UNIVERSIDAD NACIONAL DEL ALTIPLANO

ESCUELA DE POSGRADO

DOCTORADO EN CIENCIAS DE LA COMPUTACIÓN

TESIS

FRAMEWORK PARA EL DESARROLLO ESCALABLE DE SISTEMAS DE INFORMACIÓN EN LA NUBE ORIENTADO A MYPES

PRESENTADA POR:

EDWIN FREDY CALDERON VILCA

PARA OPTAR EL GRADO ACADÉMICO DE:

DOCTORIS SCIENTIAE EN CIENCIAS DE LA COMPUTACIÓN

PUNO, PERÚ

2020

DEDICATORIA

A mis padres por ser guía y ejemplo de persona, y a mis hermanos por el constante apoyo de cada día en todos los aspectos de la vida, principalmente en lo profesional.

A mi Esposa y mis hijos, por su gran apoyo y motivación que día a día me dan para alcanzar mis metas.

AGRADECIMIENTOS

- A la Universidad Nacional del Altiplano por haberme formato y permitirme realizar mi investigación de doctorado.
- A los docentes del doctorado de Ciencias de la Computación tanto nacionales e internacionales, por impartir sus conocimientos, y así formarme para poder realizar mi trabajo de investigación.
- A los miembros de Jurado y asesor de la presente investigación; por la paciencia y dedicación al guiarme mediante todas las sugerencias y correcciones.

ÍNDICE GENERAL

Puno, 31 de enero de 2020

ÁREA: Ciencias de la ingeniería.
TEMA: Desarrollo escalable de sistemas de información.
LÍNEA: Sistemas, Computación e Informática.

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INDICE DE ANEXOS

RESUMEN

Las empresas están optando cada vez más por utilizar los servicios de las infraestructuras computacionales en la nube, plataformas y softwares, para la implantación de sus sistemas de información, los cuales pueden ser iniciados y liberados de forma automática sin intervención del proveedor del servicio; estos sistemas de información conforman un factor de éxito en el crecimiento de las organizaciones, permitiendo ser competitivos e innovadores. Actualmente las pequeñas y medianas empresas peruanas (MYPES), implementan sus sistemas utilizando una arquitectura clásica de cliente servidor, sin aprovechar los recursos y servicios ofrecidos por la computación en la nube. El objetivo de este trabajo de investigación es desarrollar un framework para el desarrollo escalable de sistemas de información sobre infraestructura en la nube, para ello se utilizó modelos de programación, manejo de infraestructura en la nube, modelos de calidad de desarrollo de software y uso de herramientas de pruebas de carga, para evaluar el rendimiento del framework. Tenido como resultado de esta última, una mejora de 34% en el caso de prueba de un sistema de información implementado bajo el framework propuesto, en comparación a un sistema implementado en una arquitectura tradicional cliente servidor.

Palabras clave: Computación en la nube, escalabilidad, framework, mypes, rendimiento.

ABSTRACT

Companies are increasingly choosing to use the services of cloud computing infrastructures, platforms and software for the implementation of their information systems, which can be started and released automatically without the intervention of the service provider; these information systems are a success factor in the growth of organizations, allowing them to be competitive and innovative. Currently, small and medium-sized Peruvian companies (MYPES) implement their systems using a classic client-server architecture, without taking advantage of the resources and services offered by cloud computing. The objective of this research work is to develop a framework for the scalable development of information systems on cloud infrastructure, for which programming models, cloud infrastructure management, software development quality models and use of load testing tools, to evaluate the performance of the framework. As a result of the latter, an improvement of 34% in the test case of an information system implemented under the proposed framework, compared to a system implemented in a traditional client-server architecture.

Keywords: Cloud computing, framework, mypes, performance, scalability.

INTRODUCCIÓN

El uso de sistemas de información son un factor de éxito en el crecimiento de las organizaciones, permitiendo ser competitivos e innovadores. Según la Sociedad de Comercio Exterior del Perú ComexPerú (2017) en el Perú las MyPes representan el 96.5% de las empresas. La adopción de las Tecnologías de la Información y Comunicación (TIC) se da con más frecuencia en las grandes empresas que tienen mayor capacidad de inversión y recursos, las pequeñas y medianas empresas le siguen gradualmente, según INEI (2018) de 82,249 empresas peruanas el 94.2% de las empresas hicieron uso de computadoras, el 92,6% del servicio de internet y el 18,9% hicieron uso de intranet, este último porcentaje indica que tienen adoptado el uso de sistemas de uso interno, y solo el 12.4% de empresas peruanas desarrolla su propio software para dar soluciones a sus necesidades particulares.

Actualmente las TIC's, se han convertido en uno de los instrumentos clave para medir el desarrollo de las empresas del país (Álvarez *et al.*, 2019), además una empresa generalmente tiene como objetivo expandirse, o crecer en números de clientes, ventas, etc. Lo que se traduce en crecimiento de peticiones y consumo de recursos del servidor, los cuales son limitados de acuerdo al modelo de programación de un solo servidor para varios clientes.

Las pequeñas empresas utilizan generalmente un modelo clásico de cliente servidor, donde todo el poder de computo, se ofrecía sobre un súper computador llamada mainframe o comúnmente llamado server, y los clientes usaban los computadores personales que se conectaban al servidor y actuaban como terminales, permitiendo ejecutar aplicaciones del servidor localmente (Hayes, 2008).

Por otro el modelo de computación en la nube, permite utilizar bajo demanda recursos ofrecidos como procesamiento, almacenamiento, aplicaciones y servicios. Siendo las MyPes las que más se benefician de esta tecnología, al no ser necesario de presupuestos altos para la adquisición y administración de esta tecnología, pero si necesitan de herramientas que les ayuden implementar sus soluciones de sistemas de información (Henriquez *et al*., 2015).

Existiendo variedad de alternativas de proveedores de servicios de computación en la nube, empresas de tecnologías de la información como Google, Amazon, Microsoft, IBM.

Cada empresa tiene formas particulares de proveer los recursos computaciones, donde resalta una jerarquía de los servicios computacionales en la nube, que contiene los siguientes servicios: Infraestructura como servicio, Plataforma como servicio y Software como servicio (Zhang & Zhou, 2009).

La computación en la nube utiliza un modelo de despliegue, que especifica el modo de acceso, conexión y quien es el propietario del servicio, definiendo 4 tipos de proveedores de servicio en la nube: (Nube privada, Nube publica, Nueve comunitaria y Nube híbrida). Teniendo como principales ventajas y desventajas del uso de computación en la nube que ayudarán a tomar decisiones de uso para los usuarios en general (Fujita *et al*., 2013).

El desarrollo de sistemas de información sobre un entorno web, provee a grandes masas de usuarios en un canal de comunicación como es el internet (Kendall, 2011), con probabilidades de alto crecimiento de usuarios y peticiones de las aplicaciones, la computación la nube permite la elasticidad de recursos bajo demanda, basados en una arquitectura multi-tenant, donde los clientes o usuarios comparten la misma aplicación (Bustamante Granda, 2017), sin embargo la administración de estos servicios y recursos necesita de un conocimiento de administración de servidores, el cual dificulta la implantación eficiente de los sistemas de información.

Según La & Kim (2009) Para poder desarrollar una aplicación que tiene el objetivo de servir a una gran cantidad de clientes con una sola instancia de aplicación, sobre una arquitectura en la nube, se tiene que considerar algunos requisitos técnicos, como son: Ejecución de múltiples clientes de un solo servicio, balanceo de carga, personalización y monitoreo.

En el desarrollo de aplicaciones en la actualidad generalmente se usan metodologías agiles, con el fin de dar importancia a la velocidad de desarrollo y a la funcionalidad, Scrum es un modelo de desarrollo de software ágil, caracterizado por el desarrollo incremental y manejo de equipos auto organizados con calidad en el proceso (Menzinsky *et al*., 2016). Para evaluar el rendimiento de dichos sistemas se usó la herramienta JMeter (Software Apache, n.d.), probando distintas tipos de cargas al servidor.

En cuestión de investigaciones existe pocos estudios referentes a esta necesidad de las MyPes, dichas investigaciones son más orientadas, a la arquitecturas del software, administración de plataforma (Tang *et al*., 2010), descripción de los factores de

evaluación de escalabilidad de sistemas sobre arquitecturas en la nube (Pattabhiraman *et al*., 2011), lo cual motivó el presente trabajo de investigación, el cual tiene como objetivo el desarrollo de un framework que proporcione un marco de trabajo para el desarrollo sistemas de información, que permita escalabilidad sobre una arquitectura de computación en la nube.

El trabajo de investigación se ha estructurado en cuatro capítulos de la siguiente manera:

En el primer capítulo, se desarrolla la base teórica que guía la investigación, el marco teórico y los antecedentes de la investigación.

En el segundo capítulo, se refiera a la problemática de la investigación, formulación del problema, objetivos y justificación de la investigación.

En el tercer capítulo, se describe los métodos e instrumentos, métodos teóricos para desarrollo de la investigación, y métodos para evaluar la investigación.

En el cuarto capítulo, se exponen los resultados, y la discusión de la investigación estructuradas por etapas, análisis, diseño y desarrollo del framework, implementación y evaluación el rendimiento en una infraestructura en la nube.

Finalmente se describen las conclusiones y recomendaciones de las investigación.

CAPÍTULO I
REVISION DE LITERATURA

1.1. Marco Teórico

1.1.1. Computación en la nube

El uso de la computación ha cambiado, las organizaciones están migrando sus centros de datos a lugares geográficamente distantes, utilizando internet como principal medio de comunicación. Los proveedores de los servicios han trasladados sus operaciones a infraestructuras en la nube, también los softwares son ofrecidos como un servicio independientemente de la plataforma que se utiliza el usuario, que a través de un navegador web puede administrador los recursos computacionales (Hayes, 2008).

Según Nieto (2013), la computación en la nube, es un modelo de computación que permite utilizar bajo demanda recursos ofrecidos como procesamiento, almacenamiento, aplicaciones y servicios, los cuales pueden ser iniciados y liberados de forma automática sin esfuerzo del proveedor ni del usuario.

1.1.1.1. Características del modelo de computación en la nube.

La computación en la nube son un conjunto características principales, que los distinguen de otros modelos computacionales que lo preceden, como es el modelo cliente-servidor (Nenartovič, 2017).

Recursos a demanda: El usuario puede iniciar servicios de uso de determinados recursos computacionales, de forma automática y sin necesidad de la interacción humana, siempre que estos estén configurados de esta forma.

Interconexión: Los diferentes servicios deben estar disponibles, a través de distintos medios de comunicación, que utilicen la comunicación estándar, lo cual permitirá el acceso desde distintos dispositivos tecnológicos.

Virtualización: Los recursos computacionales deben ser capaces de servir a múltiples usuarios, aumentando o disminuyendo su capacidad (también llamado elasticidad), para ser asignados dinámicamente, de acuerdo a la necesidad.

Control (Medición): El uso de los recursos también debe de controlarse, a través de mecanismos que permiten el monitoreo trasparente para el usuario, mediante el uso de estadísticas, alertas, etc.

1.1.1.2. Jerarquía de la computación en la nube

Existen variedad de alternativas de proveedores de servicios de computación en la nube, empresas de tecnologías de la información como Google, Amazon, Microsoft, IBM, etc. Cada empresa tiene formas particulares de proveer los recursos computaciones, donde resalta una jerarquía de los servicios computacionales en la nube, que contiene los siguientes servicios: Infraestructura como servicio, Plataforma como servicio y Software como servicio, la Figura 1 muestra la jerarquía de la computación en la nube (Tsai *et al.*, 2010).

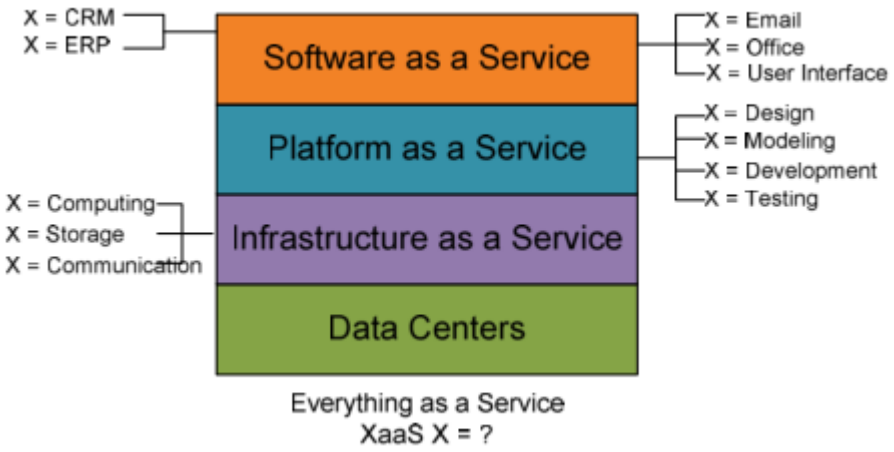


Figura 1. Jerarquía de la computación en la nube

Fuente: Tsai *et al.* (2010)

Centro de datos (Data Centers): Es el hardware que sirve para el funcionamiento del servicio de computación en la nube, generalmente esta instalados en zonas geográficas de energía eléctrica barata y baja probabilidad de desastres.

Infraestructura como servicio (IaaS - Infrastructure as a Service): Está ubicada encima de la capa de centro de datos, en esta capa los usuarios pueden escalar recursos bajo demanda, como máquinas virtuales, almacenamiento, recursos de red, procesamiento, sistemas operativos, CPU, memoria, almacenamiento.

Plataforma como servicio (PaaS - Platform as a Service): Provee una plataforma y herramientas orientado a los desarrolladores de software, como servicios de diseño, desarrollo, testing, despliegue y almacenamiento en la nube; generalmente requiere de aplicaciones externas instalables.

Software como servicio (SaaS - Software as a Service): Los usuarios finales pueden usar bajo demanda el software, generalmente es usado a través de un navegador web, con capacidad de crear varias sucursales y múltiples clientes a la vez.

1.1.1.3. Modelo de despliegue

El modelo de despliegue se refiere al modo de acceso, conexión y quién es el propietario del servicio (Ercolani, 2017).

Existen 4 tipos de proveedores de servicios en la nube:

Nube privada: La infraestructura de la nueve está instalada dentro de la organización o es físicamente propietaria, puede ser gestionada por un tercero, y tiene la ventaja de mayor control, seguridad, y uso sin restricciones; sin embargo, el costo de instalación puede ser inalcanzable para pequeñas empresas.

Nube pública: La infraestructura de la nube es propiedad de una organización que ofrece los servicios de computación en la nube, ofrecida al público en general, pero puede ser gestionada de manera segura por la organización que ha adquirido el servicio; tiene la ventaja de ser accesible y barata, en comparación con la nube privada.

Nube comunitaria: La infraestructura de la nube son propiedad de varias organizaciones y da soporte a una comunidad específica. Puede ser administrada y alojada por las mismas organizaciones o por un tercero; es parecida a la nube privada, con la diferencia que es propiedad de un grupo de organizaciones, que tienen similares objetivos.

Nube híbrida: La infraestructura de la nube es compartida entre diferentes tipos de nube (privadas, comunitarias o públicas), utilizando mecanismos que permiten una comunicación estándar, para facilitar la interconexión. Este tipo de nubes son complejas, ya que hay que considerar los mecanismos de comunicación, sincronización de información entre los diferentes tipos de nube; sin embargo, es bastante practico para las empresas, ya que pueden utilizar para información critica la nube privada, e información pública en la nube pública.

1.1.1.4. Ventajas y desventajas de la computación en la nube

El modelo de computación que provee la computación en la nube se aplica muy bien para aplicaciones basadas en Internet, pero como todas las tecnologías tienen ventajas y desventajas.

A continuación, se muestra los principales ventajas y desventajas del uso de computación en la nube que ayudarán a tomar decisiones de uso para los usuarios en general (Fujita *et al*., 2013).

Ventajas

- Eficiencia en el costo: La computación en la nube es probablemente la más eficiente en cuando al costo, tanto en su mantenimiento y escalabilidad. Una infraestructura privada tendría un costo elevado en hardware, licencias y personal encargado de la administración.

- Copias de seguridad. Las infraestructuras de la computación en la nube proporcionan mayor capacidad de almacenamiento, copias de seguridad, recuperación completa de pérdida de datos y reducción al mínimo de los tiempos de inactividad.

- Fácil manejo, Se comienza a trabajar más rápido y no es necesaria conocimientos técnicos profundos.

- Rápido despliegue: Las aplicaciones implementadas sobre infraestructura de computación en la nube suelen estar disponibles inmediatamente, en cuestión de minutos u horas.

- Actualizaciones automáticas, generalmente no afectan negativamente a los servicios adquiridos.

- Consumo eficiente. Gracias a las opciones de configuración permite el consumo eficiente de recursos, En los centros de datos tradicionales, los servidores consumen más energía de la requerida realmente.

Desventajas

- Dependencia: La centralización de los sistemas de información, y otros servicios origina una dependencia de los proveedores de servicios.

- Dependencia de internet: Las aplicaciones pueden ser accedidos utilizando el acceso a internet, por lo tanto, el acceso a los sistemas dependerá del acceso a Internet.

- Privacidad y seguridad: al no tener control físico de la base de datos del negocio, lo que podría generar una desconfianza en el proveedor del servicio de computación en la nube.

- Escalabilidad. Tener un esquema de crecimiento de usuarios, ayuda que los servidores no se sobrecarguen.

1.1.2. Modelo tradicionales y modelo en la nube

Los sistemas de información tradicionales, se instalaban en un servidor, en el cual se paga por la licencia de uso, licencia de usuarios, el servidor donde se ejecutaría, el administrador de red y servidores, y otros gastos relacionados (electricidad, internet con alto ancho de banda, actualizaciones, etc.). También se podían instalar sobre servidores alquilados, donde el problema es la capacidad que tenía el servidor de soportar la carga de peticiones de los usuarios.

1.1.2.1. Modelo cliente servidor

El modelo cliente-servidor introducido en los años 1980, y que hasta ahora sigue siendo usado pro las empresas, donde todo el poder de computo, se ofrecía sobre un súper computador llamada mainframe o comúnmente llamado server, y los clientes usaban los computadores personales que se conectaban al servidor y actuaban como terminales, permitiendo ejecutar aplicaciones del servidor localmente (Hayes, 2008).

1.1.2.2. Modelo de Software como servicio

Según Kim *et al*. (2017) , un modelo de prestación de servicio de instalación, mantenimiento, soporte del software, implementado sobre una infraestructura en la nube, este modelo permite a los usuarios, alta eficiencia, soporte inmediato, y escalado automático.

