UNIVERSIDAD TÉCNICA NACIONAL
Sede Central
Carrera de Ingeniería del Software

PROYECTO DE GRADUACIÓN

Sometida a consideración del Tribunal Examinador para optar por el grado de Licenciatura en Ingeniería del Software

ANÁLISIS DEL MODELO DE SOFTWARE COMO SERVICIO ORIENTADO A LA MICRO Y PEQUEÑA EMPRESA COSTARRICENSE DE DESARROLLO DE SOFTWARE, UTILIZANDO UN ENTORNO VIRTUAL EN LA NUBE

AUTOR: Dilan Alberto Altamirano Cerda

ALAJUELA, COSTA RICA
AGOSTO, 2022

TRIBUNAL EXAMINADOR

ANA CECILIA ODIO UGALDE (FIRMA)

Firmado digitalmente por ANA CECILIA ODIO UGALDE (FIRMA)
Fecha: 2022.08.19 18:43:05 -06'00'

MGt. Ana Cecilia Odio Ugalde
Directora de Carrera

KARLINNA VANESSA CHAVES GONZALEZ (FIRMA)

Digitally signed by KARLINNA VANESSA CHAVES GONZALEZ (FIRMA)
Date: 2022.08.22 09:34:50 -06'00'

MGt. Karlinna Chaves González
Tutora

WALTER SEQUEIRA PORRAS (FIRMA)

Firmado digitalmente por WALTER SEQUEIRA PORRAS (FIRMA)
Fecha: 2022.08.20 12:33:44 -06'00'

Lic. Walter Sequeira Porras
Lector

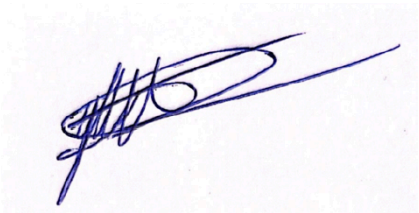
EDWIN ALONSO LOPEZ PANIAGUA (FIRMA)

Firmado digitalmente por EDWIN ALONSO LOPEZ PANIAGUA (FIRMA)
Fecha: 2022.08.19 19:54:44 -06'00'

Lic. Edwin López Paniagua
Lector

DECLARACIÓN JURADA

Yo, **Dilan Alberto Altamirano Cerda**, mayor, casado, estudiante de la carrera de Ingeniería del *Software* de la Universidad Técnica Nacional, domiciliado en Calle Santa Fe, Alajuela, portador de la cédula de identidad número 155802398030, en este acto, debidamente apercibido y entendido de las penas y consecuencias con las que se castiga, en el Código Penal, el delito de perjurio, ante quienes se constituyen en el Tribunal Examinador de mi trabajo de Tesis para optar por el título de licenciatura en Ingeniería del *Software*, juro solemnemente que mi trabajo de investigación titulado: **"ANÁLISIS DEL MODELO DE *SOFTWARE* COMO SERVICIO ORIENTADO A LA MICRO Y PEQUEÑA EMPRESA COSTARRICENSE DE DESARROLLO DE *SOFTWARE*, UTILIZANDO UN ENTORNO VIRTUAL EN LA NUBE"**, es una obra original que ha respetado todo lo preceptuado por las Leyes Penales así como la Ley de Derechos de Autor y Derechos Conexos, número 6683 de 14 de octubre de 1982 y sus reformas, publicada en La Gaceta número 226 de 25 de noviembre de 1982; incluyendo el numeral 70 de dicha ley que advierte: "Artículo 70º: Es permitido citar a un autor transcribiendo los pasajes pertinentes siempre que estos no sean tantos y seguidos, que puedan considerarse como una producción simulada y sustancial, que redunde en perjuicio del autor y de la obra original". Asimismo, estoy advertido que la Universidad Técnica Nacional se reserva el derecho de protocolizar este documento ante Notario Público. En fe de lo anterior, firmo en la ciudad de Alajuela, el día tres del mes de agosto del año dos mil veintidós.



Dilan Alberto Altamirano Cerda
Cédula de identidad: 155802398030

DEDICATORIA

Este trabajo lo dedico primeramente a mi Dios, de quien proviene mi sabiduría y fuerza para afrontar retos de este calibre.

A mi esposa Berlay Corella, cuyo apoyo incondicional me ha dado la valentía para seguir adelante y, finalmente, a mis padres: Ronald Rojas y Rosa Cerda, quienes me han impulsado a través de toda mi formación académica, por su sacrificio y honradez que han formado mis valores y principios para ser un profesional integral.

AGRADECIMIENTOS

A todas aquellas personas que me apoyaron, directa o indirectamente, para que este trabajo se realice de forma satisfactoria. En especial a los docentes, quienes me guiaron durante este proceso; a mi profesora tutora por su ayuda, paciencia y dedicación. Gracias por sus aportes y consejos.

A toda mi familia, por creer siempre en mí y haberme acompañado durante estos largos años de estudio.

CARTA DE AUTORIZACIÓN DEL TUTOR

Alajuela, 09 de agosto de 2022

Sra.
Mgtr. Ana Cecilia Odio Ugalde
Directora de la Carrera de Ingeniería del *Software*
Universidad Técnica Nacional

Estimada señora Directora:

El estudiante **Dilan Alberto Altamirano Cerda**, portador de la cédula de identidad n.º 155802398030, ha presentado para revisión el Proyecto de Graduación denominado: **"ANÁLISIS DEL MODELO DE *SOFTWARE* COMO SERVICIO ORIENTADO A LA MICRO Y PEQUEÑA EMPRESA COSTARRICENSE DE DESARROLLO DE *SOFTWARE*, UTILIZANDO UN ENTORNO VIRTUAL EN LA NUBE"**. En calidad de Tutor, se ha revisado y corregido todos los aspectos referentes a este documento. Por lo tanto, se hace constar que se encuentra listo para ser presentado a la Universidad Técnica Nacional como trabajo de graduación.

Atentamente,

KARLINNA VANESSA CHAVES GONZALEZ (FIRMA)

Digitally signed by KARLINNA VANESSA CHAVES GONZALEZ (FIRMA)
Date: 2022.08.09 09:44:20 -06'00'

Mgtr. Karlinna Chaves González
Tutora

CARTA DE AUTORIZACIÓN DEL LECTOR

Alajuela, 03 de agosto de 2022

Sra.

Mgt. Ana Cecilia Odio Ugalde

Director de la Carrera de Ingeniería del Software

Universidad Técnica Nacional

Estimada señora Directora:

Sirva la presente para saludarle y hacer de su conocimiento mi aprobación, en calidad de lector, del Proyecto de Graduación realizado por el estudiante **Dilan Alberto Altamirano Cerda**, portador de la Cédula de Identidad No. 155802398030, titulado: **"ANÁLISIS DEL MODELO DE SOFTWARE COMO SERVICIO ORIENTADO A LA MICRO Y PEQUEÑA EMPRESA COSTARRICENSE DE DESARROLLO DE SOFTWARE, UTILIZANDO UN ENTORNO VIRTUAL EN LA NUBE."**

Hago constar que se ha revisado y corregido todos los aspectos referentes a este documento; por lo que manifiesto que el mismo se encuentra listo para ser presentado a la Universidad Técnica Nacional, como trabajo de graduación.

Atentamente,

WALTER SEQUEIRA PORRAS (FIRMA)

Firmado digitalmente por WALTER SEQUEIRA PORRAS (FIRMA)
Fecha: 2022.08.03 16:21:04 -06'00'

Walter Sequeira Porras

Lector

CARTA DE AUTORIZACIÓN DEL LECTOR

Alajuela, 04 de agosto de 2022

Sra.
Mgt. Ana Cecilia Odio Ugalde
Directora de la Carrera de Ingeniería del Software
Universidad Técnica Nacional

Sirva la presente para saludarle y hacer de su conocimiento mi aprobación, en calidad de lector, del Proyecto de Graduación realizado por el estudiante **Dilan Alberto Altamirano Cerda**, portador de la Cédula de Identidad No. 155802398030, titulado: **"ANÁLISIS DEL MODELO DE SOFTWARE COMO SERVICIO ORIENTADO A LA MICRO Y PEQUEÑA EMPRESA COSTARRICENSE DE DESARROLLO DE SOFTWARE, UTILIZANDO UN ENTORNO VIRTUAL EN LA NUBE."**

Hago constar que se ha revisado y corregido todos los aspectos referentes a este documento; por lo que manifiesto que el mismo se encuentra listo para ser presentado a la Universidad Técnica Nacional, como trabajo de graduación.

Atentamente,

EDWIN ALONSO LOPEZ PANIAGUA (FIRMA)

Firmado digitalmente por EDWIN ALONSO LOPEZ PANIAGUA (FIRMA)
Fecha: 2022.08.04 09:58:46 -06'00'

Edwin Alonso Lopez Paniagua
Lector

CARTA DE REVISIÓN FILOLÓGICA

San José, 8 de agosto de 2022

Sra.
Mgtr. Ana Cecilia Odio Ugalde
Directora de la Carrera de Ingeniería del *Software*
Universidad Técnica Nacional

Estimada señora Directora:

Sirva la presente para saludarle y expresar que en mi calidad de Licenciada en Filología Española, he revisado la redacción, ortografía y estilo literario del Proyecto de Graduación titulado: **"ANÁLISIS DEL MODELO DE *SOFTWARE* COMO SERVICIO ORIENTADO A LA MICRO Y PEQUEÑA EMPRESA COSTARRICENSE DE DESARROLLO DE *SOFTWARE*, UTILIZANDO UN ENTORNO VIRTUAL EN LA NUBE"**, realizado por el estudiante **Dilan Alberto Altamirano Cerda**, para optar por el grado académico de licenciatura en **Ingeniería del *Software*** de la Universidad Técnica Nacional. Por lo que se puede dar fe del correcto español que este contiene.

MARIA DE LOS ANGELES BONILLA SEQUEIRA (FIRMA)
Firmado digitalmente por MARIA DE LOS ANGELES BONILLA SEQUEIRA (FIRMA)
Fecha: 2022.08.08 11:57:03 -06'00'

_____
María Bonilla Sequeira
Carné ACFIL n.º 0009
Filóloga y Correctora de Estilo

TABLA DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN EJECUTIVO

Dada la explosión en popularidad de las tecnologías de computación en la nube, se comenzó a analizar sus aplicaciones en varios ámbitos comerciales para el beneficio de múltiples actores (proveedores y consumidores). Es por ello por lo que uno de los objetivos de este proyecto de investigación se enfoca en el estudio de las capacidades que poseen las micro y pequeñas empresas (PYME) de desarrollo de *software* del distrito económico central de Alajuela, para aplicar modelos de computación de la nube, como el de *Software* como Servicio.

Lo anterior, conlleva la consideración fundamental de revisar si este tipo de organizaciones puede o no implementar productos de *software* de esta naturaleza, para así determinar y establecer un conjunto estándar de procedimientos y buenas prácticas que sean utilizados para finiquitar dicho objetivo. A su vez, esto permite a varios emprendimientos proveer un servicio de calidad y robusto que pueda ser entregado a un sinnúmero de clientes del área, así como a clientes en otras regiones de Costa Rica y fuera del país.

Por consiguiente, como primer paso, se procede a revisar la teoría de los fundamentos que comprenden la computación en la nube: su historia, conceptos clave, definiciones, relaciones con otros modelos de entrega, como: Infraestructura como Servicio y Plataforma como Servicio, etc. Dentro de estas definiciones se encuentra una estructura marcada que considera el uso mixto de varios modelos en productos de *software*. Por ejemplo, aplicaciones de *Software* como Servicios actualmente están desplegadas en ambientes de Infraestructura como Servicio, que ponen a disposición del anfitrión múltiples servicios que simplifican la complejidad de la creación de sistemas de información. Por otro lado, existen otros que apoyan el ciclo de vida de un producto de *software* y, a su vez, mejoran la productividad de los equipos de desarrollo para el

mantenimiento de la aplicación, su infraestructura y base de código. Esto genera un ambiente de colaboración óptimo para dar espacio a la innovación y resolución de problemas de reglas de negocio.

En segundo lugar, se busca conceptualizar las aplicaciones de aspectos de seguridad informática, integración, entrega continua y gestión de la arquitectura del *software*. Estas áreas son importantes porque son esenciales, es decir, cualquier producto en la actualidad se espera contenga estos aspectos en su estructura de funcionamiento y, además, completan el esquema del ambiente ideal para proyectos de *software* que contengan un alcance de múltiples usuarios con aspiraciones de escalar a millones.

De esta forma, también se ejecuta una recolección de información relacionada con las micro y pequeñas empresas (PYME) de desarrollo de *software* del distrito económico central de Alajuela, las cuales ejercen sus funciones desde ámbitos como: la administración de proyectos, de personal y aspectos técnicos como la infraestructura y ciclo de desarrollo de *software*. La población en estudio está asociada al Ministerio de Economía, Industria y Comercio (MEIC), por lo cual el listado de organizaciones también se corroboró con la información del Ministerio de Hacienda de Costa Rica. Parte de este trabajo se realiza por medio de un cuestionario que se aplicó a los colaboradores del sector administrativo y de tecnologías de la información de cada empresa participante.

Asimismo, se encuentra varios puntos a favor de los sujetos en estudio, entre ellos la madurez y confianza de los procesos operativos que posee. Algunos de estos cuentan con un apoyo mayoritario por parte de los colaboradores, como la implementación de la computación en la nube, la gestión de proyectos, uso de arquitecturas definidas y prácticas de desarrollo seguro. Mientras que los menos apoyados, y en los cuales este tipo de empresas tienen más deficiencias,

fueron la implementación completa de soluciones de integración y entrega continua, monitoreo sintético de aplicaciones, seguridad informática y gestión de la documentación.

Por consiguiente, lo que se busca es integrar estas áreas dentro del alcance del proyecto para eventualmente crear unas pautas que indiquen el flujo de procesos y tareas que debe tener una micro y pequeña empresa con la finalidad de generar un producto de *Software* como Servicio. Estos procesos son adaptaciones de información ya existente que se ejecuta en compañías más grandes (tanto en capital como en fuerza de trabajo).

Es por ello por lo que, para finalizar el trabajo de investigación, se centra en una guía ligera de sugerencias y buenas prácticas centradas principalmente en los siguientes puntos: en primer lugar, consideraciones generales, comprende todo aquello relacionado directa o indirectamente con la administración de proyectos, así como con recursos humanos, incluyendo los puestos de trabajo que son necesarios para contribuir con un ambiente de micro y pequeña empresa. En segundo lugar, la gestión de la infraestructura, como parte de la gobernanza que debe existir en los procesos de administración de recursos de la nube para soportar la aplicación o aplicaciones propietarias de la organización. Y, en tercer lugar, la gestión de la creación del producto de *software* y su operación diaria, incluyendo aspectos de arquitectura y prácticas para el diario soporte de las operaciones de tecnologías de la información para que el *software* esté en su máximo rendimiento.

CAPÍTULO I. DELIMITACIÓN DE LA INVESTIGACIÓN

1.1 Estado del Arte

La industria del *software* en el ámbito nacional e internacional ha evolucionado a pasos agigantados, y durante la historia se han visto grandes avances en lo que se produce tanto a nivel científico como comercial.

Como se aprecia en el estudio de Briggs y Kassner (2017), llamado *Estrategia Empresarial de Computación en la Nube*, publicado por el *Microsoft Press*, sobre los productos de *software,* se indica que:

> Propician una perspectiva muy marcada en el ámbito histórico de las Tecnologías de la Información (TIC) en un ambiente empresarial, en cuanto a "un antes y un después" de la computación en la nube, su uso y la manera en que las organizaciones comienzan su viaje utilizando equipos de cómputo, *software* de terceros y la capacidad de operarlos con sus propios medios hasta el nacimiento y adopción de la computación en la nube a inicios del siglo XX. (pp. 13–15)

Es conveniente mencionar que la computación en la nube es consecuencia de la necesidad del ser humano de compartir información, que con el nacimiento de Internet empezó a gestarse y a proyectarse hacia un futuro prometedor.

En la misma línea, se aprecia una serie de eventos cronológicos en un artículo publicado en 2018 por la *Revista de Ciencias Fundamentales y Aplicadas*, llamado "Computación en la nube: una nueva era":

> Dentro de los conceptos ahí acuñados, sobresale el término "tiempo compartido", práctica común por todos aquellos usuarios que no podían comprar y poseer una computadora central (*Mainframe* por su vocablo en inglés) para utilizarla en sus propósitos comerciales, científicos y otros, dando un banderazo de salida a lo que hoy se conoce como computación en la nube. (Namasudra, 2018, p. 116)

Por su parte, Kandukuri *et al.* (2009, citados en Namasudra, 2018) ejemplifican que, en 1969, J. C. R. Licklider introdujo la idea de una red intergaláctica de computadoras. Él estaba tratando de conectar todas las computadoras localizadas globalmente. Otros expertos acreditaron la idea del concepto de "la nube" al científico John McCarthy. (p. 116)

Dentro de esta misma línea del tiempo, el mismo artículo relaciona las contribuciones de empresas como IBM, que desde estas épocas trataban de utilizar el poder computacional y de sus recursos para cambiar el modelo de negocios entre *hardware* y *software*. De la misma manera, ejemplifica un caso importante para la industria, como el de *Salesforce*. En 1999, Salesforce.com era uno de los principales promotores de la computación en la nube y su objetivo era entregar aplicaciones vía un sitio web simple. Después, muchas compañías seguirían estos pasos en tecnología para proveer sus servicios. (Namasudra, 2018, p. 116)

Eventualmente, esto impulsó a muchos profesionales en el área a replantearse variados conceptos y, principalmente, la concepción de conocimiento como consecuencia de la aplicación de estas tecnologías emergentes en todo ámbito.

Tal es el caso que analiza Nayan Ruparelia, en su artículo titulado "Computación en la nube", publicada en el *MIT Press* (*The MIT Press Essential Knowledge Series*), donde:

> Plantea un cambio de paradigma, que lo enfoca desde tres puntos de vista diferentes: el social y personal: que refiere al impacto en el entorno del usuario; el laboral: el cómo afecta el trabajo del usuario, y el empresarial: que se refiere a la afectación de las organizaciones. Esto último, haciendo énfasis en cómo los departamentos de TI han tenido que convertirse en especialistas de compras y abogados de los servicios en la nube, dada su naturaleza de pago sobre el uso del servicio. (Ruparelia, 2016, pp. 53–66)

Por lo expresado, es indispensable apuntar hacia un modelo de entrega de servicio en la computación en la nube. Este aspecto, también, lo cubre Namasudra en su investigación *Computación en la nube: una nueva era*, en el que expone que la computación en la nube está compuesta por tres modelos de servicio, llamados *Software* como Servicio, Plataforma como Servicio e Infraestructura como Servicio. (Namasudra, 2018, p. 117)

Todos estos tienen diferentes tipos de clientes y propósitos varios, que dependerán de las necesidades de cada organización. Al mismo tiempo, este modelo se visualiza en la figura 1, a continuación:

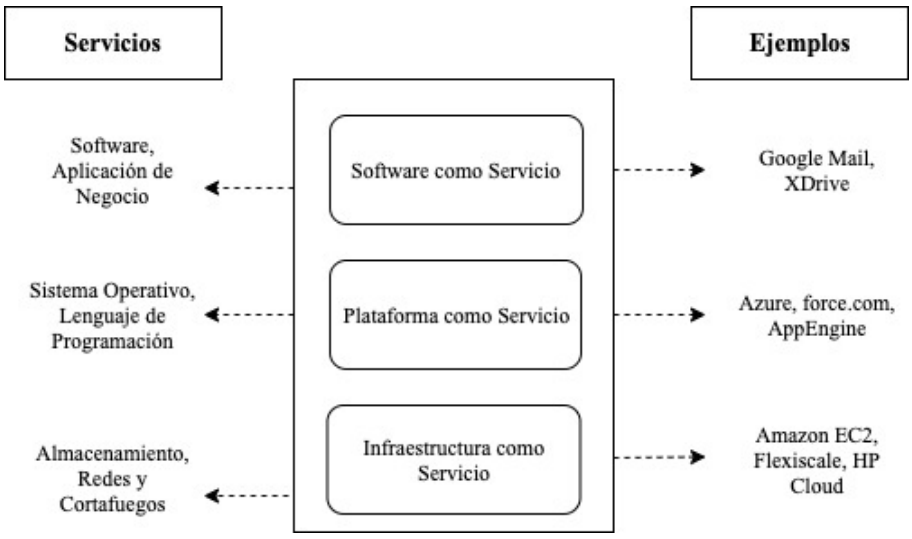


Figura 1.1. Modelo de Servicio de la Computación en la nube. Tomado de: "Cloud Computing: A new Era", elaborado por S. Namasudra, 2018, *Journal of Fundamental and Applied Sciences*, 10(2), p. 118.

Para este apartado, se pretende el enfoque hacia un método de venta y entrega de *software* en forma de servicio, dándole al cliente una oportunidad de subscribirse a este y pagar mensualmente, anualmente, etc. Por consiguiente, al mencionar artículos como el de Bussler (2002), titulado *Software Como Servicio: ASP y ASP Agregación*, que introduce conceptos como "renta" de productos de *software* y "Proveedor de Servicio de Aplicaciones" bajo el modelo de facilitar el uso de programas de tipo empresarial por medio de Internet, usando solamente herramientas como el navegador de los computadores (p. 1), se logra introducir al tema de interés.

La adopción de estos servicios ha crecido e incrementado su popularidad en las últimas dos décadas. Prueba de ello, se analiza en el artículo de *The McKinsey Quarterly*:

> En una revisión de inversiones que realizaron, vieron un incremento en las ganancias de $295 millones en 2002 a $485 millones en 2005, resultando en 18% de incremento. Este estudio se realizó en Estados Unidos y, dados los prometedores resultados, en el mismo artículo se menciona que muchos de los ejecutivos de tecnologías de la información estarían considerando un salto en la entrega de aplicaciones usando el modelo de *software* como servicio. (Dubey y Wagle, 2007)

En este mismo documento se destaca la entrega de *software* como servicio dentro de los competidores fuertes a acaparar en el mercado de tecnologías de información en un futuro cercano. Debido a su sencillo modelo de adopción, pretende lograr gran cantidad de adeptos en

el mundo de los negocios, proveyéndoles de una ayuda tecnológica con valor agregado para sus organizaciones.

Dada esta aceptación, es importante considerar los casos de negocio y utilización, que son generalmente los que se involucran en el pensamiento de las empresas en cuanto al desarrollo de un producto de este tipo y la utilización o beneficio por parte de los clientes.

Según Ruparelia (2016), existen varios ejemplos de casos de uso, como se expresa a continuación:

> Los Sistemas de Planeación de Recursos Empresariales (*ERP or Enterprise Resource Planning,* por sus siglas en inglés), Sistemas de Gestión de Relaciones con el Cliente (*CRM or Customer Relationship Management*, por sus siglas en inglés), cobros y facturación, productividad en la oficina, gestión de activos, correo y mensajería instantánea, herramientas de colaboración, gestor de contenido y otros más. (p. 158)

Estas consideraciones de uso de *Software* como Servicio tienen como característica que resuelven problemas que todas las empresas deben enfrentar durante su operación, ya sea de manera manual o con una solución de *software*, no importa su tamaño.

Por ende, también entra en juego una etapa en la que se podría relacionar los siguientes modelos para la entrega de productos de *software* de esta índole: Infraestructura como Servicio (IaaS) y *Software* como Servicio (SaaS).

Debido a que la naturaleza de la computación en la nube reside en proveer de las herramientas necesarias a los usuarios para construir aplicaciones con la menor complejidad, existe la interrogante de si un modelo depende del otro.

Ahora, estudios en el ámbito nacional, como el de Miguel Alvarado Abarca, publicado en el repositorio de la Universidad Nacional, titulado *Metodología de implementación de aplicaciones de software en Amazon Web Services (AWS)*, presenta una sugerencia de métodos para la puesta en marcha de aplicaciones, en esta ocasión haciendo uso de los servicios del proveedor Amazon. Dicho trabajo se orienta a organizaciones que no cuentan con experiencia previa en este tipo de iniciativas. (Alvarado Abarca, 2017, p. 20)

Pero, además de los proveedores que vieron una oportunidad de negocio, existen empresas que han inclinado su balanza por el uso de *software* bajo la modalidad de suscripción, especialmente por sus ventajas para quien consume el producto; entre ellas se menciona la menor inversión en desarrollo, la configuración y el mantenimiento del producto, siendo estos costos trasladados al proveedor.

Esta posición se alude en el artículo internacional de investigación avanzada de computación bajo el título *Antecedentes del software como servicio (SaaS) adopción: un modelo de ecuación estructural*, donde se señala que: "El cambio hacia aplicaciones basadas en la web para servicios de *software* atrae beneficios salientes para los emprendedores, como la reducción inicial de costos y la facilitación de innovación en el núcleo de sus negocios" (Alotaibi, 2016, p. 128). Lo anterior supone un problema inicial resuelto de forma parcial para todo negocio en sus etapas iniciales: el costo de la inversión.

Por consiguiente, para reforzar esta relación de sujetos o actores dentro del ámbito de la nube computacional, en el siguiente cuadro se puede ver esa relación del estudio realizado por Lee y Brink (2019) acerca de la "confianza en los servicios de la nube y adopción de clientes de *software* como servicio":

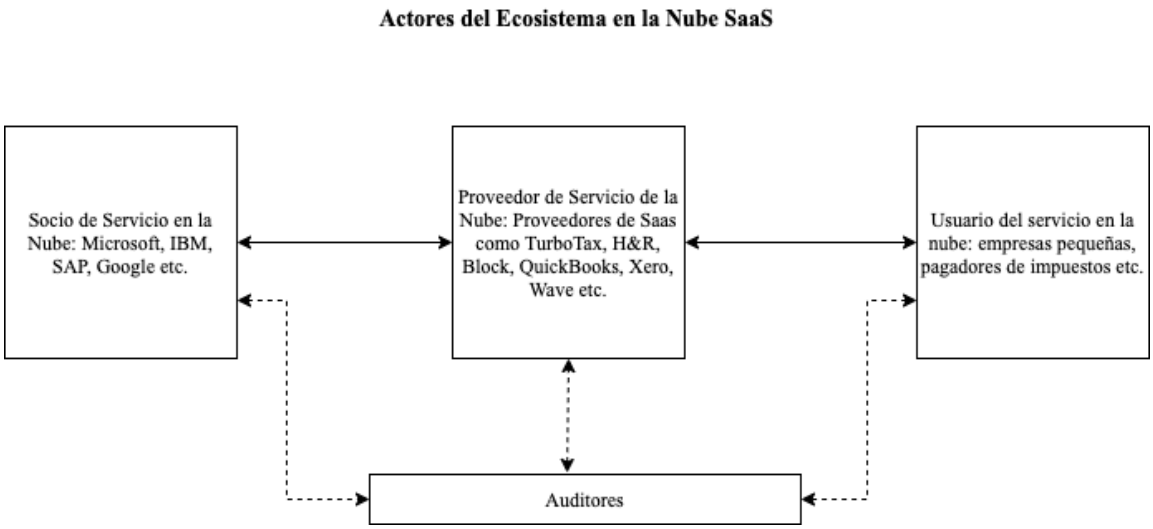


Figura 1.2. Actores de SaaS en un ambiente en la nube. Adaptado de *Confianza en los servicios de la nube y adopción de clientes de software como servicio*, elaborado por Lee y Brink (2019).

Si bien es cierto, cualquier empresa puede ser cliente de un producto de *software*, es probable que las organizaciones que empiezan su viaje en el mundo de los negocios adquieran rápidamente estas soluciones para sus necesidades tecnológicas, e inclusive, algunas ya lo hacen de forma automática y gratuita. Productos de correo como Gmail y Facebook para comunicación y promoción inicial de sus negocios está en auge, ya que permite a pequeños emprendimientos surgir sin mayores costos.

