento o en un mismo contexto. Debido a que se contextualizarán las herramientas para realizar comparaciones, evaluar resultados obtenidos mediante el uso de indicadores y métricas de evaluación, que medirán el rendimiento de los estándares orientados a servicios web SOAP, REST y GraphQL.

2.2. Tipo de Investigación

La investigación es cuantitativa tal y como la definen Toro & Parra [33], considerando que tiene como finalidad tomar decisiones exactas y eficaces que contribuyan a alcanzar el objetivo que se plantea en la investigación y tomando la idea de Hernández, Fernández & Baptista [34] donde utilizan procedimientos estructurados y herramientas formales para la recolección de datos y el estudio de la generalización mediante métodos estadísticos. El resultado de esta investigación debe ser el análisis del impacto de los estándares orientado a servicios web SOAP, REST y GraphQL en el rendimiento de aplicativos webs.

2.3. Métodos

Dentro de la presente investigación se emplea el método inductivo para establecer el comportamiento del tiempo de respuesta y carga de las arquitecturas, además de conceptualizar que la lógica inductiva parte principalmente de la observación repetida de los fenómenos, y se puede llegar a conclusiones como resultado de la inferencia de similitudes observadas en los casos estudiados.

2.4. Procedimiento de Investigación

En esta investigación se han aplicado dos técnicas, la observación científica, como menciona Dieterich [35] esta técnica permite observar y registrar el comportamiento de un fenómeno, la observación científica juega un papel importante porque es la condición principal para la recolección de datos.

La técnica *del Mapping Study*, nos permitirá el análisis documental de otras investigaciones con respecto al tema de investigación.

2.4.1. Técnica Observación Científica

Se ha aplicado la técnica de observación científica, usando herramientas informáticas para medir el comportamiento del sistema frente a una sobre carga de peticiones o datos, y el rendimiento en base al tiempo de respuesta que nos brinda cada estándar del servicio web SOAP y REST, los datos analizados aportaran con información verídica y concisa para determinar el estándar óptimo para el desarrollo de aplicaciones web.

2.4.2. Técnica Mapping Study

Esta técnica a través del procedimiento de las 3 etapas principales como son búsqueda, se obtuvo una selección inicial de estudios primarios en la que se evaluó la calidad en dos etapas. Primero se tuvo en cuenta los criterios de inclusión y exclusión, después se realizó una extracción de información para una revisión más precisa y detallada. Por último, se verificó la claridad de sus métodos y propuestas, esto se encuentra detallado en la figura 8.

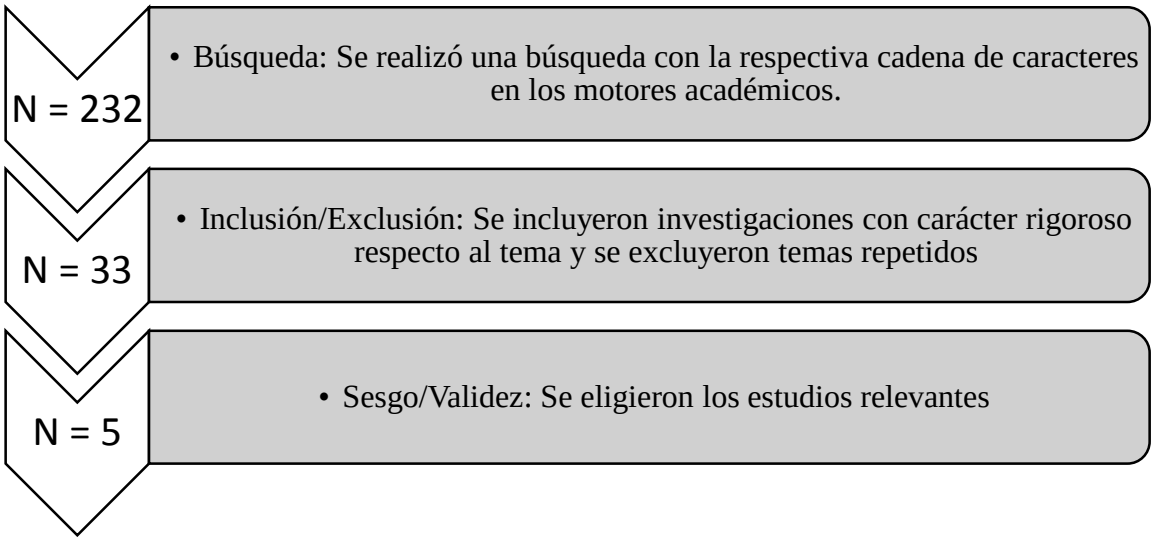


Figura 8. *Procedimiento de selección de estudios*

En la tabla 4 se muestran todos los estudios seleccionados por el análisis documental en donde se han seleccionado 232 investigaciones que se relacionan implícitamente con el tema y a través del método de inclusión y exclusión del Mapping Study, 33 estudios responden de manera rigurosa al tema, de lo cual para esta investigación se seleccionaron 5 estudios relevantes para poder resolver la interrogante de esta investigación.

Tabla 4. *Estudios primarios seleccionados*

Autores y año/Titulo	**Titulo**	**Nro. Estudios primarios**	**Tipo Investigación**	**Fuente**
Abhijit Bora & Tulshi Bezboruah. (2015)	A Comparative Investigation on Implementation of RESTful versus SOAP based Web Services	34	Experimental	IEEE
Pavan Kumar, Sanjay Ahuja, Karthikeyan Umapathy. (2012)	Comparing Performance of Web Service Interaction Styles: SOAP vs. REST	23	Experimental	IEEE
Smita Kumari & Santanu Kumar Rath. (2015)	Performance comparison of SOAP and REST based Web Services for Enterprise Application Integration	18	Experimental	IEEE
Andrea Vázquez, Juan Cruz, Francisco García	Improving the OEEU's data-driven technological ecosystem's interoperability with GraphQL	16	Experimental	IEEE
David Čechák	Using GraphQL for Content Delivery in Kentico Cloud	55	Experimental	IEEE

2.4.2.1. Discusión de métricas a evaluar

El análisis de estas investigaciones que se reflejan en la Tabla 4 se limita principalmente para identificar las métricas de medición del rendimiento en los sitios web y la forma de análisis adoptadas por los autores.

Con base en las investigaciones cabe mencionar que cada autor definió métricas para medir el rendimiento de un aplicativo web, y en ellas podemos resaltar que concordaron en dos de ellas, como se muestra en la tabla 5.

Tabla 5. *Cuadro comparativo de métricas de Mapping Study.*

Autores	**Throughput**	**Tiempo de respuesta**	**Recursos utilizados**	**Disponibilidad**
Abhijit Bora & Tulshi Bezboruah.	X	X	X	X
Pavan Kumar, Sanjay Ahuja, Karthikeyan Umapathy	X	X		
Smita Kumari & Santanu Kumar Rath.	X	X		
Andrea Vázquez, Juan Cruz, Francisco García		X	X	
David Čechák	X	X		

2.4.3. Instrumentos

Para la presente investigación se ha utilizado una ficha de observación (Anexo 1), y herramientas informáticas las cuales permitieron medir el comportamiento y rendimiento de los aplicativos webs basado en los estándares SOAP, REST y GraphQL, estas herramientas son:

- Apache JMeter: como lo describe su página oficial [36], es un software de código abierto, para medir el rendimiento y comportamiento bajo carga.

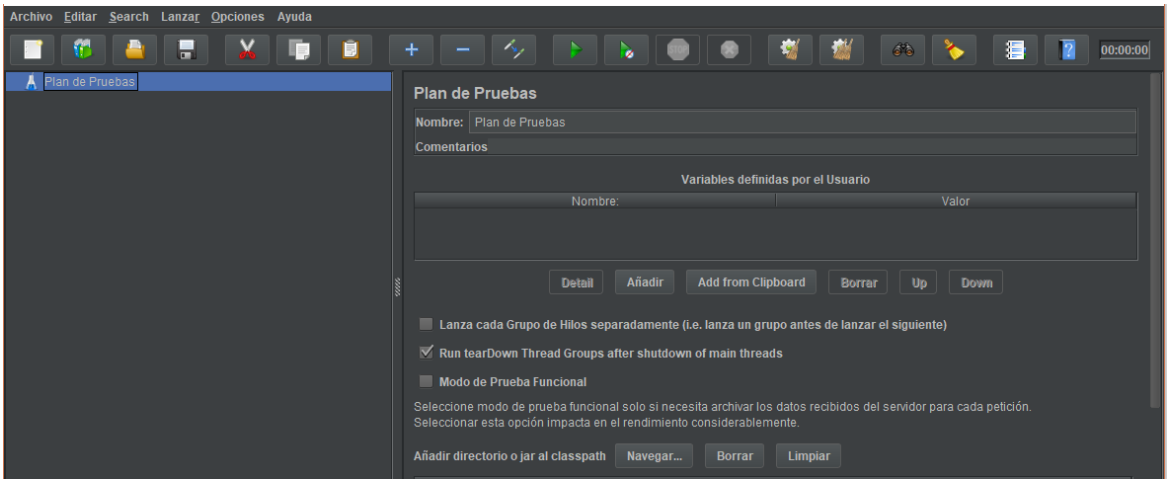


Figura 9. *Herramienta JMeter.*

- SoapUI: Es una herramienta para realización de pruebas a aplicaciones orientada a servicios web como SOAP y REST.