Las principales diferencias, entre los modelos de cliente servidor, y software como servicio, son:

Tabla 1
Diferencias entre arquitecturas

	Cliente Servidor	**SaaS**
Hardware	Propiedad del cliente o del proveedor	Propiedad del proveedor
Software	Propiedad del cliente o del proveedor	Propiedad del proveedor
Mantenimiento	Responsabilidad del cliente o proveedor	Responsabilidad del proveedor
Tiempo de implementación	Demora de acuerdo a los requisitos del sistema	Inmediato
Costo	Costoso pero de uso ilimitado	Costo es por tiempo de uso y su capacidad
Plataforma	Hardware	Virtualizado
Escalabilidad	Manual	Automática

Fuente: Adaptado de Kim *et al.* (2017)

La principal ventaja de este tipo de tecnología es que el ordenador del usuario, o la infraestructura de la organización cliente, no tiene que asumir una fuerte carga de trabajo al ejecutar aplicaciones. Así, gracias al capacidad de los servidores en

la nube, dicho esfuerzo lo hace la red de ordenadores que forman el sistema. De esta manera se agiliza el trabajo y se hace más accesible (Nenartovič, 2017).

Siendo el navegador lo único que necesita el usuario, en vez de usar una aplicación previamente instalada. Es la nube la que se encarga de todo el trabajo, simplificando el trabajo de parte del usuario y la organización. Obviamente, como toda tecnología tiene sus ventajas y desventajas:

Tabla 2
Ventajas y desventajas de SaaS

	Software como Producto	**Software como Servicio**
Distribución	Diseñado para que el cliente lo instale, administre y lo mantenga.	Diseñado para ser utilizado utilizando internet.
Desarrollo	Ciclo de desarrollo largo, múltiples parches.	Diseñado para correr miles de clientes sobre un único código fuente
Precios	Licencia + Mantenimiento	Suscripción
Precios de mantenimiento	Instalación, mantenimiento, personalización, actualizaciones.	Solo para características adicionales.
Plataforma	Múltiples versiones para diferentes plataformas	Una versión para diferentes plataformas.
Errores	Actualización lenta, y diferentes parches para diferentes plataformas.	Un parche sirve para todos los usuarios.
Instalación	Requiere instalación, y requisitos en software y hardware.	No requiere instalación, tiene 2 requisitos, internet y un navegador web moderno.
Actualizaciones	El rendimiento depende del hardware del cliente.	El rendimiento depende del proveedor de la infraestructura.

Fuente: Adaptado de (Nenartovič, 2017)

1.1.3. Arquitectura Multi tenat

Con el alto crecimiento de usuarios activos de las aplicaciones, y en consecuencia el incremente de peticiones en las transacciones de información, la computación la nube

permite la elasticidad de recursos bajo demanda, basados en una arquitectura multi-tenant, donde los clientes o usuarios comparten la misma aplicación (Bustamante Granda, 2017).

Multi-Tenant, se define como una arquitectura en la que una sola instancia de una aplicación sirve a múltiples usuarios. Siendo un cliente un Tenant. Un tenant comúnmente es un grupo de usuarios que comparten un acceso común con privilegios específicos a la instancia de la aplicación, separando los datos sensibles.

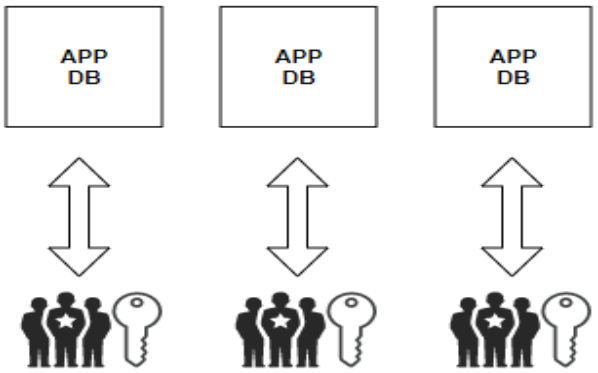


Figura 2. Arquitectura multi-tenant: Todo separado
Fuente: Adaptado de (Bustamante Granda, 2017)

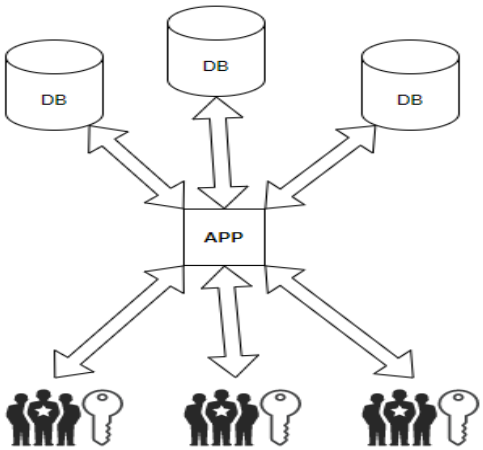


Figura 3. Arquitectura multi-tenant:
Aplicación y base de datos separados
Fuente: Adaptado de (Bustamante Granda, 2017)

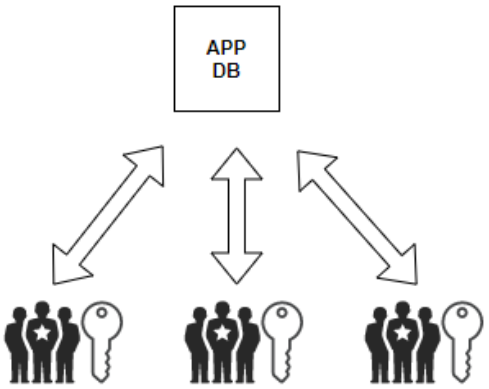


Figura 4. Arquitectura multi-tenant:
Aplicación y Base de datos juntas
Fuente: Adaptado de (Bustamante Granda, 2017)

1.1.3.1. Impacto de las arquitecturas multi-tenat

La siguiente tabla muestra el impacto de la arquitectura Multi-tenant con respecto a los criterios de costo de hardware, rendimiento de la aplicación, seguridad, personalización y capacidad de clientes:

Tabla 3
Atributos de calidad

Criterio / Arquitectura	BD y APP separados	BD separado y una sola APP	BD y APP juntas
Hardware	Cada aplicación requiere su propia BD y mantenimiento	Menos requerimiento de hardware	Mucho menos requerimiento de hardware
Rendimiento	El rendimiento no es afectado por el uso de otro cliente, si se usan servidores por aplicación.	Rendimiento afectado si comparten el mismo servidor	Rendimiento afectado si comparten el mismo servidor

Seguridad	Mayor seguridad al estar aislado la app y la BD	El DBMS debe asegurar la seguridad con privilegios	No tan seguro al compartir BD y app
Personalización	Es fácil personalizar al estar separado código y BD	Personalizar a nivel de BD	Difícil de personalizar al estar app y BD juntos
Clientes	Se hace difícil gestionar el número de clientes, ya que depende de distintos hardware.	Control de clientes a nivel de base de datos y a nivel de la aplicación	Gestión sencilla de clientes, al depender más del hardware.

Fuente: Adaptado de Nieto (2013) y Bustamante Granda (2017)

Cabe destacar, que una arquitectura es segura por la buena gestión del administrador de IT, también que este tipo de Arquitecturas a día de hoy son fundamentales en pequeñas y medianas empresas porque dan una mejora en estabilidad, a nivel económico y de seguridad.

1.1.4. Sistema de información Web

Internet ha evolucionado como una red de comunicación mundial, y también el desarrollo de las nuevas tecnologías de la información, hace que la web sea como un servicio imprescindible para la comunicación, donde las aplicaciones web tienen la posibilidad de comunicación con el usuario, interactuando para adaptarse a sus requerimientos y gustos.

Según Kendall (2011), Un sistema de información implementado sobre un entorno web, provee a grandes masas de usuarios en un canal de comunicación como es el internet. Los sistemas de información web han creado un espacio para la interacción entre el usuario y el sistema, creando una plataforma para la integración de servicios existente que funcionan en un entorno con internet, dichos sistemas de información web, deben contener los siguientes elementos:

- Usuarios y configuración de preferencias.
- Módulos de entrada y salida de la información.
- Bases de datos de información y conocimiento.
- Módulos de procesamiento.

Se define entonces como sistema de información al "conjunto de elementos relacionados y ordenados, según ciertas reglas que aporta a la organización a la que sirve y que marca sus directrices de funcionamiento la información necesaria para el cumplimiento de sus fines; para ello, debe recoger, procesar y almacenar datos, procedentes tanto de la organización como de fuentes externas, con el propósito de facilitar su recuperación, elaboración y presentación" (Kendall, 2011).

Actualmente, los sistemas de información basados en web y en un entorno de internet, son accedidos por grandes masas de usuarios; creando un nuevo modelo de comunicación, en el que los usuarios interactúan directamente con los sistemas de información para satisfacer sus necesidades de información.

1.1.5. Requisitos de las aplicaciones SaaS

Para poder desarrollar una aplicación SaaS, que tiene el objetivo de servir a una gran cantidad de clientes con una sola instancia de aplicación, se tiene que considerar algunos requisitos técnicos, que ayudan a desarrollar una solución basada en el modelo SaaS (La & Kim, 2009).

1.1.5.1. Multi-tenant

Se refiere al principio de que una sola instancia de software, ejecutándose en un servidor, pueda servir a múltiples organizaciones o clientes. Cada cliente tiene su propio entorno que le da independencia, definiendo la personalización, usuarios u otras configuraciones. Para que una aplicación sea de tipo multi-tenat, tiene que ser diseñada con ese fin, y de esta forma pueda optimizar los recursos del servidor, y pueda servir de ayuda a en la gestión del mantenimiento de la instancia.

1.1.5.2. Escalable y proveer mecanismos de balaceo de carga

El sistema debe soportar el incremento en cara da peticiones de carga por parte del incremento de usuarios o clientes, no por un incremento de la capacidad de hardware si no dándole más recursos a los tenants.

1.1.5.3. Personalización

Al utilizarse una misma instancia para múltiples clientes, es necesario dar el poder de personalización, para permitir la cubrir sus necesidades particulares de cada cliente.

1.1.5.4. Monitoreo

Es necesario tener el control de uso de los tenants, recursos consumidos, cantidad de registros, para evaluar la suscripción y su posterior facturación.

1.1.6. Metodología Scrum

Las organizaciones cada vez más competitivas necesitan de estrategias de lanzamientos de productos tangibles en el tiempo más corto posible, con flexibilidad de cambios que piden los clientes.

1.1.6.1. Manifiesto Agil

En marzo del 2001, profesionales de software fueron reunidos por el creador del libro Metodología Extreme Programming, para discutir sobre los procesos empleados por los equipos de programación (Beck, 1999).

Los integrantes de la reunión resumieron cuatro principios que se han denominado manifiesto ágil.

“Estamos poniendo al descubierto mejores métodos para desarrollar software, haciéndolo y ayudando a otros a que lo hagan. Con este trabajo hemos llegado a valorar:

- A los individuos y su interacción, por encima de los procesos y las herramientas.
- El software que funciona, por encima de la documentación exhaustiva.

- La colaboración con el cliente, por encima de la negociación contractual.
- La respuesta al cambio, por encima del seguimiento de un plan.

Aunque hay valor en los elementos de la derecha, valoramos más los de la izquierda." (*Manifiesto Por El Desarrollo Ágil de Software*, n.d.)

Scrum es un modelo de desarrollo de software ágil, caracterizado por el desarrollo incremental y manejo de equipos auto organizados con calidad en el proceso (Menzinsky *et al*., 2016).

Este modelo está basado en procesos de retroalimentación, es decir el conocimiento viene de la experiencia y se toma decisiones en función a los resultados.

1.1.6.2. Componentes Scrum

Equipos

- Product Owner: El responsable de maximizar el valor del producto de software, encargado de comunicarse con el cliente y saber que espera del producto.
- Scrum Master: El gurú en temas del modelo scrum, responsable de promover y apoyar scrum.
- Equipo de desarrollo: Son los profesionales que realizan el trabajo de entrega (Programadores, diseñadores, documentadores, etc). El equipo tiene que ser suficientemente pequeño para que sea ágil y los suficientemente grande para completar el trabajo.

Eventos

Existen eventos predefinidos que ayudan a regularizar y minimizar la necesidad de reuniones no definidas.

- **Sprint:** Se define el incremento de software en un periodo de tiempo
- **Planificación de script:** Evento donde se define la lista de requerimientos que se entregaran.

- **Scrum diario:** Reunión diarias con el fin de planear el día de trabajo, donde se responde las preguntas ¿Qué hice ayer? ¿Qué hare hoy? ¿Tengo algún impedimento?

- **Sprint Review:** Ocurre al final de cada sprint, donde se muestra los avances con el cliente.

- **Retrospectiva:** Donde se analiza las oportunidades de mejora, para poderlos aplicar en el siguiente sprint.

Artefactos

- **Lista de producto**: Lista de todos los requerimientos necesarios para finalizar el producto.

- **Lista de pendientes**: Lista de requerimientos seleccionados para el sprint.

1.1.7. Framework

Marco de trabajo, es un conjunto de clases cooperativas que unen conceptos, prácticas y criterios, proporcionando una arquitectura que se usa como base de desarrollo de una aplicación, para enfrentar y resolver nuevos problemas de índole similar.

En el desarrollo de software, un entorno de trabajo es una estructura conceptual y tecnológica de asistencia definida, normalmente, con artefactos o módulos concretos de software, que puede servir de base para la organización y desarrollo de software. Típicamente, puede incluir soporte de programas, bibliotecas, y un lenguaje interpretado, entre otras herramientas, para así ayudar a desarrollar y unir los diferentes componentes de un proyecto (María & Haro, 2008).

Representa una arquitectura de software que modela las relaciones generales de las entidades del dominio, y provee una estructura y una especial metodología de trabajo, la cual extiende o utiliza las aplicaciones del dominio.

1.1.7.1. Modelo Vista Controlador

El patrón MVC es clave para implementar una sistema altamente configurable y escalable. La mayoría de los frameworks que implementan el patrón MVC permiten definir en ejecución la interacción entre la vista y el controlador (Harris & Ahmed, 2011).

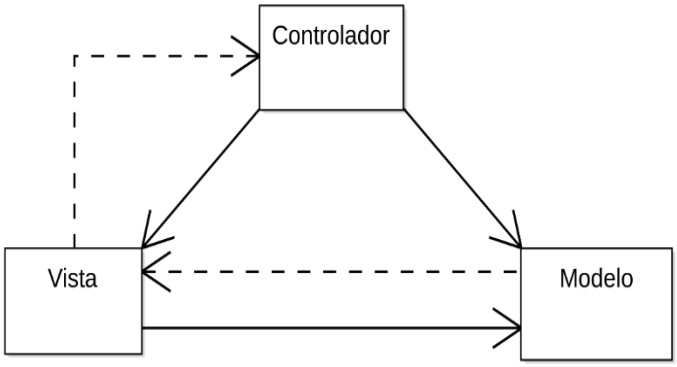


Figura 5. Estructura MVC
Fuente: María & Haro (2008)

Descripción del modelo

- El Modelo: Gestiona toda la información, proporcionando un acceso seguro respetando las especificaciones de la lógica del negocio.
- El Controlador: Proporciona acceso al sistema de información, interpretando la entrada del usuario, y de acuerdo a la lógica del negocio comunica al modelo y la vista.
- La Vista: Representa la interfaz de la lógica de negocio (Representando la salida del sistema).

1.1.8. Servicios web y APIS

Se define como canales de comunicación, usados como intermedios o interfaces de comunicación entre diferentes sistemas.

1.1.8.1. Servicio web

Son componentes de software accesibles utilizando un conjunto de protocolos que sirven para intercambiar datos entre aplicaciones. Aplicaciones de distintas plataformas, pueden utilizar los servicios web para intercambiar información. La interoperabilidad se consigue mediante la adopción de estándares abiertos. Protocolo SOAP (Simple Object Access Protocol) utilizando un lenguaje como WSDL(Web Services Description Languaje) (Mateu, 2004).

1.1.8.2. La API

Es un conjunto de definiciones y protocolo s que sirven para integrar deferentes softwares, permitiendo la comunicación sin necesidad de que estén en la misma plataforma, lenguaje de programación (Li *et al*., 2013).

1.1.9. Ingeniería de Software

Para poder desarrollar el sistema de información, es necesarios, definir un proceso de desarrollo, constando de actividades que ayudan en la construcción del software.

1.1.9.1. Modelo de análisis

En esta etapa se realiza el análisis de requerimientos de un momento determinado, para ser nuevamente analizado en una siguiente etapa donde aparezcan nuevos requerimientos, hasta que se estabilice los cambios.

Los elementos del modelo de análisis que permiten buscar los requisitos, son los escenarios (diagramas de casos de uso), diagramas de clases, y catálogo de requisitos (Pressman & Troya, 2005).

1.1.9.2. Modelo de datos

Modelar como se almacenará la información, sus relaciones y sus restricciones, representa el funcionamiento del sistema a nivel de información, para esta etapa se utiliza diagramas de entidad relación (Henriquez et al., 2015).

1.1.9.3. Modelo Entidad Relación

Es la técnica más sencilla, sirve para analizar los requerimientos a nivel de información con sus respectivas relaciones (Kendall, 2011):

Entidades. Las entidades son datos obtenidos de elementos que serán almacenados, puede ser una persona, un lugar, o una cosa, que es necesario almacenar su información. Las entidades se convierten en tablas en el diseño de la base de datos.

Relaciones. Las relaciones representan la conexión entre entidades, habiendo una entidad primaria que tiene una clave principal que identifica el registro, y sirve para conectarse a sus entidades hijos.

Atributos. Los atributos describen los datos de una entidad. Contienen detalles de la entidad y hacen que cada registro sea único con respecto a otros registros de la misma tabla.

1.1.9.4. Modelo relacional

La relación es el elemento central del modelo. Las relaciones tienen un nombre que los identifica, un conjunto de atributos que los datos que se almacenaran y un conjunto de registros o tuplas. Una relación pude ser representada como una tabla de dos dimensiones (las columnas son los atributos de la relación y las filas son las tuplas) con un único valor en cada celda intersección (Kendall, 2011).

1.1.9.5. Modelado UML

El Lenguaje Unificado de Modelado (UML -Unified Modeling Language) es un conjunto de artefactos que sirven para especificar, visualizar, construir y documentar el funcionamiento de los sistemas software, también es una herramienta para el modelado del negocio y otros sistemas no software (Grau & Segura, 2013).