Según el periódico *La República* en Costa Rica, "Un 90% de los emprendimientos puede aprovechar las redes sociales para impulsar ventas. El otro 10% corresponde a empresas que requieren una página como vitrina virtual y cuyas ventas se realizan de la forma tradicional; es decir, cara a cara". (Castro, 2019, párr. 3)

Sin embargo, esta noticia no es un movimiento que acaba de comenzar en Costa Rica; por el contrario, refleja de manera precisa lo que ha pasado en la última década, el crecimiento de consumo de productos de naturaleza similar basados en la computación en la nube.

Estudios como el de la Universidad de Costa Rica, *Programa Sociedad de la Información y el Conocimiento*, en esta ocasión hace referencia a su informe publicado en 2010, el cual, en el capítulo cinco, incorpora un análisis de la adopción y uso de los servicios de computación en la nube. Inclusive, se citan casos específicos (lado del proveedor y lado del consumidor), como el de RACSA (Radiográfica Costarricense), que se menciona a continuación:

> Uno de principales servicios que ofrece RACSA es el *Escritorio Virtual*, solución conformada por una serie de herramientas de trabajo de oficina dentro de las que se incluyen navegadores de Internet, correo electrónico, procesadores de texto, presentaciones y hojas de cálculo, además de un sistema operativo y almacenamiento virtual, pues a partir de un disco duro virtual se puede respaldar cualquier tipo de archivo, con una capacidad de 5GB o de 10GB. Además del escritorio virtual, RACSA ofrece servicios con terminales delgadas que son dispositivos que sustituyen a las computadoras personales y cuya información está alojada en un centro de datos especializado, el servicio *Secure E-Mail* para asegurar la información que los usuarios comunican por correo electrónico y el servicio de Voz IP a partir del cual se da una reducción de costos y ampliación de las fronteras pues la señal de voz viaja a través de Internet. (PROSIC, 2010, pp. 173-174)

Estos mismos servicios, casi una década después, siguen vigentes, según el blog oficial del sitio de RACSA. En su entrega de junio de 2019, titulada: *Beneficios de elegir un Virtual Desktop de RACSA*, se detallan las ventajas competitivas que posee este servicio para sus clientes, entre las cuales está la escalabilidad y principalmente, la abolición de la necesidad de incurrir en gastos de reemplazo de equipos de cómputo a corto plazo. (RACSA, 2019b, párr. 4)

Además, describe el ofrecimiento de otro tipo de herramientas de *software* como RACSA DOCS, como gestor documental; proporciona evidencia de la contribución y compromiso de esta empresa nacional a la transformación digital para facilitar "soluciones digitales", como le suelen llamar oficialmente, para los clientes dentro del territorio costarricense. (RACSA, 2019a, párr. 6)

Lo anterior se menciona en el artículo en línea de 2019, publicado bajo el nombre *Cuidados a la hora de elegir un gestor documental adecuado*. Por lo tanto, gran parte de este crecimiento, como se puede deducir, tiene sus orígenes hace años en Costa Rica y, eventualmente, es un sector que se proyecta con un futuro sumamente favorable en términos de crecimiento y retorno de la inversión.

Se puede determinar también que, del lado del consumidor, existe una clara inclinación por la utilización de alguno de los modelos de servicio de computación en la nube, los cuales se evidencian en el gráfico de la figura 1.3, del informe de la Universidad de Costa Rica basado en su propia encuesta hecha al año 2011, como se aprecia a continuación:

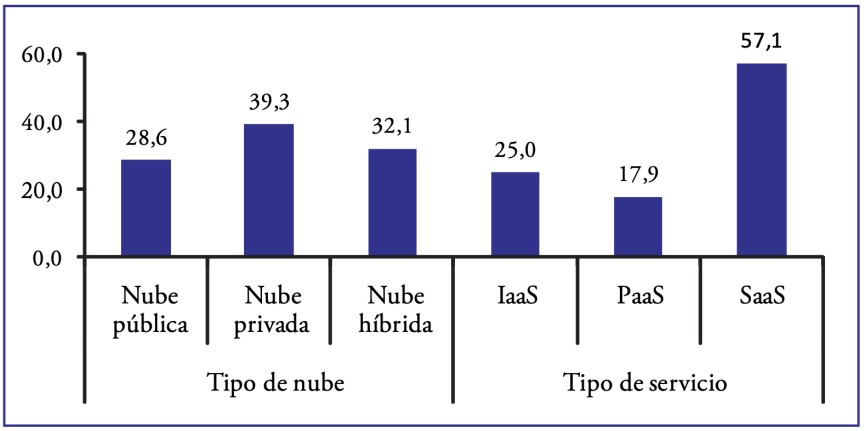


Figura 1.3. Características según tipo de nube y principal servicio que utilizan. Tomado de "Computación en la nube en Costa Rica", elaborado por PROSIC, 2010, *Informes hacia la sociedad de la información y el conocimiento*, p. 18.

De la misma forma, un estudio de 2019 publicado en la *Revista de Investigación Asiática en Ciencias de la Computación* llamado "Un estudio de la adopción de *Software* como Servicio (SaaS) en negocios en línea - pequeñas y medianas empresas en Sri Lanka", afirma lo siguiente:

> La nube computacional es una de las tecnologías que provee soluciones para sobreponerse a los problemas existentes identificados y que encaran muchas pequeñas y medianas organizaciones (o por sus siglas en inglés, *Small and Medium Enterprises SME´s*) en particular. La tecnología de la computación en la nube ofrece muchos beneficios significativos para los negocios con una referencia específica a la reducción de costo: inversión en *hardware*, costos de mantenimiento y bajo consumo de energía eléctrica. (Ayoobkhan y Asirvatham, 2019, p. 3)

Siguiendo en esta misma línea, existen otros factores, según Buyya *et al.* (citados en Ayoobkhan y Asirvatham, 2019), que indican que algunos de los costos a eliminar incluyen: actualización del sistema, almacenamiento de datos, mejoras, compra de equipos de *hardware* y mantenimiento (p. 4). Tal y como se mencionó, son aspectos que probablemente se trasladen directamente al proveedor del servicio.

Es decir, en este estudio se observa y analiza cuáles son las principales razones por las cuales empresas pequeñas y medianas abrazan aplicaciones de *software* como servicio para ayudarse en sus objetivos gerenciales y organizacionales. Además, tiene un fuerte enfoque en el uso de procesos sistemáticos, como la teoría de Difusión e Innovación (Rogers, 1962, citado en Ayoobkhan y Asirvatham, 2019, p. 5) y el marco de trabajo de Tecnología, Organización y Ambiente. (Tornatzky y Fleischer, citados en Ayoobkhan y Asirvatham, 2019, p. 5)

Del mismo modo, Hendro Lukman en su artículo “Un marco de trabajo conceptual de implementación de la nube computacional en emprendimientos con enfoque” publicado en la revista *IOP Conference Series: Materials Science and Engineering*, trata de:

> Examinar y evaluar la forma en que micro, pequeñas y medianas empresas adquieren el uso de esta tecnología utilizando un método como el marco de trabajo Tecnológico, Organizacional y Ambiente (TOE por sus siglas en inglés de *Technological, Organizational and Environmental Framework*). Según el autor, existen aspectos importantes básicos para la toma de decisiones, entre los cuales Infraestructura como Servicio, Plataforma como Servicio, *Software* como Servicio resaltan como candidatos fuertes para esta adopción de productos. (Lukman, 2020, p. 12176)

Por consiguiente, una vez identificadas estas publicaciones cercanas al tema de interés, que es la computación en la nube, y una de sus particulares aplicaciones, como es el *Software* como Servicio, se busca identificar situaciones en el territorio nacional costarricense que reflejen condiciones semejantes en cuestión de necesidad tecnológica, accesible en los procesos de micro y pequeñas empresas (PYME).

Lo anterior se basa en el artículo de CAMTIC sobre un estudio realizado por IOCIT consultores, donde se indica que el 80% de las PYME consultadas en territorio nacional reportan la utilización de más de dos servicios (aplicaciones) en la nube. (CAMTIC, 2012, párr. 7)

1.2 Justificación

La computación en la nube ha venido a impulsar y modernizar la manera como se diseña, desarrolla, entrega y comercializa *software*, con una rápida adopción que parece no conformarse con estar en la cúspide del mundo de las tecnologías de la información; así como también en el comercio tecnológico.

Esta tecnología busca expandir sus fronteras ofreciendo nuevos servicios y productos, los cuales podrían, hasta cierto punto, ser creados tomando como base los existentes, que ya les proporcionan a sus clientes. Como lo evidencia *International Trade Administration* (2015), se proyecta que las organizaciones gasten USD$191 billones en servicios de computación en la nube para el año 2020, comparado con los USD$74 billones que solían destinar en 2014. (Citado en Senarathna, Wilkin, Warren, Yeoh y Salzman, 2018, pp. 1-2)

Sumado a esto, la flexibilidad que ofrecen los modelos de servicio de la nube, como la Infraestructura como Servicio y *Software* como Servicio, provee cierta libertad creativa para generar productos nuevos. Muchos modelos de negocio son moldeados y desarrollados por la computación en la nube (Armbrust *et al*., 2010, citados en Namasudra, 2018, p. 114). En el futuro, esta misma puede ser utilizada como modelo computacional de negocios. (Namasudra, 2018, p. 114)

Por consiguiente, dado el potencial y la popularidad de esta tecnología, se intenta analizar los beneficios de una solución de *software* como servicio basado en la nube, desde el punto de vista técnico de implementación y tratamiento de datos, ante una solución de *software* local, desarrollada a la medida para la organización.

Estudios como el de Namasudra (2018), Briggs, Kassner (2017) y Ruparelia (2016), entre otros, presentan las características principales de estos modelos de servicio, además de una serie de beneficios a nivel empresarial. Sin embargo, se trata de abarcar una perspectiva amplia en el ámbito de comparación con los retos que puedan enfrentar profesionales que decidan emprender en el mundo del desarrollo del *software* con un producto o que ya sean una PYME en el sector tecnológico, pero que no posean un servicio de esta índole.

Como resultado, si el aprovechamiento de estos servicios es útil, otro punto importante a tratar sería el efecto que tendría en las operaciones de esas PYME.

Como bien lo expone Namasudra (2018), las tecnologías de la nube por sí mismas desarrollan los modelos de negocios; entonces, si se lleva a cabo la modalidad de *software* como servicio, se procura indagar las potenciales consecuencias que repercutan en términos de factibilidad técnica, económica y financiera en sus organizaciones.

Si bien es cierto, estudios como el de Alvarado Abarca (2017) buscan ayudar a usuarios completamente nuevos a guiarse en la implementación de aplicaciones en la nube de Amazon, proporciona un inicio del camino para determinar la reacción que pueda tener la operación de una solución como servicio en las PYMES en Costa Rica.

También, se aspira a comprender si al construir y operar productos (ciclo de vida de desarrollo) de *software* como servicio, en términos de arquitectura, el modelo de microservicios es un candidato factible a nivel técnico en el ámbito de escalabilidad y estandarización para este tipo de empresa.

Tizzei, Nery, Segura y Cerqueira exponen la arquitectura de SaaS como:

> Un producto de *software* como servicio debería ser multi-instancia para soportar múltiples clientes, escalable al maximizar la concurrencia y uso de recursos de manera eficiente. También, afirma que la *Arquitectura basada en Servicios* (SOA, por sus siglas en inglés) es una técnica común para crear este tipo de productos. En resumidas cuentas, con lo citado anteriormente, se ejemplifica el por qué de la interrogante en cuanto a esta arquitectura de sistemas de *software*. Asimismo, los Microservicios son un descentralizado estado de la práctica del enfoque SOA que se basa en pequeños y autónomos servicios que trabajan en conjunto. (Tizzei *et al*., 2017, p. 205)

Por otra parte, existe un aspecto que se desea conocer sobre este modelo de entrega de servicio de computación en la nube: la relación y efecto que tiene la modalidad de *software* como servicio en la implantación de productos novedosos impulsados por emprendedores en la actualidad. Este punto es importante porque se relaciona directamente con el crecimiento que ha tenido esta tecnología durante las últimas dos décadas.

Por el momento, no se tiene claridad sobre el impacto en las PYMES; sin embargo, organizaciones más grandes, como el caso de RACSA (RACSA, 2019a) en el ámbito nacional, y Amazon (Alvarado Abarca, 2017), en el ámbito internacional, que se reinventaron y empezaron a ofrecer soluciones digitales, demuestran la tendencia a innovación que se pretende justificar en este apartado.

Resulta clara la trascendencia que posee el modelo de *Software* como Servicio, así como las implicaciones que tenga para las organizaciones, especialmente pequeñas y medianas empresas.

De esta forma y dado el vasto número de datos e información existente se busca recopilar, concebir y desarrollar una serie de prácticas o métodos útiles para que organizaciones de poca experiencia en el desarrollo de *software*, puedan esclarecer su camino en la puesta en marcha de un producto de *software* como servicio en un ambiente de computación en la nube 100% virtual, estandarizando este mecanismo de entrega de *software* en Costa Rica.

1.3 Objetivos

1.3.1 Objetivo general.

- Proponer una guía de implementación de un producto de *Software* como Servicio orientada a los emprendedores costarricenses (PYME) del sector tecnológico, utilizando un entorno virtual de infraestructura en la nube mediante la estandarización de conocimientos, metodologías y procesos que permitan la agilización de la operación y la entrega de *software* al cliente.

1.3.2 Objetivos específicos.

- Identificar las pautas y prácticas utilizadas por las aplicaciones en la industria del desarrollo del *software* dentro y fuera del ámbito nacional, para comprender cómo funciona y opera el modelo de *software* como servicio utilizando infraestructuras en la nube, mediante la consulta pública de autores de artículos, revistas, libros y demás información de referencia inmediata.

- Conocer las operaciones de tecnologías de la información del sector PYME a nivel nacional, en las cuales su actividad se relacione con el desarrollo de *software*, por medio de interacciones con los miembros que las conforman, identificando la capacidad en recursos y habilidades que poseen para la implementación de *software* como servicio como parte de sus productos o servicios.

- Analizar los datos recopilados sobre la capacidad y habilidades que presentan los emprendedores costarricenses del sector PYME dedicados al desarrollo de *software* por medio del contraste de las prácticas, técnicas y procesos utilizados actualmente por la

industria de las tecnologías de la información para el descubrimiento del marco de trabajo, conocimiento, métodos y procesos necesarios y aplicables al sector en estudio.

- Producir una guía sistemática para la implementación de un producto de *Software* como Servicio mediante la estandarización de conocimiento en metodologías y procesos para traer beneficios operativos y financieros a los emprendimientos costarricenses del sector PYME tecnológico como proveedor, así como también a los consumidores.

1.4. Problema de Investigación

Actualmente, existen diversas formas mediante las cuales se puede poner en marcha un producto de *software* como servicio, principalmente como medio de emprendimiento. Con la ayuda de tecnologías en la nube y modelos utilizados por grandes compañías, como Salesforce.com o WebEx, se ha demostrado que este modelo de entrega de productos de *software* es viable y valoran la proposición de este modelo. (Dubey y Wagle, 2007, p. 3)

Además, respecto de lo dicho por Gartner (2019, citado en Seppänen, 2020), SaaS se establece como uno de los segmentos de mercado más grandes del mundo en los servicios públicos de la nube (p. 7). Dado lo anterior, puede ser de interés en el sector nacional, el formar parte de esta tendencia y entregar productos de *software* como servicio.

Sumado a ello, las PYMES en Costa Rica han estado adoptando este tipo de tecnologías basadas en la nube a un ritmo parecido al de países desarrollados, aludiendo al fácil acceso y operación de estas herramientas, esto último basado en el estudio de la empresa consultora Iocit. (CAMTIC, 2012, párrs. 5-7)

Sin embargo, surge la duda de si ¿los emprendedores que se dediquen al desarrollo de *software* en Costa Rica tienen la capacidad, los conocimientos, las metodologías y los procesos definidos para crear y poner en marcha un producto de *software* como servicio?

Lo anterior es relevante en empresas cliente de un mismo sector comercial, sea restaurantes, abastecedores, tiendas de compra y venta de bienes o mueblerías, las cuales posiblemente no cuenten con capital suficiente para invertir en soluciones más complejas y costosas.

Dado lo anterior, se plantea como base de esta investigación la siguiente pregunta:

¿Es posible y se adecua a sus necesidades organizacionales, la estandarización e implementación de un producto de *software* como servicio para una micro y pequeña empresa costarricense del sector de tecnología, específicamente de desarrollo de *software*, mediante el uso de infraestructura y servicios basados en la nube?

1.5 Hipótesis

Los diferentes mecanismos y técnicas que comprenden la puesta en producción de un producto de *software* son conocidos por la mayoría de los profesionales en el área de tecnologías de la información, sobre todo durante el tiempo y el surgimiento de marcos de trabajo ágiles y estándares para el manejo de la integración y entrega continua.

Sobresale en esa línea el concepto que ha surgido durante los últimos años como respuesta a una mejora constante en el proceso de desarrollo y entrega de *software*: DevOps (que traducido al español es la combinación de desarrollo y operaciones). Este último, "es un conjunto de prácticas para automatizar los procesos entre el equipo de desarrollo y otros equipos de TI, para que puedan construir, probar y entregar *software* más rápido y de manera robusta" (Atlassian, s. f., párr. 1)

Habitualmente, se espera ver este tipo de mecanismos empleados de varias maneras, por lo cual se le llamará adaptaciones, para la necesidad de equipos de distintos tamaños. Asimismo, los recursos que se necesitan para liberar un *software* como servicio con soporte multi-instancia

para muchos clientes es vasto, y se puede evidenciar en el conocimiento y variedad de procesos que pueden existir para poner en marcha soluciones de este calibre.

Por lo tanto, para efectos de esta investigación se plantea la siguiente hipótesis:

La estandarización de la implementación de un producto de *Software* como Servicio, en un ambiente completamente virtual en la nube computacional, ayudará a los emprendedores costarricense (PYME) del sector de tecnología a tener un modelo de negocio en el cual puedan desarrollar, entregar y operar, de modo ágil, una solución al alcance de las necesidades de sus clientes.

1.6 Alcance

Para el ámbito de estudio de este apartado, se pretende establecer una guía de implementación de un producto de *Software* como Servicio (SaaS, por sus siglas en inglés de *Software as a Service*).

Esta deberá contemplar los aspectos técnicos como infraestructura necesaria para operar una aplicación de este tipo, usando servicios de computación en la nube en su totalidad; así también, abarcará aspectos de arquitectura de sistemas de información aplicables a este modelo de entrega de *software*.

Producto del análisis de la realidad de los emprendimientos que se dedican al desarrollo de *software* en el territorio nacional (probablemente empresas registradas como PYMES), se espera abarcar aspectos como la capacidad técnica, de recurso humano y la viabilidad del modelo de entrega de *software* con la finalidad de propiciar un modelo que les permita la creación y entrega de servicios de *software*, de forma ágil y rápida; de modo que les permita suministrar aplicaciones en la nube para uso de la misma empresa PYME en Costa Rica.

En consecuencia, los aspectos que se mencionan en el estudio de la creación de esta solución deben acoplarse fácilmente a esta población de organizaciones costarricenses. Además, las mejores prácticas se deben adaptar al contexto en el cual se desenvuelven, para así proveerles de una ventaja competitiva en cuanto a la adopción de servicios de computación en la nube.

1.7 Limitaciones

Dentro del ámbito de desarrollo de *software* y computación en la nube, esta investigación cuenta con puntos específicos e importantes, los cuales no están cubiertos dentro de este trabajo en detalle.

Primero, porque no interesa desarrollarlos, ni se tiene el recurso en tiempo y esfuerzo; sin embargo, se incentiva el interés como punto de partida para otras investigaciones que en el futuro tengan oportunidad de acrecentar el conocimiento en esas áreas.

Inicialmente, una restricción que se visualiza en este estudio es la facilidad de obtener información de las PYMES participantes. Aspectos que incluyen los derechos de autor, seguridad y confidencialidad de la gestión de los datos, secretos comerciales y demás aspectos relacionados con la divulgación de información específica de las organizaciones, se tornan en un panorama difícil en cuanto a obtención de respuestas a las interrogantes del estudio. Pese a lo anterior, es posible que algunas de ellas faciliten datos generalizados que podrían afectar los indicadores de medición que se detallan más adelante, en el capítulo tres de este documento.

Segundo, tampoco es de interés el análisis de los otros modelos de entrega de computación en la nube como el de Infraestructura como Servicio y Plataforma como Servicio para la adquisición de aplicaciones para PYMES que no sean de desarrollo de *software* para su utilización.

Tampoco, esta investigación pretende adentrarse en el mundo de la adopción de productos de *software* como servicio, ni generar un marco de trabajo para ejecutar ese proceso; si

no, más bien, utilizar este último como indicador de posibles soluciones de *software* que buscan las PYMES para construir un producto ajustado a su condición.

Con respecto a los precios y licenciamiento, no es de importancia para el objetivo primordial de investigación. Simplemente se muestra un camino y, como parte de esa guía, se muestran ejemplos de plataformas, pero no se introduce en temas de precios. Eso queda sujeto a cada proceso de selección de plataformas que es propio de cada empresa y único de cada contrato con proveedores en ambientes de computación en la nube.

Respecto del desarrollo de aplicaciones como tal, solamente se adentrará en el tema de arquitectura de manera generalizada, para reforzar el razonamiento explicativo que existe detrás de la creación de un producto de este calibre.

No procura adentrarse en aspectos relacionados con los lenguajes de programación, ni cuestiones concernientes a diagramas de clases, ni modelos entidad-relación y otras herramientas de diseño. Al contrario, se busca cubrir el diseño y operación del sistema como un todo, acoplando la infraestructura y aplicación de *software* (desarrollo y operaciones).

En última instancia, no se pretende crear modelos que se amoldan a cualquier tipo de empresa, tampoco reemplazar la forma en que empresas, mucho más grandes y con amplia trayectoria, operan y entregan *software*. No obstante, el estudio de la experiencia de este tipo de entidades permitirá adecuar las mejores prácticas y procesos a un ámbito más reducido, como es el de pequeños emprendedores en el territorio nacional, para que así sus limitaciones sean gestionables y propicien su acción bajo este marco de trabajo de computación en la nube.

1.8 Matriz de Congruencia

Título	Problema	Objetivo General	Objetivos Específicos	Preguntas de Investigación
Análisis del modelo de *software* como servicio orientado a la PYME costarricense de desarrollo de *software*, utilizando un entorno virtual en la nube.	¿La estandarización de la implementación de un producto de *Software* como Servicio para una PYME costarricense en el sector de tecnología, utilizando una infraestructura y servicios basados en la nube, es posible y se adecua a sus necesidades organizacionales?	Proponer una guía de implementación de un producto de *Software* como Servicio orientada a los emprendedores costarricenses (PYME) del sector tecnológico, utilizando un entorno virtual de infraestructura en la nube mediante la estandarización de conocimientos, metodologías y procesos que permitan la agilización de la	Identificar las pautas, prácticas, productos y proveedores utilizados en la industria del desarrollo del *software*, dentro y fuera del ámbito nacional, para comprender cómo funcionan y operan los modelos de *Software* como Servicio utilizando infraestructuras en la nube mediante la consulta pública de	¿Cuáles son los beneficios de una solución de *software* como servicio basado en la nube, desde el punto de vista técnico de implementación y tratamiento de datos, ante una solución de *software* local desarrollada a la medida para la organización?

		operación y entrega de *software* al cliente.	autores de artículos, revistas, libros y demás información de interés. Conocer las operaciones y tendencias del sector PYME a nivel nacional, cuya actividad se relacione con el desarrollo de *software*, por medio de entrevistas y conversaciones con los miembros que las conforman, identificando la capacidad en recursos y habilidades que poseen para la implementación	¿De qué manera el *software* como servicio impacta la micro y pequeña empresa (PYME) en Costa Rica si se usa como opción de modelo de entrega de *software*, en términos de factibilidad técnica y operacional?

			de *Software* como Servicio como parte de sus productos o servicios. Analizar los datos recopilados sobre la capacidad y habilidades que presentan los emprendedores costarricenses del sector PYME dedicados al desarrollo de *software* por medio del contraste de las prácticas, técnicas y procesos utilizados actualmente por la industria de las tecnologías de la	¿Cómo impacta al proveedor, el uso de una arquitectura de *software* basada en microservicios en el ciclo de vida de desarrollo y operación de un producto de *software* como servicio a nivel técnico en términos de

			información para el descubrimiento del marco de trabajo, conocimiento, métodos y procesos necesarios y aplicables al sector en estudio.	escalabilidad y estandarización?
			Producir una guía sistemática para la implementación de un producto de *Software* como Servicio mediante la estandarización de conocimiento en metodologías y procesos para traer beneficios operativos en los	¿Es concebible que las micro y pequeñas empresas (PYME) de desarrollo de *software* en Costa Rica puedan aplicar un marco de

			emprendimientos costarricenses del sector PYME tecnológico como proveedor, así como también a los consumidores.	trabajo simplificado en cuanto a la implementación de un *software* como servicio como parte de su catálogo de servicios?

CAPÍTULO II. MARCO TEÓRICO

2.1 Computación en la Nube

Las tecnologías de la información han impactado de gran forma el mundo de los negocios en las últimas décadas. Uno de sus conceptos o prácticas mayormente aceptados fue la interconexión de ordenadores para compartir datos. En un inicio, fue marcado por el nacimiento de la Internet, término con el cual la mayoría de los seres humanos hoy en día se encuentran muy familiarizados, pero que en sus primeros pasos era un mundo desconocido, con capacidades todavía no descubiertas.

Así es como, a través de la Internet, se empieza a gestar la idea de compartir recursos en red para facilitar la colaboración entre personas. Eventualmente, entidades de mayor tamaño adoptan este modelo, ya no simplemente para compartir archivos, sino otro tipo de activos hasta llegar a proveer servicios. Como una novedosa consecuencia, la nube computacional viene a crear una nueva forma de cooperación y trabajo, además de replantear a los departamentos de tecnología el modo emergente de tener, configurar y soportar tanto infraestructura como aplicaciones.

De acuerdo con NIST (Instituto Nacional de Estándares y Tecnología, por sus siglas en inglés), se define la computación en la nube como un modelo que permite, de manera eficiente, rápido acceso bajo demanda a la red para distribuir una banda de activos de computadoras configurados. (NIST, citado en Diaby y Rad, 2017, p. 50)

En otras palabras, la nube computacional sirve para compartir cualquier activo de tecnología que exista a través de Internet como un servicio; es decir, solo se paga por usar el beneficio durante un periodo de tiempo específico. Todo esto sin que el cliente tenga que

desembolsar grandes cantidades de dinero en responsabilidades que están abstraídas por el proveedor del servicio de computación en la nube.

2.1.1. Antecedentes e historia de la computación en la nube.

El crecimiento de la demanda por recursos distribuidos en entornos empresariales en la década de los 90's y los 2000 es un buen punto de referencia para empezar a rastrear el nacimiento de esta nueva modalidad de servicio. Sin embargo, la computación en la nube tiene antecedentes más antiguos, que se remontan a la década de los 50's.

Como se menciona a continuación, en 1955, John McCarthy acuñó y propuso la teoría de "tiempo compartido" en un grupo entero de usuarios, esto debido a que poseer computadores era un lujo muy caro, y maximizar su uso era una prioridad para aquellas organizaciones que le estaban apostando a la tecnología. (ECPI University, 2020, párr. 3)

Lo anterior también se evidencia en un artículo publicado por IBM, que curiosamente es uno de los grandes actores y promotores en el desarrollo de esta tecnología, el cual sostiene que:

> La computación de *mainframes* (supercomputadoras centrales, por su significado en español) fue la que dio inicio a toda esta evolución tecnológica, en la que múltiples usuarios fueron capaces de acceder computadores centrales a través de terminales tontas, cuya única función era de proveer una interfaz de acceso al nodo principal. Una vez más, el alto costo era uno de los principales obstáculos para las organizaciones. (Neto, 2019, párr. 3)

Por consiguiente, debido a estos eventos, se puede descubrir una necesidad bastante obvia por la cual se gestan estas soluciones y es la base de lo que hoy le da a la computación en la nube

una ventaja competitiva abismal ante el alto costo de adquisición y mantenimiento de computadores propios en una escala que no es rentable.