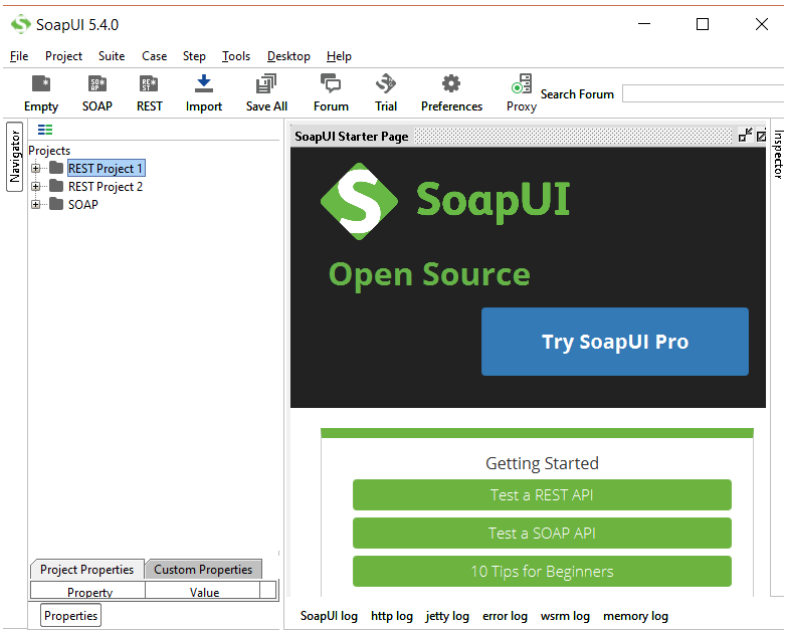


Figura 10. *Herramienta SoapUI.*

2.5. Definición Conceptual y operacionalización de variables

2.5.1. Definición de parámetros a comprobar

Para definir los siguientes parámetros y analizar los servicios web, en el desarrollo de las aplicaciones, está basado en investigaciones anteriores [4], [37], [38], afirman que el mejor parámetro para el análisis de los estándares es el rendimiento ya que en el podremos medir el tiempo de respuesta que brinda cada servicio y la capacidad que puede soportar cada estándar, en la tabla 6 se detallan los parámetros a utilizar.

Tabla 6. *Descripción del parámetro*

Parámetro	**Descripción**
Rendimiento	El tiempo mínimo de respuesta que utiliza un estándar orientado a servicio web para suministrar datos
Capacidad	Número de usuarios concurrentes

2.5.2. Determinación de variables de comparación

Para observar el comportamiento de las arquitecturas orientadas a servicios web y mostrar los tiempos de respuesta de todas las transacciones y capacidad que soporta el servicio web, se analiza la variable de la cual depende el rendimiento y la capacidad en este caso es el tipo de servicio como se muestra en la tabla 7 y tabla 8.

Tabla 7. *Operacionalización de variable dependiente*

Variable Dependiente	Descripción	Dimensiones	Indicadores	Técnicas e instrumentos	Unidades de medida
Rendimiento	Tiempo de respuesta que nos ofrece cada estándar de servicio web	Eficacia de tiempo	Tiempo promedio que se tarda en dar una respuesta cada estándar de servicio web	Observación del caso de prueba	Milisegundos

Tabla 8. *Operacionalización de variable independiente*

Variable Independiente	Descripción	Dimensiones	Indicadores	Técnicas e instrumentos	Unidades de medida
Backend	Capacidad que soporta el Backend de un lenguaje de programación	Eficacia de desempeño	Tiempo promedio que se tarda en dar una respuesta cada estándar de servicio web	Observación del caso de prueba	Milisegundos
			Capacidad	Observación del caso de prueba	Entero

2.6. Ambiente de pruebas

Para aplicar las herramientas informáticas y testear el rendimiento de los estándares de los servicios web SOAP, REST y GraphQL como se muestra en la figura 11, se ha definido un ambiente de pruebas ideal para así evitar la pérdida de datos y conexión, con el propósito de no alterar los resultados.

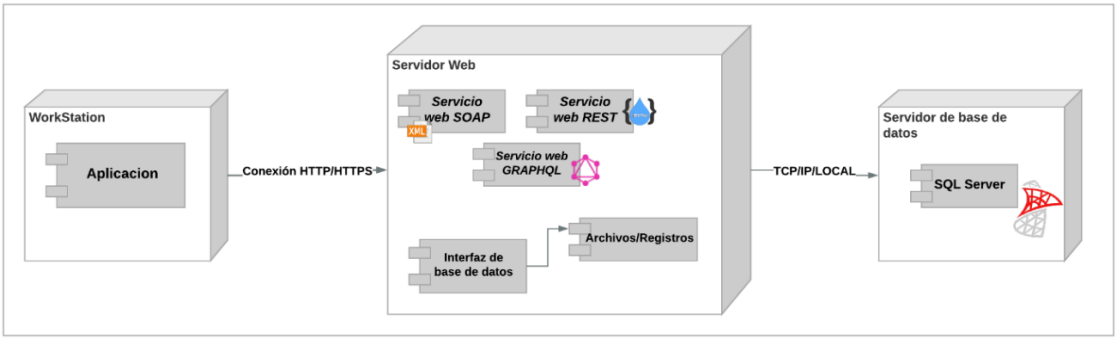


Figura 11. *Diagrama de despliegue del ambiente de pruebas*

A continuación, en la tabla 9 se detallan las características del equipo donde se ejecutaron las pruebas.

Tabla 9. *Especificaciones del equipo de pruebas*

Hardware	Software
Aspire R5-571TG Intel® Core™ i7-7500U CPU @ 2,70 GHz 2,90 GHz, RAM 12 GB, SO 64 Bits	Sistema operativo Windows 10

2.7. Metodología Benchmark

La calidad en el servicio es uno de los factores más importantes, la satisfacción al cliente provoca que éste permanezca en ellas creando un síntoma de identidad o bien decida cambiarse a otra que presumiblemente le ofrezca mejor calidad o comodidad en el servicio[39].

Al fin de obtener información que ayude a mejorar el desempeño, el benchmark es un proceso continuo que nos permite medir y comparar sistemas o uno de sus componentes.

Relacionando las metodologías de Patil & Joshi [22] y Kunhua Zhu [40], realizan procedimientos para las actividades principales de las pruebas de rendimiento, y es así como se definen las siguientes actividades del benchmark para la presente investigación:

a) Identificar los entornos de pruebas.
b) Planificar y diseñar las pruebas.
c) Configurar el entorno de pruebas.
d) Ejecutar las pruebas.
e) Analizar los resultados.

2.7.1. Identificar el entorno de pruebas.

Para la presente investigación se determina el siguiente entorno de prueba:

a) Prueba de Rendimiento. – Se mide el tiempo de respuesta (response time) de los servicios web que se han desarrollado en 3 lenguajes de programación (Java, C#, PHP), la prueba consiste en simular 50, 100, 500, 1000, 1500 usuarios virtuales como se muestra en el *anexo 1* respectivamente.

2.7.2. Planificar y diseñar las pruebas.

Para el proceso de las pruebas de los servicios web se dispuso de la herramienta SOAPUI para verificar el adecuado funcionamiento de los endpoint de cada servicio y Apache JMeter para realizar las pruebas de estrés y carga a cada aplicativo, para ello se definieron las siguientes métricas dadas por el Mapping Study:

- Response time: Se calcula el tiempo promedio en milisegundos para un conjunto de resultados.
- Throughput: Rendimiento medido en Kb/s de la respuesta del servidor.

2.7.3. Configurar el entorno de pruebas

Para el entorno de pruebas se utilizó una máquina virtual en la cual se instaló Windows Server 2016, con la ayuda del servidor HTTP Apache se desplegaron los contratos de los servicios web.

2.7.4. Ejecutar las pruebas

Las pruebas se las ejecuta utilizando las herramientas ya definidas en el punto *3.4.3. Instrumentos*, en la cual se usará:

- Apache JMeter: es una herramienta que permite probar el rendimiento tanto de recursos estáticos como dinámicos. Simula usuarios virtuales para emular una carga pesada en un servidor, red u objeto para analizar el rendimiento general bajo diferentes tipos de carga.

CAPÍTULO III: ANÁLISIS E INTERPRETACIÓN DE LOS RESULTADOS

3.1. Análisis de datos experimentales y evaluación.

Para comparar el rendimiento de aplicaciones realizadas bajo los estándares de los servicios web SOAP, REST y GraphQL, se ejecutaron las pruebas diseñadas en el capítulo anterior; que consiste en simular 50, 100, 500, 1000, 1500 usuarios virtuales concurrentes.

Para poder evaluar los resultados se utilizaron 3 componentes provistos por la herramienta.

- Summary Report: Permite visualizar un resumen de la prueba realizada.
- Agreggate Graph: Permite visualizar un resumen de la prueba realizada de una manera precisa, utiliza más memoria que los otros reportes, ya que este calcula la mediana y la línea del 90%, 95% y 99%, la cual requiere que todos los datos estén almacenados, los datos son:
    - Label: Etiqueta de la muestra.
    - Muestras: Cantidad de thread utilizados para la URL.
    - Promedio: Tiempo promedio en milisegundos para un conjunto de resultados.
    - Mediana: Valor en tiempo del percentil 50.
    - Línea de 95%: Tiempo máximo utilizado por el 95% de la muestra.
    - Error: Porcentaje de requerimientos con errores.
    - Min: Tiempo mínimo de la muestra de una determinada URL.
    - Max: Tiempo máximo de la muestra de una determinada URL.
    - Rendimiento: Rendimiento medido en los requerimientos por segundo.
    - Kb/sec: rendimiento medido en Kbytes por segundo.

Se analizaron los resultados a través de un intervalo de confianza con un nivel de confianza del 95%. Para un primer análisis, se supone que la población tiene una distribución Normal, para un segundo análisis, dado que la muestra es grande, no se requiere hacer la suposición de que la muestra tiene una distribución Normal ya que por el Teorema Central del Límite (TCL), las medias de muestras grandes y aleatorias son aproximadamente normales.