1.1.9.6. Casos de uso

Los casos de uso representan los requisitos, mediante una documentación y no necesariamente solo diagramas, y el modelado de casos de uso es la acción de representar la funcionalidad del sistema.

UML define un diagrama de casos de uso para ilustrar los nombres de casos de uso y actores, y sus relaciones.

Primeramente, un actor representa el comportamiento de un rol que interactúa con el sistema de información; Ejemplo, en un cajero. El escenario es una representación de acciones entre los actores y el sistema a estudiar; se define instancia de caso de uso al escenario de éxito o fallo, por ejemplo, a la acción de compra de artículos que se paga en efectivo, o al fallo al comprar debido al rechazar la transacción con la tarjeta de crédito. Entonces, un diagrama de casos de uso es un conjunto de escenarios de acciones con éxito y fallo relacionados, que interactúan con los actores utilizando un sistema. En resumen, los casos de uso permiten recopilar los requisitos funcionales que hará el sistema (Kendall, 2011).

1.1.9.7. Modelo del Dominio

Según Kendall (2011), El Modelo del dominio permite mostrar el dominio del problema, mediante el diagrama de clases, siendo el diagrama más importante en la etapa de análisis en la metodología orientada a objetos, Utilizando la notación UML, un modelo del dominio representado por un diagrama de clases donde las clases no especifican ninguna método u operación. Estos diagramas deben tener:

- Clases conceptuales o dominios
- Relaciones o Asociaciones entre las clases.
- Los atributos de las clases.

1.1.9.8. Modelo del diseño

Este modelo tiene 4 componentes importantes para la representación a nivel diseño del sistema, Modelado de datos, Estructura arquitectónica, Diseño de interfaz, y diseño de componentes (María & Haro, 2008).

1. Modelado de datos. Tiene el objetivo de representar la estructura de la información basándose en el modelo de dominio, representado en la etapa de análisis, las estructuras de datos son la base para implementar el sistema. Las entidades, relaciones y restricciones definidas en el diagrama, y la especificación que se da en el diccionario de datos proporcionan la base para

los siguientes diagramas de diseño y muy fundamental para la etapa de desarrollo.

2. Estructura arquitectónica. Representa la relación que existe entre las distintas funcionalidades del sistema. Este diagrama se obtiene a partir de la etapa de análisis donde se detectan los subsistemas y sus interacciones.

3. Diseño de interfaz. Describe la interfaz de comunicación entre el sistema y el usuario.

1.1.10. Pruebas de rendimiento

Para cualquier tipo de aplicación es importante preparar un plan de pruebas de rendimiento de software, con el objetivo de cumplir con los requerimientos, como la rapidez de trabajo en algunos ambientes o condiciones establecidas (Juan Francisco Sánchez, 2018).

Además de la demostración de rapidez, también se verifica algunos atributos de calidad como la escalabilidad, fiabilidad, uso de recursos.

Tipos de prueba de rendimiento

- Pruebas de carga: medir el comportamiento bajo una cantidad esperada de peticiones, con usuarios concurrentes, demostrando tiempos de respuesta.
- Pruebas de estrés: se utiliza para probar con cuanta carga se rompe la aplicación, realizando una carga extrema.
- Prueba de estabilidad: Con el objetivo de determinar si la aplicación soporta cargas continuas y no falle los recursos.
- Pruebas pico: Con pruebas de carga de cambios drásticos en el número de usuarios, ayuda al administrador de servidores a monitorizar los cambios.

1.1.11. Modelo de calidad de producto de software

Para Pressman & Troya (2005) la calidad de software tiene que ver con la concordancia de los requisitos funcionales y los rendimientos establecidos con los estándares de desarrollo documentados y las características esperadas implícitamente.

El estándar ISO 9126 del IEEE para la evaluación de Software, supervisado por el proyecto SQuaRE, del ISO 25000:2005, El estándar está dividido en 4 partes: modelo de calidad, métricas externas, métricas internas y calidad en las métricas de uso (Sicilia, 2009).

1.1.11.1.Modelo de calidad

Se identifica seis atributos clave de calidad de software, como son la funcionalidad, fiabilidad, usabilidad, eficiencia, mantenibilidad y portabilidad.

Tabla 4
Atributos de calidad

Atributos de calidad	Subatributos
Funcionalidad Es el grado de satisfacción de las necesidades que indica los siguientes atributos	• Idoneidad • Corrección • Interoperabilidad • Conformidad • Seguridad
Fiabilidad Es el tiempo que el sistema estará disponible para su uso	• Madurez • Tolerancia a fallos • Facilidad de recuperación
Usabilidad Las interfaces del sistema tienen que lograr una óptima comunicación con el usuario.	• Facilidad de comprensión • Facilidad de aprendizaje • Operatividad
Eficiencia Grado en que el sistema hace optimo el uso de los recursos disponibles para su uso.	• Tiempo de uso • Recursos utilizados
Mantenibilidad Facilidad en la modificación del sistema, pueda ser realizada en el futuro.	• Facilidad de análisis • Facilidad de cambio • Estabilidad • Facilidad de prueba
Portabilidad La posibilidad de que el sistema pueda funcionar en diferentes entornos.	• Facilidad de instalación • Facilidad de ajusto • Facilidad de adaptación
Conformidad	Aplicado a todos los anteriores atributos

Fuente: Sicilia (2009)

1.2. Antecedentes

En la presente investigación se tiene como antecedentes trabajos de investigación científicos publicados en el ámbito nacional y mundial.

El Kafhali & Salah (2018) presenta un modelo matemático de colas para estudiar y analizar el rendimiento de las máquinas virtuales multinúcleo que alojan aplicaciones SaaS en la nube. Nuestro modelo analítico estima bajo cualquier carga de trabajo ofrecida la cantidad de instancias de VM multinúcleo necesarias para satisfacer los parámetros de calidad de servicio (QoS).

Luna Pérez *et al.* (2018) en su artículo propone un desarrollo progresivo de diferentes etapas que inicia desde la recopilación de requerimientos hasta la elaboración de un prototipo funcional, haciendo uso de visualizaciones de gráficas estadísticas para que el usuario final tenga un mejor panorama del comportamiento de sus clientes, se realizan casos de estudio y análisis de escenario con el objetivo de tener una estrategia bien entendida y un software de calidad que lleve a las PyMES.

Para Nenartovič (2017) estudia en profundidad las arquitecturas Cloud multi-tenant definiendo modelos de referencia detallados para las Arquitecturas Multi-tenat (AMTs) y su capa de datos, lo cual no se encuentra descrito de forma explícita en la literatura. Además, partiendo de dichos modelos se propone una extensión que permite a las AMTs tradicionales desplegar múltiples funcionalidades de forma selectiva y en una única instancia de software. Esta extensión busca fomentar la reusabilidad entre componentes, reducir nuevamente costes y disminuir la complejidad en el uso de sistemas. Como conclusión indica que los AMT son una tecnología clave para el éxito de SaaS como un nuevo modelo para la entrega de software en la computación en la nube. La tenencia múltiple es un patrón arquitectónico para aplicaciones SaaS que permite que varios clientes se consoliden en el mismo sistema operativo.

Vidhyalakshmi & Kumar (2017), presenta la investigación donde propone un framework de evaluación de confiabilidad orientada al cliente (CORE). AHP, uno de los métodos MCDM, se utiliza en el marco que acepta las preferencias de atributos comparativos y asigna sus pesos, en función de los cuales se calculan los valores de confiabilidad. El marco CORE facilita a los clientes realizar un proceso eficiente de

toma de decisiones sobre productos SaaS en función de los requisitos comerciales y sus preferencias de atributos. El marco también ayuda a los proveedores a comprender la posición de su producto y las expectativas de los clientes.

Ercolani (2017), presenta un análisis de los factores utilizados por las PYMES para la toma de decisiones al optar por un SaaS. Se muestra la evaluación de los atributos que identifican y caracterizan los componentes de beneficios y preocupaciones relativos a la capacidad técnica del producto evaluado, así como su importancia en el ámbito empresarial específico.

Ercolani (2017) presenta un estudio analiza la computación en nube para la pequeña y mediana empresas (PYMEs) como una solución de las cuestiones relativas a la introducción o la evaluación de las nuevas tecnologías que pueden beneficiar a la empresa. El resultado del proceso de análisis expuesto genera el IPA (numero valorado entre 1 y 4) que indica el nivel de utilidad para la empresa a la adopción del software SaaS analizado.

Area *et al.* (2016). presenta los aspectos financieros de la participación de la nube en la actividad empresarial. La consideración financiera es importante para este sector, ya que siempre trabajan con un presupuesto reducido. Esto brinda detalles sobre las diferentes aplicaciones en la nube disponibles junto con su retorno de la inversión.

Kiran *et al.* (2015) en su investigación explica algunos de los desafíos y experiencias para identificar una solución adecuada para realizar pruebas de rendimiento para aplicaciones web, con caso de estudio una plataforma de autentificación única desarrollada con capacidad para admitir múltiples mecanismos de autenticación y se puede integrar a cualquier aplicación web para proporcionar una solución de inicio de sesión único (SSO).

Shaikh & Patil (2014) propone un sistema basado en un modelo SaaS de múltiples inquilinos que hace posible tener múltiples tiendas en línea desde la base de datos y el código único. Esta solución permite una estrategia comercial de comercio electrónico innovadora porque permite compartir los costos de desarrollo y mantenimiento.

Kouki & Ledoux (2013) propone SCAling, una plataforma y un enfoque impulsado por el acuerdo de nivel de servicio (SLA) para el escalado automático de la nube. Con

experimentos de simulación que indican que el modelo mantiene con éxito la mejor compensación entre las ganancias del proveedor de SaaS y la satisfacción del cliente sin requerir reglas de escala predefinidas.

Nieto (2013) tiene como objetivo investigar el diseño de aplicaciones SaaS sobre Plataformas como Servicio, analizando las características propias de una aplicación SaaS en combinación con los servicios que ofrecen los diferentes proveedores de PaaS, de forma tal de evidenciar la problemática técnica que involucra esta combinación, con el fin de identificar conceptos de diseño y de arquitectura que permitan construir soluciones óptimas. Como conclusión final indica que una construcción de una aplicación web como SaaS implica entender los conceptos de multi-tentat, configurabilidad y escalabilidad, y esto implica tener conocimientos avanzados en programación muy diferentes a aplicaciones web tradicionales.

Lee & Choi (2012), describe un framework de aplicaciones web multi-tenants para SaaS. El marco propuesto admite personalizaciones en tiempo de ejecución de las interfaces de usuario y la lógica comercial mediante el uso de espacios de nombres, herencia y polimorfismo a nivel de archivo. En el framework, varios inquilinos comparten un único servidor de aplicaciones y esto reduce los costos operativos.

López *et al*. (2012) Expone los beneficios más importantes y las desventajas más significativas de la computación en la nube, tomando como objeto de referencia las PyMEs argentinas. Basado en el análisis hecho sobre distintos trabajos de investigación y diversas páginas de proveedores de servicios de Computación en la Nube, además se presenta un conjunto de sugerencias que ayuden al lector a valorar debidamente el impacto que en la actualidad conlleva la incorporación de Computación en la Nube en cualquier pequeña o mediana empresa argentina.

Madisha & Van Belle (2011) informa sobre la investigación sobre la preparación y adopción de SaaS en Sudáfrica como economía emergente. Este estudio se centra en organizaciones pequeñas y medianas, que incluyen tanto empresas como organizaciones sin fines de lucro.

Pattabhiraman *et al.* (2011) Propone nuevos modelos gráficos formales y métricas para evaluar el rendimiento de SaaS y analizar la escalabilidad del sistema en las nubes.

En Tsai *et al*. (2011) introduce al desarrollo de un framework simplificado que provee alternativas para construir múltiples clientes, los scripts automatizan la solución, mediante el uso de plantillas que provee una fácil personalización acorde a los requerimientos del cliente. El artículo científico tiene como conclusión un marco SaaS que adopta un diseño centrado en el consumidor y un diseño orientado al servicio con características como un marco de personalización con un sistema de recomendaciones, un sistema de indexación con redundancia y un esquema de recuperación a nivel de SaaS, en lugar de a nivel de PaaS. Estas características facilitan el desarrollo rápido de aplicaciones SaaS. La efectividad del enfoque propuesto demostrada por un estudio de caso.

Khalid (2010) destaca algunos de los problemas más críticos de la implementación de computación en la nube en las pequeñas empresas, junto con algunos pasos de mitigación para lograr una implementación gratificante. Esto también describe algunos trabajos de desarrollo futuro del concepto de colocación subyacente.

Xiao & Cheng (2010) proporcionan algunas soluciones para resolver los problemas de gestión de las CRM que utilizan el modelo SaaS, basadas en la teoría de la seguridad informática, la computación en la nube y la minería de datos.

Tang et al.(2010), describe una investigación sobre la plataforma de administración SaaS y propone un marco para admitir diferentes modelos comerciales, así como múltiples arquitecturas de aplicaciones. El framework cubre los requisitos técnicos y comerciales de manera separada pero interconectada, y tiene en cuenta consideraciones no funcionales, como la disponibilidad, la seguridad, la escalabilidad y el rendimiento. Validando mediante un estudio de caso que aborda un par de escenarios típicos de gestión de SaaS.

Wang *et al.* (2009), propone un framework de identidad abierta, que aprovecha el protocolo de identidad abierta como OpenID y OAuth. Proporcionando beneficios a todos los roles involucrados en el ecosistema de una manera no intrusiva y centrada en el usuario.

CAPÍTULO II
PLANTEAMIENTO DEL PROBLEMA

2.1. Identificación del problema

Los sistemas de información basados en computadoras son un factor de éxito en el crecimiento de las organizaciones, permitiendo ser competitivos e innovadores, suministrando una plataforma de información para la toma de decisiones.

Según la Sociedad de Comercio Exterior del Perú ComexPerú (2017) en el Perú las MyPes representan el 96.5% de las empresas. La adopción de las Tecnologías de la Información y Comunicación (TIC) se da con más frecuencia en las grandes empresas que tienen mayor capacidad de inversión y recursos, las pequeñas y medianas empresas le siguen gradualmente, según INEI (2018) de 82,249 empresas peruanas el 94.2% de las empresas hicieron uso de computadoras, el 92,6% del servicio de internet y el 18,9% hicieron uso de intranet, este último porcentaje indica que tienen adoptado el uso de sistemas de uso interno, y solo el 12.4% de empresas peruanas desarrolla su propio software para dar soluciones a sus necesidades particulares.

Actualmente las TIC's, se han convertido en uno de los instrumentos clave para medir el desarrollo de las empresas del país (Álvarez *et al*., 2019), además una empresa generalmente tiene como objetivo expandirse, o crecer en números de clientes, ventas, etc. Lo que se traduce en crecimiento de peticiones y consumo de recursos del servidor, los cuales son limitados de acuerdo al modelo de programación de un solo servidor para varios clientes.

La computación en la nube, siendo un modelo de computación que permite utilizar bajo demanda recursos ofrecidos como procesamiento, almacenamiento, aplicaciones y servicios. Siendo las MyPes las que más se benefician de esta tecnología (Fons Gómez,

2014), al no ser necesario de presupuestos altos para la adquisición y administración de esta tecnología, pero si necesitan de herramientas que les ayuden implementar sus soluciones de sistemas de información (Henriquez *et al*., 2015).

Las MyPes, al no disponer recursos suficientes y herramientas que faciliten el desarrollo rápido, y que aproveche los recursos disponibles en una infraestructura en la nube, se encuentran en una desventaja competitiva. A la hora de desarrollar un sistema de información, es vital escoger un Framework que se adapte a las necesidades de la empresa, las tecnologías actuales y la infraestructura; Por lo cual el desarrollo se vuelve cada vez más complejo, al no tener un esquema que conecte las tecnologías.

2.2. Enunciados del problema

2.2.1. Problema general

La presente investigación se realizó en base a la formulación del problema ¿Es posible desarrollar un Framework escalable para sistemas de información en la nube?

2.2.2. Problemas específicos

- ¿Cuáles son los requerimientos y las arquitecturas para el framework para sistemas de información en la nube?
- ¿Es posible implementar el framework para el desarrollo escalable de sistemas de información en la nube?
- ¿Cuál es el rendimiento del framework en la implementación de un sistema de información en la nube?

2.3. Justificación

Los beneficios de la computación en la nube son múltiples, principalmente para las pequeñas y medianas empresas, al no requerir de altos presupuestos para su implementación; de la misma forma la implementación de sistemas de información, requiere de personal especializado en la administración de servidores en la nube. Por lo que es necesario desarrollar un esquema que ayude a hacer uso eficiente de los recursos proveídos por la infraestructura en la nube.

Existe una creciente necesidad de las MyPes, de sistemas de información que ayuden en la gestión de sus negocios, de esta forma ser más competitivos con las grandes empresas.

Actualmente las MyPes implementan de sus sistemas de información sobre servidores compartidos, o servidores con capacidades limitadas, este tipo de soluciones tienen las desventajas de no poder controlar el crecimiento de usuarios (procesos), espacio en disco limitado, medir los recursos utilizados.

Por lo tanto, se propone el un framework que proporcione un marco de trabajo para desarrolladores de sistemas, con posibilidades soportar múltiples usuarios, para múltiples empresas, y espacio en disco para base de datos o copias de seguridad a petición, para ello se utiliza las APIs de servicios de computación en la nube, siguiendo los procesos de desarrollo de software.

En cuestión de investigaciones existe pocos estudios referentes a esta necesidad de las MyPes, además dichas investigaciones teóricas y donde no se describen el esquema para el desarrollador de código fuente de sistemas de información, lo cual motivó la implementación de un framework que proporcione un marco de trabajo para sistemas de información sobre computación en la nube.

La presente investigación tiene como caso de estudio, un sistema de información de facturación electrónica desarrollado sobre el framework propuesto, este sistema es usado por diversos clientes y usuarios de la región de Puno.

La importancia de utilizar un framework orientado al desarrollo de sistemas de información para computación en la nube, es que otras MyPes puedan tener como referencia el marco de trabajo, y puedan implementar sus soluciones sobre las nuevas tecnologías que ofrece la computación en la nube.

Con los resultados de la presente investigación se puede realizar futuras investigaciones usando el framework para dar solución a otros tipos de necesidades organizacionales.