Después, solamente unos años más tarde, en 1967, IBM logra virtualizar los sistemas operativos dando la capacidad a múltiples usuarios de compartir un recurso. Eventualmente, en 1969, ARPANET (red de agencias de investigación avanzada de proyectos, por sus siglas en español) es lanzada por el Departamento de Defensa de los Estados Unidos, que fue el predecesor de lo que hoy se conoce como Internet. (BCS, 2019, párr. 2)

Estos dos eventos dieron un impulso significativo a la segunda característica de la fundación de la computación en la nube: la comunicación de datos a través de la red a bajo costo. La capacidad de ir más allá y expandir las operaciones de la virtualización en conjunto con el crecimiento de nodos en la red ayudó a catapultar la manera en que las compañías harían negocios en las siguientes décadas.

Posteriormente, estas dos tecnologías seguirán su desarrollo. Como se menciona en el mismo artículo de la BCS, esta fusión de tecnologías facilitó acciones como transferencias entre instituciones financieras, uso de señales olvidadas de televisión para el envío de datos, más nodos conectados a la red y mayor capacidad de almacenamiento. (BCS, 2019, párr. 3)

Ciertamente, la comunicación a través de Internet fue propiciando un camino abierto a una serie de eventos, en los cuales las compañías tenían a mano el lienzo para pintar la mejor obra de arte que serviría para crear productos nuevos, mejorados y eficientes para contribuir en lo que ya se colocaba como la era digital en sus inicios. También, se puede apreciar que el origen de la computación en la nube tiene estricta relación con el ámbito corporativo.

Sin embargo, no sería hasta los años 90 que las compañías empezarían un trabajo más robusto hacia la nube, centrándose primeramente en negocios basados en Internet, un ejemplo de ello es Amazon.com. Siguiendo la línea del artículo de la BCS, en 1997, el profesor Ramnath Chellapa de la Universidad de Emory mencionó el término "nube" en uno de sus artículos. Aunque fue acuñado, muchos afirman que es el predecesor del modelo *Grid Computing* (computación en malla, adaptado al español). (BCS, 2019, párr. 5).

De acuerdo con el autor J. Lee, en su publicación *Una vista de la computación en la nube*, en la computación en malla, una supercomputadora virtual está compuesta de muchas computadoras en red desacopladas para realizar tareas demandantes en recursos en conjunto; asimismo, para lidiar con ciertos picos en la demanda. (J. Lee, 2013, p. 3)

Por su parte, K. Chandrasekaran, en su libro *Essentials of Cloud Computing* (*Principios de la computación en la nube,* por su adaptación al español), sostiene que la computación en malla es una red de computadores o máquinas procesadoras administradas por un *software* tipo *middleware* (*software* intermediario) para acceder y usar los recursos remotamente. (K. Chandrasekaran, 2014, p. 4)

Del mismo modo, este autor también afirma que la idea detrás de la computación en malla era darle uso al poder computacional que poseen organizaciones, el cual se está desperdiciando, a las que no tienen la misma capacidad, y aumentar el retorno de la inversión en tecnología. (K. Chandrasekaran, 2014, p. 4)

Ahora bien, tiene sentido la afirmación acerca de la computación en malla. En realidad, en la mayoría de libros, artículos y demás material en el que la computación en la nube salga a relucir, siempre está presente este concepto. La idea de tener varios nodos realizando tareas de

manera colaborativa se asemeja al modelo de la nube en la que varios computadores actúan en conjunto para proveer una infraestructura sólida como servicio a otros consumidores.

Luego, se puede ver el ejemplo de Salesforce.com en 1999. Su manera simple de entregar y proveer aplicaciones vía un navegador de Internet era un paso agigantado para el público en ese momento. Como producto, muchas otras compañías en el mercado seguirían sus pasos. (Namasudra, 2018, p. 116)

Después, se entraría en una de las décadas cuando la computación en la nube se convirtió en un servicio demandante: los 2000. Esta escaló a tal punto, que muchas grandes empresas tuvieron que reacomodar su plan principal de negocios para seguir compitiendo. Algunos ejemplos de ello son Apple Inc., IBM y Microsoft, que tienen algo en común: un servicio de computación en la nube. Apple con su iCloud, IBM con su IBM Cloud y Microsoft con Azure.

En cambio, retrocediendo un poco el tiempo a inicios del nuevo milenio, la empresa que apostó a lo grande en este campo fue Amazon, con su subsidiaria de computación en la nube: Amazon Web Services (Servicios Web de Amazon, por su adaptación en español).

Como lo menciona la BCS, en 2002, Amazon Web Services lanzó su nube pública. No había casi ningún competidor en este punto, pero mientras que los beneficios de la nube eran evidentes, como la elasticidad y escalabilidad, tenía que abrirse más camino. (BCS, 2019, párr. 6)

Más adelante y como lo menciona Amazon en su sitio web oficial, “en 2006, Amazon Web Services comenzó a proporcionar servicios de infraestructura de TI para empresas en forma de servicios web, más conocido hoy como informática en la nube”. (Amazon Web Services, 2021, párr. 1)

La capacidad computacional estaba creciendo y ganando mercado rápidamente a través de más compañías que adoptaron este nuevo modelo, tanto clientes como proveedores. Como se expresó anteriormente, empresas importantes como IBM, Apple, Microsoft e, incluso, Google, empezaron a darse cuenta del potencial que tenía este modelo de servicio.

A esto último hace referencia Surbhi Rastogi en su libro titulado: *Cloud Computing Simplified* (*Computación en la nube de manera simplificada,* por su adaptación al español), donde afirma que, en 2009, Google Apps fue lanzado para entregar servicios en la nube, disponible en web y para teléfonos inteligentes. En 2010, lo hizo Microsoft con el lanzamiento de Azure. Después, lo hizo IBM con el lanzamiento de SmartCloud en 2011. (Rastogi, 2021, p. 16)

Retrospectivamente, cuando se analiza desde un punto de vista de mercado, la computación en la nube no ha dejado de expandirse durante la última década. La presencia de más actores en el juego ha propiciado un cambio tecnológico con tendencia a la migración a la nube. En la actualidad, todas las compañías están desechando sus infraestructuras *on-premise*, término en inglés utilizado para referirse a todo *hardware* y *software* administrado y alojado directamente en centros de datos propios de las empresas, y cambiándolos parcial o completamente por entornos virtuales en la nube.

Lo anterior, se evidencia en lo que aluden Attaran y Woods en su artículo publicado en el *Journal of Strategic Innovation and Sustainability* (*Revista de Investigación de Innovación Estratégica y Sustentabilidad*, por su traducción en español):

> El uso de modelos de servicio de computación en la nube ha mostrado un crecimiento significativo. De acuerdo con un reporte de 2017 de Gartner, Infraestructura como

> Servicio (IaaS) está proyectado a crecer un 32% para alcanzar los $5.5,1 billones. El mercado de *Software* como Servicio (SaaS) se espera que experimente un menor crecimiento debido a la madurez de las ofertas de este producto y la aceleración en la compra de servicios como CRM y aplicaciones financieras. (Reporte de Gartner, 2017, citado en Attaran y Woods, 2018, p. 99)

En relación con el estado de la computación en la nube durante estos años, según Varghese y Buyya (2018), las aplicaciones ahora apuntan a empoderar la infraestructura en la nube haciendo uso de recursos heterogéneos de múltiples proveedores. Esto contrasta con cómo se usaba tradicionalmente un proveedor único o un centro de datos años atrás. (Varghese y Buyya, 2018, p. 1)

Sin ninguna duda, aquí hay un patrón que se ha repetido durante generaciones de adelantos tecnológicos y, hasta en algunos casos, relacionados con otros avances e inventos en otros campos. La necesidad de hacer entornos más complejos se refleja en el párrafo anterior, en el cual ya un único proveedor de servicios en la nube no es suficiente, sino que surge la tendencia de poseer entornos mixtos. Esto permite el uso de distintos servicios que convergen en arquitecturas flexibles y robustas para las necesidades actuales de las organizaciones modernas.

De vuelta a la línea de Varghese y Buyya, en ese mismo artículo se habla de un entorno de nube múltiple. Ellos afirman que la noción tradicional de la nube múltiple se aprovechaba de varios centros de datos de un proveedor. Luego, las aplicaciones evolucionaron a utilizar recursos de más de un proveedor. (Varghese y Buyya, 2018, p. 2)

Además, de acuerdo con una encuesta de Rightscale, una compañía que se dedica a la venta de *software* como servicio, se estima que las organizaciones modernas utilizan, en

promedio, seis proveedores diferentes (Rightscale, 2016, citado en Varghese y Buyya, 2018, p. 2). Por ende, no es sorpresa que sea una práctica que las empresas adopten con más frecuencia.

Los beneficios de un entorno múltiple de nubes se evidencian en las soluciones de *software* que surgen como aplicaciones novedosas y que rápidamente escalan en uso, lo cual evidentemente requiere mayor complejidad en escalabilidad y flexibilidad de la infraestructura de TI: servidores, redes privadas virtuales (VPC's, por sus siglas en inglés), almacenamiento, bases de datos, librerías de contenido, etc.

2.1.2 Características de la computación en la nube.

Es conveniente conocer los principales fundamentos por los cuales la tecnología de la computación en la nube es tan extensamente popular y aceptada. Del mismo modo que todo avance en el campo de la informática, este modelo se basa en propiedades que describen este concepto de forma específica, dando a conocer sus fortalezas.

En referencia a lo anterior, en el Instituto Nacional de Estándares y Tecnología del Departamento de Comercio de Estados Unidos (NIST, por sus siglas en inglés), Peter Mell y Timothy Grace afirman que está compuesto por cinco características esenciales, tres modelos de servicio y cuatro modelos de despliegue. Dentro de las características están: "autoservicio por demanda, amplio acceso a través de la red, reservas de recursos en común, rápida elasticidad y servicio controlado". (Mell y Grance, 2011, p. 2)

2.1.2.1 Autoservicio por demanda.

Un pilar básico es el autoservicio, primeramente, porque requiere interacción completamente de parte del cliente con el servicio de la nube que se necesita consumir; esto puede ser algún servidor de base de datos, un servidor web, un cortafuegos, un balanceador de carga, una librería de contenido etc. En otras palabras, el proveedor no tiene que estar presencialmente ni obligado a provisionar los recursos para el cliente, sino que él mismo consume lo que ocupa.

Según la NIST, lo define como "un consumidor puede provisionar recursos computacionales, ejemplo: un servidor o almacenamiento; automáticamente, como sea necesario, sin requerir interacción humana de cada proveedor de servicios. (Mell y Grance, 2011, p. 2)

2.1.2.2 Amplio acceso a través de la red.

La nube está hecha para utilizarse a través de Internet, cualquier servicio será entregado por este medio. La conectividad es remota, es decir, que los recursos estarán en una ubicación virtual o física totalmente apartada del centro de datos del cliente y, aun así, los podrá consumir sin importar nada más que la conexión a la red.

Por su parte, la NIST, en su estándar, lo expresa de manera similar: "la capacidad computacional está disponible a través de la red y accedida por medio de mecanismos estándar que promueven el uso heterogéneo de plataformas de clientes". (Mell y Grance, 2011, p. 2)

A manera de aclaración, por "uso heterogéneo" se entiende que es la capacidad que tiene la nube de entregar su servicio a un sinnúmero de dispositivos electrónicos, así como plataformas

de *software* de cualquier tipo, siempre y cuando puedan comunicarse usando el protocolo TCP/IP.

2.1.2.3 Reservas de recursos en común.

De acuerdo con la NIST, los recursos computacionales son asignados para servir a múltiples clientes utilizando un modelo *multi-tenant* (multi-inquilino, por su adaptación al español), con diferentes recursos virtuales o físicos que son dinámicamente asignados o reasignados basados en la demanda de los consumidores. (Mell y Grance, 2011, p. 2)

En otras palabras, al igual que un servicio de taxis, cuando se toma un viaje de un lugar A hacia otro B, se utiliza una unidad con su chofer particular y con su placa única. Al mismo tiempo, cuando se llega al destino B, se desocupa el taxi y otra persona se monta al taxi que acaba de quedar libre para ir del lugar B a otro destino C. Esto ocurre *n* cantidad de veces por tiempo indefinido.

Las reservas de recursos que pone a disposición un proveedor de la nube también tienen un mecanismo de administración similar, aunque su fuerte sea la virtualización, esta capa todavía es dependiente de los recursos computacionales físicos de los centros de datos de los suministradores de estos.

2.1.2.4 Rápida elasticidad.

Como resultado y en relación con la anterior característica, la elasticidad es un aspecto sumamente importante en este ámbito porque permite que las aplicaciones escalen exponencialmente, de manera instantánea, debido a que la nube abastece, administra y reconoce cuando necesita subir y bajar la oferta respectivamente para satisfacer la demanda.

2.1.2.5 Servicio controlado.

Cabe considerar que la NIST muestra lo siguiente acerca de esta característica: los sistemas de la nube automáticamente controlan y optimizan el uso de los recursos con su primordial forma de cobro, el cual es "pague solamente por lo que usa", a cierto nivel de abstracción apropiado para el tipo de servicio. (Mell y Grance, 2011, p. 2)

Es evidente que estas últimas tres características se enfocan en un elemento crucial para que exista computación en la nube: los recursos; por lo tanto, los proveedores desarrollaron sus plataformas y servicios con varias funcionalidades que permiten a los clientes controlar lo que consumen para evitar grandes costos al aprovechar estos para aplicaciones de *software*, infraestructura de TI, etc.

Es por ello que la gestión de reportes en este tipo de servicio es "debe de estar", ya que ayuda en gran manera a las organizaciones a rastrear sus gastos en la nube; sin embargo, dado que la mayoría de infraestructura de TI es necesaria para darle valor agregado a las empresas, entonces se cambia la descripción de gasto por inversión. Desde una perspectiva más general, es una inversión a largo plazo, puesto que las ganancias de las instituciones dependen de qué tan buenos sean sus sistemas y tecnologías que pongan en uso para apoyar el negocio.

2.1.3 Modelos de servicio.

La computación en la nube ha expandido sus fronteras y adaptado sus ofertas para cubrir los diferentes casos de uso que el mundo de los negocios requiere. Parte de esta evolución ha llevado a tener tres tipos de servicio, bien definidos, en los cuales se ha centrado la mayoría de los productos existentes; estos son: Infraestructura como Servicio (IaaS), *Software* como Servicio (SaaS), y Plataforma como Servicio (PaaS).

2.1.3.1 Infraestructura como servicio.

Como lo especifica la NIST, se trata del aprovisionamiento de capacidad de procesamiento, almacenamiento, red y otros recursos de computación fundamentales, donde el cliente es apto para desplegar y correr *software* arbitrariamente, lo cual incluye sistemas operativos y aplicaciones. (Mell y Grance, 2011, p. 3)

Por su parte, un ejemplo arbitrario de uso de este modelo lo propician Briggs y Kassner en su estudio llamado *Estrategia Empresarial de la Nube*, donde aseguran que es posible correr un servidor de SQL o muchos otros productos en este modo; esto es, dentro de máquinas virtuales. En este caso, el cliente debe mantener el sistema operativo, *software* de base de datos y demás. (Briggs y Kassner, 2017, p. 91)

Como se puede apreciar, ambos autores hacen hincapié en el uso de recursos computaciones de manera virtualizada para poder instalar y correr otros productos de *software* dentro de estas instancias. Todo el *hardware* y su capacidad es abstraída en servicios virtualizados que logran capturar la atención del cliente, y le dan la suficiencia para poder administrar estos recursos y así construir su infraestructura basado en ellos. Esto le brinda solución a su entorno con la utilización mínima de *hardware*.

De igual forma, es importante recalcar que este modo suele ser para usuarios especializados en tecnologías en información, ya que requiere una experiencia técnica para poder configurar, administrar y mantener todos estos recursos computacionales que se ponen a disposición.

2.1.3.2 Plataforma como servicio (PaaS).

Sin lugar a duda, es uno de los principales referentes de la computación en la nube. La plataforma como servicio se ha introducido como una opción atractiva para grandes organizaciones, esto permite a otros consumidores extender y desplegar soluciones personalizadas que respondan a necesidades específicas.

En este caso, también se hace referencia a la definición oficial de la NIST, en la cual se especifica que es la facultad dada al cliente de desplegar en la infraestructura de la nube, aplicaciones creadas o adquiridas por ellos usando lenguajes de programación, librerías, servicios y herramientas soportadas por el proveedor. (Mell y Grance, 2011, pp. 2-3)

De forma similar, el autor Cusumano, en su publicación *La nube como innovación de plataforma para el desarrollo de software*, "se refiere a las funcionalidades que los usuarios acceden a través de la nube mientras administran sus propias aplicaciones de *software* y datos en servidores internos de las compañías". (Cusumano, 2019).

Por su parte, Namasudra afirma que en este modelo los usuarios no se tienen que preocupar de la escalabilidad, ni de cualquier *software* o *hardware*. Tampoco por la cantidad de memoria usada para correr el *software*, ya que provee una plataforma completa para el desarrollo del *software*. (Namasudra, 2018, p. 118).

Según lo anterior, la principal característica de este modelo es brindar un ambiente óptimo para el desarrollo de aplicaciones de *software*. Se sabe que este es el enfoque principal, ya que las necesidades de negocio pueden variar entre organizaciones. Además, proporciona ciertas ventajas en el ciclo de vida de creación de una aplicación, ya que abrevia el tiempo y esfuerzo en el que un producto de *software* puede estar funcionando en un ambiente productivo.

Aunque, en tal sentido, después del análisis de lo afirmado por Cusumano, se está en desacuerdo con que siempre los recursos deban estar en servidores internos de la compañía, esto debido a que, productos más actuales como App Engine de Google, son un tipo de plataforma como servicio completamente entregado en la nube. Inclusive, no se deja de lado la probabilidad de que, en un futuro cercano, la utilización de centros de datos locales pase a ser un porcentaje muy bajo de la infraestructura total de las empresas, a excepción de los que se dediquen a ser proveedores de servicios en la nube o agencias gubernamentales.

Como se puede apreciar en la documentación de Google Cloud y para ejemplificar un poco acerca de App Engine, "es una plataforma sin servidores completamente administrada para desarrollar y alojar aplicaciones web a gran escala. Puedes elegir entre varios lenguajes, bibliotecas y marcos de trabajo para desarrollar tu aplicación". (Google, 2021, párr. 1)

De este modo, se reitera la naturaleza del modelo de plataforma como servicio y, además, sirve como ejemplo de un servicio completamente administrado por un proveedor de la nube. Sin duda, una forma novedosa de entregar *software* que existe actualmente.

2.1.3.3 Software como servicio (SaaS).

Como objetivo central de este estudio, el *software* como servicio es una pieza fundamental en la manera en que se entregan aplicaciones en la actualidad. Parte de su éxito se atribuye a su facilidad con la cual sus usuarios pueden no solamente hacer uso de la herramienta, sino que pueden obtener beneficios mayores a sus operaciones que resolver el problema principal del origen del *software*.

Los autores Peter Mell y Timothy Grance, en el estándar de NIST, manifiestan que es la capacidad dada al consumidor para utilizar las aplicaciones del proveedor corriendo en la

infraestructura en la nube. Estas son accesibles desde varios dispositivos clientes a través de interfaces, como un navegador web, o un programa de interfaz. (Mell y Grance, 2011, p. 2)

Algo semejante ocurre con Ian Sommerville, quien en su libro titulado *Ingeniería del Software*, expresa que, en computación en la nube, el *software* como servicio (SaaS) es un modelo en el cual productos de *software* son entregados a clientes a través de Internet y cobrado basado en el uso. (Sommerville, 2016, citado en Hoang Pham *et al.*, 2017, p. 285)

En concordancia, Nayan Ruparelia expone que los clientes no tienen por qué preocuparse de instalar la aplicación, el sistema operativo en el cual corre o ninguna otra dependencia. Crucialmente, tampoco de mantener el *software* actualizado porque, estando como servicio, siempre estará con la última versión. (Ruparelia, 2016, p. 29)

Por tal motivo, es que este modelo de servicio es tan conveniente para empresas cuyo capital de inversión no es muy alto, especialmente si están empezando su travesía en el mundo del emprendimiento. Esto se puede abordar desde dos puntos muy específicos, los cuales son: costo de uso de la aplicación y costo del recurso humano para mantener la aplicación; siendo este último, quizá, uno de los más influyentes a la hora de adquirir herramientas empresariales que sean realmente costosas.

Habitualmente, la sencillez con la que se entrega *software* funcional a través de una interfaz sencilla, como un navegador o una aplicación móvil en los dispositivos móviles, como los celulares y tabletas, permite una interacción concisa entre el sistema y el usuario. Le ahorra mucho espacio en almacenamiento local, instalación y configuración de manera local; evita la preocupación de que las especificaciones del *hardware* no satisfagan las del requerimiento del *software*, entre otros beneficios que posicionan este modelo en el *top* de los más frecuentemente usados.

Además, en el lado del fabricante, permite lanzar actualizaciones a una sola instancia del *software* que permite una publicación global y compartida entre todos los usuarios por igual. No existe distinción de ninguna clase, lo cual facilita también la entrega de nuevas funcionalidades, así también como corrección de errores de manera rápida y eficiente.

Esta característica es compartida por Haigh Smith y Tanner, quienes plantean que el *software* como servicio provee una plataforma de entrega para el desarrollo de *software* con acceso global a los usuarios con acceso a Internet. Asimismo, las actualizaciones de *software* ocurren en el lado del servidor. (Haigh Smith y Tanner, 2016, p. 126)

Al mismo tiempo, como señalan estos autores, los desarrolladores de *software* pueden usar API's prestadas a través de SaaS para integrar como componente de su *software*, promover la reutilización de código y estandarización, además de ser una potencial fuente de ganancias para el proveedor del API[1]. (Haigh Smith y Tanner, 2016, p. 126)

En relación con lo anterior, es muy importante resaltar la capacidad de los programas entregados bajo este modelo que tienen estandarizado facilitar la comunicación con otros sistemas. Esto se ejemplifica con la suite de servicios web puestos a disposición por plataformas como Workday (encontrado en: https://community.workday.com/sites/default/files/file-hosting/productionapi/index.html), o Google GSuite (llamado Google Workspace, encontrado en: https://developers.google.com/workspace). Por ende, la integración entre sistemas se da de manera sencilla y con acciones que, regularmente, tienen una sola responsabilidad.

[1] API: Interfaz de programa de aplicación.

2.1.4 Proveedores de servicio de computación en la nube.

Con respecto a los distribuidores de los servicios de computación en la nube, actualmente existe una variedad alta de ofertas que contienen un número amplio de propuestas que realizan una acción definida en pos de la necesidad de los consumidores.

Según Microsoft, define el proveedor de servicio de la nube como una compañía tercera que ofrece una plataforma basada en la nube, infraestructura, aplicaciones y servicios de almacenamiento. Similar a tener una casa, se paga por servicios como electricidad o gas, los consumidores solo pagan por los servicios que usan. (Microsoft, 2021a, párr. 1)

En este sentido, concuerda dicha definición con los modelos de servicio que anteriormente se mencionaron en este documento. Esto supone englobar varios requisitos que se le "exigen" a un proveedor para considerarse una opción viable para las compañías en la actualidad, especialmente aquellos cuya clasificación cae dentro de las grandes corporaciones. Es decir, la oferta del proveedor tendría que poseer una combinación de varios modelos para considerarse competitivo. Un ejemplo es Google, el cual posee Google Cloud Platform, su oferta de proveedor de infraestructura (IaaS) y plataforma (PaaS). Inclusive, esta misma empresa también se adentra en el *software* como servicio con sus productos de GSuite.

De acuerdo con lo anterior, los proveedores de servicios en la nube pueden usar esta capacidad y poder computacional para formar varios modos de servicio dentro de PaaS, SaaS y IaaS (Onar, Oztaysi y Kahraman, 2018, p. 278), lo cual refuerza, una vez más, la necesidad de los proveedores de tener la creatividad para ofrecer servicios llamativos que incluyan estos tres modelos de manera integrada.

2.1.5 Importancia de la computación en la nube.

Concretizando, la revolución que ha traído la computación en la nube durante esta era digital y tecnológica se evidencia en el crecimiento de consumidores alrededor del mundo. Las compañías ahora buscan reducir diferentes costos de operaciones en los que normalmente incurrirían si decidieran ser dueños de toda la infraestructura tecnológica que conlleva tener una aplicación, sistema, almacenamiento, redes, entre otros.

Por ende, el atractivo recae en simples factores, la mayoría del tiempo, como la reducción del costo de inversión y mantenimiento, además de la fiabilidad que pueden llegar a ser las tecnologías basadas en computación en la nube. También, se debe resaltar lo robusto que se ha vuelto el operar completamente en la nube. Parte de estos beneficios se dan por la rapidez con la que se puede migrar un consumidor de usar soluciones *on-premise* a unas en la nube.

Según un artículo de la revista del CPA (Contadores Públicos Certificados), llamado *Administrando el impacto de la computación en la nube*, expresa que estos beneficios incluyen administrar y tercerizar costos que son difíciles de actualizar, así como de gestionar una infraestructura propia; racionalizando y escalando almacenamiento, *software* y soporte de aplicaciones; aumentando la rapidez de procesamiento, reduciendo costos. (Stein, Campitelli y Mezzio, 2020, p. 20).

Por otro lado, esto supone una oportunidad de crecimiento para la pequeña y mediana empresa porque puede acceder, de forma más sencilla, a servicios tecnológicos de esta índole para soportar el negocio de manera robusta. Sin embargo, más adelante en este documento, se amplía otros riesgos y limitaciones que surgen, especialmente para este sector empresarial, el

cual es el poseer recursos humanos capacitados para operar de manera independiente un ambiente totalmente en la nube.

En efecto, Attaran y Woods (2018) exponen acerca de la limitante mencionada anteriormente, sobre cual afirman que:

> Normalmente en estas firmas no tienen soporte avanzado de TI en el sitio, y es raro que los dueños de pequeños negocios se desvíen de toma de decisiones desinformadas a escuchar el consejo de profesionales de TI. Además, sucede que los pequeños negocios típicamente no tienen recursos económicos para contratar profesionales de esta índole. (p. 101)

Para ejemplificar este escenario, se podría suponer que se tiene una PYME, tipo pequeña empresa, que se dedica a la elaboración y distribución de jabones; tiene una fuerza laboral de 25 empleados, de los cuales el dueño, dos socios y tal vez algún par de empleados del área administrativa hacen uso de correo y herramientas ofimáticas para llevar registros de ventas y demás. Por lo tanto, se puede inferir que no ocupan esta mano de obra calificada como parte de sus gastos.

Sin embargo, conforme avanza la tecnología en el sector comercial, es muy posible que empiecen a perder oportunidades de negocio que podrían potenciar su crecimiento por carecer de un adecuado sistema de inventario o, inclusive, de gestión de compras y gastos. Con este ejemplo, para ellos no sería una opción viable ni contratar empleados de TI, ni invertir en infraestructura, tanto propia como en la nube. A pesar de ello, la computación en la nube tiene un modelo que puede ser un candidato perfecto para ayudar a este tipo de cliente: *software* como servicio.

Debido a esto es que se dice que la computación en la nube es revolucionaria, ya que posee soluciones para todo tipo de clientes, que se adaptan a sus necesidades, no solo de negocio sino monetarias, que implican una reducción de desembolse por parte de estas entidades. Con ello, se le proporciona una ventaja competitiva a la pequeña y mediana empresa de empezar a buscar servicios que realmente le agreguen valor a sus negocios para impulsarlos al siguiente nivel.

Como se puede apreciar, en el artículo llamado "Tecnología de computación en la nube: una opción viable para pequeña y mediana empresa", publicado en la *Revista de Innovación Estratégica y Sostenibilidad*, se sostiene que el *software* como servicio permite a los pequeños negocios accesar muchos servicios a un precio bajo, especificamente, para administrar varias actividades como: contabilidad, nómina, recursos humanos, gestión de relación de clientes, entre otros. (Attaran y Woods, 2018, p. 103)

Como instancia final, es posible ver que la computación en la nube tiene una flexibilidad enorme con respecto a las soluciones que ofertan para sus consumidores. Como se ha venido afirmando, impulsa una nueva generación de empresas proveedoras y clientes que toman ventaja de estas oportunidades para crear productos novedosos que impacten el mercado en la actualidad, y mucho más en las décadas que vienen.