3.1.1. Análisis de datos experimentales con el lenguaje de programación C Sharp

Para el primer análisis experimental, los tres servicios web son consumidos bajo el lenguaje de programación C Sharp, en ella se ha realizado un promedio de los tiempos de respuesta y el rendimiento que ofrece bajo ciertos niveles de usuarios concurrentes.

Tabla 10. *Comparación de resultados experimentales con C Sharp.*

		SOAP		**REST**		**GRAPHQL**	
VUs	***Parámetro***	***Promedio***	***Conexión rechazada %***	***Promedio***	***Conexión rechazada %***	***Promedio***	***Conexión rechazada %***
50	Tiempo de respuesta Throughput	74000 7144	0	56800 3603,1	0	32200 1420,8	0
100	Tiempo de respuesta Throughput	346800 11580	0	269600 6981	0	186400 3520,4	0
500	Tiempo de respuesta Throughput	18850000 25156,2	0	16742000 15939,2	0	13140000 10437,1	0
1000	Tiempo de respuesta Throughput	54604000 229165,3	6,38	44516000 177568	0	22524000 122351,4	0
1500	Tiempo de respuesta Throughput	134550808 197131,6	42,06	98164457 225905,5	0	54057600 162104,3	0

3.1.1.1. Interpretación de los datos experimentales con C Sharp

En la tabla 10 se puede observar mediante los valores de las métricas analizadas por la herramienta, que el protocolo SOAP con 1000VU comienza a generar un bajo rendimiento con 6% de rechazo de conexión, al llegar a los 1500VU el protocolo comienza a aumentar el rechazo de conexión a un 42%, en el caso de la arquitectura REST y el lenguaje de esquemas GraphQL, se observan que los servicios están libres de errores y estables hasta los 1500VU.

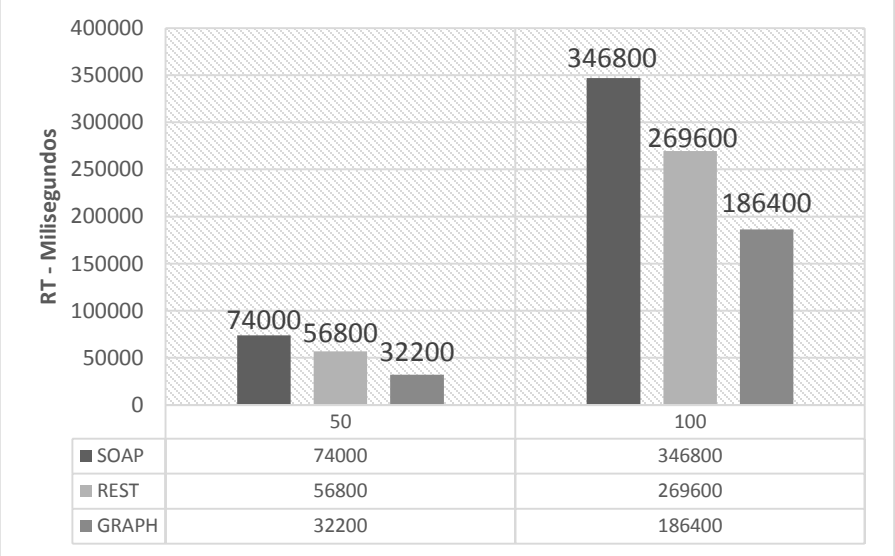


Figura 12. *Tiempo de Respuesta en el consumo de los servicios web 50-100 VU's con C Sharp.*

En la figura 12 bajo la implementación del lenguaje C Sharp muestra que la diferencia de tiempos de respuesta con 50 y 100 usuarios concurrentes es un poco convergente por lo que el usuario final no notará gran diferencia.

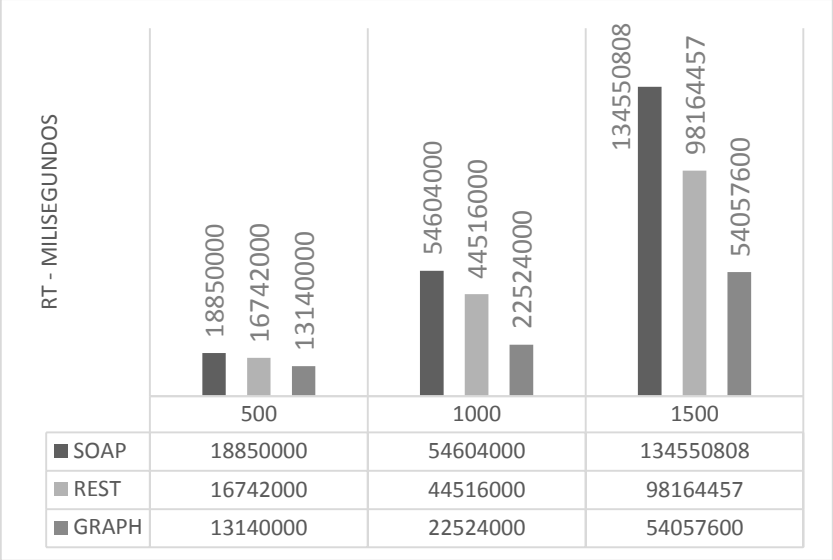


Figura 13. *Tiempo de Respuesta en el consumo de los servicios web 500-1000-1500 VU's con C Sharp.*

En la figura 13 se puede apreciar que a partir de una cantidad mayor a los 500 VU concurrentes se puede notar una gran diferencia, esto quiere decir que entre más usuarios concurrentes existan mayor será el tiempo de respuesta, por lo que desde los 1000 usuarios el protocolo SOAP comienza a generar un mayor tiempo de respuesta y guiado por la tabla 10 donde dicho servicio web comienza a producir un rechazo de conexión del 6% por lo que se le dificulta atender varias solicitudes y genera un bajo rendimiento, en los 1500 usuarios pasa algo semejante pero en esta cantidad el rechazo de conexión es mucho mayor generando un 42% por lo que los tiempos de respuesta llegan a ser muy notorios.

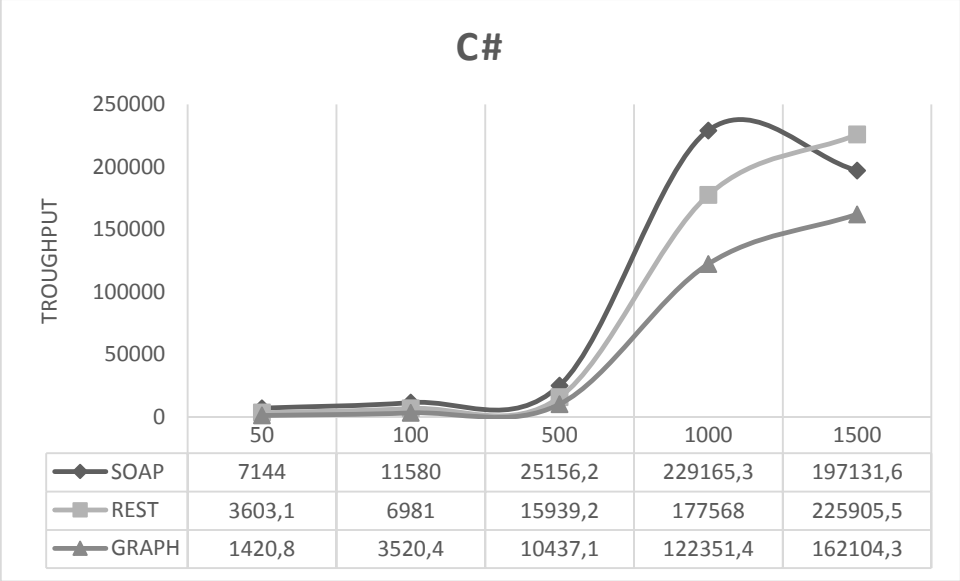


Figura 14. *Rendimiento en el consumo de los servicios web con C Sharp.*

En la figura 14 se evalúa el rendimiento que ofrece cada servicio, en ella podemos apreciar que los tres marcos no presentan una gran diferencia a partir de una cantidad menor a los 500 usuarios, llegando a los 1000 usuarios concurrentes donde se presenta un rechazo de conexión del 6% en SOAP, quiere decir que en este punto el rendimiento comienza a disminuir y en los 1500 VU's en SOAP ya presenta un bajo rendimiento por lo que el rechazo de conexión llega al 42% lo que significa que se deja de atender varias solicitudes de usuarios a no llegar a completar las peticiones.

3.1.2. Análisis de datos experimentales con Java

Para el segundo análisis los servicios web son consumidos bajo el lenguaje de programación Java, al igual que el anterior análisis se ha realizado el promedio de los tiempos de respuesta y el rendimiento que este nos ofrece de cada servicio web.

Tabla 11. *Comparación de resultados experimentales con Java.*

		SOAP		**REST**		**GRAPHQL**	
VUs	***Parámetro***	***Promedio***	***Conexión rechazada %***	***Promedio***	***Conexión rechazada %***	***Promedio***	***Conexión rechazada %***
50	Tiempo de respuesta	127600	0	107400	0	54000	0
	Throughput	18520,2		12476,6		3683,6	
100	Tiempo de respuesta	483600	0	392000	0	206000	0
	Throughput	22872,3		15749,1		8151,7	
500	Tiempo de respuesta	21358000	0	16962000	0	11734000	0
	Throughput	34385,9		17659,9		20587,8	
1000	Tiempo de respuesta	46152000	4,16	41212000	0	24988000	0
	Throughput	206897,3		119968,7		78462	
1500	Tiempo de respuesta	97706000	43,34	64224000	0	44010000	0
	Throughput	178950,1		143948,1		79553,1	

3.1.2.1. Interpretación de los datos con Java

En la tabla 11 se puede observar mediante los valores de las métricas analizadas por la herramienta, que el protocolo SOAP con 1000VU comienza a generar un bajo rendimiento con 4.1% de rechazo de conexión, al llegar a los 1500VU el protocolo comienza a aumentar a un 43.3%, en el caso de la arquitectura REST y el lenguaje de esquemas GraphQL, se observa que los servicios están libres de errores y estables hasta los 1500VU.