2.4. Objetivos

2.4.1. Objetivo general

Desarrollar un Framework para el desarrollo escalable de sistemas de información en la nube orientado a MyPes

2.4.2. Objetivos específicos

- Analizar los requerimientos y diseñar el framework para sistemas de información en la nube.
- Implementar el framework para el desarrollo escalable de sistemas de información en la nube.
- Evaluar el framework en la implementación de un sistema de información en la nube.

2.5. Hipótesis

2.5.1. Hipótesis general

El funcionamiento del Framework permite el desarrollo escalable de sistemas de información en la nube orientado a MyPes.

2.5.2. Hipótesis especificas

- El framework permite mejorar el tiempo de respuesta del sistema de información implementado sobre una infraestructura de computación en la nube.
- El framework permite mejorar la estabilidad en la respuesta del sistema de información implementado sobre una infraestructura de computación en la nube.
- El framework permite mejorar el rendimiento del sistema de información implementado sobre una infraestructura de computación en la nube.

CAPÍTULO III
MATERIALES Y MÉTODOS

3.1. Lugar de estudio

La presente investigación se realizó sobre la plataforma de computación en la nube del proveedor Amazon (Amazon Inc., n.d.), que dispone de múltiples recursos propios de una infraestructura en la nube. Se ha implementado la aplicación desarrollada con el Framework Escalable para sistemas de información en la Nube orientada para MyPes, con soporte a múltiples peticiones de clientes y usuarios.

3.2. Población

La población está conformada por simulación de clientes que generarán peticiones del sistema de información.

3.3. Muestra

El tipo de muestra que se usó es no probabilístico, muestreo por conveniencia, por los criterios de accesibilidad y proximidad para el investigador. Teniendo como muestra en este caso 10 simulaciones de clientes del sistema de información, estas serán representadas por peticiones las cuales serán evaluadas con la herramienta JMeter, esta herramienta está orientada a medir el rendimiento de aplicaciones web.

3.4. Método de investigación

La investigación cuantitativa nos da la posibilidad de generalizar ampliamente los resultados adquiridos en la investigación, otorgándonos control sobre los variables de los fenómenos, con posibilidad de réplica, y un punto de vista de estadísticas de los fenómenos, facilitando la comparación entre los estudios similares.

Por lo cual la presente investigación corresponde al enfoque de investigación cuantitativa debido a su naturaleza que presenta y ser desarrollado en forma objetiva.

Para el estudio donde se manipulan una o más variables intencionalmente, la variable independiente, el Framework para el desarrollo escalable, y analizar las consecuencias de la manipulación de la variable dependiente, el desarrollo de un sistema de información en la nube orientado MyPes, analizando las consecuencias de la manipulación de las variables dependientes, se utiliza el diseño experimental, donde se da el control al investigador para su control.

3.5. Descripción detallada de métodos por objetivos específicos

3.5.1. Método para desarrollar el objetivo específico 1

3.5.1.1. Ingeniería de requerimientos

Las investigaciones revisadas en el marco teórico permitieron establecer un marco de trabajo a seguir para el proceso de desarrollo de software y así cumplir con el objetivo de la investigación. Para el modelado de análisis y diseño se ha usado el UML, modelo de datos, diagrama de clases y el diseño de la interfaz.

3.5.1.2. Procedimiento de recolección de Requisitos

La recolección de características funcionales es necesaria para la elaboración del catálogo de requisitos y los diagramas de casos de uso, diagrama de clases para luego ordenarlos por subsistemas y prioridades.

3.5.1.3. Herramientas para el Desarrollo del Sistema

Se ha preferido utilizar herramientas libres para el desarrollo del framework, ya que existe abundante material sobre estas herramientas, además de que se puede replicar la investigación.

3.5.1.4. Amazon AWS (Amazon Web Services)

Es una plataforma de computación en la nube, que contiene una serie de servicios de computación en una nube publica, AWS está situado en las 18 regiones geográficas:

- EE.UU. Este: Norte de Virginia, Ohio

- EE.UU. Oeste: Norte de California, Oregón
- Asia Pacífico: Bombay, Seúl, Singapur, Sídney, Tokio
- Canadá: Central
- China: Pekín, Ningxia
- Europa: Fráncfort, Irlanda, Londres, París
- América del Sur: São Paulo
- AWS GovCloud (US-West)

Las instancias de amazon están disponibles en la mayoría de las zonas de disponibilidad de las regiones de AWS y proporcionan un modelo para implementar su aplicación en los centros de datos para lograr alta disponibilidad y confiabilidad.

Distintas organizaciones utilizan Amazon para implementar sus sistemas de información, tales como Netflix, Nasa, Pinterest, etc. Ya que Amazon tiene la suficiente madurez.

3.5.1.5. Servicios Web de Amazon (AWS)

Existen variedad de servicios proveídos por la infraestructura de Amazon, en la Figura 7 se muestra de la estructura necesaria de servicios.

- Amazon EC2: Amazon Elastic Compute Cloud, permite administrar la plataforma de computación, con la posibilidad de elegir el microprocesador, almacenamiento, redes, sistema operativo (Linux, Ubuntu, Windows, etc.), admitiendo distintos procesadores (Intel, AMD y ARM), con redes Ethernet de 400 Gbps.
- Amazon RDS: Con Amazon Relational Database Service, ofrece una administración sencilla al utilizar y escalar una base de datos relacional en la nube. Automatizando las tareas de administración, como el aprovisionamiento de hardware, la configuración de bases de datos y la

creación de copias de seguridad, puede utilizar distintos sistemas de base de datos como:

  - MySQL
  - PostgreSQL
  - Oracle
  - SQL Server
  - Amazon Aurora
  - DynamoDB.

- Amazon VPC: Amazon Virtual Private Cloud le permite controlar el entorno e redes virtuales, como su ubicación de los recursos, conectividad, y seguridad, en el cual se puede agregar recursos, como instancias de Amazon Elastic Compute Cloud (EC2) y Amazon Relational Database Service (RDS).

- Amazon S3: Amazon Simple Storage Service, es un servicio de almacenamiento de archivos que se ofrece bajo disponibilidad, escalabilidad, seguridad y rendimiento. Con clases de almacenamiento rentables y características de administración fáciles de utilizar.

- Almacenamiento y copias de seguridad: La nube de AWS ofrece varias opciones para almacenar, acceder y realizar copias de seguridad de datos y archivos de aplicaciones web, tal como Amazon Elastic Block Store (Amazon EBS), es un servicio de almacenamiento en bloque fácil de usar, escalable y de alto rendimiento.

- Herramientas para desarrolladores: Son un conjunto de servicios diseñados para permitir que los desarrolladores y los profesionales de operaciones de TI gestionar los servicios de forma rápida y segura.

- Grupo de Seguridad y control de acceso: Un grupo de seguridad actúa como un firewall virtual para que sus instancias EC2 controlen el tráfico entrante y saliente. En AWS, los dispositivos de red como firewalls,

enrutadores y balanceadores de carga para las aplicaciones de AWS ya no residen en dispositivos físicos y se reemplazan con soluciones de software.

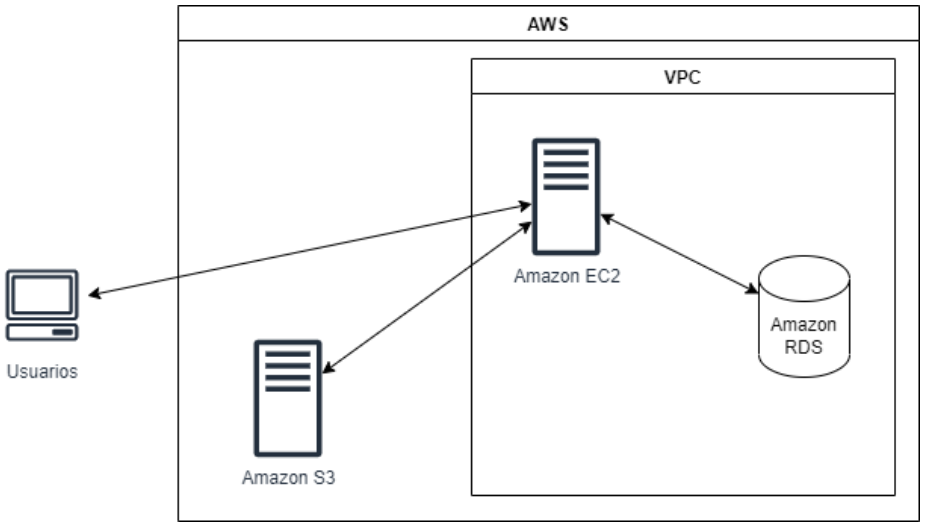


Figura 6. Estructura básica de servicios Amazon

Fuente: Adaptado de (Amazon Inc., n.d.)

Esta estructura de servicios en la nube nos permite escalar a medida que crece la base de usuarios, tal como se muestra la Figura 8, agregando más servidores rápidamente, esto se llama Escalabilidad, ademas permite reducir la cantidad de servidores eliminandoslos en consecuencia, esto se conoce como Elasticidad.

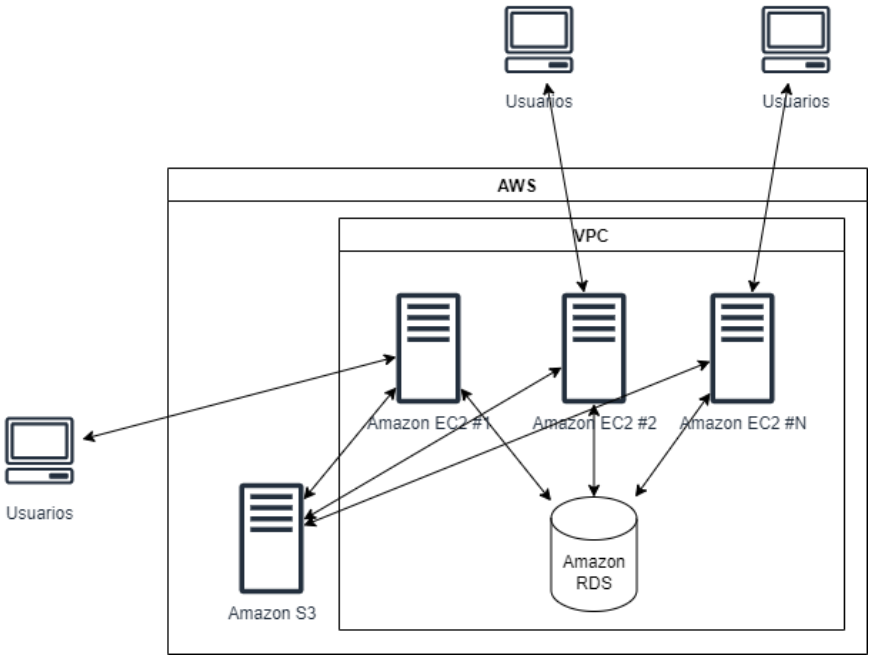


Figura 7. Estructura de escalamiento de servicios

Las VPC, Virtual Private Cloud es su sección privada de AWS, donde puede colocar recursos de AWS y permitir/restringir el acceso a ellos, esto se utiliza para escalar a demanda del sistema de información.

3.5.2. Método para desarrollar el objetivo específico 2

Según las características del sistema a desarrollar se descartó la metodología tradicional, debido al tiempo y la disponibilidad de los usuarios a colaborar con el desarrollo, motivo por el cual se optó por la metodología ágil Scrum, destacando las entregas o prototipos, los cuales permiten recolectar rápidamente los requisitos, y así poder perfeccionar mediante un proceso de retroalimentación.

Figura 8. Roles, artefactos y eventos principales de Scrum
Fuente: Deemer *et al*. (2009)

3.5.2.1. Componentes Scrum

Equipos

- Product Owner: El responsable de maximizar el valor del producto.
- Scrum Master: El gurú en temas del modelo scrum.
- Equipo de desarrollo: Son los profesionales que realizan el trabajo de entrega.

Eventos

- **Sprint:** Se define el incremento de software en un periodo de tiempo
- **Planificación de script:** Evento donde se define la lista de funcionaidad que se entregaran.
- **Scrum diario:** Reunión diarias con el fin de planear el día de trabajo.
- **Sprint Review:** Ocurre al final de cada sprint

Artefactos

- **Lista de producto**: Lista de todos los requerimientos necesarios para finalizar el producto.
- **Lista de pendientes**: Lista de requerimientos seleccionados para el sprint.

3.5.2.2. Flujo de trabajo de SCRUM

Generación del Lista de producto

Se lista de manera ordenada, en módulos, categorías, o prioridad todas las tareas por realizar para el desarrollo del producto.

Planificación del Lista de pendientes

Se planifica las tareas a realizar por cada Sprint, que generalmente puede tomar 2 a 4 semanas, cada nuevo Sprint debe de empezar después de la conclusión del Sprint anterior, luego de definir el sprint se empieza a dividir las tareas entre los miembros del equipo.

Reunión diaria

Se realiza al empezar el trabajo diario, con el fin de preparar el contexto para el resto del día, con el objetivo de intercambiar información entre los miembros del equipo.

Prueba y demostración del producto

Antes de presentar el producto generado por el sprint, es necesario la revisión del sprint, realizar pruebas de funcionamiento, de necesitar cambios se debe añadir al desarrollo.

Planificación retrospectiva

Al finalizar el sprint, el quipo Scrum realiza una mirada retrospectiva al Sprint, y si se pueden realizan mejoras, o solucionar errores cometidos, para que el siguiente sprint pueda tener mejores resultados.

3.5.3. Método para el desarrollo del objetivo específico 3

3.5.3.1. Pruebas de rendimiento

El plan de pruebas de software consta de todos los pasos que deben ejecutarse en el script, conformado por ThreadGroup, Sampler y Listener, midiendo el rendimiento utilizando usuarios virtuales en forma de hilos que llegan al servidor.

3.5.3.2. jMeter

Apache JMeter está basado en Java y es Open Source, está diseñado para la evaluación de aplicaciones web, en cuanto a su rendimiento, donde se puede simular una carga pesada en un servidor de red o para analizar el rendimiento general bajo diferentes tipos de estrategias, además nos dará un análisis gráfico.

Características

- Capacidad para cargar y probar el rendimiento de aplicaciones web
- IDE con funciones de grabación y plan de pruebas
- Modo CLI, desde línea de comandos.
- Reportes
- Portabilidad
- Subprocesos múltiples
- Almacenamiento en cache

3.5.3.3. Flujo de trabajo de jMeter

JMeter utiliza un PROXY para capturar todas las peticiones que hace el navegador web y a partir de ahí generar un plan de pruebas basado en las peticiones realizadas, tal como se muestra en la Figura 9.

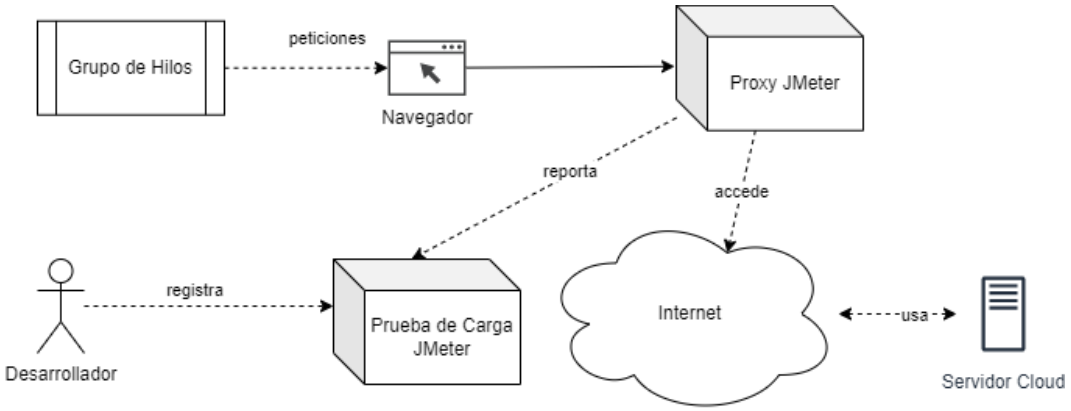


Figura 9. Flujo de trabajo JMeter

Fuente: Adaptado de (Software Apache, n.d.)

CAPÍTULO IV
RESULTADOS Y DISCUSIÓN

4.1. Resultados del primero objetivo específico

En la etapa de análisis, se analizó los requisitos que son necesarios para el desarrollo del framework, refinándolos y estructurándolos. Consiguiendo una comprensión más precisa de los requisitos y una descripción de los mismos, para que sea fácil de administrar y que nos proporcione un entendimiento de la estructura y su arquitectura.

Para la etapa de diseño se modeló y diseño el sistema y su arquitectura, y que tengan coherencia con los requisitos funcionales planteados.

4.1.1. Análisis de requerimientos

Se ha realizado estudio de diferentes investigaciones con respecto a las características que debe de tener un sistema de información orientado a la nube (Harris & Ahmed, 2011), mediante técnicas de ingeniería de software y UML se describió los requerimientos funcionales para el desarrollo del framework, teniendo como resultado el siguiente diagrama:

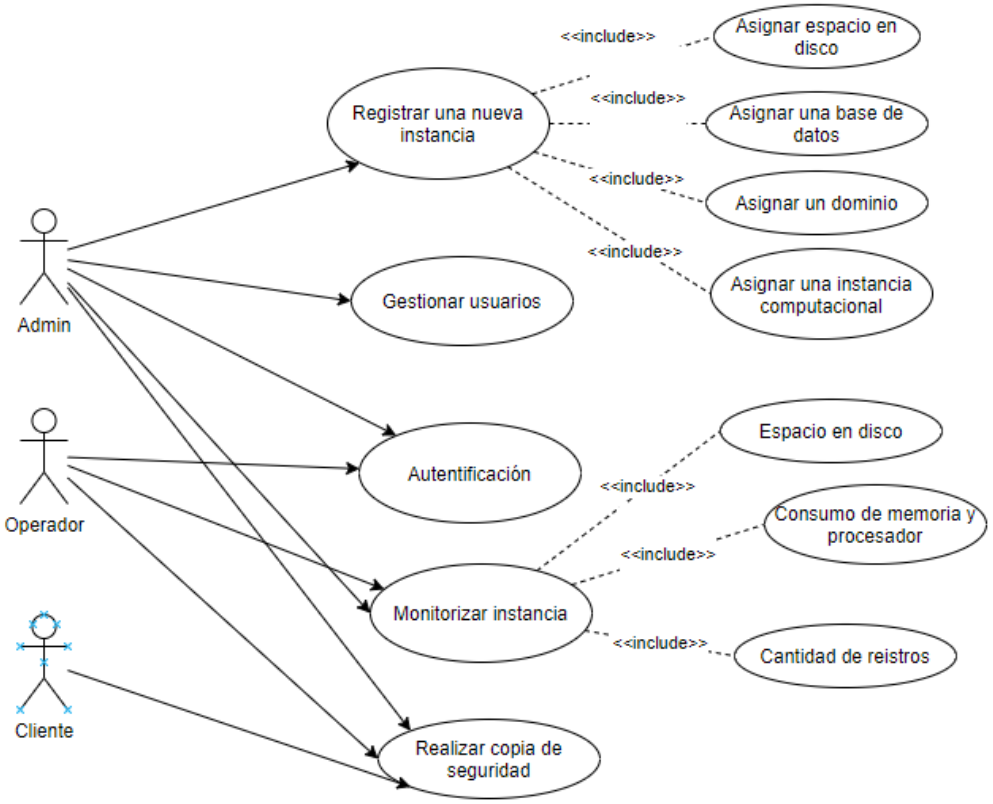


Figura 10. Casos de uso del Framework

En este caso se identificó casos de uso como el registro de instancias que representan un espacio de trabajo del sistema de información para una empresa, la gestión de usuarios que representan usuarios de una empresa, la autentificación, monitorizar instancia y realizar copias de seguridad, tal como se describen en las siguientes tablas.