2.2 Arquitectura y Gestión de Entornos de Computación en la Nube

Ciertamente, el diseño de soluciones que hagan uso de servicios en la nube conlleva el involucramiento de aspectos nuevos, tanto en habilidades de TI necesarias, así como configuración, pruebas, integración, compatibilidad, etc. Esto debido a que utilizar servicios de

infraestructura y aplicaciones de *software* en la nube es totalmente diferente a los denominados *on-premise*.

La computación en la nube puede dividirse en dos secciones: la capa de presentación y la capa de datos y lógica de negocio (*Cloud Computing Architecture*, citado en Jadeja y Modi, 2012, p. 878). Estos están conectados a través de la red, usualmente Internet. La capa de presentación es lo que el cliente ve, mientras que la capa de datos y lógica de negocio es el sistema de computación en la nube (Jadeja y Modi, 2012, p. 878). Esta capa, a su vez, la comprenden los diferentes modelos de servicio de la nube. (Odun Ayo, Ananya, Agono y Goddy Worlu, 2018, p. 3)

Desde una perspectiva más general, esta definición compartida por varios autores no representa gran diferencia en cuanto a elementos que la componen de los modelos en capas que siempre se han visto en los sistemas y aplicaciones de *software* en el mercado durante las últimas tres a cuatro décadas. Por ejemplo, tiene similitud, de una manera más reducida, con el modelo de cinco capas y, también, el patrón de modelo-vista-controlador, los cuales son ampliamente conocidos en mundo de la arquitectura de *software*.

Dentro de este orden de ideas, se podría estar hablando de una similitud en componentes, pero una diferencia significativa del origen de estos. En todo caso, haciendo la aclaración de que los servicios expuestos por los proveedores de la nube son encapsulados completamente, es decir, no se puede saber con certeza su composición interna ni estructura, por lo cual se cuentan como un producto terminado que reemplaza muchos elementos que solían existir dentro de un esquema tradicional: como la configuración de servidores, bases de datos, elementos de red: como enrutadores, puentes, etc.

2.2.1 Definición de arquitectura de computación en la nube.

En concordancia con lo expuesto anteriormente y tomando en cuenta los factores que menciona la NIST acerca de la nube: autoservicio bajo demanda, amplio acceso a la red, aprovisionamiento de recursos, elasticidad y servicio controlado (Mell y Grance, 2011, p. 3), es indispensable pensar en una definición que tenga en consideración estas cualidades de la computación en la nube aplicadas a la estructura de sus servicios y cómo sirven a la comunidad.

Por ende, se puede exponer que una arquitectura de computación en la nube es la composición de elementos, tanto de acción recíproca con el usuario, así como aquellos procesos automatizados que hacen posible el tratamiento, procesamiento y almacenamiento de datos, los cuales son envueltos en servicios que forman parte de sus modelos: IaaS, SaaS y PaaS. Asimismo, describe la relación entre ellos de proveer funciones altamente disponibles, de menor costo y con gran retorno de la inversión para sus consumidores.

Otros autores, como Chandrasekaran (2014), en su estudio: *Essentials of Cloud Computing* [*Esenciales de la Computación en la Nube*], expresa que cualquier modelo tecnológico consiste en una arquitectura basada en cuáles funciones modela, que es la vista jerárquica de describir la tecnología. A su vez, también describe el mecanismo de trabajo. (p. 28)

Por consiguiente, siempre existe la relación de componentes, o estructura de la solución, y la interacción entre ellos para proveer la función específica que vaya a soportar. El problema que resuelvan en algunas ocasiones remedia problemas comunes como almacenamiento de datos y manipulación de estos; sin embargo, en muchas otras ocasiones, se concentra en acertar soluciones personalizadas para distintas necesidades de negocio.

2.2.2 Características y elementos que componen la arquitectura de un ambiente en la nube.

Generalmente, la necesidad de ver la arquitectura de una tecnología es fundamental para el aprendizaje de esta y, más aún, para la mejora continua que pueda sufrir durante los años subsecuentes a su nacimiento. Se trata de visualizar sus piezas desde un punto de vista general, ya que existen infinidad de enfoques que sirven para diferentes necesidades.

De acuerdo con un artículo publicado como parte de las actas de la conferencia *In 2015 IEEE 6th Control and System Graduate Research Colloquium (ICSGRC)*, se sostiene que la descripción de la arquitectura está dada por elementos como: la capa base de responsabilidades, el acceso a la red, sistemas operativos, el flujo de trabajo del *software*, la seguridad y privacidad. (Moghaddam, Rohani, Ahmadi, Khodadadi y Madadipouya, 2015, p. 5)

Por su parte, el autor Chandrasekaran plantea la arquitectura desde una perspectiva de capas. Esta se puede dividir en cuatro con base en el acceso del usuario: capa cliente, capa de red, capa de administración de la nube y capa de recursos de *hardware.* (Chandrasekaran, 2014, p. 29)

A pesar de las diferencias en los modelos de servicio, se puede observar claramente una separación de componentes dependiendo del uso e interacción que tienen dentro de la estructura de la arquitectura. Todos los modelos de servicio de operación de la nube comparten los mismos principios, todos ellos dependen de las conexiones a la red a través de Internet, también lo hacen en el *hardware* del lado del proveedor y las aplicaciones de alto nivel que estos ponen a disposición de los clientes por medio de técnicas como la virtualización, entre otros.

En otras palabras, en cuestión de arquitectura se debe tener en cuenta el problema que enfrenta para adecuar los servicios disponibles, desde uno solo o varios proveedores de la nube. Esto último es algo muy frecuente que ha estado sucediendo en los últimos años y de lo cual surge otro término llamado *inter-cloud* (internube en español), que se refiere a la interconexión de nubes o redes de redes (Odun Ayo *et al*., 2018, p. 4). Por ende, se confirma que el acceso a la red es una característica que debe estar mandatoriamente presente para considerarse parte de la arquitectura de la nube.

En la figura 2.1 se puede apreciar un ejemplo de arquitectura en un ambiente en la nube multirregional utilizando el proveedor de la nube pública Google:

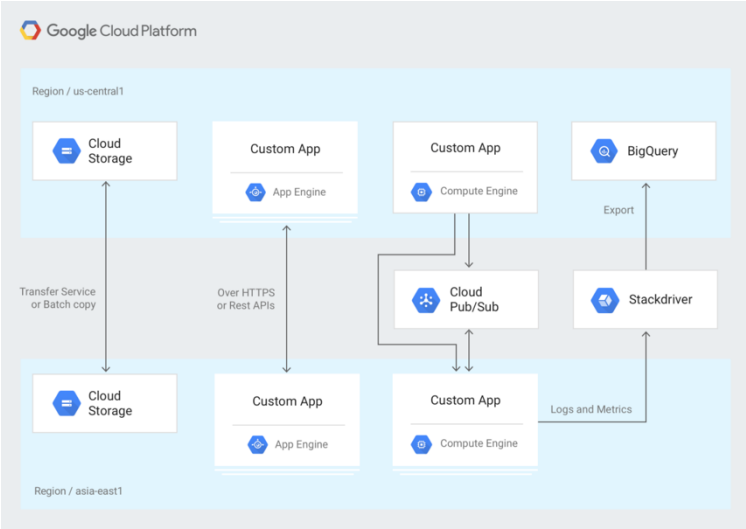


Figura 2.1. Arquitectura multirregional de Google Cloud. Tomado de *Elección de la arquitectura correcta para la distribución de datos globales*, Google, 2021, párr. 38. Recuperado de: https://cloud.google.com/architecture/global-data-distribution?hl=es. CC BY 4.0

En conclusión, se puede evidenciar el uso de diferentes servicios, así como las tecnologías que los conforman, más su interacción entre estos. Esto concuerda con las características de la arquitectura mencionadas anteriormente.

2.2.3 Gestión de la infraestructura de un ambiente en la nube.

La infraestructura en un ambiente de computación en la nube juega un papel importante, no importa el tamaño de la organización. La administración y configuración de los sistemas tiene que ser lo menos tediosa posible, para que no represente gastos, tanto en dinero como en recursos humanos, significativos para las empresas. Esta necesidad de tener métodos y herramientas que permitan una orquestación ágil de estos ambientes ha introducido conceptos, aplicaciones y servicios que facilitan en gran manera este trabajo.

De este modo, desde un punto de vista de actores, se puede hablar acerca de dos categorías de herramientas que ayudan a cada figurante en su cometido: para proveedores y para consumidores. Esto debido a que no es lo mismo administrar la infraestructura completa de un proveedor para dar su servicio, a un ambiente de un consumidor que hace uso de estos para completar su conjunto de sistemas para su beneficio.

En referencia al proveedor, el núcleo de la administración de la nube es la gestión adecuada de recursos. Esto envuelve diferentes tareas internas, como calendarización de recursos, aprovisionamiento y balanceo de cargas, manejado por un *software* referido como "sistema operativo de la nube" (Chandrasekaran, 2014, p. 38). Por su parte, citando a Sotomayor *et al*., manifiesta que el *software* responsable de la orquestación se llama administrador virtual de infraestructura (VIM, por sus siglas en inglés), ya que agrega recursos de múltiples

computadores, presentando una vista única al usuario y aplicaciones. (Sotomayor *et al.*, citados en Buyya, Broberg y Goscinski, 2011, p. 17)

Por lo tanto, se puede hablar de una herramienta o *software*, da igual, que ayuda al proveedor del servicio de la nube a gestionar su infraestructura para dar su servicio eventualmente a sus consumidores. El término "sistema operativo de la nube", así como otros términos incluidos "software de infraestructura compartida" y "motor virtual de infraestructura", se refieren a ésta (Buyya, Broberg y Goscinski, 2011, p. 17). Por supuesto, se basa mucho en la virtualización, procurando mantener las características que hacen a la nube computacional un recurso altamente confiable, disponible y con la capacidad de aprovisionar múltiples usuarios escalando a niveles que, en un esquema tradicional, tendría mucha complejidad, sin mencionar el alto costo.

Algunos ejemplos de administradores virtuales de infraestructura son: ApacheVCL, VMWare vSphere, Citrix Essentials, Nimbus, OpenNEbula, Eucalyptus, etc.; los cuales, por supuesto, presentan propiedades tanto compartidas como únicas para cada caso de uso, incluyendo tecnologías con las cuales son compatibles y soportan. Tal es el caso de Apache CloudStack, cuyos proveedores de servicios en la nube y organizaciones lo utilizan para configurar infraestructura como servicio bajo demanda, así como elástica (Opensourceforu.com, 2013, citado en Kumar y Jain, 2014, p. 113), ya que soporta extensivamente *hypervisors* como XenServer, VMWare, OracleVM, vSphere y KVM (Sosinsky, s. f., citado en Kumar y Jain, 2014, p. 112)

Por otro lado, si se apunta a la administración de una infraestructura en un ambiente de la nube del lado del consumidor, se toma en cuenta otros aspectos, entre ellos, la capacidad de administrar a alto nivel, recursos como almacenamiento, instancias de servidores virtuales,

componentes de red como balanceadores de carga, redes privadas virtuales, traductores de NAT, etc. Asimismo, la velocidad y facilidad con la que una organización puede configurar su infraestructura para escalar sus aplicaciones con base en la demanda, para no incurrir en gastos innecesarios ni pérdida de confiabilidad para con sus sistemas.

Como resultado, se puede concluir que la gestión a alto nivel consta de la parte de *hardware*. Se enfatiza este término porque en un ambiente en la nube solo existe la versión virtualizada del *hardware* y de *software*, como tal. Tal afirmación de Jenkins (2021) introduce dos conceptos: herramientas de configuración de infraestructura y la infraestructura como código, que son similares en funcionalidad; la primera se enfoca en la configuración del *software* del recurso, mientras que la segunda en el aprovisionamiento automatizado de estos. (párr. 3-6)

Ahora bien, en cuanto al primer concepto mencionado, las herramientas de administración de la configuración (CMT, por sus siglas en inglés) son usadas para gestionar y automatizar configuraciones de *software*, para cumplir con sus características de nodos computacionales y provisionar actividades de nodos en infraestructuras de computación distribuidas (Rahman, Enck y Williams, 2020, p. 1). La administración de la configuración se ha convertido en una necesidad para reducir el porcentaje de error y dificultad, la cual ha llegado a ser el mantener y desplegar componentes en infraestructuras grandes. (Goh, 2013, p. 12)

Dentro de este grupo de herramientas se pueden encontrar los siguientes: Chef, Puppet, Ansible, CF Engine, entre otros. Cada uno de ellos posee una estructura similar en la cual tienen un estado esperado de la configuración de los recursos, es decir, una serie de instrucciones que se ejecutan para alcanzar el objetivo, y también de agentes (*software* que se instala en los nodos clientes, generalmente un componente de la infraestructura que se puede administrar). Así pues, se asemeja a un modelo de arquitectura cliente-servidor, en el cual los agentes simplemente

ejecutan instrucciones que el servidor central les ordena, y también pueden enviar estados de la configuración de vuelta para actualizar y ajustar de acuerdo con la condición esperada del recurso. En la figura 2.2 se muestra, a nivel general, la arquitectura de la CMT:

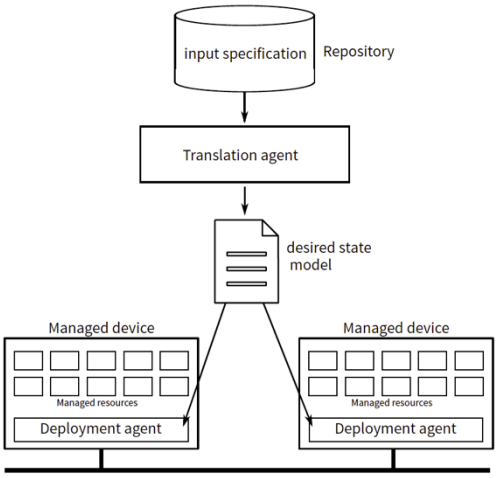


Figura 2.2. Arquitectura de referencia para herramientas de la administración de la configuración. Tomado de: "Configuration management as multi-cloud enabler", elaborado por Vanbrabant, B. y Joosen, W., 2014, *In Proceedings of the 2nd International Workshop on CrossCloud Systems*, p. 2.

En segundo lugar, se abarcará la infraestructura como código, como concepto y utensilio clave en la automatización y gestión de ambientes en la nube. En este caso, mientras que varios autores y profesionales en el campo incluyen el *software* de administración de la configuración como infraestructura como código, lo cual tienen razón desde el punto de vista en el que se basa en escribir instrucciones del estado deseado de los recursos en un lenguaje específico (Hummer, W., Rosenberg, F., Oliveira, F. y Eilam, T., 2013; Patrick Debois, 2012, citado en Vanbrabant y

Joosen, 2014, p. 2); en este segundo caso, se enfoca en aquel *software* capaz del abastecimiento de elementos que conforman parte del ecosistema del ambiente en la nube.

A manera de ejemplo, si se quisiera desplegar una aplicación enorme utilizando los servicios de un proveedor público de la nube, como Amazon o Google Cloud, se ocuparían varias instancias de servidores, tanto web como de base de datos, cubetas para almacenamiento, balanceadores de carga, ponerlos dentro de redes privadas virtuales y demás para realizar una separación lógica y "física" de estos elementos. Ahora bien, la implementación una única vez de esta tarea suena sencillo, pero si se le agrega la complejidad de escalar la aplicación a varios centros de datos y entre servicios, entonces ya no suena fácil. Para ello se requerirá una forma simple de diligenciar los miles de recursos que rápidamente puede requerir una aplicación como la del ejemplo; ahí es donde entra en juego este tipo de herramientas.

Dentro de este marco y haciendo referencia a un documento llamado *Iniciando con Terraform*, se expresa lo siguiente:

> Existe una cierta necesidad de una herramienta de configuración que opere un nivel superior que el establecimiento de un solo servidor, un instrumento que permita escribir un plan de diseño que definiría todas piezas a un alto nivel en una sola acción: servidores, servicios de la nube, y aún, productos de *Software* como Servicio externos. Un artefacto como este tiene nombres varios: orquestador de infraestructura, aprovisionador de infraestructura, creador de plantillas de infraestructura, entre otros. No importa el nombre que se le dé, en un cierto punto del tiempo, la organización lo va a necesitar. (Shirinkin, 2017, p. 12)

Por ende, dentro de esta categoría de productos de *software* de nivel superior se puede encontrar Terraform, AWS Cloud Formation, Pulumi, Ansible, entre otros. Tal como lo expresa

Shirinkin, manejan los recursos como creadores y distribuidores de estos dentro de un ambiente en la nube, no importa el proveedor que se esté utilizando, en el caso de Terraform, pero también existe de manera privada, como lo es AWS Cloud Formation, que solamente se puede encontrar entre los servicios ofrecidos por el gigante tecnológico Amazon Web Services.

2.2.4 Implementación de aplicaciones de *software* en una infraestructura en la nube.

La puesta de aplicaciones en ambientes de la nube se puede realizar de varias maneras. Esto depende del modelo de servicio que se adopte por parte del proveedor, lo cual conlleva a la utilización de servicios más específicos que se usan combinados para tener el *software* funcionando y disponible para el uso de sus consumidores. Todo esto es posible gracias a la diversidad de servicios que ofrecen los proveedores en los modelos de entrega de computación en la nube.

De esta manera, actualmente existen servicios de computación, como máquinas virtuales, también bases de datos, manejo automatizado de infraestructura, redes, inteligencia artificial, almacenamiento, realidad aumentada, Internet de las cosas, publicidad y mercadeo, video y audio, integración y entrega continua, para desarrollo de aplicaciones de *software*, seguridad informática y demás. Cada vez, los mismos proveedores inventan nuevos servicios tomando como base los ofrecimientos que tienen en su núcleo de portafolio, incrementando la oferta y añadiendo la combinación de modelos de entrega de la computación en la nube.

Para ejemplificar, en este caso usando productos de Google Cloud, se puede construir una infraestructura que contenga unas máquinas virtuales que funcionen como servidores web y de base de datos (Google Compute Engine y Google SQL, respectivamente), como el contenedor

principal de la aplicación en su capa de presentación y lógica de negocios con datos, sumado al uso de servicios de almacenamiento de Warehouse, como BigQuery y DataLab para el procesamiento de datos y predicciones para reportes especializados, entre otros. Puede darle a la aplicación un mixto del modelo de servicio que utiliza, ya sea IaaS con PaaS, o IaaS con SaaS o PaaS con SaaS, etc.

Por consiguiente, el operar una aplicación de *software* en la nube no es muy diferente de un ambiente bajo demanda en el aspecto de estructura, es decir, los elementos que se necesitan para hacer funcionar el *software* siguen la misma lógica que la vieja forma; por ejemplo: utilizar servidores y redes lógicas para separar componentes de la infraestructura. En ese sentido, se comprende que la diferencia radica en la manera en que estos componentes son entregados para ser usados: el usar *hardware* en contra de usar *software*, es decir, la forma virtual de cada uno de los elementos físicos, pero con el mismo objetivo.

Ahora bien, ciertamente, el uso de esta tecnología tampoco remueve riesgos que el personal de TI tendrá que solucionar a la hora de operar un producto de *software* en un ambiente virtual como la nube. Dicho en palabras de Mohan, en su trabajo titulado *Migrating into a cloud* [*Migrando hacia la nube*, por su traducción al español], quien expresa que los riesgos se pueden clasificar según su tipo, ya sea que impacten el lado operativo como el desempeño o cumplimiento de regulaciones; más aquellos relacionados con seguridad, como un posible robo de datos a nivel de centro de datos. (Mohan, 2011, p. 54)

Sin embargo, lo anterior no quiere decir que la aplicación requiera algún tipo de elemento especial que pase a formar parte de la infraestructura en la que corre esta. Incluso, si los elementos físicos se convirtiesen en servicios autoadministrados, seguirían teniendo el mismo

objetivo, pero con diferente forma de gestión y entrega. El mismo principio es transportarse de un lado a otro en la ciudad, lo cual se puede realizar en carro propio, taxi, tren, bus, a pie, etc.

2.2.5 Modelos de Entrega e Integración continua en una infraestructura en la nube.

La industria del desarrollo de *software* ha batallado a través de los años en contra de las malas prácticas que terminan en pésima calidad del producto y problemas cuando funciona en producción. Las pruebas de *software*, por un lado, han tenido gran impacto en la calidad de las aplicaciones que se crean. Otro punto es la intervención manual que, en muchas ocasiones, influye en errores que se presentan cuando se le otorga el programa al consumidor; mayormente, tecnología emergente y demanda de nuevas ideas a la brevedad posible. Es por ello, que los modelos de entrega e integración continua han apoyado a los equipos de desarrollo, considerablemente, para evitar caer en estos fallos.

Empleando las palabras de Arachchi y Perera (2018), se dice que se relaciona con los principios de las metodologías ágiles donde se busca satisfacer al cliente con la entrega temprana y continua de *software* con valor, debido a que provee más eficiencia y productividad en procesos ágiles (p. 2). Así pues, se resume en ayudar a la capacidad de apoyar el proceso de mejora continua en un producto de *software*. Ejemplos como la entrega de mejoras, arreglo de errores e introducción de nuevas funcionalidades se pueden encajonar dentro de esa definición.

Ahora bien, se pretende, entonces, definir ambos procesos que parecieran ser lo mismo, ya que apoyan el mismo objetivo general, pero tienen responsabilidades distintas. Comenzando con la integración continua, según el autor Ravi Teja Yarlagadda, en un artículo publicado en el *Journal of Emerging Technologies and Innovative Research* [*Revista de Tecnologías Emergentes*

e Investigación Novedosa], se define este concepto como una metodología de desarrollo de *software* en la cual los programadores están a cargo de la incorporación de actualizaciones al código en un repositorio único y compartido, seguido de revisiones y compilaciones automáticas, para encontrar y corregir errores tempranamente. (Yarlagadda, 2018, p. 1421)

A continuación, para el concepto de la entrega continua, los autores Somya Garg y Satvik Garg (2019), en un artículo publicado en el acta de congreso *In 2019 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)* [*Conferencia sobre el procesamiento y recuperación de información multimedia en 2019 IEEE*], manifiestan que es una política de la ingeniería del *software* que se centra en reducir el tiempo requerido para producir un *software* (…) es decir, un enfoque en desplegar cambios pequeños de manera más frecuente. (p. 2)

Volviendo al punto de la demanda, cabe destacar una conclusión de lo anteriormente mencionado: que la característica principal de esto es la automatización. Se aprecia que la entrega continua no es más que un elemento, sumamente importante, que viene a formar parte de una serie de pasos que se tiene que realizar antes de desplegar una actualización de código en el ambiente de producción. Esto supone tener listos procesos que sean capaces de hacer migraciones de código al ambiente de producción de forma ágil, segura y eficaz, sin intervención humana.

Por consiguiente, a la hora de colocar controles sistemáticamente dentro de línea de desarrollo y pruebas de código, así como de la migración de este al ambiente de producción, se puede incrementar la productividad de los equipos involucrados en el ciclo de desarrollo de *software* para preocuparse menos por temas administrativos y más por aquellos que requieren mayor atención, como el terminar los entregables comprometidos y también el corregir errores que aparecen dentro de las operaciones de las aplicaciones.

Ahora bien, estas prácticas no son ajenas a la operación de la creación y entrega de *software* utilizando ambientes en la nube. Inclusive, no existe incompatibilidad por la cual herramientas como Jenkins, Bamboo, Circle CI, Git, GitLab, BitBucket, CodeShip tienen una integración sumamente completa con tecnologías de la nube de la mayoría de los proveedores existentes en el mercado. Además, de la misma forma, estos mismos proveedores han creado sus propios servicios, los cuales proveen de funcionalidades de integración y entrega continua completas o parcial, es decir, se complementan con alguna otra herramienta.

La figura 2.3 muestra la integración de estas herramientas con los servicios ofrecidos por el proveedor de la nube, en este caso, el de Google.

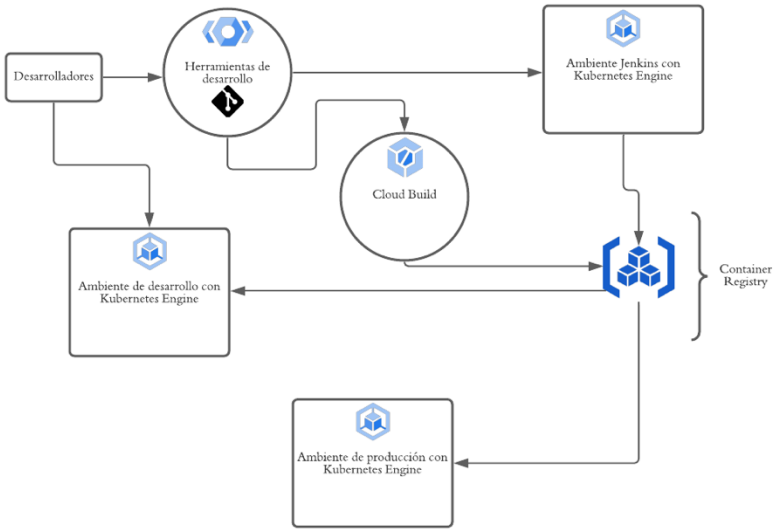


Figura 2.3. Implementación continua usando servicios de la nube de Google. Adaptado de "Implementación continua en Google Kubernetes Engine mediante Jenkins", Google, 2021, *Cloud Architecture Center*, párr. 1. Recuperado de: https://cloud.google.com/architecture/continuous-delivery-jenkins-kubernetes-engine. CC BY 4.0.

De esta manera, gracias a la utilización de interfaces de programación de aplicaciones (API's), la integración de estos servicios se realiza de forma rápida y eficiente. Una de las ventajas recae en la compatibilidad con las tecnologías con las cuales la organización esté trabajando. No importa si no se tiene licencia para operar con BitBucket como repositorio de control de versionamiento, si se tiene acceso a GitHub, entonces, se tendrá las mismas capacidades de acceso a consumir servicios de la nube. Esto aplica para cualquier herramienta de terceros que se tenga para administrar el canal de integración y entrega continua.

Debido a esto, el punto es demostrar la flexibilidad que existe entre tecnologías y cuál se puede aprovechar de gran manera, contrario a utilizar ambientes locales que, potencialmente, pueden tener problemas con versiones no soportadas entre sí, etc. Es por esto que se puede ver en la figura 2.3, una pequeña variación entre usar Jenkins o un servicio directo de Google llamado *Cloud Build* ("Constructor de la nube"), inclusive, ambos al mismo tiempo, lo cual no afecta los otros componentes de la infraestructura.

2.2.6 Gestión de operaciones de aplicaciones de *software* en una infraestructura en la nube.

La operación de una aplicación de *software* en un modelo de infraestructura en la nube tradicional no es más que la preocupación de dos aspectos muy importantes: primero, el gestionar la infraestructura, preferiblemente de manera automatizada, para soportar la carga y el trabajo que genera el aplicativo de *software*; y segundo, la gestión del *software* como tal, que puede ser divido, a su vez, en dos grandes partes, como el ciclo vida de desarrollo, así como el monitoreo del comportamiento de las transacciones y eventos que la aplicación produce como parte de su funcionamiento normal en tantos ambientes esta esté presente.

Considerando los puntos anteriores, es donde los modelos de servicio de la nube pueden mezclarse para proveer las herramientas necesarias a los consumidores dependiendo de su capacidad financiera y técnica. Por ejemplo, como lo expresa Chandrasekaran (2014), una desventaja de administrar completamente toda la infraestructura es la carga de trabajo en el mantenimiento y la inversión de dinero que los proveedores deban destinar a este aspecto. (p. 220).

En otras palabras, no siempre es bueno hacer uso de la infraestructura como servicio, plataforma como servicio o *software* como servicio por completo. Cada caso de uso debe analizarse cuidadosamente para verificar la factibilidad de uno sobre el otro en ocasiones específicas. Lo que para una empresa grande puede funcionar, puede que no sea una opción viable para otra más pequeña.