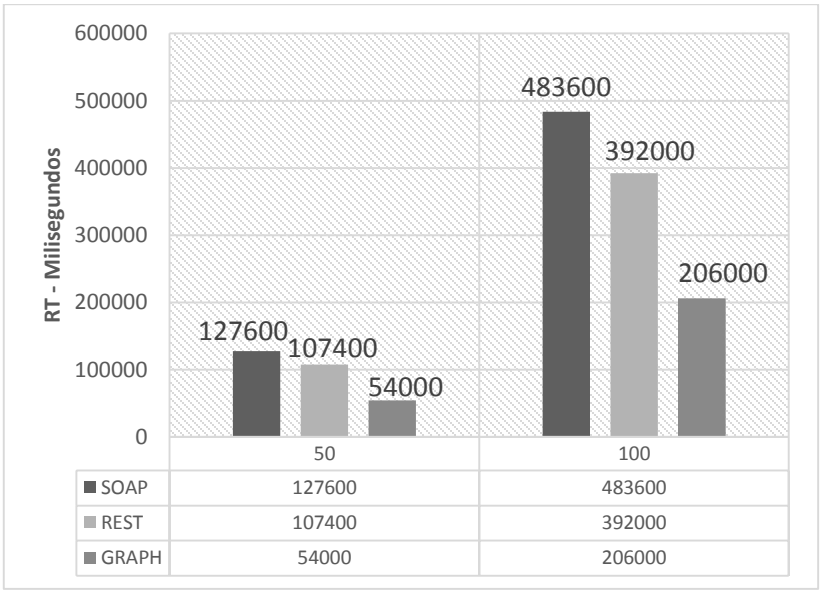


Figura 15. *Tiempo de Respuesta en el consumo de los servicios web 50-100 VU's con Java.*

En la figura 15 se puede apreciar lo mismo que en la figura 14, ya que los servicios no presentan un mayor y significativo cambio y el usuario final no notará gran cambio o diferencia en los tiempos de respuesta.

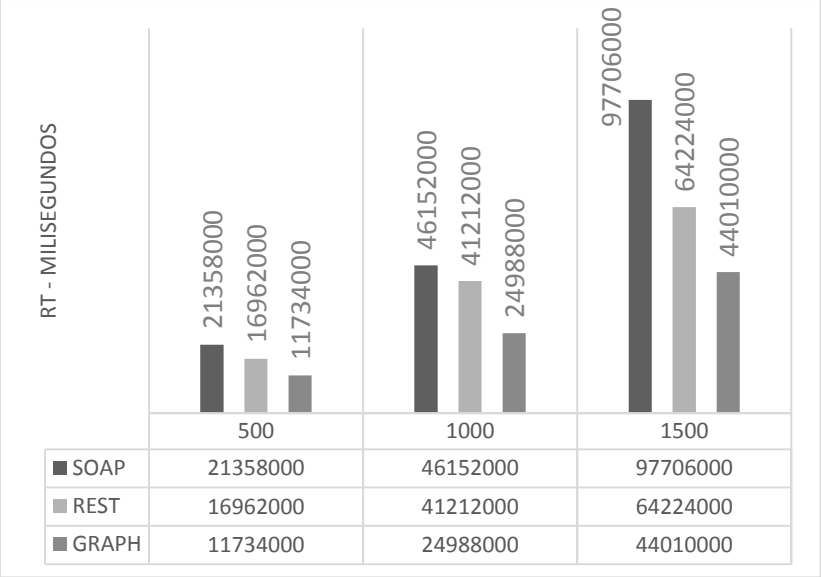


Figura 16. *Tiempo de Respuesta en el consumo de los servicios web 500-1000-1500 VU's con Java.*

En la figura 16 se muestra que los tiempos de respuesta llegan a ser notorios a partir de una cantidad mayor de los 500 usuarios concurrentes, desde los 1500 VU's comienza a generar mayores tiempos y guiado por la tabla 11 el protocolo SOAP genera un 43% de rechazo de conexión, mientras que los otros servicios no producen errores y se mantienen estables.

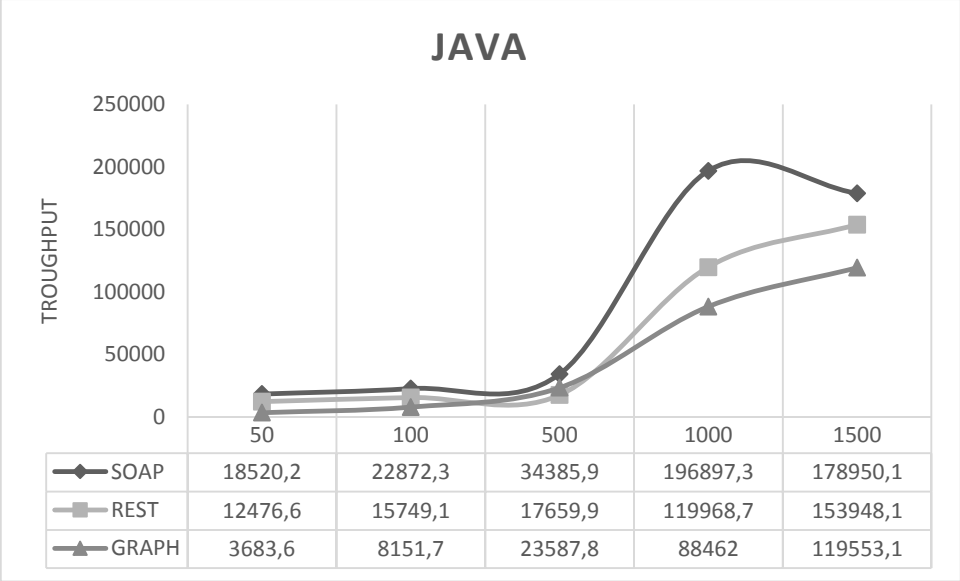


Figura 17. *Rendimiento en el consumo de los servicios web con Java.*

En la figura 17 se puede apreciar el rendimiento que producen los tres servicios con el lenguaje de Java, por ende, no genera una mayor diferencia a partir de una cantidad mayor a los 500 usuarios, y es que a partir de los 1000 usuarios el protocolo SOAP comienza a generar errores y su rendimiento comienza a bajar por lo que no puede procesar el 43% de las solicitudes asignadas.

3.1.3. Análisis de datos experimentales con PHP

Para el tercer análisis de los servicios web, se ha llegado a consumir bajo el lenguaje de programación PHP, se realiza un promedio de los tiempos de respuesta y el rendimiento que ofrece dichos servicios.

Tabla 12. *Comparación de resultados experimentales con PHP.*

		SOAP		**REST**		**GRAPHQL**	
VUs	***Parámetro***	***Promedio***	***Conexión rechazada %***	***Promedio***	***Conexión rechazada %***	***Promedio***	***Conexión rechazada %***
50	Tiempo de respuesta	107800	0	102800	0	48800	0
	Throughput	5938,1		3339,7		2680,9	
100	Tiempo de respuesta	417600	0	284000	0	228000	0
	Throughput	15263,1		6854,7		5946,1	
500	Tiempo de respuesta	30722000	0	25638000	0	19658000	0
	Throughput	50205,7		37743,5		48727,5	
1000	Tiempo de respuesta	62980000	6,04	48556000	0	31416000	0
	Throughput	354937		223645,7		151329,2	
1500	Tiempo de respuesta	173522000	45,38	109007422	0	72050000	0
	Throughput	267412,2		311580,6		164182,8	

3.1.3.1. Interpretación de los datos con PHP

En la tabla 12 se puede observar mediante los valores de las métricas analizadas por la herramienta, que el protocolo SOAP con 1000VU comienza a generar un bajo rendimiento con 6% de rechazo de conexión, al llegar a los 1500VU el protocolo comienza a aumentar a un 45%, en el caso de la arquitectura REST y el lenguaje de esquemas GraphQL, se observan que los servicios están libres de errores y estables hasta los 1500VU.

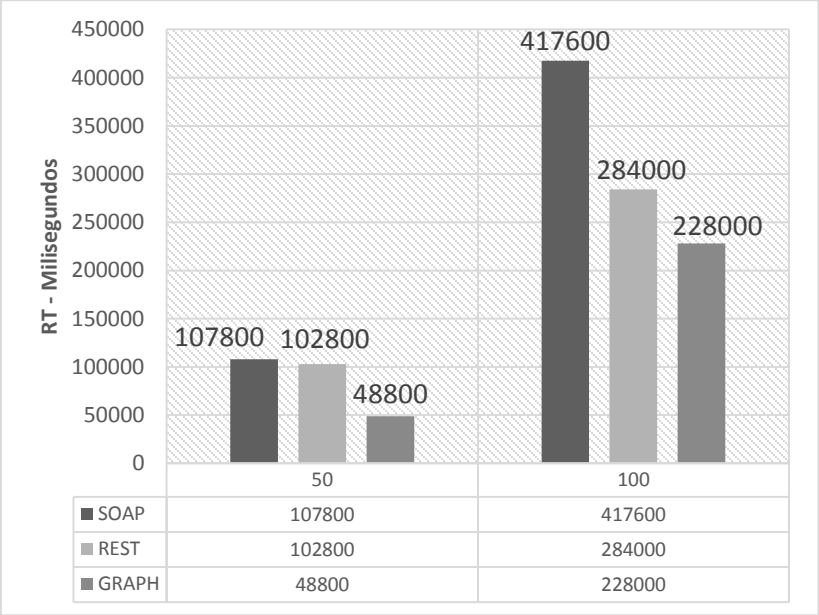


Figura 18. *Tiempo de Respuesta en el consumo de los servicios web 50-100 VU's con PHP.*

En la figura 18 no cambia mucho el panorama comparándolo con las figuras 16 y figura 17, en la cual se puede definir que los servicios web con una cantidad leve de usuarios no genera una gran notoriedad en los tiempos de respuesta.