Tabla 5

Registrar nueva instancia

Descripción de caso de uso	
Actores	Admin
Descripción	Solo el administrador tiene la opción de crear una nueva instancia del sistema, asociada a un código fuente base único, con dominio propio, base de datos propia, espacio en disco y de ser necesario en otra máquina o instancia computacional.

Precondiciones	• El usuario debe de estar autentificado mediante el uso de usuario y contraseña. • Debe haber disponible instancias computacionales o de lo contrario generar una instancia computacional.
Flujo de eventos	• Tiene la opción de crear el un sistema para el nuevo cliente, llenando campos como nombre de la empresa, usuario y correo del administrador del sistema. • Hay un listado de clientes del sistema, mostrando sus instancias computacionales. • Se puede escoger una instancia computacional nueva, para el sistema de información. • Tiene la opción de configurar los espacios asignados, datos del de la cuenta del sistema.
Precondiciones	El cliente ya puede hacer uso de su espacio asignado para el sistema de información.

Tabla 6

Autentificación

Descripción de caso de uso	
Actores	Admin, Cliente, Operador
Descripción	Se permite a los usuarios validar su identidad ante el sistema.
Precondiciones	Ninguno.
Flujo de eventos	• Ingresa su usuario y contraseña • Se verifica los datos ingresado, así como la instancia computacional asignada al cliente. • Si no se recuerda la contraseña, tiene la opción de recordar contraseña.
Precondiciones	El usuario puede realizar las opciones disponibles

Tabla 7

Monitorizar instancia

Descripción de caso de uso	
Actores	Operador, Admin
Descripción	Muestra información del estado del sistema, como cantidad de registros en base de datos, espacio utilizado en disco, promedio de consumo de memoria y procesador.
Precondiciones	• Validación de usuarios. • Seleccionar una cuenta de sistema de información.
Flujo de eventos	• Opción de ver el estado del sistema • Enviar alerta de estado del sistema al cliente • Suspender o dar de baja al cliente.
Precondiciones	-

Tabla 8

Realizar copias de seguridad

Descripción de caso de uso	
Actores	Operador, Admin, Cliente
Descripción	Realiza copias de seguridad de todo el sistema, de la instancia computacional, de la base de datos o de los archivos almacenados en disco.
Precondiciones	• Validación de usuarios. • Seleccionar una cuenta de sistema de información.
Flujo de eventos	• Selecciona una cuenta para realizar una copia • La copia de seguridad solo puede restablecer el admin. • La copia de seguridad para el cliente son solo de los archivos y el registros en base de datos.
Precondiciones	-

Tabla 9

Gestionar usuarios

Descripción de caso de uso	
Actores	Admin
Descripción	Administra cuentas de usuarios, de operadores y clientes, con opciones de crear nuevas cuentas, suspender o eliminar.
Precondiciones	• Validación de usuarios.
Flujo de eventos	• Tiene la opción de crear cuentas y seleccionar el rol de esa cuenta. • Editar, eliminar o suspender cuenta. • Notificar mediante mensajes a una cuenta.
Precondiciones	-

4.1.2. Diseño arquitectónico

Según los requerimientos analizados, se plantea la siguiente arquitectura de la Figura 11, que describe de manera abstracta los elementos computacionales que lo conforman, y la comunicación entre ellos.

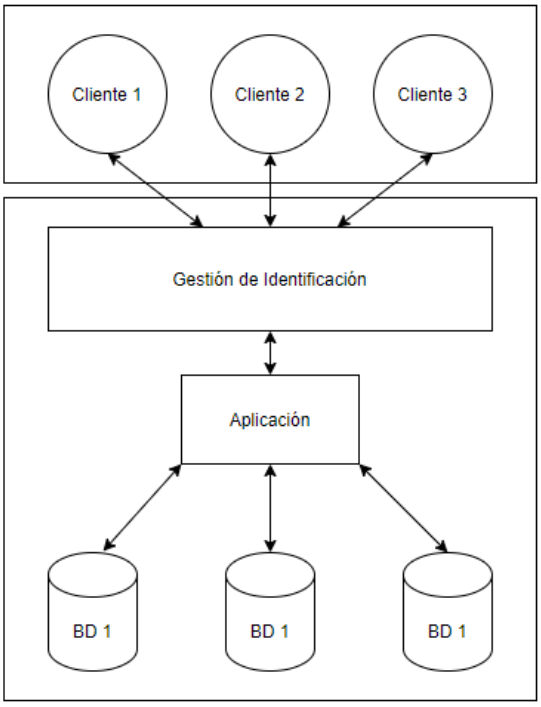


Figura 11. Arquitectura del framework

Tabla 10

Arquitectura del framework

Cliente	Usuario cliente
Gestión de aplicaciones	Administrador de aplicaciones para empresas
Aplicación	Instancia de aplicación
Datos	Base de datos por empresa

4.1.2.1. Arquitectura de base de datos

El almacenamiento de datos de instancias en bases de datos separadas es el enfoque que se utiliza para el aislamiento de datos, permitiendo la separación de información privada entre instancias, tal como lo muestra la Figura 12.

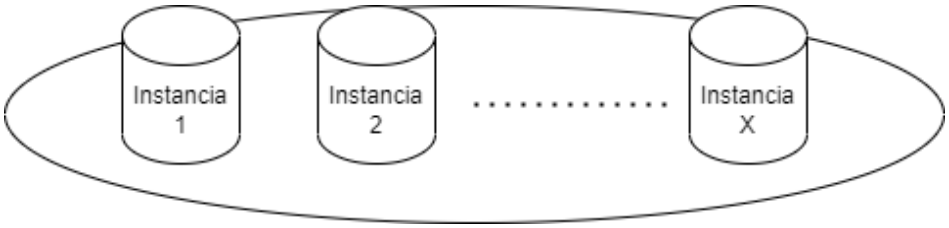


Figura 12. Arquitectura de base de datos separadas

Los recursos informáticos y el código de la aplicación generalmente se comparten entre todas las instancias en un servidor, pero cada instancia tiene su propio conjunto de datos que permanece lógicamente aislado de los datos que pertenecen a todas las demás instancias. La seguridad de la base de datos evita que cualquier instancia acceda accidental o malintencionadamente datos de otros.

Cada instancia de base de datos independiente, facilita la escalabilidad del modelo de datos de la aplicación, para cumplir las necesidades individuales, y la restauración de los datos a partir de copias de seguridad en caso de falla es una tarea relativamente simple.

Desafortunadamente, este enfoque tiende a generar costos más altos para mantener el equipo y respaldar la información de las instancias. Los costos de hardware también son más altos que con enfoques alternativos, ya que la cantidad de instancias que se alojan en un servidor de base de datos.

Los clientes en áreas como la banca o la gestión de registros médicos suelen tener muy fuertes requisitos de aislamiento de datos, siendo útil para organizaciones que prefieran tener sus datos aislados de otras bases de datos.

4.1.2.2. Multi-tentat

También conocido como multi cliente, cada cliente tiene su propio ambiente de uso del sistema de información, con acceso a una base de datos asignada y una instancia de la aplicación de un único código fuente.

Se considera los siguientes aspectos para el desarrollo del framework:

a) Un único código fuente

El mantenimiento de un solo código fuente para el sistema e información, lo hace más, sencillo y barato, haciendo posible que todos los clientes se beneficien de las actualizaciones. Pero necesita de un esfuerzo a la hora de planificación, diseño y desarrollo.

b) Aislamiento entre los diferentes instancias

Para lograr el aislamiento de cuentas de cliente, es necesario aislar completamente la información, asignando cada cuenta a una nueva base de datos o una nueva instancia de base de datos, esto permite que los usuarios de una cuenta del sistema no interfieran en el correcto funcionamiento de otra cuenta de usuario.

Sin embargo, un aislamiento completo, solo logra cuando toda la instancia de la aplicación, incluido el código fuente está dentro de otra infraestructura o ambiente virtual de máquina.

4.1.2.3. Escalabilidad

Una característica importante es que las aplicaciones puedan escalar de forma automática, o con poca intervención del administrador.

4.1.2.4. Configurabilidad

Los sistemas de información en la nube tienen la característica de poder personalizar su información, como son datos propios del cliente, además de otros aspectos.

4.1.3. Diseño del framework

4.1.3.1. Diagrama de base de datos

El modelo de datos es un punto de partida para entender la estructura de la información y la funcionalidad del sistema de información, esta sección se analizó el Modelo Relacional, tanto de la base de datos del framework como del cliente.

Base de datos del framework

El objetivo de esta tarea es realizar el diseño del modelo de datos a partir de los requisitos del sistema.

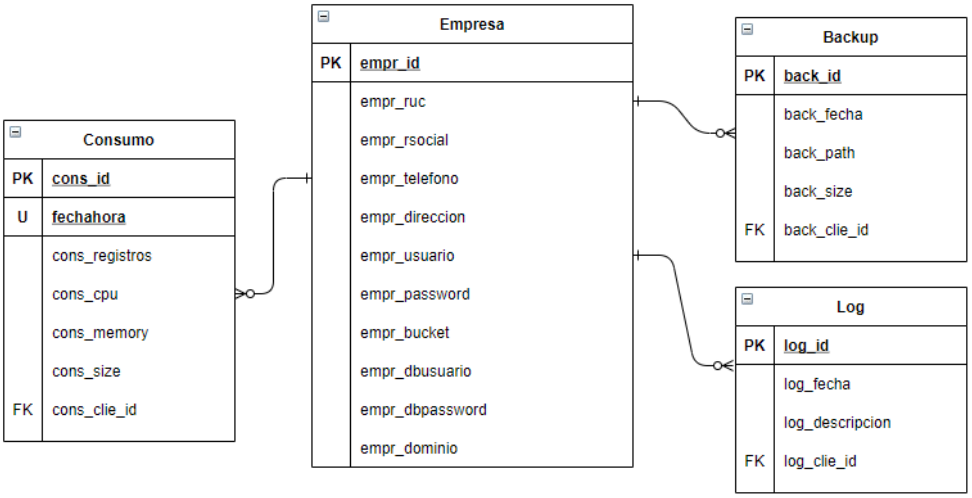


Figura 13. Modelo de base de datos multi-tenat

Fuente: Elaboración propia

Base de datos del cliente

Siendo un sistema para funcionar para múltiples clientes, cada uno con su estructura propia según la Figura 14, y de acuerdo a los requerimientos es necesario tener una estructura de base de datos distribuida, es decir tener una base de datos por cada instancia creada por el framework.

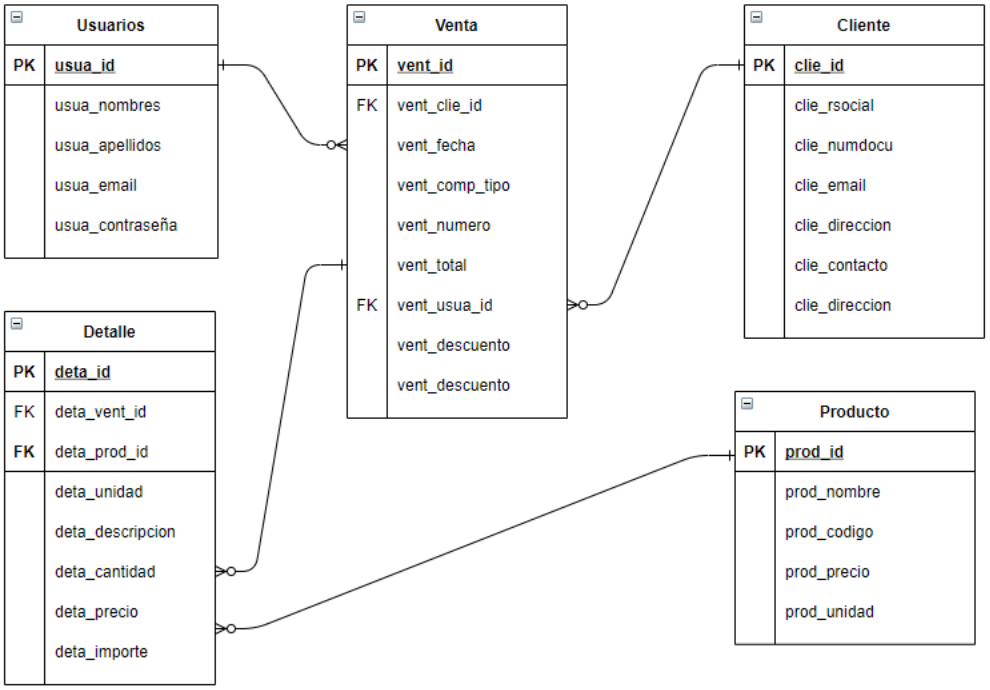


Figura 14. Modelo de base de datos cliente

4.1.4. Discusión

Para la obtención de requisitos de considero los criterios de La & Kim (2009), que ayudaron desarrollar la estructura del framework, además existen varias alternativas al momento de trabajar con bases de datos (Tsai *et al*., 2011), en este caso se escogió la arquitectura que utiliza una instancia de aplicación conectada a varias bases de datos correspondientes a cada cliente, por ser más administrable y segura.

4.2. Resultado del segundo objetivo específico

4.2.1. Dominio de cliente

Todas las instancias de los clientes trabajaran en base a un solo fuente de código, donde cada cliente entraría al sistema con un parámetro de identificación de instancia distinto.

4.2.2. Base de datos compartidas

Al registrar un nuevo cliente del sistema, el proceso de aprovisionamiento crea las tablas específicas para el nuevo cliente en una base de datos nombrándola con prefijo para poder identificar a qué cliente pertenece esta tabla.

Esta opción reduce los costos de administración, puesto que se permite que cada instancia del cliente pueda extender sus tablas, agregando nuevos campos y relaciones, sin interferir con otros clientes.

También es necesario tener toda la información estadística de todos los clientes, que unifiquen la información. Para esta solución necesitaremos definir una nueva base de datos que contendrá la información relacionada con cada cliente.

4.2.3. Arquitectura del sistema

El modelo vista controlador MVC, En la primera capa cada usuario podrá tener acceso a la aplicación por medio de la interfaz de presentación la que permitirá transferir las peticiones a la segunda capa, ésta se encargará de procesar la lógica de la aplicación a través de los scripts desarrollados en el lenguaje de programación PHP y así estructurar la información devuelta por el servidor de base de datos, perteneciente a la tercera capa.

La implementación se ha realizado utilizando el lenguaje PHP, tiene una estructura que permite a los desarrolladores crear proyectos mucho más rápidos, incluye un conjunto de librerías, y métodos simples para acceder a sus librerías. Para las bases de datos se ha utilizado la base de datos MariaDB.

4.2.3.1. Estructura del Framework

Es un Framework para construir aplicaciones web, utilizando el lenguaje PHP, fácil de aprender y con una gran colección de librerías.

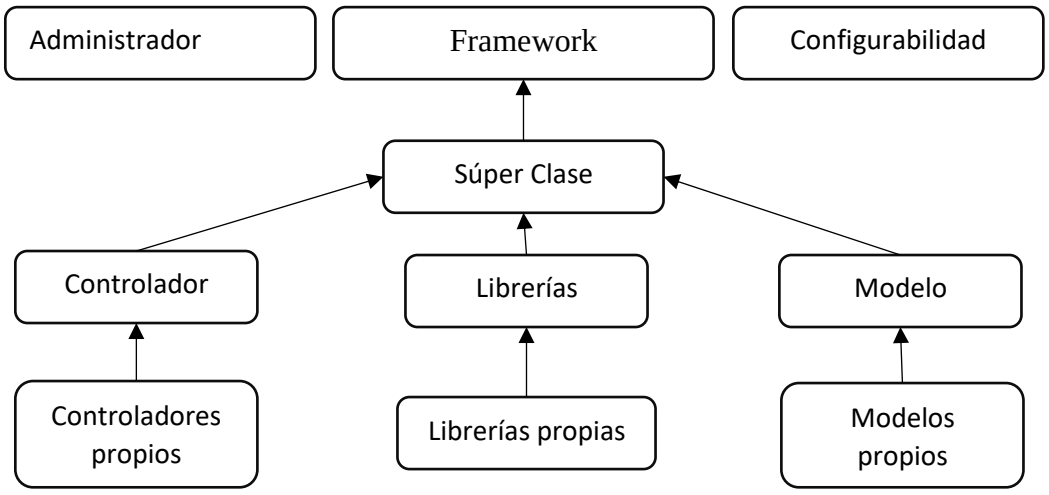


Figura 15. Estructura de clases

Características

- Administración basada en MVC (Modelo vista controlador)
- Facilita el trabajo colaborativo entre varios desarrolladores.
- Permite la habilitación y la incorporación de nuevas librerías.

4.2.3.2. Estructura de archivos

La estructura de archivos ayuda a que las aplicaciones de las instancias estén ordenadas, y permitan escalabilidad, utilizando el modelo MVC, para ubicar rápidamente los archivos de modelo, vista y controlador; además de los archivos de librerías, plugins, copias de seguridad, código fuente del nucleo.

- ADM: Administración de las instancias de la aplicación
- ADPP: Código fuente para crear instancias de la aplicación, con estructura para el MVC.
- BACKUP: Archivos de copias de seguridad, logs y estadísticas.