Por ende, al combinar varios servicios del proveedor en la nube que precisamente se clasifican en modelos separados, entonces, el operar una aplicación que está en la nube se convierte en una aventura caótica si no se está familiarizado con la plataforma. Podría hablarse de que este punto es una situación que considerar cuando se adquieren los servicios de un proveedor A o B; en otras palabras, tener el entrenamiento indicado por plataforma. Lo anterior, debido a que algunos proveedores tienen herramientas específicas para monitorear algunos de sus servicios, especialmente aquellos bajo la modalidad de PaaS y SaaS.

2.3 Seguridad Informática en Relación con la Computación en la Nube

La era de la información y computación en la nube ciertamente ha introducido nuevos riesgos en la protección del activo más preciado de toda organización: los datos. Con las

increíbles invenciones tecnológicas, las amenazas crecen y mutan para tratar de penetrar en los sistemas que, de alguna u otra forma, aseguran una parte, pero dejan al descubierto otra.

Según Whitman y Mattord (2009, citados en Monteiro Dias, Oliveira Zacarias, de Lima Varella y Pereira dos Santos, 2022), se define la seguridad de la información como la protección de los datos y sus elementos críticos, incluyendo los sistemas y *hardware* que usan, almacenan y transmiten esa información. Identifican varias características críticas como: confidencialidad, integridad y disponibilidad (triada CID). (p. 2)

Es decir, la confidencialidad se refiere a la protección de los datos de aquellos que no están autorizados, la integridad tiene que ver la confianza que se tiene en estos, y la disponibilidad es tener la información por cuanto sea posible (Neogy, 2017, p. 1). Por consiguiente, todos los demás principios y elementos se derivan de estas características que forman parte de la seguridad de la información.

En tal sentido, para una mayor comprensión de aspectos de seguridad en los servicios de computación en la nube se sugiere revisar los controles que dicta la norma ISO 27000, específicamente la ISO 27017. Este estándar proporciona a proveedores y consumidores del servicio, la habilidad de implementar controles de seguridad para servicios de computación en la nube. También, es una extensión de la norma ISO 27002, que se enfoca en seguridad en la nube. (Hsu, 2018, sección de ISO 27017 y ISO 27018, párr. 2).

Ahora bien, como afirma la ISO/IEC (2015), es conveniente mencionar que:

> Los nuevos siete controles que trae esta regla son: roles y responsabilidades compartidos en el ambiente de computación en la nube, eliminación y retorno de activos del cliente del servicio de la nube, segregación en ambientes de computación virtuales, seguridad

> operacional del administrador, monitoreo de servicios, alineamiento de la gestión de la seguridad para las redes físicas. (Citado en Arafat, 2018, p. 4)

Por otro lado, OWASP, organización no lucrativa con alcance global que tiene la misión de aportar para la constante mejora de la seguridad en aplicaciones y sistemas de *software*, proporciona una lista de los riesgos de seguridad en los servicios de computación en la nube en la cual los controles de esta norma ISO 27017 sirven para mitigar el riesgo. Por ejemplo, según este proyecto:

Existen riesgos como: R4 – Continuidad del negocio y resiliencia, R6 – Servicio e integración de datos, y R9 – Seguridad de la infraestructura, en la cual controles de la norma ISO 27017 como: S17 – aspectos de seguridad de la información de continuidad del negocio, S10 – criptografía, y S9 – control de acceso; respectivamente, pueden ser aplicados y eliminar esos riesgos. (Ivkic, Wolfauer, Oberhofer y Tauber, 2017, pp. 3-4)

Asimismo, se puede decir lo siguiente acerca de la computación en la nube en términos de seguridad informática:

> Coloca datos de las empresas en las manos de proveedores externos, lo cual consecuentemente, provoca que el cumplimiento de los controles regulatorios se vuelva riesgoso y más complejo que cuando los sistemas son administrados y alojados en centros de datos propios. La pérdida de visión directa significa que la compañía cliente debe verificar que el proveedor del servicio está trabajando para asegurar que la seguridad e integridad de los datos está dentro de las prioridades y aspectos básicos de sus ofertas. (Chandrasekaran, 2014, p. 328)

Otro aspecto por mencionar radica en que la administración de la seguridad de la infraestructura de una organización es una parte fundamental para el éxito de un producto de *software* que esté soportado por esta, por lo cual, el cumplimiento de los controles de seguridad para los datos debe estar en primer lugar; además de que esto también llega de la mano de regulaciones gubernamentales de cada país en los cuales se presta el servicio. Inclusive, esta situación ha supuesto un reto para la computación en la nube en un ámbito global, ya que países cuyas leyes son muy estrictas, son mercados difíciles de acaparar debido a esta naturaleza.

Sin embargo, el principio de seguridad primero es una responsabilidad de todos los participantes del entorno de la computación en la nube, esto incluye a ambos: consumidores y proveedores; es decir, un modelo de responsabilidad compartida, como lo menciona la norma ISO 27017. Por su parte, los proveedores deben brindar controles de seguridad en sus servicios; tanto a nivel de almacenamiento de datos como a nivel de manipulación de estos a través del *software*. Al final, son responsables directos de la información y del producto que ellos desarrollaron y entregaron a sus consumidores. No obstante, los clientes deben tener claro que la gestión correcta de estos servicios, en conjunto con los controles de cómo éstos son usados para administrar su propia información, es también crucial para una adecuada práctica de seguridad dentro de un ambiente de computación en la nube.

Por consiguiente, en cuanto al uso de estos controles, se puede mencionar que el acceso a los recursos que contienen los datos es un nivel de seguridad, y el acceso a la información como tal, es otra capa de seguridad que se debe cumplir para poder satisfacer adecuadamente las regulaciones que dictan, en primer lugar, el mercado, y segundo, las buenas prácticas.

En cuanto al acceso a los recursos existen distintos métodos, como la administración de identidad con herramientas sofisticadas de gestión de acceso y permisos a las aplicaciones. Así,

también, los proveedores cuentan con servicios nativos de esta naturaleza, como son los permisos granulares por dominio; en otras palabras, que se enfocan en seguir el principio de *Least Priviledge Access* (acceso de menor privilegio), que consiste en dar permisos de acceso a cuentas de usuario o cuentas de servicio, solamente a los recursos e información que necesitan, ni más ni menos. Algunos ejemplos son: el servicio *AWS Identity And Access Management* (gestión del acceso e identidad de AWS), y de forma similar *Google Identity And Access Management* (gestión del acceso e identidad de Google), entre otros.

Por otro lado, en términos de seguridad de datos se sostiene que:

> El acceso de control que se centra en la información (contrario a las listas de control de acceso) puede ayudar a balancear accesibilidad mejorada con riesgo, asociando reglas de acceso con objetos de datos diferentes con una plataforma abierta y accesible, sin perder usabilidad inherente de esa plataforma. (Morrow, 2011, p. 576)

En otras palabras, se converge en el mismo punto de la flexibilidad de estos mecanismos de adaptarse con otros sistemas de manera sencilla y dinámica. Esto supone un gane, de nuevo, para las organizaciones en esfuerzo, tiempo y, mucho más importante, dinero. Ahora bien, eso supone un ahorro en la implementación y adaptación de una tecnología a otra, pero se sigue teniendo el reto de evaluar el nivel de cumplimiento de estos aspectos por parte de la empresa proveedora del servicio en la nube.

Otras áreas en las cuales la seguridad es importante son la de virtualización y redes de computadores. En cuanto a virtualización, según Chandrasekaran (2014), esta altera la relación que existe entre el sistema operativo y el *hardware* (…) lo que conduce a un potencial riesgo es el hipervisor de las máquinas virtuales que lo convierte en un objetivo de explotación primario (p. 334). Por ende, mantener actualizadas todas estas herramientas, con respecto a su fabricante,

se vuelve una prioridad para salvaguardar la integridad de la virtualización de las máquinas virtuales y, consecuentemente, la de la información que guarda.

En relación con este tema, el impacto de la seguridad en la red de comunicación es otro aspecto primordial, sobre todo porque la computación en la nube basa su intercambio de datos por medio de Internet; en otras palabras, haciendo uso de redes de computadores. Bien lo menciona el autor Chandrasekaran en su propuesta de computación en la nube, que algunos de los retos en este campo comprenden: el desempeño de las aplicaciones, la flexibilidad del despliegue de equipos, cumplimiento de políticas, complejidad de topologías dependientes, reescritura de aplicaciones, dependencia de ubicaciones y complejidad multirred. (Chandrasekaran, 2014, p. 337)

Como última instancia, sin duda, un aspecto importante a considerar por parte de los consumidores y proveedores de servicios de computación en la nube se sustenta en los acuerdos de nivel de servicio (SLA, por sus siglas en inglés). De acuerdo con ServiceNow (s. f.), los acuerdos de nivel de servicio son un contrato entre el proveedor del servicio y el cliente, definiendo los tipos y estándares de servicios a ofrecer (párr. 1); es decir, un conjunto de reglas que todas las partes involucradas en la prestación y recepción de un servicio deben de seguir muy estrictamente.

Se plantea, entonces, que el reto de establecer SLA's en seguridad es la naturaleza cualitativa de los parámetros de seguridad. Cuantificar estos parámetros permite a terceros monitorear y velar por el complimiento de los niveles de servicios de seguridad entre proveedores y consumidores. (Rahulamathavan *et al.,* 2014).

Citando a la Cloud Security Alliance [Alianza de Seguridad en la Nube] (2011), se ha identificado que especificar seguridad en los acuerdos de nivel de servicio (dando paso al

término "acuerdo de nivel de seguridad") es provechoso para modelar y evaluar la seguridad ofrecida por el proveedor del servicio. (Citado en Luna García, Langenberg y Suri, 2012, p. 103)

Esto, a su vez, permite el sano ejercicio de la prestación de servicios de computación en la nube, incluyendo el de *software* como servicio. Además, la transparencia de los proveedores con respecto a este tema facilita la generación de confianza en el producto o servicio que se adquiere por parte de los consumidores.

Además, se pudo observar que existen actualmente mecanismos de estandarización, como la norma ISO 27017, que se enfocan en la seguridad de información en aplicaciones y ambientes de computación en la nube, que dan credibilidad a los servicios de la nube y, como proveedor, es importante tener este aspecto en cuenta porque da garantía a los clientes de que el producto de *software* que se les vende es confiable y seguro.

2.4 Modelo de Servicio de Computación en la Nube: Software como Servicio

Como se introdujo en las primeras secciones del documento, incluyendo el estado del arte, el *software* como servicio es el propósito principal de esta investigación por encima de otros modelos, como el de infraestructura o plataforma como servicio. En este aspecto, el *software* como servicio ha popularizado su empleo debido a los bajos costos que impone el comprar su uso, más el mantenimiento de la aplicación, lo cual es nulo y, en cierta forma, trasladado al proveedor como tal.

Sin embargo, la simpleza del modelo recae en la entrega, en cómo el cliente lo percibe y lo utiliza. La misma situación ocurre con sistemas operativos como Windows o Mac, en los cuales los usuarios tienen una mayor aceptación del *software* porque les permite realizar sus

tareas de forma sencilla y, mejor aún, no les demanda mucho conocimiento técnico ni procesos tediosos para llevar a cabo una acción; por ejemplo: guardar un documento, crear una carpeta, instalar una aplicación o tomar una fotografía.

Eventualmente, poco se habla del cómo se diseña, construye, mantiene y distribuye a los clientes, por eso es que en los siguientes puntos se trata de definir lo que es un *software* como servicio, además de los elementos que lo componen, así como su estructura para entender cómo están hechas las aplicaciones que hacen uso de este modelo.

2.4.1 Características del modelo.

Así como los otros modelos, este también tiene propiedades que determinan la naturaleza del servicio de entrega, entre las cuales se pueden mencionar las que se citan a continuación:

Es un servicio que se ofrece bajo la modalidad de suscripción; es decir, los clientes no compran el *software* completo, sino que adquieren una membresía, la cual se paga según la frecuencia ofertada por el proveedor que le da el derecho al consumidor de utilizar el *software* y todas sus características y funciones.

Se accede completamente a través de la red. Como muchos de los servicios de computación en la nube, el método más común de entrega se realiza por medio de Internet, siendo la interacción únicamente a través del navegador. A su vez, esto representa una ganancia para el cliente, ya que no se preocupa por la compatibilidad de su sistema operativo con el *software*, ni tampoco la ubicación geográfica, ya que puede acceder a la aplicación siempre y cuando tenga conexión a Internet.

Es autoescalable con la demanda de tráfico y carga. Cuando se hace uso de un *software* de este tipo, no es necesario estar preocupándose por la carga de trabajo que se pone en manos

del aplicativo, ya que normalmente, estas responsabilidades caen en los hombros de los proveedores del servicio.

Es multi-inquilino; en otras palabras, se puede tener múltiples instancias de la misma aplicación completamente aisladas para servir a un cliente por instancia. Esto supone un contenedor privado para cada cliente, en el cual sus datos van a existir solamente para ellos y no para varios consumidores al mismo tiempo dentro del mismo espacio lógico.

Teniendo en cuenta estos aspectos, se hace referencia a lo expuesto por Fox y Paterson (2014) en su libro titulado *Engineering software as a service: an agile approach using cloud computing* [*Ingeniería del software como servicio: un método ágil utilizando computación en la nube*], para ejemplificar estos principios que debe seguir todo *software* como servicio, donde se indica que:

> El *software* como servicio establece tres demandas en la infraestructura de las tecnologías de la información: la comunicación: para permitir a cualquier cliente interactuar con el servicio; escalabilidad: ya que las instalaciones principales que corren el servicio deben lidiar con las fluctuaciones en demanda durante el día y tiempos populares durante el año para ese servicio y nuevos para agregar usuarios de manera rápida; disponibilidad: en la que ambos, tanto el servicio como el vehículo de comunicación, deben continuamente estar disponibles: todos los días, 24 horas al día ("24x7"). (Fox y Patterson, 2014, pp. 49-50)

Así, pues, se establece una importancia que se le da a la disponibilidad del servicio, no solo en términos de "ininterrupción", sino, también, en la facilidad de acceso al servicio mismo. Como se mencionó anteriormente, siguiendo esta lógica de pensamiento, se habla entonces de un producto altamente alcanzable para cualquier cliente en cualquier parte del mundo, sin importar

mucho su entorno de equipos electrónicos para consumir el servicio. Por ejemplo, solo basta tener una computadora, una tableta o un celular para ingresar a la aplicación y utilizarla para obtener un resultado.

De esta misma manera, Namasudra (2018) expresa lo siguiente acerca de este punto, en un modelo de SaaS, una versión del *software* licenciado es proveída por el proveedor. Cualquier compañía o usuario puede comprar este *software* bajo demanda (p. 117). Sin embargo, en este sentido difiere un poco en el término "licencia" que utiliza el autor, aunque no equivocadamente, se piensa que hace referencia más bien a la suscripción para usar el *software* sin requerir elementos adicionales, ni poseer *software* de forma local.

Esto supone una ventaja para los proveedores de productos bajo este modelo debido a que, en primer lugar, rápidamente pueden servir a muchos clientes a la misma vez con el mínimo esfuerzo. Esto incluye las actualizaciones y mejoras que se entregan en todo *software*, no importa de qué tipo sea. Al ser los dueños de la aplicación, tener la administración de la infraestructura y código fuente, se tiene mayor control sobre lo que sucede en todo momento.

En segunda instancia, la adopción del servicio es significativamente más sencilla y rápida que cualquier otra aplicación entregada bajo otra modalidad. Claro está, haciendo la aclaración de que existen aplicaciones gigantes a nivel empresarial, las cuales tienen un poco más de complejidad al adoptarlas, como por ejemplo los sistemas de tipo *Customer Relationship Management* [Gestión de la relación de clientes], los *Enterprise Resource Planning* [Planificación de Recursos Empresariales] o los *Human Capital Management* [Gestión del Recurso Humano]. No obstante, el principio de implementación de rápido acceso y uso no deja de existir.

En tercer lugar, se puede mencionar las capacidades que tienen estos productos de integrarse con otros sistemas a través del uso de API's, *Middlewares* y otro tipo de servicios y herramientas que gestionan líneas de comunicación por mensajes para conectar aplicaciones en un ambiente empresarial compuesto por varios sistemas de *software* de esta naturaleza. Sin embargo, existe una desventaja con respecto a este punto, la cual reside en lo siguiente: aunque varios proveedores de SaaS han desarrollado API´s integrados en sus productos, esto introduce retos como el de codificar y mantener la solución que conecta esos servicios web, y además existe poca estandarización en la estructura de estos. (Pethuru, 2011, p. 62)

Luego, otra consideración en el aspecto de la integración que menciona este mismo autor es la poca experiencia que puedan tener las empresas al sincronizar los datos entre aplicaciones de este tipo en su ambiente, una tarea que se puede convertir en una pesadilla. Esto, en cierta forma, tiene razón, ya que crear un proceso de integración entre aplicaciones puede llegar a ser todo un proyecto aparte y tan tedioso como desarrollar la aplicación misma.

Por el contrario, aquí es donde ese bloqueo se puede llegar a solventar con ayuda de la orientación del *software* basado en una arquitectura de microservicios, en la cual las interfaces de programación de servicios web sea la base de la comunicación. Además, se puede tomar ventaja de otros servicios en la nube que proveedores ya ofertan, como por ejemplo, Amazon provee el servicio de mensajes de *Simple Message Queue* (SQS) [cola de mensaje simple], también el de *Lamda*; Google por su parte tiene *App Engine* [Motor de aplicación], *Google Functions* [Funciones de Google] entre otros que facilitan en gran manera el implementar servicios de integración que se ejecuten solamente con ciertos eventos y que se comuniquen por API´s.

Para resumir sobre el punto anterior, el principio de uso de servicios prediseñados y ofrecidos nativamente por los proveedores de computación en la nube se promueve como la

opción preferida para utilizarse en conjunto con *software* como servicio para construir un ambiente altamente integrado y sincronizado en tiempo casi real.

2.4.2 Sustentabilidad del software como servicio.

El objetivo principal de los sistemas de *software* es solventar una problemática que tengan los usuarios, o bien organizaciones para las cuales operan. Dicho esto, se tiene que pensar en la forma en la que dicho *software* es administrado, ya que en esto recae un sinnúmero de características que este modelo posee y que son cruciales para su funcionamiento, como: la disponibilidad, escalabilidad, confiabilidad, etc. El no cumplir con esas demandas puede acarrear serias consecuencias en las operaciones en las cuales estén involucradas.

La sustentabilidad está fuertemente relacionada con el *software* que vive por mucho tiempo, en el cual, sin importar el énfasis que tenga en el desarrollo de *software*, este se verá influenciado en el sistema con calidad pobre (Ahmad, Hussain y Baharom, 2015, p. 2). Por ende, se tiene que valorar este concepto primero con la duración del producto en buen estado, y segundo, con el funcionamiento de este durante el uso que se le dé. Los errores, las mejoras, los cambios que producen deuda técnica en un producto de *software* no solo degradan su imagen, sino que cuestan mucho dinero a las organizaciones.

En este mismo sentido, otros autores expresan que la sustentabilidad del *software* describe las prácticas, ambas técnicas y no técnicas, que permiten al *software* continuar operando como se espera en el futuro. Un nivel constante de esfuerzo se requiere para mantener la operación (Hettrick, 2016, p. 7). Acá se evidencia que no solamente se refiere al desarrollo de un *software* capaz de tener errores mínimos y el menor mantenimiento posible, sino que el factor

humano entra en juego al ponerlo como actor principal dentro de la definición misma para que se cumpla en un producto de *software*.

Basado en lo anterior, cuando se traslada al modelo de *software* como servicio se evidencia que tiene una ventaja relativa con respecto a una operación de una aplicación en un modelo tradicional. Se podría decir que la primera es la gestión del *software* que está en responsabilidad de una entidad en lugar de dos; contrario a lo que pasa en un esquema tradicional en el cual se tiene que ver involucrado, en la mayoría de las ocasiones, el equipo de tecnologías de la información o alguna persona encargada del cliente, en conjunto con el proveedor.

Si se apela a un ejemplo aplicable al *software* como servicio para alcanzar la sustentabilidad, se debe incrementar el número de usuarios substancialmente (Foster, Vasiliadis y Tuecke, 2013, p. 4). Acá existe un reto para este modelo y es la adopción por parte de los usuarios en ciertos campos. Notoriamente, no todos los problemas de negocio se pueden resolver fácilmente por un producto de este tipo, esto debido a las limitantes que existen alrededor del *software* como servicio puro, como es la desventaja de cubrir cada una de las necesidades de negocio existentes dentro de su público meta.

Por lo tanto, los usuarios forman parte crucial en este modelo, ya que sin usuarios no hay ingresos, y sin ingresos no existe forma rentable de seguir manteniendo completamente el *software*. Es de esta forma que los autores Foster, Vasiliadis y Tuecke (2013), pensando en esta limitación, expresan lo siguiente para proponer una solución válida para este escenario, ya que la sustentabilidad demanda una mayor cantidad de usuarios; entonces, para alcanzar varios usuarios se debe ser capaz de comunicar el valor de un producto a los clientes, la verdadera definición de mercadeo. (p. 4)

Ahora bien, Hustad y Olsen (2021), en un artículo publicado en la Revista *Procedia Computer Science*, expresan lo siguiente:

> Cuando se implementa SOA [Arquitectura orientada a servicios], la infraestructura de TI se parte en diferentes servicios. Estos grupos independientes de servicios interactivos ofrecen interfaces bien diseñadas que permiten la interacción con otros servicios. Estos servicios representan un proceso de negocio o una parte de uno. Esta arquitectura permite la reutilización de servicios enlazados para crear nuevos. Un servicio puede comprometer una o más funcionalidades específicas para las cuales puede ser usado cuando estas se requieren. (p. 599)

Lo anterior refleja la estrecha relación que tiene este modelo con esta arquitectura y por qué tiene un alto grado de confiabilidad al crear un ambiente deseado para que exista sustentabilidad del producto, una característica esencial para obtener una menor inversión en la gestión del *software*.

En primer lugar, el objetivo de separar y desacoplar, tanto que se escucha en la teoría y práctica del desarrollo de *software*, se cumple en este principio porque ofrece una flexibilidad de implementación del o los servicios para que construya un todo, donde la comunicación y envío de datos se da de forma casi “estándar”, que funciona con cualquier tipo de tecnología.

En segundo lugar, abre espacio para el manejo de errores de forma adecuada. Al tener que concentrarse en una sola tarea, los servicios se pueden revisar y corregir de forma más ágil, ya que no requiere esfuerzo de ir a toda la aplicación monolítica para saber qué parte está fallando. Esto garantiza que existan métodos de respaldo para el flujo del proceso en el cual la aplicación siga funcionando, aun cuando uno o varios servicios no estén disponibles.

En este sentido, se puede comprender que la combinación de SOA con la computación en la nube puede incrementar la sustentabilidad, llegando a ser más apta, a ser óptima en utilización de recursos en un largo plazo (Hustad y Olsen, 2021, p. 601). Por ende, se convierte en un candidato fuerte a llevarse un galardón, en cuanto a este aspecto se trata, por el simple hecho de facilitar la administración de recursos para asegurar una experiencia tanto de usuario como de proveedor lo más agradable posible, reduciendo costos y complejidades técnicas en las operaciones del producto de *software*.

Recapitulando, la llave del éxito de la sustentabilidad del *software* como servicio está en la característica del modelo como tal, es decir, en cómo se diseña y desarrolla, y en cómo se entrega al cliente. El mantenimiento que se le da al *software* se vuelve menos tedioso si se usa una arquitectura basada en servicios, más aún si estos se combinan con los ya ofrecidos por proveedores públicos de computación en la nube. Proporciona una serie de beneficios para las operaciones del *software* en el cual menos recursos son invertidos para mantenerlo corriendo.

Además, fue posible expresar la importancia que tiene la base de clientes para la vida a largo plazo de los sistemas entregados bajo esta modalidad. El retorno de la inversión se verá impactado por la adopción de los consumidores y qué tan rápido éstos se integren al servicio.

2.4.3 Relación de los modelos de *Software* como Servicio y la Infraestructura como Servicio.

A pesar de que ambos modelos de entrega de computación en la nube sirven para diferentes propósitos, es viable decir que existe una relación que los envuelve dentro del mismo

grupo de ofertas de servicios que buscan facilitar la administración de procesos y recursos tecnológicos de una forma óptima.

Por ejemplo, ambos ofrecen una suscripción a sus servicios en vez de vender el producto como un todo. Se aprovisiona recursos para cumplir con la demanda y se elimina recursos no utilizados para los dos. Entonces, se habla de una semejanza en la operación de estos, donde los usuarios optan fácilmente por consumir uno o el otro, o los dos al mismo tiempo.

Sin embargo, en esta sección se busca encontrar la combinación de estos modelos para entregar una solución completamente en la nube, que cumpla de igual forma con todas esas características propias de los servicios de la nube.

Así lo hace notar el autor Chandrasekaran (2014), quien expresa que debido a que la mayoría de los servicios SaaS aprovechan PaaS y IaaS para su desarrollo y despliegue, entonces asegura tener más escalabilidad que el *software* tradicional (p. 87). En este sentido, se está de acuerdo con que el camino deseable para construir aplicaciones como servicios robustas, fiables, escalables y sustentables es la combinación de todos los servicios y modelos para tomar ventaja de las fortalezas de cada uno y así, proveer una solución estable.

En vista lo anterior, esto es contrario al esquema tradicional y semitradicional, en el cual un *software* como servicio está construido, desplegado y operado en un ambiente totalmente local, o en un modo híbrido, en el cual cierta parte está migrada y operada en la nube, y otra en un centro de datos propio o rentado de algún tercero. Ciertamente, ofrece control manual sobre ciertos procesos y recursos, pero aumenta la carga para la organización. Esto último contradice los principios de la computación en la nube, ya que su meta principal siempre será el facilitar la gestión de los recursos tecnológicos en todas las formas posibles.

2.5 Arquitectura y Tecnologías de Aplicaciones de Software como Servicio

En esta sección se pretende abarcar un aspecto importante a considerar cuando se diseña y construye un *software* de este calibre: la arquitectura. Y es que, para el *software* como servicio, se ha encontrado popular el principio de que "todo es un servicio", para pasar de aplicaciones llamadas monolíticas a un entorno orientado a servicios.

Para nadie es un secreto que el fundamento de "divide y vencerás" es sumamente utilizado en muchas disciplinas y escenarios para resolver problemas complejos y grandes. Esto proporciona una ventaja a la hora de revisar parte por parte hasta encontrar la causa raíz, ya que no requiere invertir un esfuerzo mayor por buscar en un conjunto de elementos mucho más grande. Por ejemplo, no es lo mismo buscar un número en un arreglo de *n* cantidad de elementos todo en memoria, a procesar partes de ese arreglo, ejecutar la búsqueda y seguir procesando la siguiente parte; en especial cuando se trata de un arreglo con trillones de elementos.

Dentro de este orden de ideas, es por lo cual se considera que esta analogía aplicada a los servicios viene a ejemplificar este concepto dentro del mundo de las arquitecturas de *software*. Para evitar tener aplicaciones con muchas dependencias en las cuales hacer un cambio, se vuelve una tarea tediosa que demanda mucho tiempo por parte de los desarrolladores; además, de que se está más propenso a introducir errores que corregirlos.

2.5.1 Definición de arquitectura de *software*.

En primer lugar, para llegar al enfoque que el *software* como servicio posee, se tiene que entrar en la definición de lo que es la arquitectura de *software* como tal. A pesar de que esta industria ha estado en el mercado durante bastante tiempo ya, el concepto que tiene que ver con

el diseño de los sistemas y su relación con el ciclo de vida de *software* es fundamental para cumplir con la demanda que los usuarios y organizaciones tienen del *software* que se les entrega.