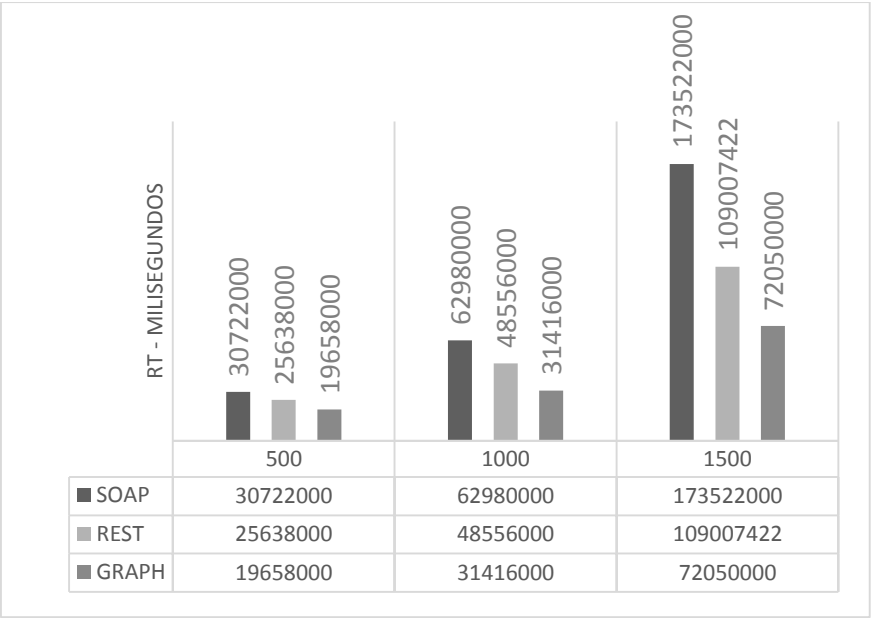
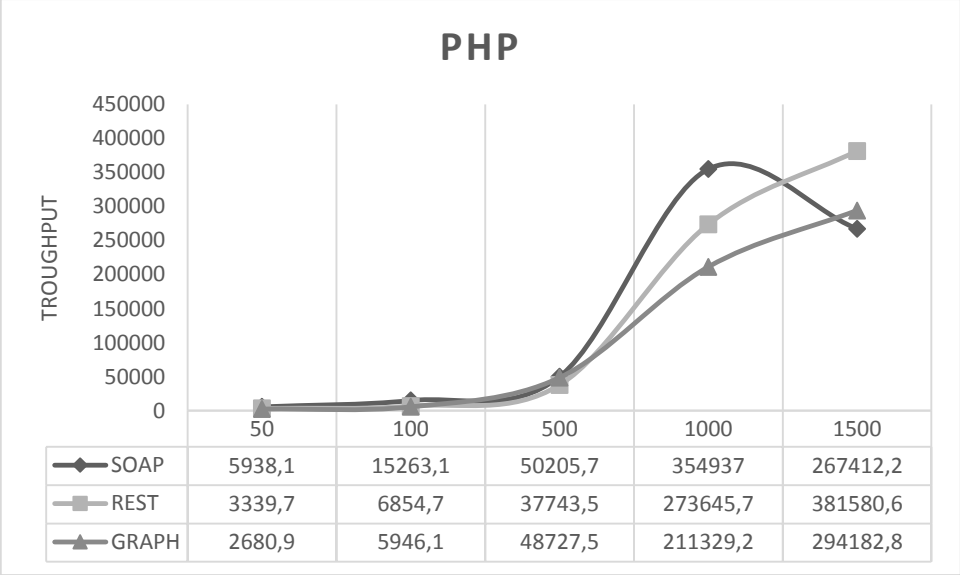


Figura 19. *Tiempo de Respuesta en el consumo de los servicios web 500-1000-1500 VU's con Java.*

La figura 19 nos demuestra que a partir de los 1000 usuarios se refleja que los tiempos de respuesta comienzan a verse notorios, guiándonos por los datos de la tabla 12 en la que SOAP comienza a generar un 45% de rechazo de conexión, aun en el lenguaje de PHP, SOAP no puede atender por completitud las peticiones dadas, mientras que sus contrapartes no muestran errores y generan menores tiempos.



	50	100	500	1000	1500
SOAP	5938,1	15263,1	50205,7	354937	267412,2
REST	3339,7	6854,7	37743,5	273645,7	381580,6
GRAPH	2680,9	5946,1	48727,5	211329,2	294182,8

Figura 20. *Rendimiento en el consumo de los servicios web con PHP.*

En la figura 20 se analiza el rendimiento que tiene cada servicio a través del lenguaje de PHP, en ella se refleja que el protocolo SOAP comienza a disminuir su rendimiento cuando llega a la cantidad de 1500 usuarios concurrentes, esto se debe a que existe un rechazo de conexión del 45%, mientras que los otros servicios se encuentran estables sin errores, notando que el lenguaje de esquemas maneja un mejor rendimiento por delante de REST.

CAPÍTULO IV: DISCUSIÓN

Entre los resultados obtenidos en esta investigación, se revela que el tiempo de respuesta del lenguaje de esquemas GraphQL es mucho mejor que el tiempo de respuesta de la arquitectura REST y el protocolo SOAP en los tres lenguajes en los que fueron examinados, mientras que los servicios web basados en REST muestran mejores tiempos que SOAP.

Si bien la investigación de Bora & Bezboruah [4] titulada "A Comparative Investigation on Implementation of RESTful versus SOAP based Web Services" realiza pruebas de rendimiento a dos servicios web en los cuales los resultados arrojados coinciden en gran parte con la presente investigación, en ella se puede apreciar que los tiempos de respuesta y el Throughput del servicio web REST son menores a comparación del servicio web basado en SOAP, al igual que en la investigación de Vázquez, Cruz & García [6] en la cual realizan un estudio de análisis de los tiempos de respuesta de la arquitectura REST y el lenguaje de consultas GraphQL, muestran los resultados de rendimiento en la que refleja que GraphQL maneja mejores tiempos de respuesta que REST.

En las tablas 10, 11, 12 se observa que los valores de los atributos de rendimiento como el Throughput y el tiempo de respuesta en el lenguaje de esquemas, son mucho menores que los atributos de rendimiento de la arquitectura REST y el protocolo SOAP. Se puede detallar que el servicio web basado en SOAP es estable entre los 850VU y 930VU y no presenta ningún error, pero ofrece un bajo rendimiento hasta los 1500VU, con una media 43.59% de rechazo de conexión. Mientras que en el caso de GraphQL y REST los servicios se encuentran libre de errores y se mantienen estables hasta los 1500 usuarios virtuales. Por lo tanto, se enfatiza que GraphQL genera una sobre carga menor haciéndolo eficiente por delante de REST y SOAP.

El análisis estadístico de las métricas de rendimiento registradas nos permite observar que el tiempo de respuesta repentinamente aumenta o disminuye para varios niveles de estrés, esto puede deberse a la liberación parcial de restos de datos del lado del servidor, lo que aumenta la tensión del servidor.

Los servicios web basados en el protocolo SOAP consumen el archivo WSDL del proveedor que procesa los mensajes XML para sus comunicaciones, y REST utiliza el URI de HTTP

del recurso disponible en Internet que funciona como la metodología normal de solicitud y respuesta HTTP, mientras que GraphQL es mucho más flexible ya que se pueden describir exactamente cómo debe ser la respuesta, combina entidades conectadas dentro de una consulta de datos GraphQL. No se necesitan viajes al servidor adicionales como lo hacen sus contrapartes SOAP y REST.

CAPÍTULO V: CONCLUSIONES

En esta investigación, se presenta un estudio para analizar el impacto que tiene el estándar empleado en el servicio web sobre el rendimiento de ejecución en aplicaciones, para definir y compararlos se llegó a implementar un aplicativo con tres lenguajes de programación; Java, C# y PHP, estos lenguajes consumen los servicios web cada uno (SOAP, REST y GraphQL), en las cuales se realizó las 4 funciones básicas de un software CRUD (Create, Read, Update, Delete). En este análisis de resultados se ha tratado de responder la pregunta de investigación descrita en la sección del Planteamiento del Problema, donde se seleccionaron las métricas y herramientas como Jmeter para la realización del testing, concluyendo que, SI impacta el estándar de servicio web empleado sobre el rendimiento de ejecución en aplicaciones, solamente cuando la cantidad oscile entre los 800 y 900VU el usuario notará perspicazmente el rendimiento del sistema web.

El diseño de pruebas se enfoca en las métricas definidas por la pregunta de investigación, las cuales son el Throughput y Response Time, con la ayuda de la herramienta Apache JMeter se logra obtener los valores pertinentes de cada atributo.