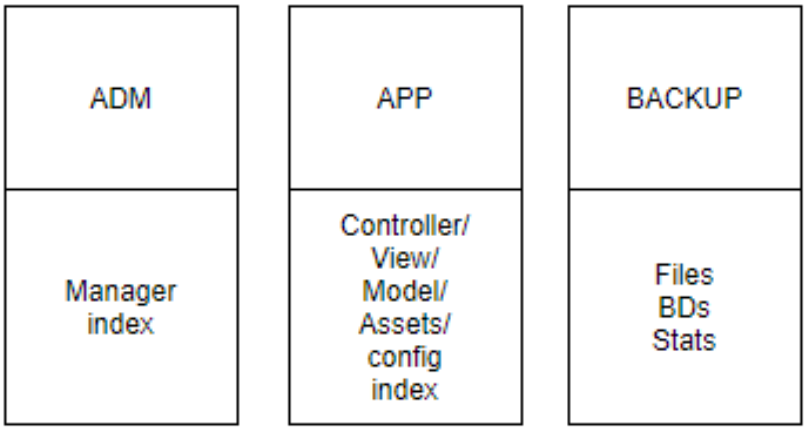


Figura 16. Estructura de archivos

4.2.3.3. Lista de Sprints

Tabla 11
Lista de Sprints

Sprints	Funciones del framework	Esfuerzo	Tiempo
Sprint 1	Crear instancia computacional	90	10
	Crear instancia de base de datos	90	5
	Crear dominio de cliente	20	3
	Crear espacio en disco	50	2
Sprint 2	Anular instancia computacional, base de datos, dominio y espacio en disco	20	2
	Realizar copia de seguridad	40	4
	Eliminar registro de instancia	10	2
Sprint 3	Autentificarse	20	4
	Configurar copias de seguridad automáticas	20	2
	Monitorear disco	10	2
	Monitorear memoria y procesador	10	2
	Monitorear base de datos	10	2

4.2.4. Discusión

La metodología Scrum nos proporciona un marco de trabajo orientado a la entrega inmediata (Rodríguez, 2008), de forma iterativa y adaptable a cambios, la presente investigación los Sprint están agrupados las tareas que representan los requerimientos. Además, se usa las listas de tareas pendientes valorados por esfuerzo y tiempo, permitiendo gestionar adecuadamente la culminación del objetivo de desarrollo del framework.

4.3. Resultado del tercer objetivo específico

Después de haber desarrollados el framework escalable para el desarrollo de sistemas de información en la nube, se procedió a la evaluación del Framework, para lo cual se ha hecho pruebas de rendimiento sobre un sistema de información implementado con el framework.

Para las pruebas se comparó el sistema sin el uso del framework, y con el uso del framework sobre una arquitectura en la nube, con el fin de comparar sobre la misma plataforma, además el framework depende de servicios como almacenamiento y base de datos y la creación dinámica de instancias computacionales que le dan ventaja sobre el desarrollo tradicional.

4.3.1. Características de la plataforma

El framework se instaló sobre una arquitectura en la nube con las siguientes características:

- Zona: EE.UU. Este (Ohio)
- Sistema Operativo: Amazon Linux 64bits
- Tipo de instancia: EC2 t2.medium
- Disco duro: SSD 8GB
- Almacenamiento de archivos en S3
- Servicio de gestión de base de datos RDS (MySQL db.r5.large)

4.3.2. Modelo de pruebas

El modelo de pruebas consiste en tomar datos del informe de resultados, de diferentes pruebas sobre el sistema de información, incrementando el número de usuarios, y el número de peticiones en un determinado tiempo tal como lo muestra la Figura 17, de esta manera se obtiene diferentes valores de rendimiento, tiempo medio, máximo, mínimo de ejecución, y otros valores que permiten analizar.

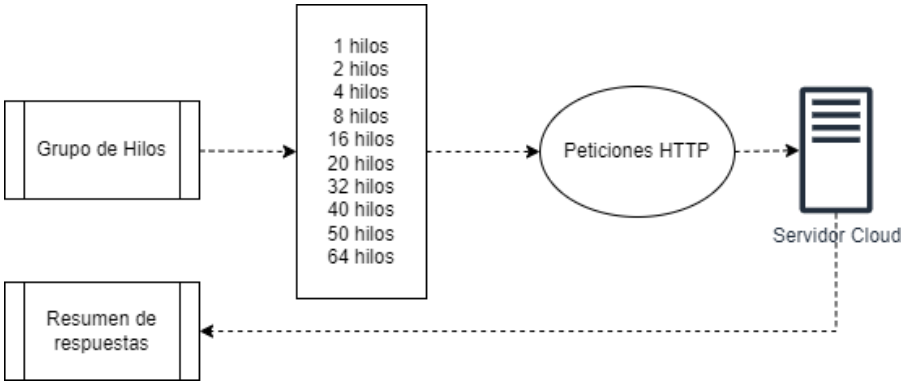


Figura 17. Modelo de pruebas

4.3.3. Tiempo de respuesta

La prueba de tiempo de respuesta, consiste en realizar peticiones del sistema, y medir el tiempo de respuesta.

Los resultados representan el tiempo de respuesta del sistema de información en un esquema tradicional de cliente servidor y haciendo uso del framework.

Tabla 12

Comparación de tiempo de respuesta

Nro. de usuarios	Sin Framework	Con Framework	Diferencia	% de mejora
10	662	592	70	10.57
10	659	665	-6	0.00
10	632	541	91	14.40
10	636	503	133	20.91

10	612	654	-42	0.00
10	657	632	25	3.81
10	608	503	105	17.27
10	647	650	-3	0.00
10	643	518	125	19.44
10	641	561	80	12.48

Como muestra los resultados hay un promedio de 9.89% de mejora en el tiempo de respuesta del sistema de información implementado con el framework propuesto, frente a un sistema tradicional de cliente servidor.

4.3.4. Estabilidad en la respuesta

Las pruebas de estabilidad de respuesta o pruebas de stress, consisten en realizar altas peticiones con el fin de hacer caer el sistema, para esta prueba se realizaron 100 peticiones simultaneas en las diferentes opciones del sistema.

Los resultados son porcentajes de error en la respuesta del sistema de información en un esquema tradicional de cliente servidor y haciendo uso del framework, que depende de los distintos servicios de la nube.

Tabla 13

Comparación de estabilidad de respuesta

Nro. de usuarios	Sin Framework	Con Framework	Diferencia
100	19.00	0	19.00
100	27.25	0	27.25
100	14.50	0	14.50
100	18.60	0	18.60
100	34.17	0	34.17
100	22.15	0	22.15
100	30.50	0	30.50
100	28.65	0	28.65
100	31.34	0	31.34
100	18.80	0	18.80

Como muestra los resultados hay un promedio de mejora del 100.00% en la estabilidad de respuesta del sistema de información implementado con el framework propuesto, frente a un sistema tradicional de cliente servidor.

4.3.5. Rendimiento

Las pruebas de carga de peticiones de diferentes clientes, consistieron en realizar 10 test de 1, 2, 4, 8, 16,20, 32, 40, 50 y 64 hilos (clientes) cada segundo.

Los resultados de rendimiento son medias de tiempo expresados en milisegundos, de diferentes tipos de carga, que representan el rendimiento del sistema de información en un esquema tradicional de cliente servidor y haciendo uso del framework, que depende de los distintos servicios de la nube.

Tabla 14

Comparación de rendimiento

Nro. de usuarios	Sin Framework	Con Framework	Diferencia	% de mejora
1	993	812	181	18.23
2	932	905	27	2.90
4	980	912	68	6.94
8	1212	921	291	24.01
16	3019	1399	1620	53.66
20	2981	1651	1330	44.62
32	3489	1996	1493	42.79
40	4308	1974	2334	54.18
50	4504	2084	2420	53.73
64	4862	2838	2024	41.63

Como muestra los resultados hay un promedio de 34.27% de mejora en el rendimiento del sistema de información implementado con el framework propuesto, frente a un sistema tradicional de cliente servidor. En la Figura 18 se puede observar que con 1 hilo (usuarios) la mejora es de 18.23% y con 54 hilos es de 41.63%.

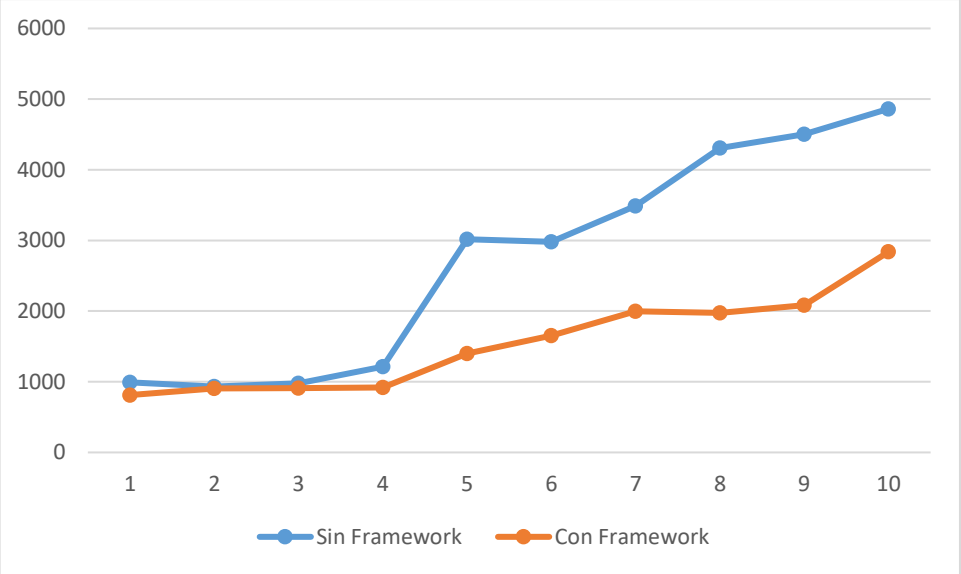


Figura 18. Rendimiento del framework

4.3.6. Discusión

Según Diaz *et al*. (2012) describe las distintas pruebas de rendimiento con el objetivo de optimizar el tiempo de respuesta servidor, en cada etapa se verifica la correcta configuración del servidor y la red, este proceso se realiza previo la comparación de rendimiento, ya que una mala configuración del servidor pueden dar valores incorrectos de rendimiento.

En el presente trabajo de investigación se usó la herramienta Jmeter, para evaluar el rendimiento del framework sobre una infraestructura en la nube, el objetivo fue simular distintos accesos, y múltiples peticiones.

Los resultados obtenidos de rendimiento del sistema de información implementado sobre el framework, frente a un sistema tradicional de cliente servidor, muestran que hay una gran mejora, esto debido a que el framework hace uso de los diferentes servicios de la nube, distribuyendo la carga a otras instancias computacionales.

4.4. Prueba de hipótesis

Para la prueba de hipótesis con respecto al Framework escalable para sistemas de información en la nube, se ha utilizado la prueba de hipótesis denominada t-student, por lo cual se planteó las siguientes hipótesis:

4.4.1. Tiempo de respuesta

Hipótesis Nula (Ho): El Framework no permite mejorar el tiempo de respuesta del sistema de información implementado sobre una infraestructura de computación en la nube.

Hipótesis Alterna (Ha): El Framework permite mejorar el tiempo de respuesta del sistema de información implementado sobre una infraestructura de computación en la nube.

Tabla 15

Prueba de tiempo re respuesta

	Sin framework	Con framework
Media	639.7	581.9
Varianza	342.2333333	4221.877778
Observaciones	10	10
Varianza agrupada	2282.055556	
Diferencia hipotética de las medias	0	
Grados de libertad	18	
Estadístico t	2.70551368	
P(T<=t) una cola	0.007240528	
Valor crítico de t (una cola)	1.734063607	
P(T<=t) dos colas	0.014481056	
Valor crítico de t (dos colas)	2.10092204	

Decisión:

- Si la probabilidad obtenida P(T<=t) dos colas <= a, se rechaza Ho y se acepta Ha
- Si la probabilidad obtenida P(T<=t) dos colas > a, se rechaza Ha y se acepta Ho.

Para el nivel de significancia escogido par al a prueba de hipótesis es de 5%, siendo a = 0.05 y 18 grados de libertad, donde se obtiene el calor t critico de 2.10, y que el nivel de significancia obtenido en la prueba es de 0.014 < 0.05 siendo menor al error permitido de 5%, por lo tanto la hipótesis nula se rechaza y se acepta la hipótesis alterna, que señala que el framework permite mejorar el rendimiento del sistema de información implementado sobre una infraestructura de computación en la nube.

4.4.2. Estabilidad de respuesta

También conocido como prueba de stress, se utiliza normalmente para romper la aplicación para determinando la solides de la aplicación.

Hipótesis Nula (Ho): El Framework no permite mejorar la estabilidad en la respuesta del sistema de información implementado sobre una infraestructura de computación en la nube.

Hipótesis Alterna (Ha): El Framework permite mejorar la estabilidad en la respuesta del sistema de información implementado sobre una infraestructura de computación en la nube.

Tabla 16

Prueba de estabilidad de respuesta

	Sin framework	Con framework
Media	24.4930000	0
Varianza	44.8982844	0
Observaciones	10.0000000	10
Varianza agrupada	22.4491422	
Diferencia hipotética de las medias	0.0000000	
Grados de libertad	18.0000000	
Estadístico t	11.5591821	
P(T<=t) una cola	0.0000000	
Valor crítico de t (una cola)	1.7340636	
P(T<=t) dos colas	0.0000000	
Valor crítico de t (dos colas)	2.1009220	

Decisión:

- Si la probabilidad obtenida P(T<=t) dos colas <= a, se rechaza Ho y se acepta Ha
- Si la probabilidad obtenida P(T<=t) dos colas > a, se rechaza Ha y se acepta Ho.

Para el nivel de significancia escogido par al a prueba de hipótesis es de 5%, siendo a = 0.05 y 18 grados de libertad, donde se obtiene el calor t critico de 2.10, y que el nivel de significancia obtenido en la prueba es de 0.00 < 0.05 siendo menor al error permitido de 5%, por lo tanto la hipótesis nula se rechaza y se acepta la hipótesis alterna, que señala que el framework permite mejorar el rendimiento del sistema de información implementado sobre una infraestructura de computación en la nube.

4.4.3. Rendimiento

Hipótesis Nula (Ho): El Framework no permite mejorar el rendimiento del sistema de información implementado sobre una infraestructura de computación en la nube.

Hipótesis Alterna (Ha): El Framework permite mejorar el rendimiento del sistema de información implementado sobre una infraestructura de computación en la nube.

Tabla 17

Prueba de rendimiento

	Sin framework	Con framework
Media	2728	1549.2
Varianza	2502438.222	457682.4
Observaciones	10	10.00
Varianza agrupada	1480060.311	
Diferencia hipotética de las medias	0	
Grados de libertad	18	
Estadístico t	2.166633346	
P(T<=t) una cola	0.021963446	
Valor crítico de t (una cola)	1.734063607	
P(T<=t) dos colas	0.043926892	
Valor crítico de t (dos colas)	2.10092204	

Decisión:

- Si la probabilidad obtenida P(T<=t) dos colas <= a, se rechaza Ho y se acepta Ha
- Si la probabilidad obtenida P(T<=t) dos colas > a, se rechaza Ha y se acepta Ho.

Para el nivel de significancia escogido par al a prueba de hipótesis es de 5%, siendo a = 0.05 y 18 grados de libertad, donde se obtiene el calor t critico de 2.10, y que el nivel de significancia obtenido en la prueba es de 0.043 < 0.05 siendo menor al error permitido de 5%, por lo tanto la hipótesis nula se rechaza y se acepta la hipótesis alterna, que señala que el framework permite mejorar el rendimiento del sistema de información implementado sobre una infraestructura de computación en la nube.

CONCLUSIONES

Cumpliendo con los objetivos propuestos en la presente investigación, se presenta las siguientes conclusiones para trabajos futuros que derivan de la presente investigación:

- El sistema de información desarrollado bajo el framework propuesto, frente a una arquitectura tradicional de cliente servidor, se concluye que el sistema mejora su rendimiento en un 34%, según el reporte generado por la herramienta JMeter.
- El framework para el desarrollo escalable de sistemas de información, sirve de base en el desarrollo de aplicaciones en la nube, definiendo una arquitectura que ayuda en el proceso desarrollo.
- El análisis de los requerimientos permitió comprender cómo las necesidades del framework para el desarrollo de sistemas de información para la nube, para poder cumplir los criterios de calidad como la alta disponibilidad, seguridad, rendimiento, escalabilidad, eficiencia y rendimiento.
- La implementación del framework para aplicaciones en la nube ayuda en el proceso de desarrollo y a los clientes tener una independencia y seguridad en su sistema y base de datos.
- La evaluación del framework en las instancias creadas de la aplicación, mediante indicadores de rendimiento, permitió validar el framework propuesto.

RECOMENDACIONES

Habiendo finalizado la investigación sobre el desarrollo del Framework escalable para sistemas de información en la nube orientado a MyPes, se recomienda a los futuros investigadores los siguientes:

- Se recomienda el uso de Framework orientados a computación distribuida o que funciones sobre plataformas sobre la nube, para aprovechar los recursos disponibles ofrecidos por servidores en la nube.
- para el desarrollo de aplicaciones para la nube, frente a Framework que están solo orientados a una sola aplicación, ya que aprovechan mejor los recursos ofrecidos por la computación en la nube.
- Se recomienda el uso de metodologías agiles, y la programación colaborativa, para cumplir con los requerimientos obtenidos en la fase de análisis.
- Se recomienda utilizar base de datos no relacionadas, para que ayuden a mejorar el rendimiento y flujo de la información en un entorno distribuido., y además utilizar mecanismos de programación para lograr integridad de datos.

BIBLIOGRAFÍA

Álvarez, I., Complutense, I., & Internacionales, D. E. (2019). *Indicadores de tecnología para medir la presencia global de un país - Elcano*. *1*, 1–14. http://www.realinstitutoelcano.org/wps/portal/rielcano_es/contenido?WCM_GLOBAL_CONTEXT=/elcano/elcano_es/zonas_es/ari115-2019-alvarez-natera-marin-indicadores-de-tecnologia-para-medir-la-presencia-global-de-un-pais

Amazon Inc. (n.d.). *AWS | Elastic compute cloud (EC2)*. https://aws.amazon.com/es/ec2/

Area, I., Vihar, S., & Delhi, N. (2016). *SaaS as a Business Development tool for MSMEs SaaS AS A BUSINESS DEVELOPMENT TOOL SaaS as a Business Development Tool for MSMEs Vikas Kumar Asia-Pacific Institute of Management ,*. *April*.