Desde la posición de Perry y Wolf, se puede definir como:

> La combinación de un conjunto de elementos arquitectónicos, ejemplo: procesamiento, datos y conectividad, la forma de estos elementos, como principales guías de la relación entre los elementos y sus propiedades, y la razón fundamental para escoger los elementos y su forma en cierta manera. (Perry y Wolf, 1992, citados en Babar, Brown y Mistrik, 2014, p. 4)

Así, de esta manera, se puede ver que la arquitectura del *software* va más allá de solamente describir la forma como los elementos están distribuidos y que son parte del producto. Por eso, y como se mencionó en la parte anterior de arquitectura de computación en la nube, el flujo de trabajo se vuelve parte crucial, siendo esta la razón de la existencia de varios diagramas que modelan estos elementos de forma completa. Por ejemplo, los diagramas UML son de gran importancia en la etapa temprana de gestación del *software* para describir el sistema o aplicación a construir.

Al comparar esta definición con la de otros autores, se expone una perspectiva similar en la cual, la arquitectura de *software* de un sistema es un conjunto de estructuras necesitadas por una razón acerca del sistema, que compromete elementos del *software*, las relaciones entre ellos y propiedades de ambos (Bass, Clements y Kazman, 1998, citados en Babar, Brown y Mistrik, 2014, p. 4). Entonces, es certero pensar que la definición de arquitectura describe a cabalidad cómo un sistema cumple la función para la cual fue construido.

Por lo tanto, una buena arquitectura de *software* es el resultado de un conjunto de principios bien definidos y prácticas que son aplicadas a través del tiempo de vida del proyecto. Es resiliente, ya que los cambios son inevitables, y también provee guía. (Kaisler, 2005, p. 205)

En tal sentido, es conveniente trabajar con los problemas a resolver y hacer todo un proceso en retrospectiva que conduzca a la elección correcta de la arquitectura del *software*. El realizar esta tarea bien traerá un sinnúmero de beneficios que apoyarán el crecimiento de las aplicaciones en tal medida que, sin importar el número de consumidores que posea, el *software* será capaz de escalar sin problemas y funcionar en un entorno complejo lleno de retos de desempeño y extensibilidad.

Ahora bien, como los problemas son varios, por eso existen distintas formas de diseñar un *software*, para que permita solventarlos en una manera única y eficiente. Los patrones de diseño, por ejemplo, son una serie de prácticas que tienen como meta determinar una resolución común que sirva para muchos escenarios. Por eso es que en esta etapa se suelen observar procesos de búsqueda, selección y adopción de la arquitectura más apta para cierta aplicación.

Como señala Gomaa (2011) en su estudio acerca de modelado y diseño de *software,* los patrones de estructura de arquitectura son aplicados para diseñar la parte general de la arquitectura de *software*, lo cual guía cómo el sistema se estructura en subsistemas (p. 228). Esto ayuda a que surjan nuevas y mejoradas formas de modelar *software* como un todo, combina estos conocimientos para crear uno nuevo, y que salgan aplicaciones mucho más eficientes y robustas.

2.5.2 Tipos de arquitectura de *software.*

Habitualmente, debido a la naturaleza del desarrollo del *software*, existen varias formas de diseñarlo. Por ende, las arquitecturas de *software* poseen enfoques específicos que tienen

denominadores comunes, los cuales las caracterizan en términos descriptivos y de funcionalidad. A continuación, se ejemplificarán algunas de las más utilizadas en el mercado.

En primer lugar, se encuentra la arquitectura orientada a objetos; en segundo lugar, la arquitectura cliente/servidor; en tercer lugar, la arquitectura orientada a servicios (cuya definición se amplía más adelante como foco de atención para la investigación); en cuarto puesto está la arquitectura de *software* de tiempo real y concurrente; y por último, la arquitectura de línea de producto de *software*.

Como se puede apreciar, todos sus nombres indican claramente el objetivo principal que se usa para describir los sistemas desarrollados bajo esas estructuras. También, y no es de sorprenderse, se evidencia que sus nombres expresan la cualidad en común en la cual se basa su propuesta de solución ante interrogantes comunes, frecuentemente referidas como subsistemas. Como señala Gomaa (2011):

> Un subsistema puede satisfacer más de un criterio de estructuración. Los subsistemas generalmente son ilustrados con el estereotipo "subsistema". Para ciertas arquitecturas que consisten en subsistemas basados en componentes distribuidos, el estereotipo "componente" es utilizado como subsistema, y en arquitecturas orientas a servicios que consisten en subsistemas de servicio, el estereotipo "servicio" es empleado para este. (p. 220)

2.5.3 Arquitectura de *Software* Orientada a Servicios (SOA).

Es fundamental entender el origen y definición de la arquitectura orientada a servicios, ya que proporciona la base de cómo el *software* como servicio se ha caracterizado como un pilar importante, especialmente para negocios que apenas empiezan y de tamaño pequeño. Sus

características proveen un escenario ventajoso no solo para el *software* como servicio, sino para cualquier otro tipo de aplicación que se desee construir.

Expresado en las palabras de Anandamurugan y Priyaa (2014), en su libro titulado *Service Oriented Architecture* [*Arquitectura orientada a servicios*], se indica lo siguiente:

> La arquitectura orientada a servicios es un paradigma arquitectónico utilizado para permitir consumidores y proveedores de servicios interactuar vía servicios. Los servicios son el núcleo facilitador para la arquitectura orientada a servicios. Estos tipos de servicios son independientes de cada vendedor, producto o tecnología. Varios tipos de servicios pueden ser combinados con otros para proporcionar una funcionalidad completa para una aplicación más grande de *software*. (p. 68)

Por consiguiente, se entiende que se hace una segregación de responsabilidades para una mejor distribución de los componentes propios de la aplicación y, así, obtener cierta ganancia con respecto del mantenimiento o desarrollo de estos. Esto, también converge en un solo repositorio central que hace posible tener a todo un sistema funcionando bajo este modelo.

Asimismo, otros autores como Borges, Holley y Arsanji (2004), en un artículo publicado en el *Journal of Management Information Systems* [*Revista de la Administración de los Sistemas de Información*], señalan que la arquitectura de *software* como servicio se refiere a un estilo de arquitectura que soporta servicios interoperables altamente desacoplados para habilitar la flexibilidad y agilidad del negocio. (Citado en Li y Madnick, 2015, p. 105)

De acuerdo con estas afirmaciones, es posible, entonces, construir un *software* que se base en componentes llamados servicios que constituirán las entidades y transacciones que sea posible realizar a estas; por ejemplo, un servicio de facturación que soporte y comunique con el

servicio de inventario para el manejo de los productos vendidos. Esto, en un modelo tradicional, hubieran sido módulos que se encuentran en la lógica de negocios de una aplicación monolítica.

2.5.3.1 Ventajas del uso de una arquitectura orientada a servicios.

Al introducir estos conceptos sobre cómo la arquitectura orientada a servicios es una gran opción para reducir la complejidad de la aplicación, se debe tener en cuenta varios aspectos que la hacen ventajosa para ciertas ocasiones.

Según Anandamurugan y Priyaa (2014), la arquitectura orientada a servicios (SOA) habilita el incremento de la agilidad del negocio, mejora los flujos de trabajo, es una arquitectura extensible, mejora la reutilización y un tiempo largo de vida de las aplicaciones (p. 84). De esta forma, genera confianza y soporta el ciclo de vida de desarrollo de *software* y el del sistema a lo largo del uso que se le dé en la organización.

Asimismo, estos mismos autores exponen las siguientes características que se pueden ver en la tabla 2.1, a continuación:

Tabla 2.1

Ventajas del uso de la arquitectura del software orientado a servicios

Descripción	
Pérdida de acoplamiento	Reusabilidad
Mobilidad del Código	Enriquecido en pruebas
Transparencia de la ubicación	Desarrollo en paralelo
Mejor retorno de la inversión	Mayor disponibilidad y mejor escalabilidad
Roles enfocados y específicos	Soporta múltiples tipos de clientes

Nota: Adaptado de *Service Oriented Architecture*, elaborado por Anandamurugan, S. y Priyaa, T., 2014, pp. 84-87.

Con el objeto de apoyar la mejor distribución de componentes y reducir la complejidad del mantenimiento requerido, su uso se ha popularizado mucho en las últimas décadas. Por otra parte, otras dos ventajas que se desean resaltar de la lista anterior son la reusabilidad y el desarrollo en paralelo. Esto debido a que soportan mucho mejor los procesos de desarrollo y reducen por mucho la tediosa tarea de compartir repositorios y el error humano al construir nuevas funcionalidades sin afectar profundamente otras partes.

2.5.3.2 Relación de SOA con la computación en la nube.

Ciertamente, en este sentido son dos conceptos completamente separados, ya que SOA es simplemente un estilo de arquitectura de *software* mientras que la computación en la nube va más allá que un solo paradigma y estilo, ya que envuelve varios modelos de entrega de *software* e infraestructura en la que corre el *software*.

Bajo esta premisa se podría decir que la arquitectura de *software* orientada a servicios, de cierta forma es una generalización de lo que es la computación en la nube. Este último adopta estos conceptos y los pone en práctica para entregar servicios que sirven a sus clientes en demanda por infraestructura, aplicaciones y plataformas enteras. Al final, la computación en la nube no es más que un modelo de entrega de *software* basado en servicios que se pueden consumir a través de servicios web.

En cuanto al diseño se refiere, se dice que los desarrollos actuales del acercamiento de SOAD (diseño de arquitectura de *software* como servicio) son usados en la computación en la

nube y también son reusables en el modelo de decisión de arquitectura. (Anandamurugan y Priyaa, 2014, p. 132)

Es por ello por lo cual se comparte el principio de convertir los componentes del *software* en un servicio. En este ejemplo, el proveedor de la nube transforma su infraestructura en servicios separados capaces de interactuar entre sí por medio de la red. Estos sirven a varios clientes al mismo tiempo y su estructura está desacoplada, es decir, si el día de mañana deciden que el servicio de almacenamiento lo van a sustituir, simplemente se conecta por medio de API al nuevo servicio que provee el acopio de los datos en otros contenedores.

2.5.4 Arquitectura de *software* de microservicios.

Uno de los componentes más importantes de este trabajo de investigación es la arquitectura de *software* basada en microservicios. Debe señalarse que, a pesar de ser un concepto ampliamente aplicado en el desarrollo de *software*, tiene gran relación con el modelo de entrega de *software* como servicio y, por ende, con la computación en la nube misma, lo cual lo hace un candidato perfecto para ahondar en temas de diseño de aplicaciones altamente sustentables en este ámbito.

De acuerdo con Nadareishvili, Mitra, McLarty y Amundsen (2016), la definición de microservicio se expresa de la siguiente manera:

> Un microservicio es un componente desplegable independiente de un alcance compartido que soporta interoperabilidad a través de comunicación basada en mensajes. La arquitectura de microservicios es un estilo de ingeniería de sistemas de *software* altamente automatizados, y propensos a evolución compuestos por microservicios capaces y alineados. (p. 6)

Además, como lo expresa Sourabh Sharma en su libro titulado *Mastering Microservices with Java 9* [*Dominando los microservicios con Java 9*], los microservicios comparten muchas características en común con SOA, como el enfoque en servicios y como un servicio está desacoplado del otro (…) sin embargo, son menos complejos, ya que la comunicación se basa en puntos inteligentes. (Sharma, 2017, p. 7)

También, hay que hacer notar otra definición que se encuentra en la misma referencia, dada por Tony Pujals, quien señala lo siguiente: "en mi modelo mental, yo pienso en procesos livianos autocontenidos (como contenedores) comunicándose a través de HTTP, creados y desplegados con un pequeño esfuerzo relativo y ceremonioso, proveyendo API´s enfocadas a consumidores. (Citado en Sharma, 2017, p. 7)

De modo que, con base en estas definiciones, se puede ver que la arquitectura de microservicios es una versión alterna de la arquitectura orientada a servicios. No se va a discutir si una es mejor que la otra, por eso se evita el calificativo "versión mejorada", sino que se entiende como una opción adicional que se basa en la orientación a servicios con un objetivo más específico y granular.

Por consiguiente, aquí entran en juego las características principales que acuerpan y definen este estilo de arquitectura. Una es la descentralización, que significa que lo pesado del trabajo realizado en el sistema ya no será controlado ni administrado por un núcleo central (Nadareishvili, Mitra, McLarty y Amundsen, 2016, p. 8). La otra es la autonomía, que se puede expresar como cada servicio está a cargo de su propio modelo de datos que podría liderar la replicación. (Cerny, Donahoo y Pechanec, 2017, p. 230)

En otras palabras, llevan los servicios a otro nivel de interacción y desacoplamiento que juntos hacen posible construir el "todo" de una aplicación de *software*. Lo interesante sucede al

quebrar un elemento importante para las arquitecturas tradicionales de *software*, el cual es mantener uno o dos repositorios centrales de datos con los cuales comunicarse.

Cada servicio posee sus propias capas y bases de datos, y están empaquetados en un archivo diferente. Todos estos servicios desplegados proveen sus API´s como Clientes, Reservaciones o Cliente, los cuales están listos para consumirse (Sharma, 2017, p. 11). Eventualmente, introduce nuevos retos y dificultades que este debe superar, y pone a evaluar su uso a diversos arquitectos en sus organizaciones.

2.5.4.1 Ventajas del uso de una arquitectura de microservicios.

En este sentido, las ventajas son muy similares a las de la arquitectura orientada a servicios. La escalabilidad, por ejemplo, sigue siendo un pilar fuerte en este estilo de arquitectura. Esto porque cada servicio es una entidad separada de las demás; entonces, se puede alterar a gusto sin impactar drásticamente el resto de la aplicación.

Otro aspecto importante es la flexibilidad, ya que no solo las bases de datos son separadas para servicio, sino que la tecnología o lenguaje de programación utilizado para cada uno puede diferir sin ningún problema. Esto, al final, no afecta ni disminuye la operación del aplicativo de ninguna manera. Por ejemplo, un servicio puede utilizar Java para el servicio de reservas de tiquetes en una aplicación del cine, y el otro en C# o Python para las operaciones de administración de usuarios.

Además, los microservicios proveen la flexibilidad de retroceder los cambios realizados solamente a aquellas funcionalidades que fallaron (Sharma, 2017, p. 13). A pesar de que esto puede ser también posible en una arquitectura orientada a servicios, en esta opción es todavía mejor, ya que no existe dependencia alguna de un servicio a otro. Esto trae un escenario

favorable a la gestión de cambios y todos los procesos de regulaciones que estos imponen, ya que facilita mucho el no tener que involucrarse con administraciones complicadas.

Por otro lado, los autores Kwan, Jacobsen, Chan y Samoojh (2016), en su artículo *Microservices in the modern software world* [*Microservicios en el mundo del software moderno*], expresan otro aspecto a considerar acerca del recurso humano de la organización:

> Los equipos de desarrollo también deberían organizarse acerca de los diferentes componentes individuales de microservicios. Por ejemplo, equipos individuales deberían tener a cargo de construir y administrar un microservicio individual. Esto ayuda a las organizaciones que corren las aplicaciones; también, define muy bien los límites que permitirán a los microservicios ser desarrollados por individuos sin conocer la estructura completa de la aplicación y con las tecnologías correctas para cada escenario. Sin olvidar, el que ayuda mucho a las organizaciones que crecen rápido a medida que traen nuevos desarrolladores a trabajar en la aplicación. (p. 297)

Lo anterior pone una perspectiva un poco diferente a la del *software* construido tradicionalmente que depende de la tecnología escogida desde un inicio para el desarrollo completo y soporte del *software* hasta que llegue al fin de su vida útil. También, es interesante cómo podría llegar a ser beneficioso en el ámbito del recurso humano, ya que no siempre se puede encontrar desarrolladores expertos en una sola tecnología, además de permitir elegir la mejor para cada caso de uso. Eventualmente, esto representa una ventaja en términos de retorno de la inversión y gestión del *software*.

2.5.4.2 Relación de los microservicios con la computación en la nube.

Así como la arquitectura orientada a servicios, este estilo de servicios autónomos y descentralizados provee un escenario perfecto para acoplarse al modelo de *software* como servicio. Esto porque con sus características, puede proporcionar las mismas capacidades que le ofrece la arquitectura orientada a servicios, como la escalabilidad y multi-inquilinato.

En este sentido, estudios como el de Tizzei, Nery, Segura y Cerqueira (2017), titulado Using Microservices and Software Product Line Engineering to Support Reuse of Evolving Multi-tenant SaaS [Utilizando Microservicios y una línea de ingeniería de producto de software para la reutilización del software como servicio multi inquilino en constante evolución], señalan que al aplicar la arquitectura de microservicios a su aplicación (WISE) tipo SaaS multi-inquilino, demostró una reutilización del 62% de código entre inquilino. También redujo el esfuerzo en mantenimiento y mejoró la escalabilidad (pp. 205-214).

En este aspecto, cabe resaltar que la muestra de una mejora mostrada a favor de esta arquitectura para soportar el modelo de *software* como servicio puede ser beneficioso para aquellas organizaciones en términos de esfuerzos necesarios para construir y operar una herramienta de *software* de esta índole.

Posteriormente, se puede relacionar con servicios que son dados por los proveedores de la computación en la nube, lo cual brinda capacidades híbridas en las cuales ciertas operaciones de almacenamiento y análisis de datos pueden aislarse de la aplicación, adoptarse por un servicio de estos y consumirse en el sistema. Evidentemente, se observa la alta compatibilidad que existe entre los modelos de servicio que entregan proveedores de la nube con la arquitectura y el ambiente en el cual corren este tipo de aplicaciones.

2.5.5 Diferencias de las arquitecturas SOA y microservicios.

Aunque parecen tener en común varias características, incluyendo el objetivo principal de operar un *software* en partes llamadas servicios que se comunican entre sí, tiene diferencias marcadas que suelen estar orientadas a resolver diferentes casos de uso. Tanto la arquitectura orientada a servicios como la arquitectura de microservicios están equipadas para resolver mejor ciertos obstáculos. Es por ello que no se piensa en ellos como opciones alternas, sino de antecesor y sucesor mejorado.

Por consiguiente, la más notable debe ser la administración de los servicios como tal. Citando a Cerny, Donahoo y Pechanec (2017), los enfoques de orquestación centralizados parecen ser un patrón más común para la arquitectura orientada a servicios, mientras que el descentralizado pareciera más adaptado a los microservicios. (p. 229)

Dado lo anterior, es posible deducir que la manipulación de los servicios para completar el flujo de trabajo completo de la aplicación tiene un nivel jerárquico con un punto en común, para que los demás servicios que forman parte como un componente de la aplicación, funcionen de manera correcta. No obstante, esto se pretende eliminar con la autonomía que se les da a los microservicios, al ser gestionados de forma independiente se logra quitar esta dependencia.

En otras palabras, no existe un *software* terciario que se encargue de gestionar la comunicación a través de un bus, ni actuar de mensajero, ni mucho menos de procesamiento de datos. Más bien, los microservicios se encargan ellos mismos de hablar directamente entre sí para poder establecer, de manera efectiva, el intercambio de datos.

Ahora bien, algunos investigadores se refieren a los microservicios como nativos de la nube, mientras que la arquitectura orientada a servicios está raramente referenciada en esa

manera en la literatura (Kratzke y Quint, 2017, citado en Cerny, Donahoo y Pechanec, 2017, p. 233). Como producto, la arquitectura de microservicios se empieza a ver como un candidato con futuro prometedor en el ámbito del desarrollo del *software*.

2.6 Gestión de Operaciones de Aplicaciones de *Software* como Servicio

Dentro de las operaciones de un producto de *software* se debe tener muy en cuenta cómo se va a coordinar las mejoras o correcciones de errores que se den durante su funcionamiento, principalmente porque dependiendo del alcance del *software* y de su impacto en el servicio que brinda, se puede incurrir en serios problemas si no se hace de manera apropiada.

Por consiguiente, una adecuada gestión del cambio ayuda a mantener estos eventos controlados durante las operaciones, y que no incurran en fallas e impacto negativo para los consumidores del *software.* Además, también se debe estar en regla con las regulaciones y políticas, tanto internas como externas que gobiernan la organización.

2.6.1 Gestión del cambio.

Primero, se debe empezar por definir lo que significa la gestión del cambio. Como su nombre lo indica, es toda aquella planificación, ejecución y validación que se efectúa con cada cambio que se realiza, tanto en el *software* como en la infraestructura que soporta el sistema.

Según los autores Mahimkar, de Andrade, Sinha y Rana (2021), las tareas de la gestión del cambio comprenden cuatro fases: diseño e implementación, planeamiento del despliegue con el aseguramiento de la continuidad de la operación, la ejecución de manera cuidadosa y, por último, la verificación. (p. 1)

Ahora bien, mientras que ellos se enfocan en cambios en la red, como se puede leer en su artículo *A composition Framework for Change Management* [*Un marco de trabajo de composición para la gestión del cambio*], estas etapas no son del todo diferentes de lo que se puede aplicar a nivel de desarrollo de *software*. En realidad, se concuerda que cada una de ellas se pudiera dividir en más pasos específicos, los cuales son documentados para revisarlos posteriormente para efectos de auditoría y control; sobre todo para poder actuar rápidamente ante eventualidades.

Del mismo modo, de acuerdo con el *Institute of Internal Auditors* (IIA) [Instituto de Auditores Internos], un proceso de gestión del cambio está compuesto por pasos y debe incluir un conjunto formal de procedimientos. Los pasos y procedimientos son necesarios para administrar cambios, actualizaciones o modificaciones en la organización, incluyendo su *hardware* y *software.* (IIA, 2011, citado en Santos y Quilliam, 2015, p. 1)

Estos pasos están tan regulados que debe existir evidencia de la ejecución de cada uno de ellos. Esto ayuda a medir dos métricas muy importantes: la eficacia, la cual se obtiene de la ejecución de cada paso durante el despliegue del cambio, y la trazabilidad del cambio, que habla más de lo que sucede en cada paso y de cómo esto puede apoyar procesos posteriores, como la validación de que se ha completado el despliegue con éxito.

Esta última métrica es crucial, en especial cuando aparecen eventualidades no esperadas, como errores en el sistema. Según Jones, C. (1996), por los costos e importancia del control de la calidad, es evidente que rastrear defectos es vital para gestionar los cambios del *software*, puede extenderse durante toda la vida útil del proyecto de *software*: 20 años o más. (p. 82)

Por consiguiente, debe estar alineado con un proceso robusto dentro del ciclo de vida de desarrollo del *software.* Esto trae beneficios en ámbitos financieros, operacionales y de

administración para la organización, incluyendo el proporcionar mejor estabilidad a la aplicación y, por ende, es más sustentable.

2.6.2 Gestión del cambio enfocado en *software* como servicio.

Una correcta gestión del cambio en una organización beneficia a todos en un ámbito amplio, especialmente para aplicaciones que enfocan su servicio a clientes externos. Una actualización inesperada puede traerse abajo toda una gama asociada de servicios relacionados que puede traer consecuencias monetarias y hasta legales, en algunos casos.

En un sistema entregado bajo la modalidad de *software* como servicio, es muy importante tener esto en cuenta. Se debe extremar todavía más la gestión del cambio, ya que su alcance es muy amplio, es decir, miles de clientes, si es la realidad de la compañía, se pueden ver afectados por un cambio que afectó las operaciones de todos los consumidores.

2.6.2.1 Retos y problemas de la gestión del cambio de aplicaciones de software como servicio.

Como se mencionó anteriormente, uno de los retos que compete a las aplicaciones entregadas como servicio está en la correcta administración de varios cambios que se requieran a múltiples clientes. Esto quiere decir que el *software* como servicio está pensado como una solución "genérica" que trata de alcanzar a múltiples consumidores al mismo tiempo. Por ende, debido a la naturaleza que envuelve a cada negocio y sus reglas, es posible que en algún momento esa "personalización" requerida por los clientes empiece a influenciar e impactar de forma negativa el rumbo de la aplicación.

Es entendible que las ideas de potenciales funcionalidades que puedan ser agregadas durante la operación de un sistema de este tipo sean, en muchas ocasiones, solamente requeridas

por ciertos clientes, pero no todos. Este punto es donde los proveedores de aplicaciones en este modelo deben tener claros sus requerimientos y cómo gestionar estas ideas nuevas para adoptarlas de manera diplomática o, en otras palabras, para tratar de beneficiar tantos clientes como sea posible con una solución que siga el modelo del *software*, un "bueno para todos".

2.6.2.2 Recomendaciones y mejores prácticas para la gestión del cambio en SaaS.

En general, las recomendaciones no varían con la administración de los cambios en una aplicación en un esquema tradicional. Siempre debe existir una rigurosa aplicación de todos los pasos que concreten el despliegue a producción, cualquier cambio que impacte directamente la aplicación o la infraestructura en la cual opera.

La gestión del cambio debe enfocarse en varias áreas: plan de acción, ejecutar el plan, monitorear la ejecución del plan, validar que el plan se haya completado con éxito, en otras palabras, que los pasos implementados no están teniendo impacto negativo en la solución existente, y documentar todo lo sucedido durante las etapas anteriores.

También debe existir una herramienta que permita llevar el rastreo de los cambios que se realizan. En esta se pueden hacer las operaciones típicas de un *software* de manejo de tiquetes: levantamiento de la solicitud, revisión, aprobación y calendarización de esta. Adicionalmente, esto acarrea ciertos beneficios, como centralización de la información, lo cual permite a los administradores tener visibilidad absoluta de los cambios que se están realizando y cuándo éstos se implementan, tanto en la infraestructura como en el *software* como tal.

Al final, se trata de tener documentado un proceso sistemático para implementar cambios. Más allá de todos los pasos que conlleva, se trata de ganar control de lo que se está realizando

para tener planes de contingencia en caso de que algo no marche bien y, si todo sale bien, entonces de medir la eficacia de la organización para administrar estos cambios en la aplicación.

CAPÍTULO III. MARCO METODOLÓGICO

3.1 Marco Metodológico

En el ámbito de la investigación científica, se procura detallar todos los pasos que conllevan a la obtención de conocimiento por medio de un procedimiento estándar, el cual posee varias formas de realizarse. Según Baena Paz (2017), “la metodología ejerce el papel de ordenar, se apoya en los métodos, como sus caminos y éstos en las técnicas, como los pasos para transitar por esos caminos del pensamiento a la realidad y viceversa”. (p. 31)

De la misma manera, Cortés, Iglesias y Carmen (2005) definen la metodología de la investigación como “la ciencia que provee al investigador de una serie de conceptos, principios y leyes que le permiten encauzar de un modo eficiente y tendiente a la excelencia en proceso de investigación científica”. (p. 8)

Por consiguiente, el marco metodológico de toda investigación comprende esa guía de pasos, recomendaciones y mejores prácticas para que el objetivo sea alcanzado por parte del investigador de forma precisa y con resultados confiables. De hecho, esta parte de la confianza es crucial para todo estudio. Sin ella no existe veracidad en lo que se afirma; por lo cual, pierde el sentido de generación de información. En resumidas cuentas, es necesario demostrar cómo se recolectarán los datos, el por qué y para qué se necesita.

En vista de ello, se estará detallando dentro del desarrollo de este capítulo, una explicación sobre la naturaleza investigativa de este apartado, así como los elementos que comprenden el proceso completo de investigación y sus definiciones conceptuales.

3.2 Tipo de Investigación

El proceso de investigación, por sí mismo, varía debido a su fin. Existen varios tipos de estudios a la hora de clasificar las investigaciones. Estas categorías serán asignadas dependiendo del objetivo de la indagación, siendo una de las primeras instancias el efectuar la elección correcta del diseño de la investigación. Como indican Wentz (2014), McLaren (2014), Creswell (2013ª), Hernández Sampieri *et al*. (2013) y Kalaian (2008, citados en Hernández Sampieri, 2014, p. 128), "el diseño se refiere al plan o estrategia concebida para obtener la información que se desea con el fin de responder al planteamiento del problema".