Las tablas 10, 11, 12 proporcionan resultados comparativos de tiempo de respuesta y rendimiento entre SOAP, REST y GraphQL, resultando que la arquitectura de lenguaje de esquemas tiene un tiempo de respuesta más rápido que los servicios web basados en SOAP y que la arquitectura basada en REST, debe tomarse en cuenta que para los dispositivos móviles contienen recursos de hardware más bajos, la implementación de la arquitectura REST o GraphQL será preferible a la del protocolo SOAP ya que reduce el rendimiento general del servicio web.

Los resultados experimentales anteriores darán a los investigadores, así como a los profesionales de software, una idea del rendimiento de servicio web y otras métricas que influyen en el rendimiento general de los servicios.

CAPÍTULO VI: RECOMENDACIONES

Para la organización y selección de las distintas métricas a evaluar se recomienda una amplia revisión bibliográfica o utilizar técnicas de estudio de mapeo que permita elegir para la investigación los más representativos.

En la elaboración de un análisis comparativo íntegro es recomendable establecer los criterios comunes revisando la información de las fuentes oficiales.

Se recomienda que la metodología de benchmark se realice en escenarios similares para tener una mayor precisión en los resultados de comparación de los servicios web.

Se recomienda minimizar el tráfico de la red agrupando funcionalidades en los servicios de manera que el número de comunicaciones necesarias disminuya. Para ello se debe construir servicios generales que realicen muchas tareas y devuelvan mucha información, para que así se reduzca la sobrecarga de la red y existan mejores tiempos de respuesta de los servicios.

Para los mensajes de los servicios SOAP se recomienda controlar el tamaño lo más mínimo posible, ya que, al encontrarse con mensajes complejos, los procesos de deserialización y el inverso consumen mucho tiempo y es por eso que se deben diseñar los mensajes minimizando la complejidad.

Dificultades:

Al implementar el servicio GraphQL se presentaron varios inconvenientes debido a que es una tecnología que tiene poco tiempo publicada en la web y no cuenta con mucha documentación para el desarrollo en varios lenguajes de programación.

BIBLIOGRAFÍA

[1] J. Bixby, "Latency 101: What is latency and why is it such a big deal?," 2012. [Online]. Available: http://www.webperformancetoday.com/2012/04/02/latency-101-what-is-latency-and-why-is-it-such-a-big-deal/. [Accessed: 10-Jul-2018].

[2] D. Echeverría Perez and A. A. Paumier, "Testing como Práctica para Evaluar la Eficiencia en Aplicaciones Web," *Rev. Latinoam. Ing. Softw.*, vol. 2, no. 5, p. 307, Oct. 2014.

[3] L. N. Hidalgo Macas and M. E. Jimenez Acaro, "Estudio Comparativo De Los Servicios Web Restful Jersey Y Soap Jax-Ws Para El Desarrollo De Una Aplicación Android Con Wikitude Aplicada a La Gestión De Información Geolocalizada Del Turismo De La Provincia De Chimborazo.," Escuela Superior Politécnica de Chimborazo, 2016.

[4] A. Bora and T. Bezboruah, "A Comparative Investigation on Implementation of RESTful versus SOAP based Web Services," *Int. J. Database Theory Appl.*, vol. 8, no. 3, pp. 297–312, 2015.

[5] K. P. Pavan, A. Sanjay, and P. Zornitza, "Comparing Performance of Web Service Interaction Styles : SOAP vs . REST," *Proc. Conf. Inf. Syst. Appl. Res.*, pp. 1–24, 2012.

[6] A. Vázquez-Ingelmo, J. Cruz-Benito, and F. J. García-Peñalvo, "Improving the OEEU's data-driven technological ecosystem's interoperability with GraphQL," in *Proceedings of the 5th International Conference on Technological Ecosystems for Enhancing Multiculturality - TEEM 2017*, 2017, pp. 1–8.

[7] D. Čechák, "Using GraphQL for Content Delivery in Kentico Cloud," Masaryk University , 2017.

[8] M. C. Gallegos Valera, "Introducción a los Servicios Web," in *Los servicios web XML*, Ibarra, 2013, pp. 72–97.

[9] P. Carmona Barbero, "Plataformas De Integración Servicios Web REST Y SOAP," 2012. [Online]. Available: https://projetsii.informatica.us.es/documents/331. [Accessed: 09-Jul-2018].

[10] IBM Corporation, "¿Qué es SOAP?," *IBM Knowledge Center*, 2015. [Online]. Available: https://www.ibm.com/support/knowledgecenter/es/SSKM8N_8.0.0/com.ibm.etools.mft.doc/ac55770_.htm. [Accessed: 10-Jul-2018].

[11] A. Sierra Collado and A. Cubo Velázquez, “Capítulo 4: SOAP y WSDL,” in *Representational State Transfer (REST). Un estilo de arquitectura para Servicios Web. Panorámica y estado del arte.*, Universidad de Sevilla, p. 52.

[12] U. Urain Cerio, “Analisis de calidad de servicio de Middlewares asociados a la Norma IEC61850F,” Universidad de Cantabria, 2012.

[13] U. Díaz Arberas, “Desarrollo y pruebas en entorno real de un smart reader Wi-Fi,” Universidad del País Vasco, 2016.

[14] L. De Seta, “Introducción a los servicios web RESTful,” 2008. [Online]. Available: https://dosideas.com/noticias/java/314-introduccion-a-los-servicios-web-restful. [Accessed: 09-Jul-2018].

[15] Facebook Inc, “Introduction to GraphQL.” [Online]. Available: https://graphql.org/learn/. [Accessed: 10-Sep-2018].

[16] Facebook Inc, “GraphQL Specification Versions.” [Online]. Available: https://facebook.github.io/graphql/. [Accessed: 10-Sep-2018].

[17] M. Fowler and Martin, *Patterns of enterprise application architecture*. Addison-Wesley, 2003.

[18] K. Wagh and R. Thool, “A Comparative Study of SOAP Vs REST Web Services Provisioning Techniques for Mobile Host,” *J. Inf. Eng. Appl.*, vol. 2, no. 5, pp. 12–16, 2012.

[19] F. Nogatz and D. Seipel, “Implementing GraphQL as a Query Language for Deductive Databases in SWI–Prolog Using DCGs, Quasi Quotations, and Dicts,” *EPTCS*, vol. 234, pp. 42–56, 2017.

[20] F. A. Amo, L. Normand Martinez, and F. J. Segovia Pérez, *Introduccion a la Ingeneria del software*. Delta, 2005.

[21] J. Tuya, I. Ramos Román, and J. Dolado Cosín, *Técnicas cuantitativas para la gestión en la ingeniería del software*. Netbiblo, 2007.

[22] S. S. Patil and S. . Joshi, “Identification of Performance Improving Factors for Web Application by Performance Testing.,” vol. 2, no. 8, 2012.

[23] A. Muñoz, “¿Qué es un benchmark y para qué sirve?,” *Tecnología - ComputerHoy.com*, 2016. [Online]. Available: https://computerhoy.com/noticias/moviles/que-es-benchmark-que-sirve-

40273. [Accessed: 12-Jul-2018].

[24] C. M. Zapata and C. de J. Cardona Velásquez, “Comparación de las características de algunas herramientas de software para pruebas de carga,” *Revista Avances en Sistemas e Informática, Vol.8 No.2*, Medellín, Jul-2011.

[25] D. A. Menasce and D. A., “Load testing of Web sites,” *IEEE Internet Comput.*, vol. 6, no. 4, pp. 70–74, Jul. 2002.

[26] P. J. Denning and J. P. Buzen, “The Operational Analysis of Queueing Network Models*,” *ACM Comput. Surv. Vol. 10 N° 3*, pp. 225–261, 1978.

[27] D. A. Menascé and V. A. F. Almeida, *Capacity planning for Web services : metrics, models, and methods*. Prentice Hall, 2002.

[28] B. M. Subraya, *Integrated approach to web performance testing : a practitioner’s guide*. IRM Press, 2006.

[29] I. D. Cooper, “What is a "Mapping Study?".,” *J. Med. Libr. Assoc.*, vol. 104, no. 1, pp. 76–8, Jan. 2016.

[30] M. Petticrew and H. Roberts, *Systematic reviews in the social sciences : A Practical Guide*. Blackwell Pub, 2006.

[31] J. I. Ignacio Piovani Nora Krawczyk II, “Los Estudios Comparativos: algunas notas históricas, epistemológicas y metodológicas,” no. 3, pp. 821–840, 2017.

[32] D. P. Arias Henao, “Investigación comparativa transcontextual en relaciones internacionales,” *Rev. Relac. Int. Estrateg. y Segur.*, vol. 9, no. 2, pp. 77–99, 2006.

[33] I. D. Toro Jaramillo and R. D. Parra Ramirez, *Fundamentos epistemológicos de la investigación y la metodología*, 1st ed. Bogota, 2011.

[34] R. Hernández Sampieri, C. Fernández Collado, and M. del Pilar Baptista Lucio, “Metodología de la investigación, 5ta Ed.”

[35] H. Dieterich, *Nueva guía para la investigación científica*. Mexico, 2007.

[36] Apache JMeter™, “Apache JMeter.” [Online]. Available: https://jmeter.apache.org/. [Accessed: 12-Jul-2018].

[37] L. E. Burgos Suero, “Análisis y evaluación de las arquitecturas REST y SOAP para el

desarrollo de servicios web aplicados al ERP AdrisERP y su versión móvil en Android.," Universidad Señor de Sipán, 2017.

[38] S. Kumari and S. K. Rath, "Performance comparison of SOAP and REST based Web Services for Enterprise Application Integration," in *2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 2015, pp. 1656–1660.

[39] C. Hernández Rodríguez and M. C. Flores, "La importancia del benchmarking como herramienta para incrementar la calidad en el servicio en las organizaciones," 2017.