Beck, K. (1999). Extreme Programming Explained: Embrace Change. In *XP Series*. http://books.google.com/books?id=G8EL4H4vf7UC&pgis=1

Bustamante Granda, W. X. (2017). Multi - Tenancy. *INNOVA Research Journal*, *2*(2), 72–83. https://doi.org/10.33890/innova.v2.n2.2017.124

ComexPerú. (2017). COMEXPERU. *Semanario ComexPerú*, 1–7.

Deemer, Pete, P., Benefield, G., Larman, C., & Vodde, B. (2009). Básica De Scrum (the Scrum Primer). *Scrum Training Institute*, *1.1*, 1–20.

Diaz, F. J., Banchoff, C. M. T., & Soria, V. (2012). *Usando Jmeter para pruebas de rendimiento*. *January*, 1–14.

El Kafhali, S., & Salah, K. (2018). Performance analysis of multi-core VMs hosting cloud SaaS applications. *Computer Standards and Interfaces*, *55*, 1339–1351. https://doi.org/10.1016/j.csi.2017.07.001

Ercolani, G. (2017). Análisis del potencial del cloud computing para las PYMES : un modelo integrado para evaluar software as a service (SaaS) en la nube pública. *Proyecto de Investigación:*

Fons Gómez, F. (2014). *Cloud Computing: caracterización de los impactos positivos obtenidos por la utilización del modelo Cloud Computing por las pymes, basado en la tipología de Modelos de Negocio de este tipo de empresas*. *Cc*, 160.

https://riunet.upv.es/handle/10251/38864

Fujita, H., Tuba, M., Sasaki, J., WSEAS (Organization), International Conference on Applied Computer Science (13th : 2013 : Morioka City, J., & International Conference on Digital Services, I. and A. (2nd : 2013 : M. C. (2013). *Recent advances in applied computer science and digital services : proceedings of the 13th international conference on applied computer science (ACS '13), proceedings of the 2nd international conference on digital services, internet and applications (DSIA*. 171.

Grau, X. F., & Segura, M. I. S. (2013). *Desarrollo Orientado a Objetos con UML*.

Harris, I. S., & Ahmed, Z. (2011). An open multi-tenant architecture to leverage SMEs. *European Journal of Scientific Research*, *65*(4), 601–610.

Hayes, B. (2008). Cloud Computing. *Communications of the ACM*, *51*(7), 9–11. https://doi.org/10.1145/1364782.1364786

Henriquez, C., Del Vecchio, J. F., & Paternina, F. J. (2015). La computación en la nube: un modelo para el desarrollo de las empresas. *Prospectiva*, *13*(2), 81. https://doi.org/10.15665/rp.v13i2.490

INEI. (2018). *Comunicación perú : en las Empresas , 2017*.

Juan Francisco Sánchez. (2018). *Pruebas de rendimiento con JMeter. Ejemplos básicos | SDOS*. https://www.sdos.es/blog/pruebas-de-rendimiento-con-jmeter-ejemplos-basicos

Kendall, K. E. (2011). *Análisis y Diseño de Sistemas*.

Khalid, A. (2010). Cloud computing: Applying issues in small. *2010 International Conference on Signal Acquisition and Processing, ICSAP 2010*, 278–281. https://doi.org/10.1109/ICSAP.2010.78

Kim, S. H., Jang, S. Y., & Yang, K. H. (2017). Analysis of the Determinants of Software-as-a-Service Adoption in Small Businesses: Risks, Benefits, and Organizational and Environmental Factors. *Journal of Small Business Management*, *55*(2), 303–325. https://doi.org/10.1111/jsbm.12304

Kiran, S., Mohapatra, A., & Swamy, R. (2015). Experiences in performance testing of web applications with Unified Authentication platform using Jmeter. *2nd International Symposium on Technology Management and Emerging Technologies, ISTMET 2015 - Proceeding*, 74–78. https://doi.org/10.1109/ISTMET.2015.7359004

Kouki, Y., & Ledoux, T. (2013). SCAling: SLA-driven cloud auto-scaling. *Proceedings of the ACM Symposium on Applied Computing*, 411–412. https://doi.org/10.1145/2480362.2480445

La, H. J., & Kim, S. D. (2009). A systematic process for developing high quality SaaS cloud services. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 5931 LNCS*, 278–289. https://doi.org/10.1007/978-3-642-10665-1_25

Lee, W., & Choi, M. (2012). A multi-tenant web application framework for SaaS. *Proceedings - 2012 IEEE 5th International Conference on Cloud Computing, CLOUD 2012*, 970–971. https://doi.org/10.1109/CLOUD.2012.27

Li, J., Xiong, Y., Liu, X., & Zhang, L. (2013). How does web service API evolution affect clients? *Proceedings - IEEE 20th International Conference on Web Services, ICWS 2013, 61121063*, 300–307. https://doi.org/10.1109/ICWS.2013.48

López, M., Reynoso, P.-, & Velzi, B.-. (2012). *Ventajas y Desventajas sobre Cloud Computing para las PyMEs en Argentina*. http://sistemas.frba.utn.edu.ar/grupogemis/Trabajos/Otros/101 articulo.pdf

Luna Pérez, M. A., Orantes Jiménez, S. D., & Vázquez Álvarez, G. (2018). Desarrollo de un prototipo CRM SaaS para PyMES dirigido al sector restaurantero para la gestión de clientes. *CISCI 2018 - Decima Septima Conferencia Iberoamericana En Sistemas, Cibernetica e Informatica, Decimo Quinto Simposium Iberoamericano En Educacion, Cibernetica e Informatica, SIECI 2018 - Memorias*, *1*, 80–85.

Madisha, M., & Van Belle, J. P. (2011). Factors Influencing SaaS Adoption by Small South African Organizations. *In 11th Annual Conference on World Wide Web Applications. Durban, South Africa.*, 1–14.

Manifiesto por el Desarrollo Ágil de Software. (n.d.). Retrieved August 26, 2021, from https://agilemanifesto.org/iso/es/manifesto.html

María, J., & Haro, G. (2008). *Diseño e implementación de un marco de trabajo (framework) de presentación para aplicaciones JEE Resumen*. 1–173.

Mateu, C. (2004). *Software libre Desarrollo de aplicación web*. http://roa.ult.edu.cu/bitstream/123456789/450/1/Desarrollo Aplicaciones Web.pdf

Menzinsky, A., Gertrudis López, J. P., Iubaris, López, G., & Palacio, J. (2016). *Scrum Manager*.

Nenartovič, T. (2017). Doctoral thesis. In *Imago Mundi* (Vol. 69, Issue 1). https://doi.org/10.1080/03085694.2016.1242865

Nieto, E. (2013). *Diseño de aplicaciones SaaS sobre plataformas de Cloud Computing*. 1–111. http://sedici.unlp.edu.ar/handle/10915/46834

Pattabhiraman, P., Bai, X., & Tsai, T. (2011). SaaS Performance and Scalability in Clouds. *Proceedings of 2011 IEEE 6th International Symposium on Service Oriented System (SOSE), Sose*, 61–71.

Pressman, R., & Troya, J. (2005). *Ingeniería del software*. MCGRAW-HILL.

Rodríguez, P. (2008). Estudio de la aplicación de metodologías ágiles para la evolución de productos software. *Metodologias Agiles Para La Evoluciòn de Productos de Software*, *1*, 146.

Shaikh, F., & Patil, D. (2014). Multi-tenant e-commerce based on SaaS model to minimize IT cost. *2014 International Conference on Advances in Engineering and Technology Research, ICAETR 2014*, 2–5. https://doi.org/10.1109/ICAETR.2014.7012861

Sicilia, M. (2009). *Estándar ISO 9126 del IEEE y la Mantenibilidad - OpenStax CNX*. 2–3. http://cnx.org/contents/3d263044-60f5-4eda-a117-23660ce72819@3/Estndar-ISO-9126-del-IEEE-y-la

Software Apache. (n.d.). *Apache JMeter™*. https://jmeter.apache.org/

Tang, K., Zhang, J. M., & Jiang, Z. B. (2010). Framework for SaaS management platform. *Proceedings - IEEE International Conference on E-Business Engineering, ICEBE 2010*, 345–350. https://doi.org/10.1109/ICEBE.2010.79

Tsai, W. T., Huang, Y., & Shao, Q. (2011). EasySaaS: A SaaS development framework. *Proceedings - 2011 IEEE International Conference on Service-Oriented Computing and Applications, SOCA 2011*. https://doi.org/10.1109/SOCA.2011.6166262

Tsai, W. T., Sun, X., & Balasooriya, J. (2010). Service-oriented cloud computing architecture. *ITNG2010 - 7th International Conference on Information Technology: New Generations*, 684–689. https://doi.org/10.1109/ITNG.2010.214

Vidhyalakshmi, R., & Kumar, V. (2017). CORE framework for evaluating the reliability of SaaS products. *Future Generation Computer Systems*, *72*, 23–36. https://doi.org/10.1016/j.future.2017.02.039

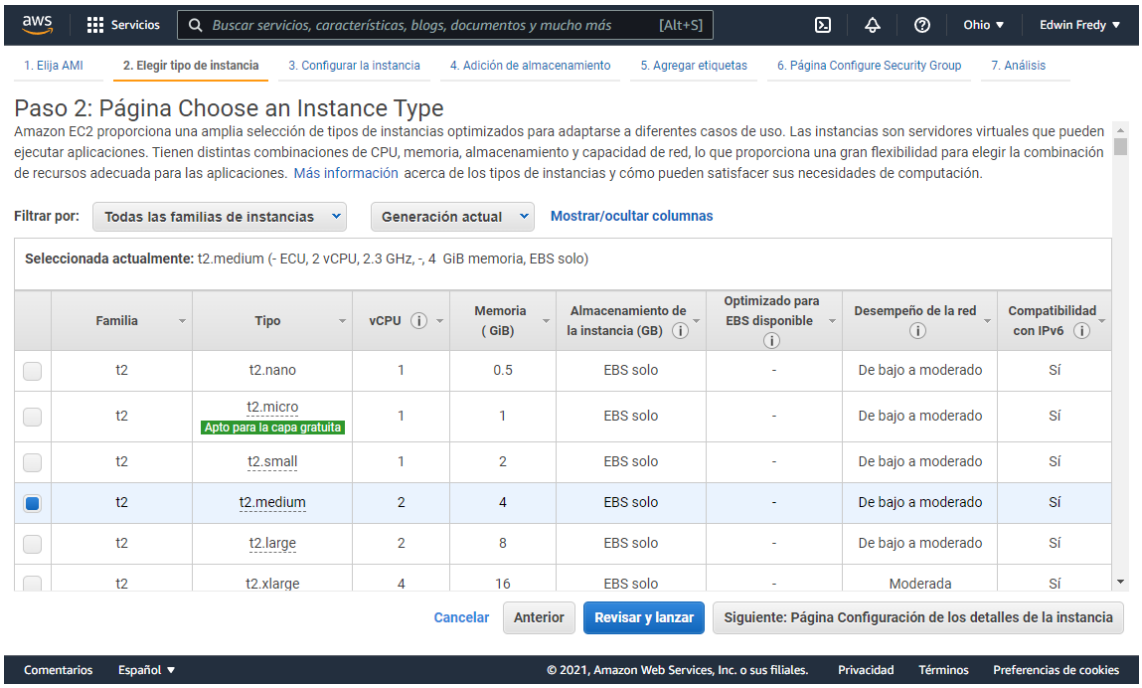
Wang, B., Huang, H. Y., Liu, X. X., & Xu, J. M. (2009). Open identity management framework for SaaS ecosystem. *Proceedings - IEEE International Conference on e-Business Engineering, ICEBE 2009; IEEE Int. Workshops - AiR 2009; SOAIC 2009; SOKMBI 2009; ASOC 2009*, 512–517. https://doi.org/10.1109/ICEBE.2009.82

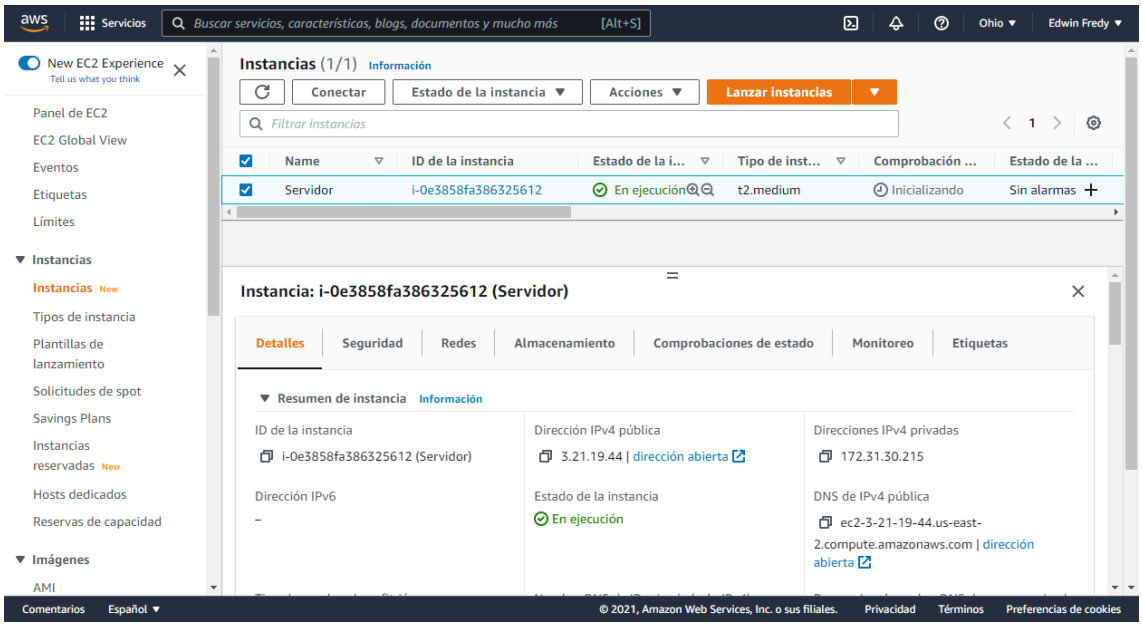
Xiao, S., & Cheng, G. (2010). Application research of CRM based on SaaS. *Proceedings of the International Conference on E-Business and E-Government, ICEE 2010*, 3090–3092. https://doi.org/10.1109/ICEE.2010.779

Zhang, L. J., & Zhou, Q. (2009). CCOA: Cloud Computing Open Architecture. *2009 IEEE International Conference on Web Services, ICWS 2009*, 607–616. https://doi.org/10.1109/ICWS.2009.144