Asimismo, el autor afirma:

> Es posible ver que, de los diferentes diseños existentes, se enfocan en una pareja de términos: experimental y no experimental. En cuanto, a los experimentales, se construye el contexto y manipula intencionalmente la variable independiente para observar el efecto en la variable dependiente. Por su parte, los no experimentales, buscan analizar relaciones entre variables o evaluar el nivel de modalidad o situación de variables en un determinado punto del tiempo. (Hernández Sampieri, 2014, pp. 128-159)

En otras palabras, los diseños experimentales tienden a impactar el resultado dependiendo del tipo de modificación o experimento que deseen realizar; mientras que los no experimentales se basan en las variables, sin alteración alguna, más que la definición o exposición de sus características, para así poder llevar a cabo un veredicto sobre lo que se piensa detalladamente de estas.

En esta misma línea, los diseños no experimentales pueden clasificarse como: transeccional o transversal, y longitudinal.

3.2.1 Diseño longitudinal.

El diseño longitudinal se define como toda investigación que envuelve la colección repetida de, al menos, una fuente de datos a uno o más puntos de tiempo. (Van Ness *et al.*, 2011; Plano Clark *et al*., 2015, citados en Zhang y Liu, 2019)

Por consiguiente, los investigadores pueden detectar desarrollo o cambios en las características de la población meta a nivel de grupo o individual. La clave es extender el estudio aparte de un momento determinado del tiempo. Como resultado, pueden establecer secuencias de eventos. (*Institute of Work & Health*, 2015, p. 2)

Este método permite tener una comparación de datos y poder medir si el paso del tiempo ha jugado un papel importante en la variación de sus valores. Para estudios históricos y demográficos es una excelente herramienta, ya que su naturaleza permite a los investigadores graficar según los años, meses o días en los que se recolectaron los datos.

3.2.2 Diseño transeccional o transversal.

Se dice que “recolectan datos en un solo momento, en un tiempo único” (Liu, 2008; Tucker, 2004, citados en Hernández Sampieri, 2014, p. 154). Un ejemplo es “Medir las percepciones y actitudes de mujeres jóvenes (18-25 años) que fueron abusas sexualmente en el último mes en una urbe latinoamericana”. (Hernández Sampieri, 2014, p. 154)

De esta forma, como su enfoque está situado en un punto en el tiempo, su alcance de estudio es delimitado por esa medición. Este diseño suele ser ventajoso para encerrar el terreno de datos del cual se extraerá la información necesaria para desarrollar la investigación.

Así también, estos diseños no experimentales poseen clasificaciones o alcances, establecidos de manera jerárquica y debido a su utilidad. Con base en lo que afirma Danhke (citado en Cortés *et al*., 2005, p. 20), cabe destacar los siguientes: exploratorios, descriptivos correlacionales y explicativos.

Respecto de estos alcances, cada uno posee enfoques donde es esencial su aplicación al proceso. La elección de uno sobre el otro, como se ha expresado anteriormente, se debe al fin o meta final del estudio. Estos se detallan a continuación:

3.2.2.1 Exploratorio.

Uno de los tipos de estudio que pretenden adentrarse en algún tema específico, sirve para responder a preguntas muy abiertas de algún tema de investigación del cual no existe grandes bases aún. Según afirman Cortés, Iglesias y Carmen, "los estudios exploratorios se efectúan, normalmente, cuando el objetivo es examinar un tema o problema de investigación poco estudiado, del cual se tienen muchas dudas o no se ha abordado antes". (Cortés, Iglesias y Carmen, 2005, p. 20)

De la misma forma, José Cegarra Sánchez, en su estudio sobre Los *métodos de investigación*, expresa que "son para averiguar si existe o no un fenómeno, como primer paso a una investigación". (Cegarra Sánchez, 2012, p. 92)

Por lo tanto, se puede explicar que cuando se requiere explorar un nuevo concepto del cual se está empezando a investigar, inclusive que no se haya examinado anteriormente, entonces

se utiliza este alcance porque está precisamente construido para dar resultados en ese tipo de estudio.

3.2.2.2 Descriptivo.

Para definir este tipo de alcance, se tendrá que recurrir a la influencia que ejerce sobre las variables con las cuales trabaja. De acuerdo con lo que explica Bisquerra, "no se manipula ninguna variable. Se limita a observar y describir los fenómenos (estudios de casos, encuestas, estudios etnográficos etc.). Pretende interpretar lo que es". (Bisquerra, 1992, citado en Martínez Ruiz, 2012, p. 16)

Por su parte, Hernández Sampieri (2014) prefiere definir el alcance descriptivo como "la indagación de la incidencia de las modalidades o niveles de una o más variables en una población". (p. 155)

Debido a esto, se puede pensar que este tipo de investigaciones siempre observan y estudian características de los sujetos (normalmente variables) que tienen como público meta, con el fin de proveer de información detallada al lector y dar una interpretación de lo que se describe.

3.2.2.3 Correlacional

Este tipo de estudio es similar al anterior, ya que permite establecer comparaciones un tanto más analíticas entre grupos dentro de la población de estudio. Según Asti Vera (2015), "consiste en la comprobación de las relaciones existentes entre dos o más variables". De esta misma forma, también relata este autor que "es posible descubrir la relación causa-efecto, tal como la concibe el investigador: la variable independiente es la causa, y la variable dependiente es el efecto". (p. 43)

Por otra parte, según Cortés, Iglesias y Carmen (2005), existe también el enfoque cuantitativo de alcance correlacional, sobre el cual menciona que su primordial uso es el de conocer el comportamiento de un concepto o variable con respecto a otras variables que se relacionan entre sí. (p. 21)

Por ende, el agrupar variables independientes o sujetos de estudio y analizar sus similitudes para explicar la relación que existen entre ellas, aún el posible impacto que tienen unas sobre otras, es la especialidad de esta clasificación. Es muy útil para describir dependencias en estructuras jerárquicas o algún otro tipo de organización entre los sujetos de una población.

3.2.2.4 Explicativo.

En el estudio de la metodología de la investigación de Cortés, Iglesias y Carmen, los autores afirman que "los estudios explicativos van más allá de la descripción de conceptos o fenómenos o del establecimiento de relaciones entre conceptos, están dirigidos a responder a las causas de los eventos, sucesos y fenómenos físicos o sociales". (Cortés, Iglesias y Carmen, 2005, p. 21)

Del mismo modo, otros autores como Cegarra Sánchez definen que "procuran encontrar la relación causa-efecto entre dos o más fenómenos" (2012, p. 92). Entonces, se centraliza en tratar de interiorizar en las consecuencias de algún evento.

Estos eventos, a su vez, simplemente son parte de los impactos que surgen al estudiar las combinaciones ya existentes de variables independientes y dependientes. Esta última característica es de suma importancia, ya que realmente identifica este concepto y, aún más, la diferencia que existe con otro tipo de investigación, como el mencionado en el principio, que hace uso de métodos experimentales.

3.2.2.5 Enfoque de la investigación.

Toda investigación debe tener algún enfoque, ya que estos están presentes para apoyar y guiar el proceso desde un punto de vista sistemático. Ejemplos de tres enfoques de investigación son: el cualitativo, el cuantitativo y el mixto. Cada uno tiene sus fortalezas, así también como sus debilidades, y se propicia su uso según el objeto de estudio de cada trabajo de investigación.

3.2.2.5.1 Cualitativo.

Por un lado, el enfoque cualitativo se puede describir, según Hernández Sampieri (2014), como "utiliza la recolección y análisis de datos para afinar las preguntas de investigación o relevar nuevas interrogantes en el proceso de interpretación. También pueden desarrollar preguntas o hipótesis antes, durante y después de la recolección y análisis de datos". (p. 7)

Asimismo, también se afirma que "el paradigma cualitativo de investigación ya no se centrará en aspectos numéricos, sino en reflexiones culturales: deducciones, razonamientos, relaciones, subjetividades. Según lo sugiere, su nombre tiene que ver con las cualidades del objeto de investigación". (Campos Ocampo, 2017, p. 16)

Es así como el enfoque cualitativo tiene su fortaleza en el descubrimiento de conocimiento en casi cualquier etapa de la fase de investigación. Sin embargo, la lógica del razonamiento que se utiliza tiene que ser descrita por el investigador para una comprensión mayor por parte de su público meta.

3.2.2.5.2 Cuantitativo.

De acuerdo con Guerrero (2015), el método cuantitativo "consiste en contrastar hipótesis desde el punto de vista probabilístico y, en caso de ser aceptadas y demostradas en circunstancias

distintas, a partir de ellas elaborar teorías generales" (p. 48). Esto sugiere un proceso que se basa en la cuantificación de los resultados que se puedan obtener al describir las variables; es decir, tiene formas numéricas y específicas de medir las interrogantes.

De manera similar, Cortés, Iglesias y Carmen (2005) lo definen como "toma como centro de su proceso de investigación a las mediciones numéricas, utiliza la observación del proceso en forma de recolección de datos y los analiza para llegar a responder sus preguntas de investigación". (p. 10)

Por ende, es un enfoque que tiene su base en la respuesta a sus interrogantes con datos. Aquí es válido hablar de porcentajes, proporciones, cantidades numéricas, etc.; es decir, todo aquello que se pueda medir a través de números y, posteriormente, se puedan realizar operaciones aritméticas para la correcta presentación y análisis de información.

3.2.2.5.3 Mixto.

Como su nombre lo podrá explicar, este enfoque pretende hacer uso de ambos métodos: cualitativo y cuantitativo, para lograr su cometido. "Los métodos mixtos representan un conjunto de procesos sistemáticos, empíricos y críticos de investigación e implican la recolección y análisis de datos cuantitativos y cualitativos, así como su integración y discusión conjunta, para realizar inferencias producto de la información recabada". (Hernández Sampieri y Mendoza, 2008, citado en Hernández Sampieri, 2014, p. 534)

Ahora bien, el Dr. Manuel Cortés Cortés y la Dra. Miriam Iglesias León (2005) definen el enfoque mixto como aquel en el que:

> El investigador utiliza las técnicas de cada uno por separado, se hacen entrevistas, se realizan encuestas para saber las opiniones de cada cual sobre el tema en cuestión, se trazan lineamientos sobre las políticas a seguir según las personas que intervengan, etc., además esas encuestas pueden ser valoradas en escalas medibles y se hacen valoraciones numéricas de las mismas, se observan las tendencias obtenidas, las frecuencias, se hacen histogramas, se formulan hipótesis que se corroboran posteriormente. (p. 11)

Por tal motivo, para el presente trabajo de investigación se ha elegido seguir con el enfoque mixto, es decir, una combinación de los enfoques cuantitativo y cualitativo para usar sus fortalezas en el estudio. A su vez, se utilizará una clasificación no experimental con un diseño transeccional descriptivo, el cual ayudará a complementar la base teórica de la información recopilada de campo con la población de estudio.

Por ende, usar las fortalezas del enfoque cuantitativo será útil para obtener una comprensión detallada de la teoría detrás de la creación e implementación de soluciones de *software* como servicio, teniendo en cuenta la descripción de las variables para entender su conceptualización e interpretación.

Ahora bien, el enfoque cualitativo se pretende utilizar para entender a la población en estudio a través de sus experiencias, conocimiento e interpretación de su realidad para realizar una comparación con las bases teóricas. Por lo tanto, esta unión será útil para ver si es posible darles respuesta a las preguntas de investigación planteadas en el primer capítulo de este documento, o bien, en el mejor de los casos, reformular las preguntas de investigación para abarcar un contexto más general, de ser posible.

Como resultado, se pretende hacer uso de las fortalezas de ambos enfoques para el refuerzo y relación de la teoría, con el estudio y refinamiento de todos los elementos que se presentan inicialmente en la investigación, como las preguntas de investigación y la hipótesis. A su vez, permite específicamente describir y extraer los conceptos y prácticas que se utilizan en el ámbito de *software* como servicio en la nube, en conjunto con la comprensión de la realidad de emprendedores en el sector de desarrollo de *software*. De este modo, se podrá tener una mayor comprensión del problema para generar una solución adecuada para los sujetos que se beneficiarán del estudio.

3.3 Fuentes de Información

Las fuentes de datos son los principales recursos que deben tener los investigadores para sustentar, respaldar, adoptar posturas y contrastar información. Se entiende como todos aquellos objetos tangibles que contienen información, tanto impresos como digitales, así también toda la interacción y recopilación de datos que se obtenga de la interacción con la población en estudio.

Según Torres Ramírez, se entiende por fuente "cualquier material o producto, original o elaborado, que tenga potencialidad para aportar noticias o informaciones o que pueda usarse como testimonio para acceder al conocimiento". (Torres Ramírez, 2002, citado en Gallego Lorenzo y Juncà Campdepadrós, s. f.)

Para efectos de esta investigación, se tienen dos tipos de información por recopilar: los datos que provienen de fuentes escritas, digitales o impresas, y los que tienen su origen a partir de la aplicación de los instrumentos de recolección de información en la población de estudio en sí, es decir, se refiere al trabajo de campo. Esto se detalla a continuación.

3.3.1 Fuentes primarias.

Basado en la obra de Gallego y Juncà (s.f.), se puede describir este concepto de la siguiente manera: “estas fuentes proporcionan información nueva, original y final en sí misma. No remiten ni complementan a ninguna otra fuente”. (Gallego Lorenzo y Juncà Campdepadrós, s. f., p. 20)

Y de la misma forma, se aprecia que Cabrera (2006) también asevera que “las fuentes de información primarias son aquellas que contienen información nueva y original, que no ha sido sometida a ningún tipo de tratamiento posterior (selección e interpretación…)”. (p. 4)

Por lo tanto, se entiende que una fuente primaria de información es aquella que ha sido concebida producto del intelecto del autor, con ideas originales, es decir, propuestas por el creador, que surgieron a partir de un procesamiento de datos que le fueron otorgados directamente de los sujetos en estudio.

Es por eso por lo cual, en esta investigación, se entiende como fuentes primarias de información las que se aplican directamente a la población en estudio, toda entrevista, conversación, cuestionario o formulario que provea datos directamente de los sujetos en estudio.

También, se considera que los libros, investigaciones científicas, trabajos de graduación (tesis, tesina, y proyecto de investigación) son consideradas fuentes primarias para el reforzamiento conceptual de la parte teórica, que comprende la computación en la nube y el modelo de entrega de *software* como servicio, de este trabajo de investigación. Como se ha expresado, esto permitirá la comparación y análisis respecto de los datos obtenidos de la población.

3.3.2 Fuentes secundarias.

Según Cabrera (2006), “son el resultado de las operaciones del análisis documental (descripción bibliográfica, catalogación, indización y, a veces resumen,), es decir, alguien ha trabajado sobre el contenido de estas. Permiten el conocimiento de documentos primarios”. (Cabrera, 2006, p. 4)

En efecto, para el carácter investigativo de este trabajo se piensa en los catálogos de biblioteca que ofrecen las bases de datos y repositorios de la Universidad Técnica Nacional, así como las que ofrecen otras universidades públicas y privadas, tanto en el territorio nacional como internacional. También bases de datos de libre acceso como la Google Académico o ERIC, siendo las búsquedas de autores y contenidos en español e inglés comprendidos dentro del alcance.

Igualmente, toda la información bibliográfica y citas respectivas que se realicen dentro de la literatura que se utiliza dentro de la investigación, y que referencian a otros autores que sean legítimos, también se clasifican dentro de esta categoría.

3.3.3 Sujetos de información.

Dentro de este punto, se describe los sujetos de información, es decir, a los participantes del estudio que se realiza. Se entiende que “se centra en qué o quiénes, es decir, los participantes, objetos o colectividades de estudio”. (Hernández Sampieri, 2014, p. 172)

En tal sentido, se describe que los sujetos para esta investigación son los miembros del Departamento de Tecnologías de la Información de las micro y pequeñas empresas (PYME) que se dedican al desarrollo de *software*, incluyendo roles como desarrolladores, arquitectos,

especialistas en calidad del *software*, encargados de infraestructura de TI, analistas de negocio y jefaturas. De esta forma, se obtiene una visión de la parte técnica y de la parte administrativa.

3.4 Definición de la Población

La población, como afirma Lepkowski (2008), "es el conjunto de todos los casos que concuerdan con una serie de especificaciones". (Lepkowski, 2008, citado en Hernández Sampieri, 2014, p. 172)

Asimismo, otros autores como Baena Paz, dentro de su libro de *Metodología de la investigación*, tiene una definición de población que comprende el número total del universo de la investigación a realizar. (Baena Paz, 2017, p. 126)

Puede, entonces, afirmarse que a la unión de todos los miembros participantes que son sujetos de estudio y que tengan características similares entre sí, en relación con el objeto de la investigación, se les puede llamar población.

De acuerdo con datos del Ministerio de Economía, Industria y Comercio (MEIC), a febrero 2021 existen alrededor de 469 PYMES activas en todo el territorio nacional, que incluyen actividades como programación informática, consultoría informática y otras actividades de tecnologías de información. (MEIC, 2021)

Como resultado, el universo que se pretende alcanzar durante esta investigación corresponde a todas aquellas micro y pequeñas empresas (PYMES) que se dediquen al desarrollo de *software* como actividad principal, o que sea parte de su actividad lucrativa, y que estén establecidas en el distrito económico del cantón central de Alajuela. Primordialmente, porque el alcance de la investigación, en términos de tiempo y recursos disponibles, solamente se centrará

en esta región geográfica. Asimismo, se puede observar en la siguiente tabla la información localizada.

Tabla 3.1

Reporte de Empresas PYME activas a febrero 2021

Tamaño	**Cantidad**
Micro	13
Pequeña	2
Total	**15**

Nota. Fuente: Ministerio de Economía, Industria y Comercio (MEIC)

En síntesis, se puede observar a simple vista que el universo de este trabajo de investigación tiene tres alcances definidos: que sean PYME, que tengan el desarrollo de *software* como parte de su actividad económica, y estén geográficamente dentro del distrito económico del cantón central de Alajuela.

3.5 Definición de la Muestra

En la mayoría de los trabajos de investigación, estudiar el universo en su totalidad puede ser tedioso y tomar mucho tiempo, no suele ser una buena opción, normalmente, porque son grandes. Para resolver este problema se utiliza una técnica llamada muestreo. Ahora bien, se suele decir que es una generalidad porque no siempre se cumple, y eso dependerá de los objetivos de la investigación.

Según Hernández Sampieri (2014), “la muestra es, en esencia, un subgrupo de la población. Digamos que es un subconjunto de elementos que pertenecen a ese conjunto definido en sus características al que llamamos población”. (p. 175)

De forma similar, Baena Paz (2017) se refiere al muestreo como un procedimiento, es decir, que es la escogencia de uno o varios miembros representativos del universo en su totalidad. Por lo tanto, provee como ventaja una inversión de recursos menor, sencillez y rapidez en la tarea. (p. 84)

3.5.1 Tipo de muestra.

Para la presente investigación se utilizará la muestra probabilística simple. De acuerdo con los autores Otzen y Manterola (2017), se puede definir de la siguiente manera:

> Este garantiza que todos los individuos que componen la población blanco tienen la misma oportunidad de ser incluidos en la muestra. Es decir, la probabilidad de selección de un sujeto a estudio “x” es independiente de la probabilidad que tienen el resto de los sujetos que integran forman parte de la población blanco. (p. 228)

Se ha optado por este método porque no existe, por el momento, preferencia en los sujetos de la población de estudio. Todos los que forman parte tienen características similares en términos de actividad económica, se dedican a la programación informática; de tamaño de empresa, micro y pequeña; y zona geográfica. Se puede afirmar que la selección aleatoria de uno, varios o todos los sujetos de estudio, sí es representativa y no altera el resultado.

3.5.2 Cálculo y tamaño de muestra.

De acuerdo con Hernández Sampieri (2014):

> Existen elementos o variables que hacen posible el cálculo de la muestra probabilística, dentro de las que poseen mayor relevancia para el cálculo se encuentran: tamaño del universo (normalmente representado con una *N*), el error máximo aceptable (expresado en porcentaje), nivel deseado de confianza (expresado en porcentaje). (p. 178)

Dado lo anterior, se puede distribuir los valores correspondientes de la selección de la muestra para esta investigación, en sus respectivas variables, para ser visualizados en la siguiente tabla:

Tabla 3.2

Tabla de valores para selección de la muestra

Variable	**Valor**
Tamaño de la población	15
Nivel de confianza (%)	99
Margen de error (%)	1
Tamaño de muestra	**15**

Es necesario mencionar que el cálculo de la muestra expuesto anteriormente se ha hecho a través del uso de la herramienta en línea de la página web de *es.surveymokey.com*, cuyo enlace completo es: https://es.surveymonkey.com/mp/sample-size-calculator/, como se aprecia en la imagen de la figura 3.1:

Figura 3.1. Herramienta de cálculo de la muestra.

Desde la perspectiva general, se puede observar que la muestra prácticamente es casi del mismo tamaño que la población. Esto se debe a que la población es relativamente pequeña. Sumado a esto, se define como nivel de confianza dentro de esta investigación un 99%, con lo cual adiciona los elementos a tomar en cuenta para el cálculo del tamaño de la muestra que realiza la herramienta en línea.

En relación con este tema, se decide hacer uso de esta página web por simplicidad del proceso y que la tarea de averiguar el tamaño de la muestra sea lo más expedita y sencilla posible.

3.6 Métodos de Recolección

La recopilación de información es una etapa crítica dentro de todo trabajo de investigación. Por ello, se debe contar con las mejores herramientas para esta tarea, las cuales provean al investigador de datos precisos, claros y relevantes para su posterior análisis. Es tan

importante que puede afectar negativamente la manipulación de la información y, como consecuencia, no ser de utilidad alguna para ayudar a responder las preguntas de investigación.

Hernández Sampieri se refiere a la recolección de datos como "la elaboración de un plan detallado de procedimientos que nos conduzcan a reunir datos con un propósito específico". (Hernández Sampieri, 2014, p. 198)

Como consecuencia, este mismo autor conceptualiza el instrumento de medición como "el recurso que utiliza el investigador para registrar información o datos sobre las variables que tiene en mente". (Hernández Sampieri, 2014, p. 199)

A continuación, en los siguientes puntos, se detallan los instrumentos de recolección a utilizar en este trabajo de investigación.

3.6.1 Cuestionario.

Con la finalidad de recolectar los datos de una forma estándar, que permita la fácil medición y evaluación de los valores que pueden almacenarse en las variables de estudio, se ha decidido optar por la utilización de cuestionarios como el principal instrumento de recopilación de información. Adicionalmente, su elaboración permitirá la expresión de datos de una forma más representativa y menos abstracta, con respecto a la capacidad y conocimiento que poseen las micro y pequeñas empresas dedicadas al desarrollo de *software*, así también como el contraste con la teoría del modelo de *software* como servicio.

Desde la posición de Chasteauneuf (2009), se alude a que "un cuestionario consiste en un conjunto de preguntas con respecto de una o más variables por medir". (Chasteauneuf, 2009, citado en Hernández Sampieri, 2014, p. 217)

Asimismo, empleando las palabras de Baena Paz (2017), “es el instrumento fundamental de las técnicas de interrogación, hay elementos que debemos considerar en la elaboración de las preguntas, tanto su clase como la manera de redactarlas y colocarlas en el cuestionario”. (p. 82)

Por ende, como se detalla más adelante en el apartado de la matriz metodológica, este instrumento será clave en la obtención de los datos para alimentar este trabajo de investigación y que esté en sincronía con sus objetivos. El uso de preguntas abiertas o cerradas, así también como escalas de medición, serán los elementos primordiales en los cuales se apoye este cuestionario.

3.7 Elaboración de Instrumentos

Como se precisó en el punto 3.6, donde se menciona explícitamente los métodos de recolección, el que se empleará durante este trabajo será el cuestionario. Para su diseño y fabricación se estará tomando los indicadores que forman parte de las dimensiones de las variables de los objetivos de la tesis de graduación, como elemento importante para elaborar las preguntas.

En adelante, se contemplará el aplicar las preguntas cerradas, en primera instancia para la mayoría de las preguntas, y dejar solamente una o dos preguntas finales abiertas. El objetivo de tal escogencia, ciertamente, está en la aplicación de las escalas a las opciones de respuesta que contendrá cada uno de los ítems. Esto permite, significativamente, que se pueda asignar valores a los indicadores y que puedan ser cuantificables y/o medidos a cierto nivel.

La definición conceptual de la pregunta cerrada, según Hernández Sampieri (2014), radica en que “Son aquellas que contienen opciones de respuesta previamente delimitadas. Resultan más fáciles de codificar y analizar”. (p. 217)

Por su parte, este mismo autor describe la pregunta abierta como “las que no delimitan de antemano las alternativas de respuesta, por lo cual el número de categorías de respuesta es muy elevado; en teoría, es infinito y puede variar de población en población”. (Hernández Sampieri, 2014, p. 220)

En tercer lugar, se emplea la técnica de escalas de medición para una clasificación cuantificable de los valores que puedan tomar los indicadores a mesurar. Dentro de las escalas de medición se encuentran: escala nominal, escala ordinal, escala de razón y escala de calificación. Se describen en los siguientes puntos:

Escala nominal: es una escala muy general y simple. Generalmente se utiliza para describir sujetos nominales. “Se puede clasificar en varias categorías, excluyentes entre sí, entre las que no es posible establecer una relación de orden y tampoco es posible operar matemáticamente”. (Hernández Martín, 2012, p. 20)

Se utilizará para medir condiciones de existencia entre los indicadores, por ejemplo: si existe una arquitectura definida a usar dentro del proceso de desarrollo de sistemas de información, también para describir elementos que forman parte de un proceso u objeto, las fases de un proceso de desarrollo de *software*, etc.

Escala ordinal: este tipo de escala es un poco más elaborada que la anterior, con la diferencia de que puede tomar más valores y el orden es un factor importante. Se puede afirmar que:

> Un conjunto de datos se denomina ordinal si a los valores u observaciones que pertenecen a él se les puede asignar un orden o asociar a una escala. Los datos ordinales pueden ser

contados y ordenados, pero no pueden ser medidos. (Cortés, Iglesias y Carmen, 2005, p. 31)

Esta herramienta será útil para la localización y clasificación de los indicadores que tiendan a tener valores para medir el nivel de importancia de uno sobre el otro con respecto a una condición específica; por ejemplo: el nivel de encriptación de datos utilizado en la comunicación a través de la red de los sistemas de las empresas.

Escala de razón: de acuerdo con Hernández Sampieri (2014), se tienen todas las características de la escala de intervalos: como unidad de medida, intervalos iguales entre categorías y aplicación de operaciones aritméticas. Además de ello, el cero es real y absoluto (no es arbitrario). (p. 2016)

Se puede inferir que, esta escala, entonces, es para la obtención de valores numéricos con los que se puede operar matemáticamente, tanto de manera sencilla como compleja, para generar resultados más precisos teniendo en cuenta que cuentan con una unidad de medida básica. Por ende, se hace el uso en indicadores de tiempo (como el tiempo de entrega), tasas porcentuales (tasa de retención de clientes) y cantidad de dinero (precios).

Escala de calificación: este tipo de escala se considera un tanto particular, ya que se utiliza mucho en la educación y otros ámbitos científicos para la evaluación de ciertos rubros o criterios de un supuesto. De acuerdo con la autora M.Sc. Laura Jiménez Aragón, en una publicación de la revista *UMBRAL* del Colegio de Licenciados y Profesores en Letras, Filosofía, Ciencias y Artes [Colypro], afirma lo siguiente:

Este tipo de matriz utiliza escalas cualitativas (exc., m.b., b., reg., insuf.) o cuantitativas (1, 2, 3, 4, 5); estas escalas deben estar integradas por más de dos valoraciones. La

> condición base para determinar la validez de la escala elegida consiste en que, si todos los criterios no pueden ser evaluados con propiedad con la escala elegida, se está frente a dos posibilidades: la primera es que se requiere buscar otra escala; o bien, la segunda es que la escala de calificación no es el tipo de matriz de evaluación requerida para valorar los criterios que se incluyeron en la matriz. (Jiménez Aragón, 2018, p. 5)

Para efectos de este documento, se utilizará tanto en la parte del trabajo de campo para ciertos indicadores, como los roles dentro las organizaciones de desarrollo de *software*, así también como en la parte posterior del estudio para evaluar la calidad, validez y veracidad de la propuesta a la problemática de investigación.