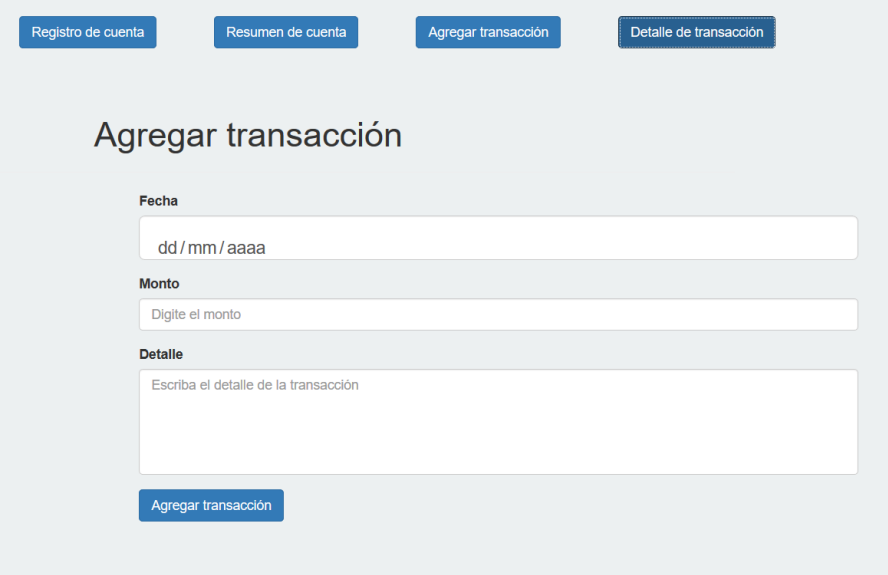
[40] Kunhua Zhu, Junhui Fu, and Yancui Li, "Research the performance testing and performance improvement strategy in web application," in *2010 2nd International Conference on Education Technology and Computer*, 2010, pp. V2-328-V2-332.

ANEXOS

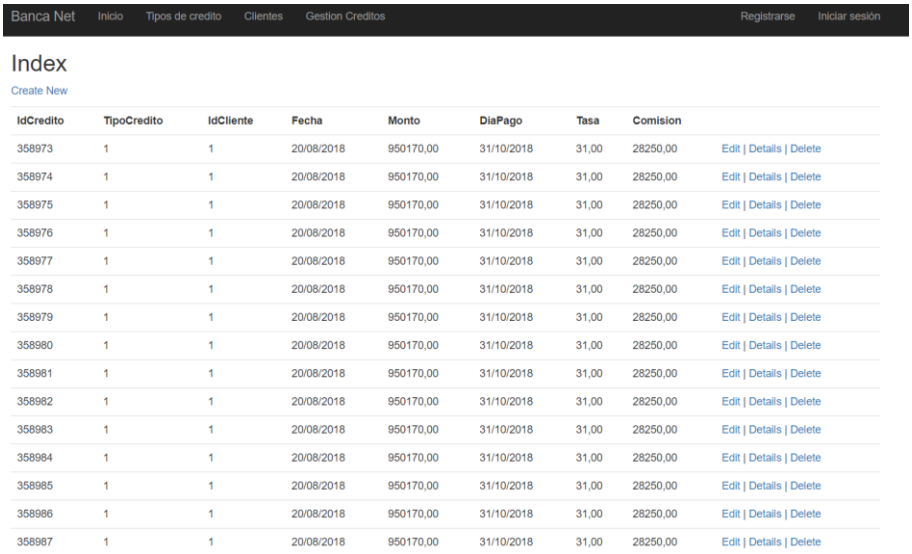
Anexo 1. *Ficha de Observación*

		RESTful WS		SOAP WS		GraphQL	
No. Usuarios Virtuales accediendo al WS.	**Parámetro**	**Promedio**	**Conexión rechazada %**	**Promedio**	**Conexión rechazada %**	**Promedio**	**Conexión rechazada %**
50	Tiempo de respuesta	ms	%%	ms	%%	ms	%%
	Throughput	ms	%%	ms	%%	ms	%%
100	Tiempo de respuesta	ms	%%	ms	%%	ms	%%
	Throughput	ms	%%	ms	%%	ms	%%
500	Tiempo de respuesta	ms	%%	ms	%%	ms	%%
	Throughput	ms	%%	ms	%%	ms	%%
1000	Tiempo de respuesta	ms	%%	ms	%%	ms	%%
	Throughput	ms	%%	ms	%%	ms	%%
1500	Tiempo de respuesta	ms	%%	ms	%%	ms	%%
	Throughput	ms	%%	ms	%%	ms	%%

Anexo 2. *Interfaz Gráfica del Software.*

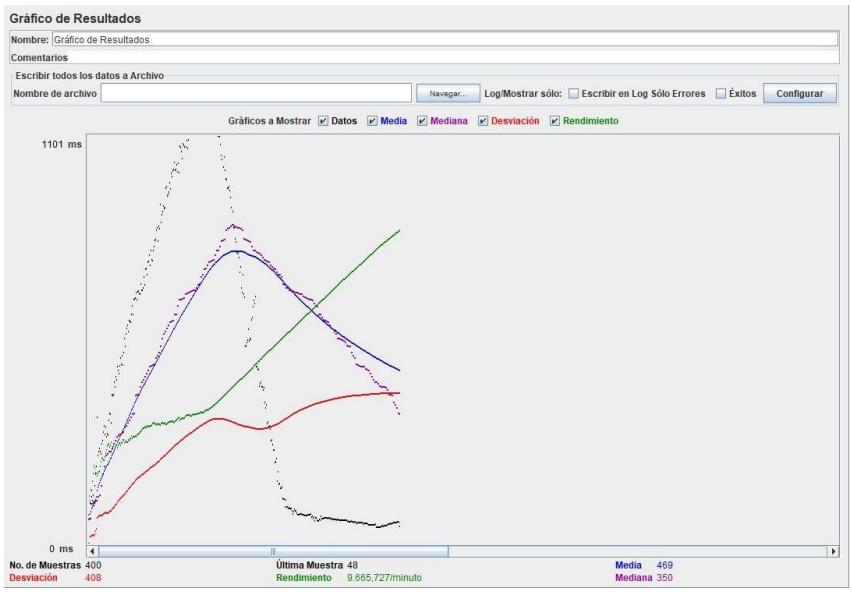


Anexo 3. *Interfaz gráfica del software general.*

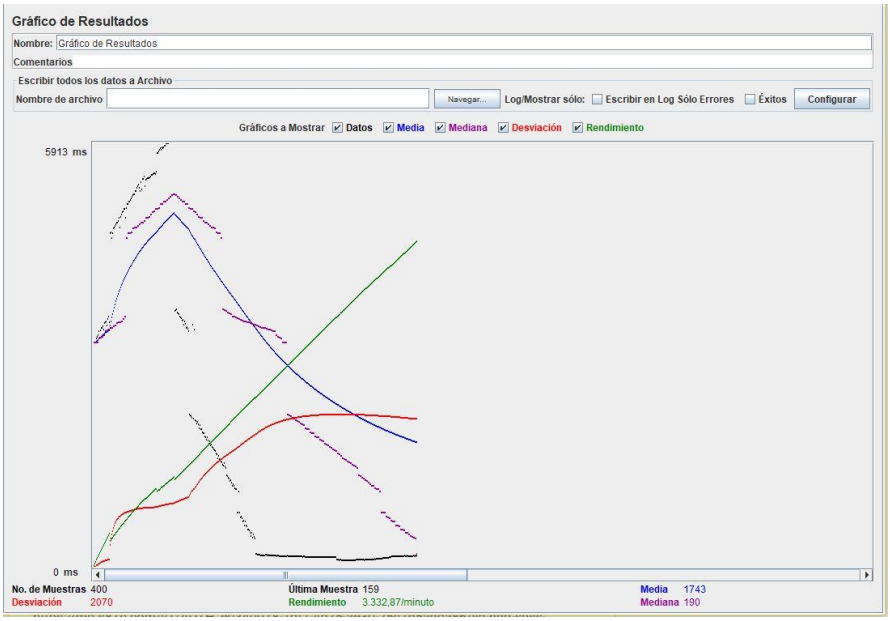


IdCredito	TipoCredito	IdCliente	Fecha	Monto	DiaPago	Tasa	Comision	
358973	1	1	20/08/2018	950170,00	31/10/2018	31,00	28250,00	Edit | Details | Delete
358974	1	1	20/08/2018	950170,00	31/10/2018	31,00	28250,00	Edit | Details | Delete
358975	1	1	20/08/2018	950170,00	31/10/2018	31,00	28250,00	Edit | Details | Delete
358976	1	1	20/08/2018	950170,00	31/10/2018	31,00	28250,00	Edit | Details | Delete
358977	1	1	20/08/2018	950170,00	31/10/2018	31,00	28250,00	Edit | Details | Delete
358978	1	1	20/08/2018	950170,00	31/10/2018	31,00	28250,00	Edit | Details | Delete
358979	1	1	20/08/2018	950170,00	31/10/2018	31,00	28250,00	Edit | Details | Delete
358980	1	1	20/08/2018	950170,00	31/10/2018	31,00	28250,00	Edit | Details | Delete
358981	1	1	20/08/2018	950170,00	31/10/2018	31,00	28250,00	Edit | Details | Delete
358982	1	1	20/08/2018	950170,00	31/10/2018	31,00	28250,00	Edit | Details | Delete
358983	1	1	20/08/2018	950170,00	31/10/2018	31,00	28250,00	Edit | Details | Delete
358984	1	1	20/08/2018	950170,00	31/10/2018	31,00	28250,00	Edit | Details | Delete
358985	1	1	20/08/2018	950170,00	31/10/2018	31,00	28250,00	Edit | Details | Delete
358986	1	1	20/08/2018	950170,00	31/10/2018	31,00	28250,00	Edit | Details | Delete
358987	1	1	20/08/2018	950170,00	31/10/2018	31,00	28250,00	Edit | Details | Delete