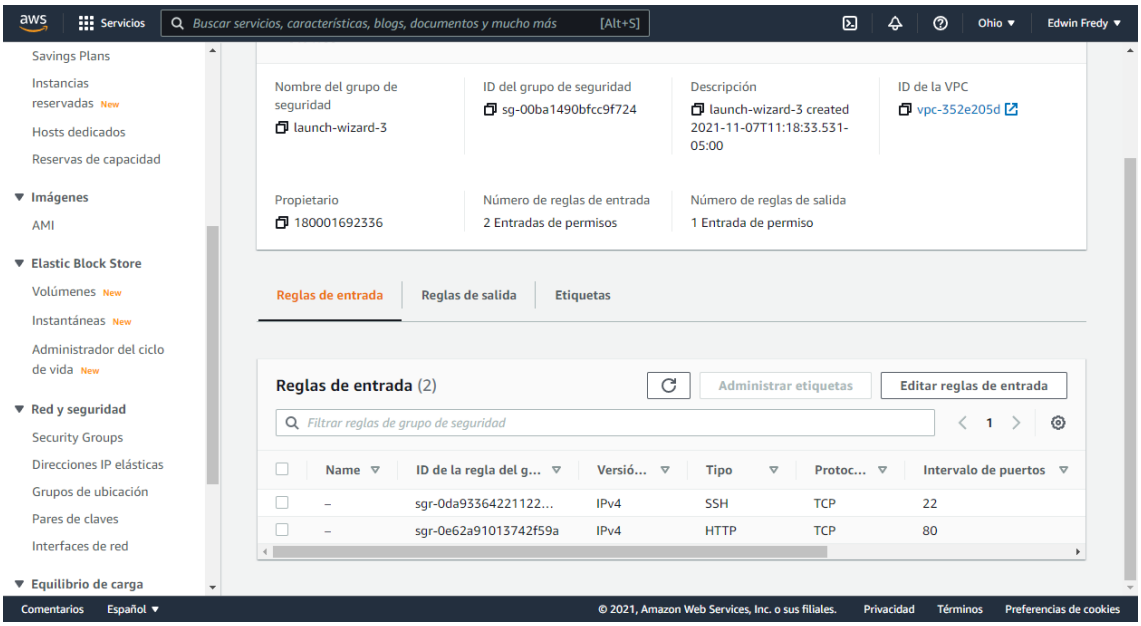
ANEXOS

Anexo 1 Configuración de la instancia computacional

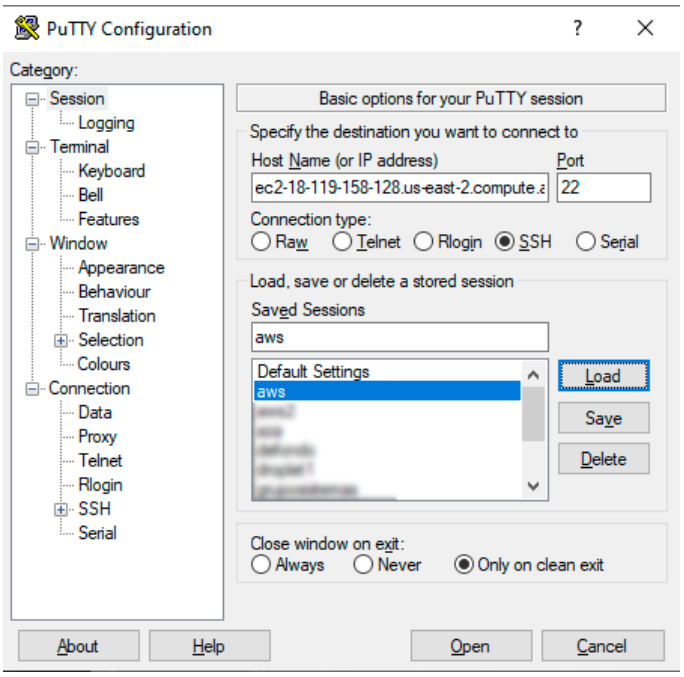


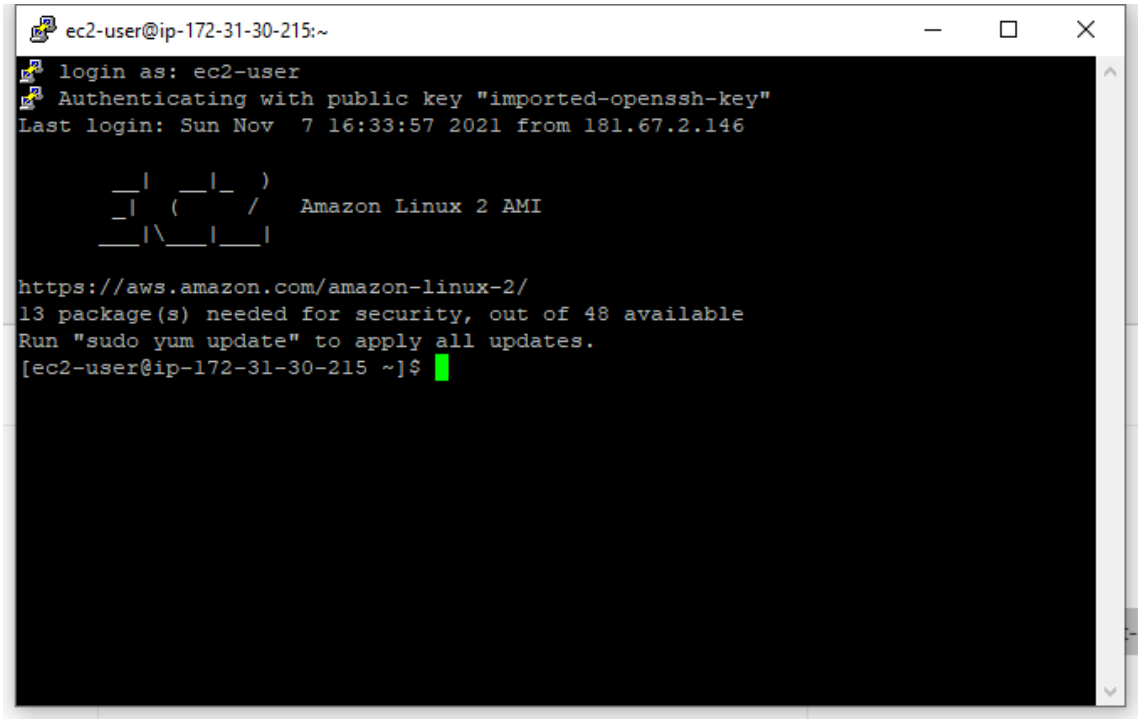


Anexo 2 Reglas de acceso

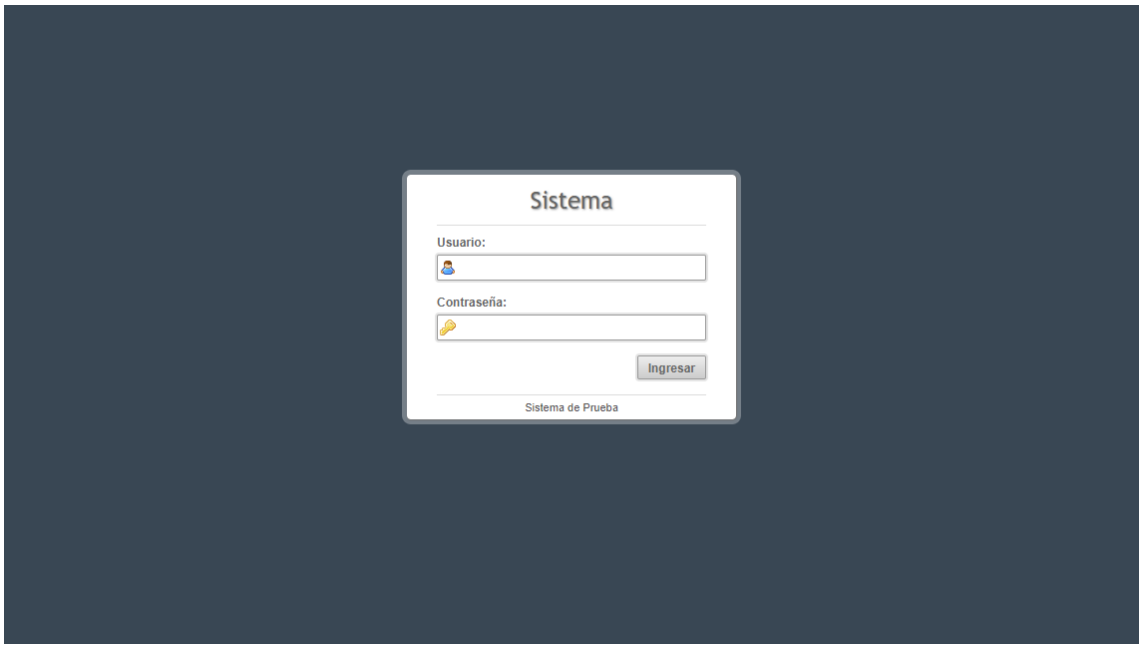


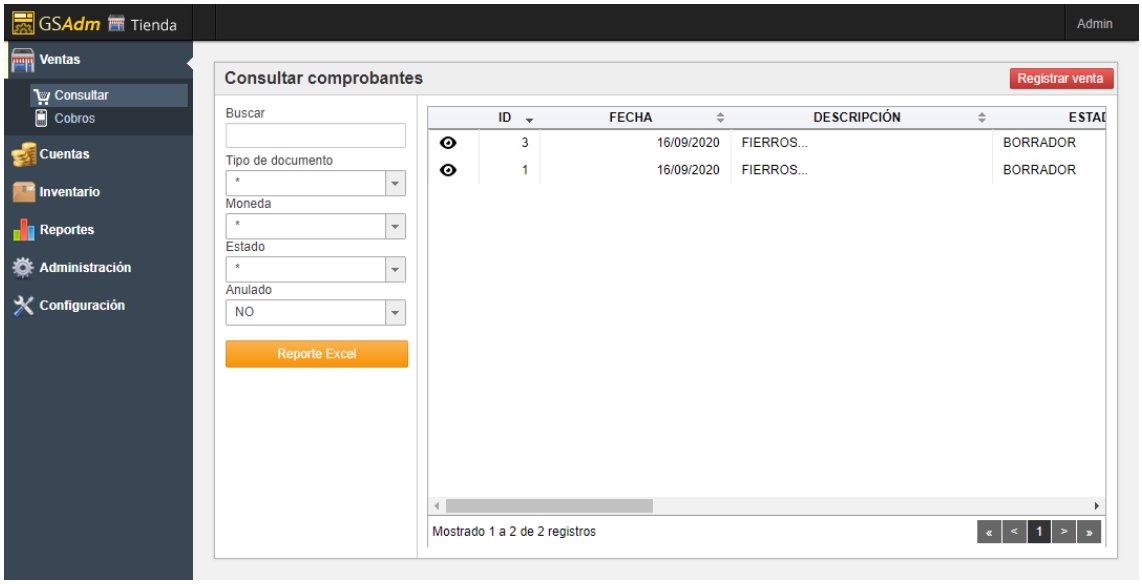
Anexo 3 Acceso vía puerto seguro



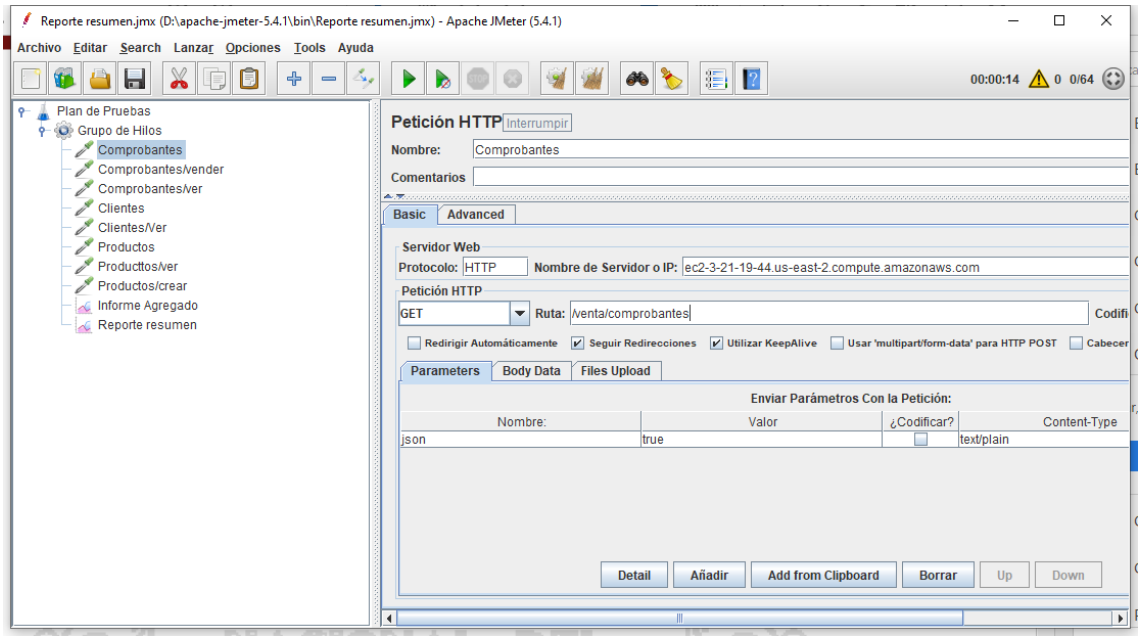


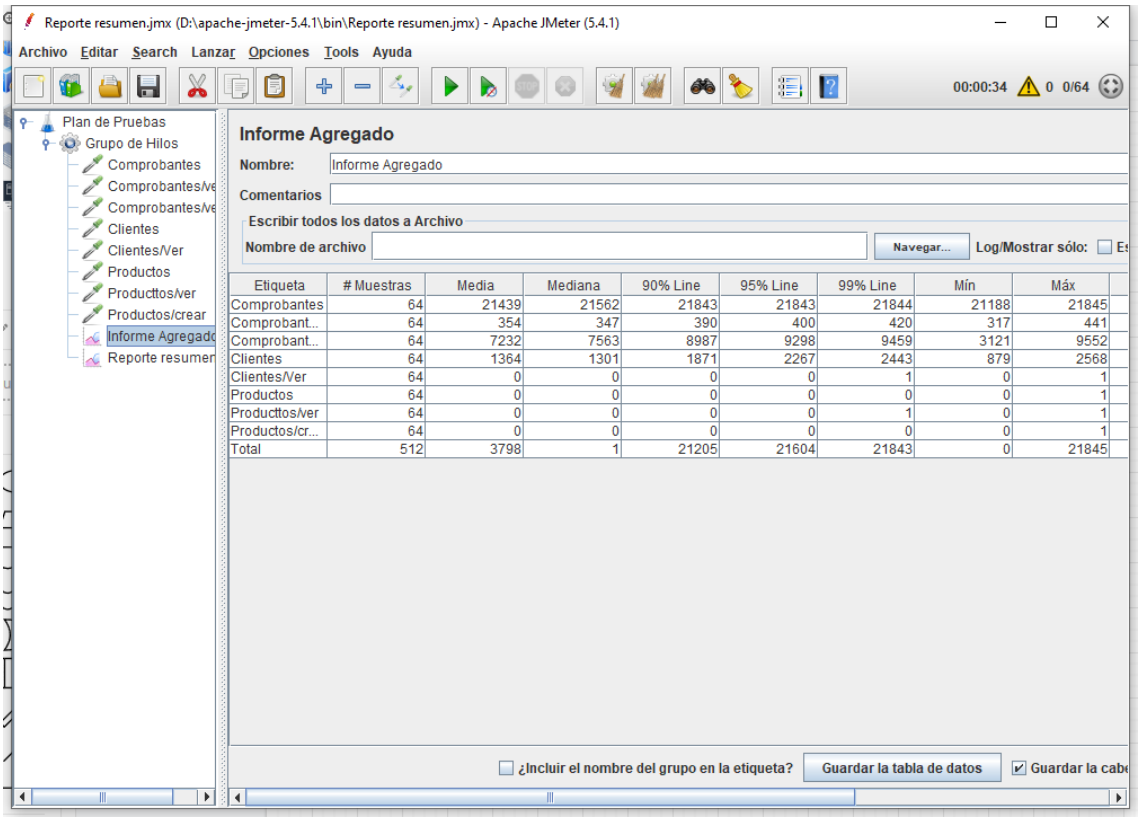
Anexo 4 Interfaz de la aplicación de prueba





Anexo 5 Configuración de pruebas de carga





Anexo 6 Código fuente del API

```php
<?php
require 'aws.phar';

use Aws\S3\Exception\S3Exception;

class Cloud
{
    var $sdk;

    function __construct()
    {
        $this->sdk = new Aws\Sdk([
            'region'   => 'us-east-2',
            'version'  => 'latest',
            'credentials' => array(
                'key'    => 'XXXXXXXXXXXXXXXXXXXX',
                'secret' => 'XXXXXXXXXXXXXXXXXXXXXXXXXXXX',
            ),
            'http'    => [
                'verify' => \cacert.pem"
            ]
        ]);
    }

    function s3_newBucket($name)
    {
        $s3Client = $this->sdk->createS3();

        try {
            $s3Client->createBucket(['Bucket' => $name]);
        } catch (S3Exception $e) {
            echo $e->getMessage();
        } catch (AwsException $e) {
            echo $e->getAwsRequestId() . "\n";
            echo $e->getAwsErrorType() . "\n";
            echo $e->getAwsErrorCode() . "\n";
            var_dump($e->toArray());
        }
    }

    function s3_newFolder($bucket, $key)
    {
        try {
            $s3Client = $this->sdk->createS3();
            $result = $s3Client->putObject([
                'Bucket' => $bucket,
```

```
                'Key' => $key . '/',
                'Body'   => "",
                'ACL'    => 'public-read'
            ]);
        } catch (S3Exception $e) {
            echo $e->getMessage() . "\n";
        }
    }

    function s3_newFile($bucket, $key, $file_Path)
    {
        try {
            $s3Client = $this->sdk->createS3();
            $result = $s3Client->putObject([
                'Bucket' => $bucket,
                'Key' => $key,
                'SourceFile' => $file_Path,
                'ACL'        => 'public-read'
            ]);
        } catch (S3Exception $e) {
            echo $e->getMessage() . "\n";
        }
    }

    function s3_delBucket($bucket, $key)
    {
        try {
            $s3Client = $this->sdk->createS3();
            $result = $s3Client->deleteObject([
                'Bucket' => $bucket,
                'Key' => $key,
            ]);
        } catch (S3Exception $e) {
            echo $e->getMessage() . "\n";
        }
    }

    function s3_delFolder($bucket, $key)
    {
        $s3Client = $this->sdk->createS3();
        $result = $s3Client->deleteObject([
            'Bucket' => $bucket,
            'Key' => $key . '/',
        ]);
    }

    function s3_delFile($bucket, $key)
    {
        $s3Client = $this->sdk->createS3();
```

```
        $result = $s3Client->deleteObject([
            'Bucket' => $bucket,
            'Key' => $key,
        ]);
    }

    function s3_listBuckets()
    {
        $s3Client = $this->sdk->createS3();
        $buckets = $s3Client->listBuckets();
        $names = [];

        foreach ($buckets['Buckets'] as $bucket) {
            $names[] = $bucket['Name'];
        }
        return $names;
    }

    function s3_listFiles($bucket, $path = '')
    {
        $s3Client = $this->sdk->createS3();
        $results = $s3Client->getPaginator('ListObjects', [
            'Bucket' => $bucket,
            'Prefix' => $path
        ]);
        $names = array();
        foreach ($results as $result) {
            foreach ($result['Contents'] as $object) {
                $names[] = $object['Key'];
            }
        }

        return $names;
    }

    function s3_zipBucket($bucket, $pathzip, $path = '')
    {
        $zip = new ZipArchive;
        $zip->open($pathzip, ZipArchive::CREATE);

        $s3Client = $this->sdk->createS3();
        $objects = $s3Client->getPaginator('ListObjects', array(
            'Bucket' => $bucket,
            'Prefix' => $path
        ));
        foreach ($objects as $result) {
            foreach ($result['Contents'] as $object) {
                $url = $s3Client->getObjectUrl($bucket, $object['Key']);
                $contents = file_get_contents($url);
```

```
                $zip->addFromString($object['Key'], $contents);
            }
        }

        $zip->close();
    }

    function s3_download($bucket, $keyname)
    {
        try {
            $result = $s3->getObject([
                'Bucket' => $bucket,
                'Key'    => $keyname
            ]);

            header("Content-Type: {$result['ContentType']}");
            echo $result['Body'];
        } catch (S3Exception $e) {
            echo $e->getMessage() . PHP_EOL;
        }
    }

    function ec2_newInstance()
    {
        $ec2Client = $this->sdk->createEc2();
        $result = $ec2Client->runInstances(array(
            'ImageId'        => 'ami-02deb1028760f2f3a',
            'MinCount'       => 1,
            'MaxCount'       => 1,
            'InstanceType'   => 't2.micro',
            'KeyName'        => 'mcedwin',
            'SecurityGroups' => array('launch-wizard-2'),
        ));
    }

    function ec2_listInstances()
    {
        $ec2Client = $this->sdk->createEc2();
        $result = $ec2Client->describeInstances();
        $minstances = [];
        $reservations = $result['Reservations'];
        foreach ($reservations as $reservation) {
            $instances = $reservation['Instances'];
            foreach ($instances as $instance) {
               // var_dump($instance);
                $instanceName = '';
                if(isset($instance['Tags']))
                foreach ($instance['Tags'] as $tag) {
                    if ($tag['Key'] == 'Name') {
```

```
                        $instanceName = $tag['Value'];
                    }
                }

                                                                $minstances[]       =
['name'=>$instanceName,'state'=>$instance['State']['Name'],'public'=>$instance['Pub
licDnsName']];
            }
        }
        return $minstances;
    }

    function ec2_stopInstance()
    {
        $result = $ec2Client->stopInstances(array(
            'InstanceIds' => $instanceIds,
        ));
    }

    function ec2_startInstance()
    {
        $result = $ec2Client->startInstances(array(
            'InstanceIds' => $instanceIds,
        ));
    }

    function ec2_restartInstance()
    {
        $result = $ec2Client->rebootInstances(array(
            'InstanceIds' => $instanceIds
        ));
    }
```