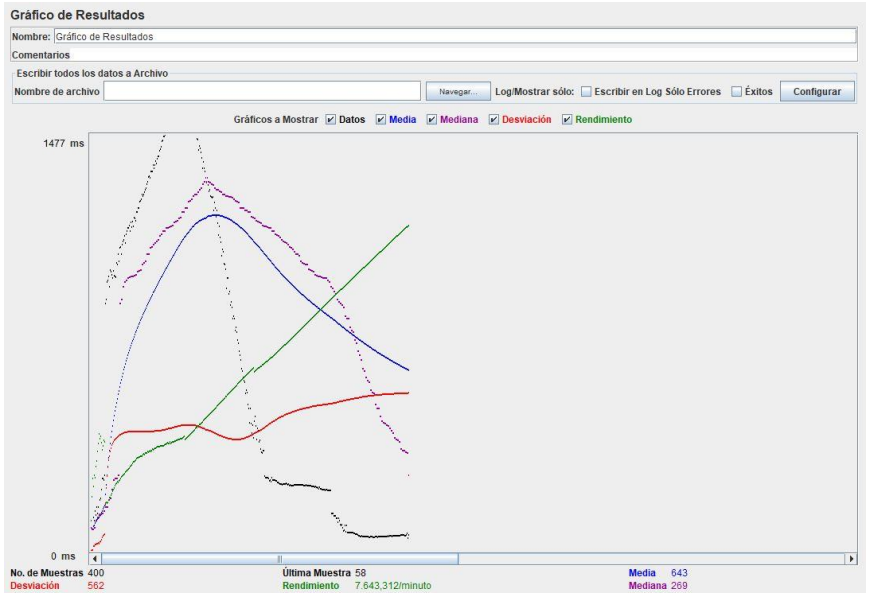
Anexo 4. *Gráfica JMeter 100 usuarios SOAP.*



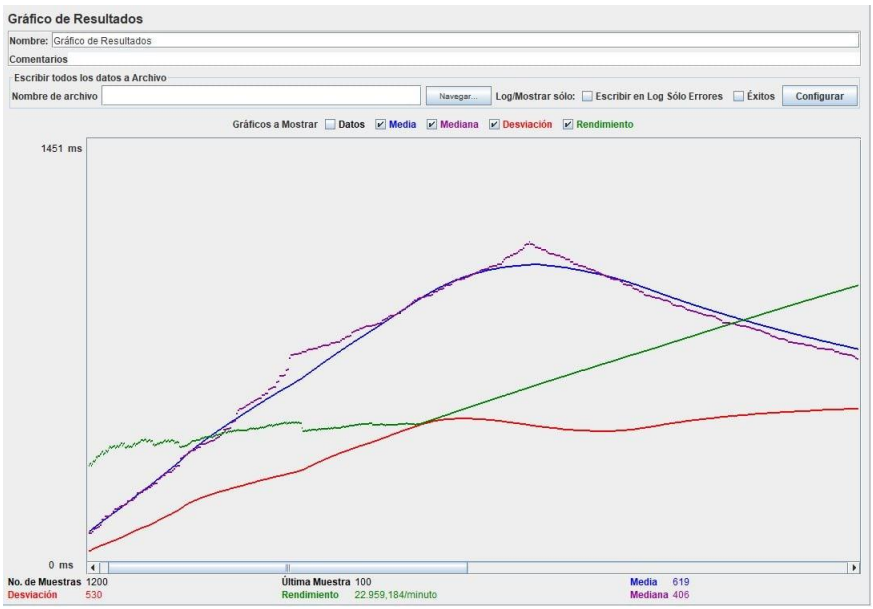
Anexo 5. *Gráfica JMeter 100 usuarios REST.*



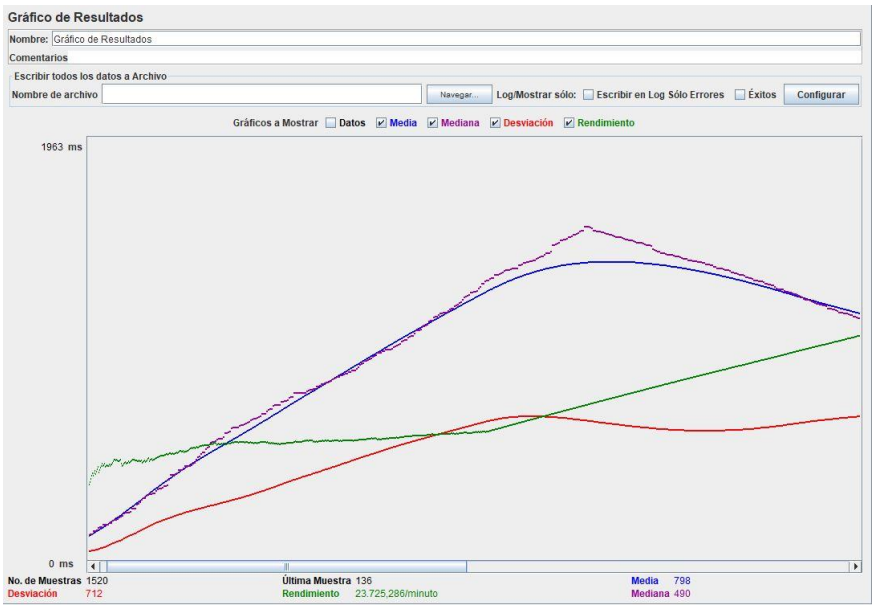
Anexo 6. *Gráfica JMeter 100 usuarios GraphQL.*



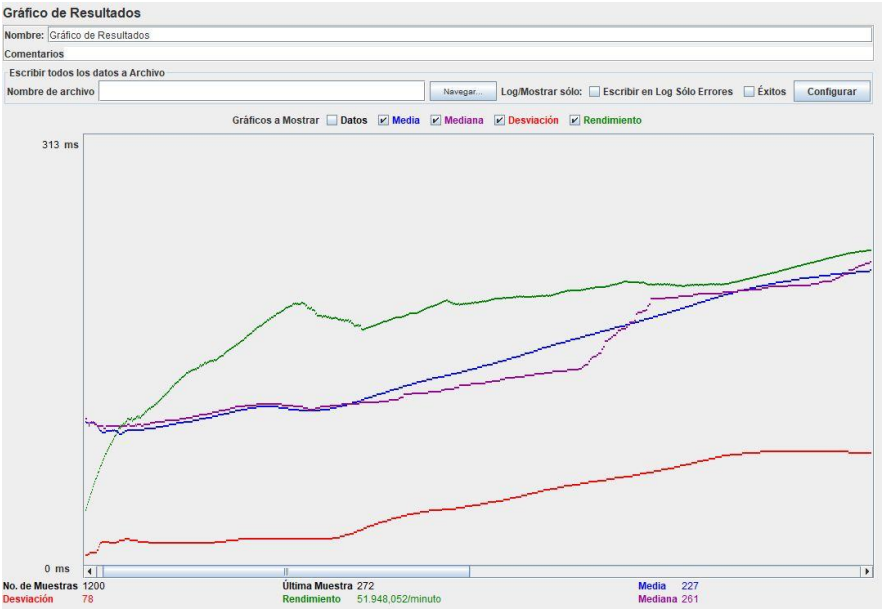
Anexo 7. *Gráfica JMeter 1000 usuarios SOAP.*



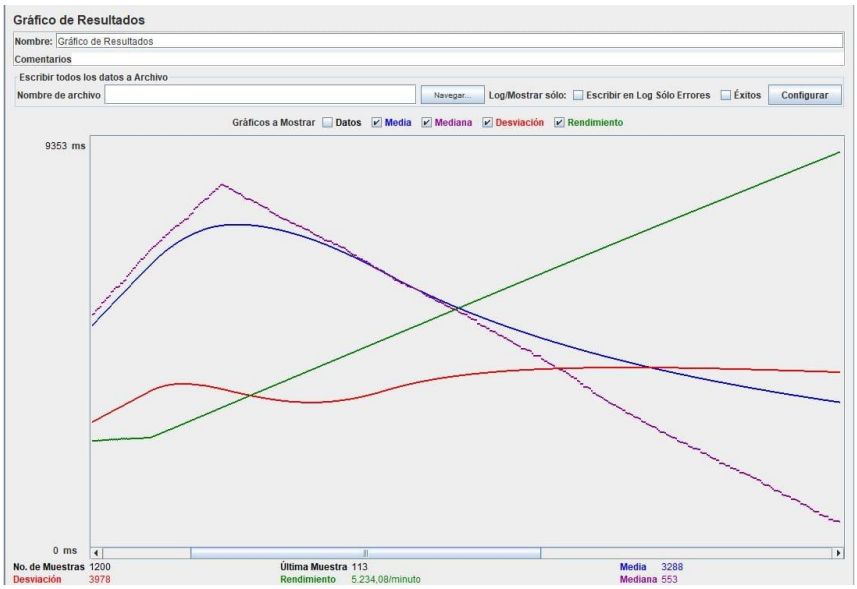
Anexo 8. *Gráfica JMeter 1000 usuarios REST.*



Anexo 9. *Gráfica JMeter 1500 usuarios SOAP.*



***Anexo 10**. Gráfica JMeter 1500 usuarios REST.*



***Anexo 11.** Gráfica JMeter 1500 usuarios GraphQL.*