3.8 Tabulación y Manejo de Información

La tarea de generación de conocimiento no es un proceso de pasos simples, sino que, en conjunto con muchos elementos más, se convierte en un proceso sistemático que requiere de paciencia, precisión y tiempo. Como aspecto importante, el sucesor de la fase de recolección es el proceso de gestión de la información y datos recopilados. A este tratamiento de datos posterior es lo que se conoce como tabulación de los datos.

Esta etapa es importante porque permite al investigador analizar los datos de forma sencilla para obtener los resultados más precisos posible, evitando el sobreesfuerzo al tratar de interpretar la información que en ellos se encuentra. Principalmente, este depende del buen diseño de los instrumentos de recolección de información, por lo que un error en alguno de estos pasos provocaría una reacción en cadena dentro de los demás pasos dependientes, como el análisis e interpretación, que podrían llevar a un resultado errado.

De acuerdo con Hernández Martín (2012), “un primer paso para sacar alguna conclusión de esta masa de datos consiste en reducirla. Para ello se procede a ordenarlos y agruparlos en categorías (este proceso se conoce como tabulación)”. (p. 13)

Por lo tanto, para esta investigación se utilizarán los gráficos y respuestas obtenidos del trabajo de campo por medio de la herramienta Google Forms, y se importarán los datos a programas de procesamiento de hojas de cálculo para su correcta tabulación y presentación. En este caso, se estará utilizando la opción de Microsoft Excel bajo la licencia de suscripción de Microsoft 365, y Power BI en su versión gratuita.

Dentro de las facilidades que ofrecen estas herramientas, una es la rápida obtención de elementos estadísticos, como: la media aritmética, mediana, moda y percentiles, gráficos, tablas de resumen de datos (ejemplo, las tablas de frecuencias), entre otros, que serán de gran utilidad para la correcta interpretación de los resultados que impulsen un análisis concreto.

Como instancia final, luego del análisis, se podrá construir nuevas tablas, gráficas y demás que permitan una selección minuciosa de información que será de ayuda para la parte final de la investigación, la cual involucra la propuesta de solución dirigida a responder las preguntas planteadas al inicio de este proceso de tesis.

3.9 Matriz metodológica

Objetivo Específico	Variables	Conceptualización	Dimensión	Indicadores y Métrica	Instrumentos
Identificar las pautas o prácticas utilizadas por aplicaciones en la industria del desarrollo del *software* dentro y fuera del ámbito nacional, para comprender cómo funcionan y operan los modelos de *Software* como Servicio utilizando infraestructuras en la nube, mediante la consulta pública de autores de artículos, revistas, libros y demás información de referencia inmediata.	Pautas o prácticas utilizadas en el desarrollo de *software* como servicio.	Conjunto de reglas o normativas existentes para la ejecución del desarrollo e implementación de un *software* como servicio.	Ejecución de desarrollo de *software*	Fases del proceso (Escala nominal)	Consultar fuentes de información impresa o digital.
			Implementación del producto de *software*	Fases del proceso (Escala nominal)	Consultar fuentes de información impresa o digital.
	Modelos de *software* como servicio.	Todo *software* que se usa bajo modalidad de subscripción donde el cliente accede a través de Internet, comúnmente por medio de los navegadores web.	Suscripción del servicio	Frecuencia de cobro y facturación (anual, mensual, quincenal)	Consultar fuentes de información impresa o digital.
				Frecuencia de adopción de clientes (Escala nominal)	Consultar fuentes de información impresa o digital.
			Comunicación a través de Internet	Canales de comunicación (Escala nominal)	Consultar fuentes de información impresa o digital.
				Nivel de encripción y protección de datos (Escala ordinal, 1-5)	Consultar fuentes de información impresa o digital.

Conocer las operaciones de tecnologías de la información del sector PYME a nivel nacional en las cuales su actividad se relacione con el desarrollo de *software* por medio de interacciones con los miembros que las conforman, identificando la capacidad en recursos y habilidades que poseen para la implementación de *Software* como Servicio como parte de sus productos o servicios.	Información de las operaciones de tecnologías de la información	Conjunto de datos que permiten describir el proceso de las operaciones de tecnologías de la información de forma integral del sector PYME a nivel nacional de desarrollo de *software.*	Operaciones de Infraestructura	¿Existe un proceso definido? (Escala ordinal, 1-5)	Cuestionario dirigido a los participantes
				¿Existe documentación del proceso? (Escala Nominal, SÍ/NO)	Cuestionario dirigido a los participantes
				Nivel de confianza en los procesos (Escala ordinal, 1-5)	Cuestionario dirigido a los participantes
				Frecuencia de modificación de la documentación (Escala nominal: anual, mensual, o quincenal)	Cuestionario dirigido a los participantes
				Cantidad de recursos (Escala de razón: número de nodos)	Cuestionario dirigido a los participantes
				Tiempo respuesta a interrupción total del servicio (Escala de razón: HH:mm:ss)	Cuestionario dirigido a los participantes
				Tiempo respuesta a degradación del servicio (Escala de razón: HH:mm:ss)	Cuestionario dirigido a los participantes
				¿Es necesaria una mejora? (Escala nominal: SÍ/NO)	Cuestionario dirigido a los participantes

				Nivel de urgencia de las mejoras (Escala ordinal, 1 -5)	Cuestionario dirigido a los participantes
				Tiempo de entrega de mejoras (Escala de razón: horas, días, meses)	Cuestionario dirigido a los participantes
			Operaciones de Desarrollo	¿Existe un proceso definido? (Escala ordinal, 1-5)	Cuestionario dirigido a los participantes
				¿Existe documentación del proceso? (Escala Nominal, SÍ/NO)	Cuestionario dirigido a los participantes
				Nivel de confianza en los procesos (Escala ordinal, 1-5)	Cuestionario dirigido a los participantes
				¿Existe una metodología de arquitectura definida o predominante? (Escala nominal: SÍ/NO)	Cuestionario dirigido a los participantes
				Frecuencia de uso de la arquitectura de software predominante (Escala nominal)	Cuestionario dirigido a los participantes
				¿Es necesaria una mejora? (Escala nominal: SÍ/NO)	Cuestionario dirigido a los participantes
				Tiempo de entrega de implementación mejoras al proceso	Cuestionario dirigido a los participantes

				(Escala de razón: horas, días, meses)	
				Uso de modelo de entrega de *software* (Escala nominal)	Cuestionario dirigido a los participantes
	Miembros de las micro y pequeñas empresas (PYMES) en Costa Rica que se dediquen al desarrollo de *software* como principal actividad económica	Grupo de colaboradores que forman parte de las micro y pequeñas empresas (PYMES) y que forman parte del Departamento de Tecnologías de Información	Recurso Humano	¿Existe personal calificado? (Escala nominal: SÍ/NO)	Cuestionario dirigido a los participantes
				Roles y responsabilidades definidas (Escala de calificación)	Cuestionario dirigido a los participantes
				Nivel académico del personal (Escala nominal)	Cuestionario dirigido a los participantes
Analizar los datos recopilados de la capacidad y habilidades que presentan los emprendedores costarricenses del sector PYME dedicados al desarrollo de *software* por medio del contraste de las prácticas, técnicas y procesos utilizados actualmente por la industria de las tecnologías de la información para el descubrimiento del marco de trabajo, conocimiento, métodos y procesos necesarios y aplicables al sector en estudio.	Marco de trabajo de aplicaciones de *software* como servicio	Consiste en una agrupación de conceptos, prácticas, técnicas y criterios para resolver problemáticas similares a través de una referencia.	Implementación y operación de la infraestructura	Guía de diseño de infraestructura (Escala de calificación)	Información de investigación propia tabulada
				Nivel de confianza en la guía de diseño (Escala ordinal, 1-5)	Información de investigación propia tabulada
				Guía de operación de infraestructura (Escala de calificación)	Información de investigación propia tabulada
				Nivel de confianza en la guía de diseño (Escala ordinal, 1-5)	Información de investigación propia tabulada
			Ejecución de desarrollo de *software*	Fases del proceso (Escala de calificación)	Información de investigación propia tabulada

			Implementación del producto de *software*	Fases del proceso (Escala de calificación)	Información de investigación propia tabulada
Producir una guía sistemática para la implementación de un producto de *Software* como Servicio mediante la estandarización de conocimiento en metodologías y procesos para traer beneficios operativos en los emprendimientos costarricenses del sector PYME tecnológico como proveedor, así como también a los consumidores.	Describir una guía de proceso de implementación en infraestructura	Se conocen los pasos a seguir como parte de un proceso sistemático para el diseño e implementación de la infraestructura de un producto de *software* como servicio	Proveedor de servicio	¿Se puede crear guía de selección de servicios o herramientas en la nube? (Escala de calificación)	Se describe la información de la investigación propia
			Diseño	¿Se puede guiar la creación del producto con una arquitectura adecuada? (Escala de calificación)	Se describe la información de la investigación propia
			Implementación	¿Se puede crear un estándar de pasos para desarrollo y configuración de la infraestructura? (Escala de calificación)	Se describe la información de la investigación propia
	Describir una guía de proceso de implementación en desarrollo	Se conocen los pasos a seguir como parte de un proceso sistemático para el diseño e implementación de un producto de *software* como servicio	Arquitectura de *software*	¿Se puede crear guía de selección de servicios o herramientas en la nube? (Escala de calificación)	Se describe la información de la investigación propia
				¿Se puede guiar la creación de un producto con una arquitectura adecuada?	Se describe la información de la investigación propia

				(Escala de calificación)	
			Implementación	¿Se puede crear un estándar de pasos para desarrollo y configuración del *software*? (Escala de calificación)	Se describe la información de la investigación propia
	Describir una guía de proceso de operaciones de un *software* como servicio	Se conocen los pasos a seguir como parte de un proceso sistemático para la operación de un producto de *software* como servicio	Operaciones	¿Se pueden describir los mejores procesos y prácticas para el manejo de las operaciones diarias? (Escala de calificación)	Se describe la información de la investigación propia
				Se pueden describir los mejores procesos y prácticas para el manejo de cambios (Escala de calificación)	Se describe la información de la investigación propia

CAPÍTULO IV. ANÁLISIS DE RESULTADOS

4.1 Análisis de Resultados

En esta fase de la investigación se describe, en detalle, los datos recopilados, así como la interpretación de estos, a través de un análisis minucioso de las respuestas obtenidas de los colaboradores del área de Tecnologías de la Información, de las micro y pequeñas empresas (PYME), que se establecen en el sector de Alajuela.

Como expectativa, el desglose de los elementos evaluados permitirá proveer una perspectiva más amplia de cómo se desenvuelven este tipo de organizaciones en el mercado costarricense. La finalidad radica en validar la capacidad de mejora y estandarización en su operación que, en apariencia, es diferente para la mayoría de estas empresas, ya que brindaría un camino más claro para todos aquellos que deseen emprender dentro del campo de desarrollo de *software*.

Por su parte, la encuesta realizada estaba compuesta de varios ítems de selección, tanto única como múltiple. Estos, a su vez, preguntaban directamente acerca de la percepción de cada colaborador con respecto a los procesos operativos de la organización a la cual pertenecen. Algunas de estas preguntas tenían como objetivo evaluar la madurez y la confianza de cada persona para con el flujo de trabajo que, normalmente, empresas de esta índole poseen en sus filas.

En este sentido, los procesos operativos de una organización forman parte crítica de cómo se desarrolla una empresa de tecnología en el mercado de la competencia laboral. La metodología es fundamental para alcanzar el éxito dentro del desarrollo de *software*; por ende, se enfoca en este aspecto organizacional para verificar, desde un punto funcional, si la operación de crear y entregar *software* es adecuada para un producto de *software* como servicio.

El por qué del enfoque en dicho modelo de entrega de *software* radica en lo ágil y adaptable que puede llegar a ser en una población de clientes donde la inversión inicial, a gran escala, no es una opción viable. No obstante, el mismo principio aplica a los proveedores del servicio o, bien, del producto. El ser capaz de canalizar un programa a través de una subscripción proporciona ingresos iniciales significativos que le dan fluidez a una organización cuya operación gira alrededor de los servicios.

Ahora bien, en la mayoría de los casos, la población estudiada manifiesta tener conocimiento alguno de como operar productos de *software* como servicio. De estas, el 60% son de tamaño micro (9 de 15), y el resto, que representa un 40% (6 de 15), es de tamaño pequeño. Esto dicta una leve pluralidad de microempresas de desarrollo de *software* que sitúan sus operaciones en el cantón y distrito económico central de Alajuela.

Al mismo tiempo, se puede señalar que al menos unas 4 compañías cambiaron su tamaño original y pasaron a ser de micro a pequeñas empresas. Esto se contrasta con las 13 microentidades representadas en el reporte del Ministerio de Industria y Comercio con corte a febrero de 2021, y la información que brinda el Ministerio de Hacienda. Aun así, esto demuestra un crecimiento bastante rápido de este tipo de organizaciones.

Mejor aún, entienden la diferencia entre los modelos de entrega de *software* que ofrece la computación en la nube para tomar ventaja de ellos, combinando servicios para obtener un producto final de alta calidad. Esto ofrece un ejemplo de lo sencillo que se ha vuelto el acceder a servicios de computación en la nube para personal que, de otra forma, no tendrían paso a invertir en soluciones completas de TI para infraestructura y servicios de *software*.

4.2 Resultados por Indicador

4.2.1 Roles presentes en las empresas.

Un aspecto importante que medir dentro de las estructuras organizacionales es la existencia de roles definidos, esto por la generalidad que tienen empresas de menor tamaño de sostener menos personal por responsabilidad; es decir, que una persona puede llegar a cubrir varias obligaciones que, de otra forma, en una organización de mayor tamaño tendría una persona por tarea específica.

En la figura 4.1 se muestra un gráfico que refleja el porcentaje de existencia de roles definidos dentro de las micro y pequeñas empresas (PYMES) de desarrollo de *software*:

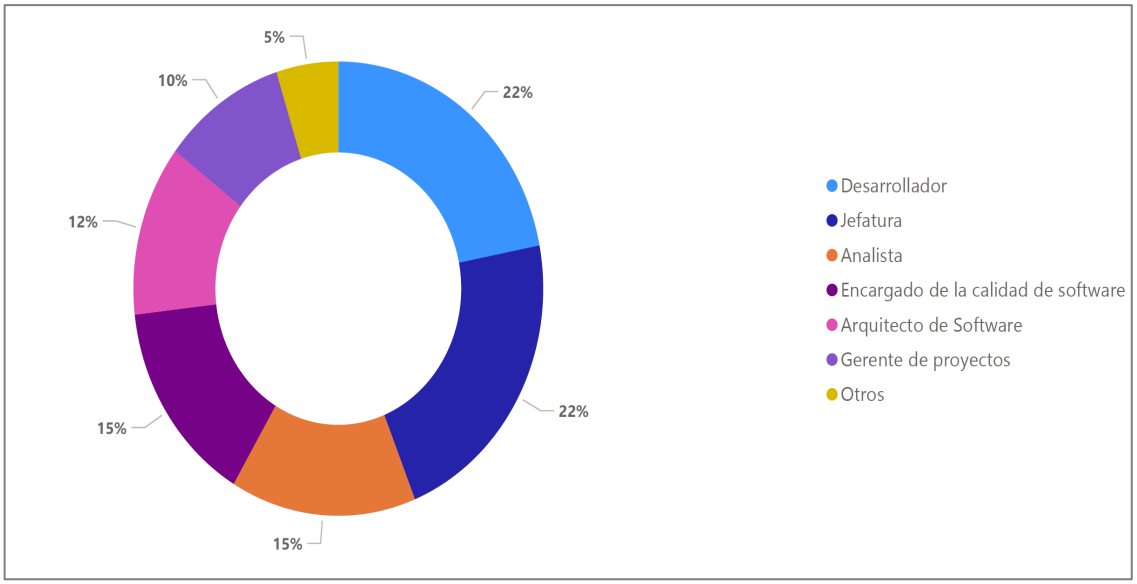


Figura 4.1. Porcentaje de existencia de roles técnicos y administrativos presentes en las micro y pequeñas empresas (PYMES) consultadas. Fuente: investigación propia.

Desde la perspectiva más general, se puede apreciar que el rol de jefatura y el de desarrollador son los más abundantes. Este porcentaje se puede traducir en un total de 9 personas

de 16 que respondieron desenvolver al menos un papel de desarrollador o jefe en la empresa para la cual laboran. Si se apela a justificar la existencia de estos roles sería simple: la mayoría de los emprendedores comienzan su negocio siendo ellos los principales autores del producto que venden. De modo similar y dependiendo del tipo de sociedad que se forme, la jefatura es necesaria para la supervisión de las operaciones dentro de una organización.

Lo anterior, a diferencia de otros roles cuya presencia disminuye en cierta medida, como es la gerencia de proyectos, entre otros, quienes no estaban enlistados en el cuestionario. Esto dicta una forma de administrar los proyectos de desarrollo de forma más ágil y susceptible a cambios. Al existir menos personal calificado para esta tarea, puede existir mayor riesgo de fallo a la hora de entregar productos finales, mejoras o arreglos a los clientes.

Si se compara la existencia de estos roles con el número de colaboradores que participaron del estudio, rápidamente se puede evidenciar la validez de la premisa de que un trabajador de este tipo de entidad ejerce al menos dos responsabilidades o más. En la tabla 4.1 se visualiza el número de personas con la cantidad de role(s) que poseen en las micro y pequeñas empresas:

Tabla 4.1

Cantidad de colaboradores por combinación de rol

Roles	**Cantidad**
Analista, Encargado de la calidad del *software*, Otros	1
Desarrollador	1
Desarrollador, Analista	3
Desarrollador, Encargado de la calidad de *software*	1

Encargado de la calidad de *software*	1
Jefatura	1
Jefatura, Arquitecto de *Software*, Analista, Encargado de la calidad de *software*	1
Jefatura, Arquitecto de *Software*, Desarrollador	2
Jefatura, Arquitecto de *Software*, Desarrollador, Analista, Encargado de la calidad de *software*, Gerente de proyectos, Otros	1
Jefatura, Arquitecto de *Software*, Encargado de la calidad de *software*	1
Jefatura, Desarrollador, Gerente de proyectos	1
Jefatura, Gerente de proyectos	2
Total	**16**

Nota. Investigación propia.

De forma similar, Ponsard y Deprez (2018) mencionan que desarrollar *software* es un reto para las pequeñas y medianas empresas, especialmente para asegurar la calidad, tiempo y limitados presupuestos. Estas difieren de organizaciones más grandes porque tienen limitados y, frecuentemente, menos recursos especializados. (p. 213)

Por consiguiente, esta estrecha relación que existe entre el número de colaboradores que pertenecen a este tipo de empresas no difiere de los números mostrados en la población de estudio. Proporciona un reto a mejorar para este tipo de negocios, el cual no es sencillo en ningún aspecto.

4.2.2 Claridad de responsabilidades.

Habitualmente, el punto anterior sobre la existencia y ejecución de varios roles por persona crea una circunstancia que puede llegar a afectar directa o indirectamente el trabajo realizado. Resulta lógico pensar que al efectuar diferentes tareas pueda llegar a ser confuso, en algún momento del tiempo, los límites de las responsabilidades que se tienen a cargo. Por esta razón, en esta sección, la medición de este ítem es clave para el comprender cómo lo perciben los colaboradores.

Ahora bien, para esta pregunta se utilizó una escala de Likert para que las opciones presentadas a los trabajadores fueran sencillas y claras de responder. En la figura 4.2 se puede visualizar las respuestas de los participantes:

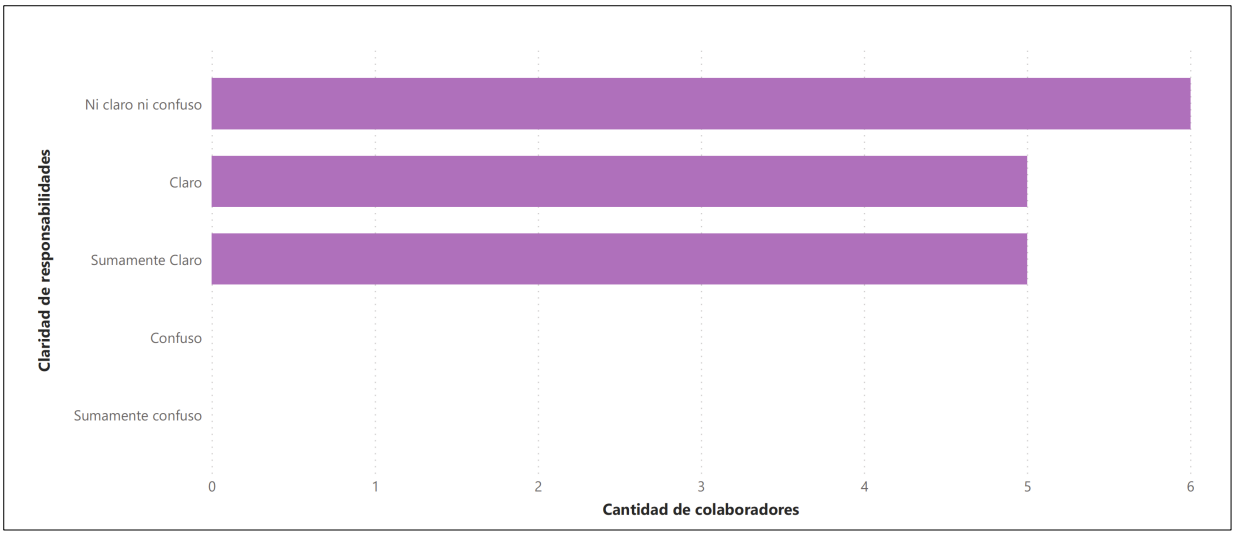


Figura 4.2. Representación gráfica de la claridad de las obligaciones por colaborador. Fuente: investigación propia.

El análisis procedente del gráfico de la figura 4.2 permite interpretar y clasificar en dos grupos: el primero, aquellos cuyos puestos tienen los límites de sus obligaciones bien definidos;

y segundo, los que no presentan esta situación, pero tampoco están tan confundidos sobre cómo proceder para cumplir con el día a día de su trabajo.

En otras palabras, se evidencia una proporción del 62,6% que representa a todos los colaboradores de micro y pequeñas empresas (PYMES), quienes identifican claramente las responsabilidades de sus puestos. Es un alto porcentaje, aun sabiendo que 13 personas dicen cumplir con dos o más roles al mismo tiempo, lo cual es alrededor del 81,3% de los encuestados.

Visto de esta forma, los datos sugieren que el desempeñar varios cargos al mismo tiempo no impacta demasiado la capacidad de comprensión de los participantes; es decir, que en un entorno donde las reglas podrían ser muy ágiles y flexibles, los colaboradores pueden identificar los pasos con precisión para llevar a cabo sus deberes laborales. Sin duda, esto es beneficioso para estas organizaciones, cuyo aporte por miembro del equipo se vuelve muy crítico, ya que impacta bastante las operaciones de estas.

A pesar de que la apreciación es muy buena comparada con una expectativa inicial, la cual era de tener personal un poco confundido con respecto de las labores que deben ejecutar; se vuelve necesario recalcar que no es lo mismo que el cumplimiento de estas funciones. Este último indicador no se abarcó ya que implicaría evaluar otras variables para llegar a la conclusión de si el rendimiento del empleado es el adecuado cuando posee dos o más cargos. Por lo tanto, el cumplimiento de las obligaciones quedó fuera del alcance de este estudio.

Teniendo en cuenta estos aspectos, se puede inferir que las micro y pequeñas empresas (PYMES) de desarrollo de *software* tienen buena comunicación de objetivos e instrucciones para lograr estos y llevarlos a la meta final. lo cual es, de cierta forma, esperado en un entorno con pocos colaboradores.

4.2.3 Nivel de educación de los colaboradores.

Otro aspecto importante es la formación académica de los participantes, indicador que sirve para medir qué preparación es la que se encuentra frecuentemente en una micro o pequeña empresa (PYME) de desarrollo de *software* en este ámbito. Mucho más cuando se trata de entregar un producto de *Software* como Servicio.

Seguidamente, en la figura 4.3 se presentan los niveles académicos alcanzados por los participantes de la encuesta:

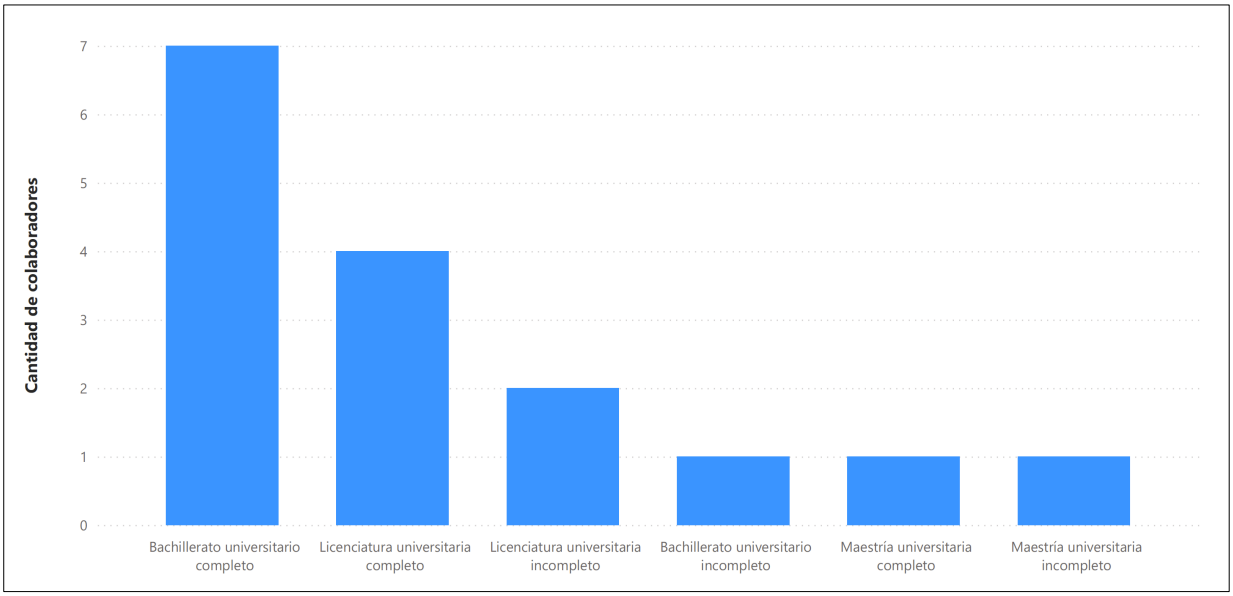


Figura 4.3. Niveles académicos presentes en los participantes. Fuente: investigación propia.

Primeramente, se valora que el total de los encuestados poseen algún grado de conocimiento de estudios superiores sin importar si terminó o no sus estudios universitarios. Este punto es interesante dentro del marco de las PYME, ya que un número alto de personas ha requerido estudios especializados dentro de estos emprendimientos.

Traducido a números, solamente una persona no cuenta con el título de conclusión de estudios universitarios, lo cual simboliza el 6,25%. De todas maneras, aunque no cuente con la acreditación respectiva, no se puede afirmar con certeza que no haya cursado y obtenido cierto conocimiento a nivel superior. En cambio, de los colaboradores que sí concluyeron sus niveles universitarios, un 68% posee al menos el bachillerato o la licenciatura, lo cual deja otro porcentaje reducido para los otros niveles especializados como la maestría; sin embargo, solo una persona la concluyó, pues deja a los demás concursantes con estos niveles sin terminar.

Por otro lado, esto contrasta con los datos mostrados por otros estudios realizados en poblaciones similares (si bien, a un nivel más general y no solamente empresas del sector tecnológico). Uno de ellos es el llevado a cabo por el Observatorio de Desarrollo de la Universidad de Costa Rica (2018), que en su informe detalla que para las microempresas solamente un 28,9% tiene educación universitaria; mientras que, en el caso de las pequeñas, este porcentaje disminuye a solo un 24,6%. (pp. 35-36)

Se adopta, entonces, una postura de que las micro y pequeñas empresas (PYME) de tecnología que se dedican al desarrollo de *software* en el distrito económico de Alajuela, son más propensas a tener en sus planillas personal con un alto grado de formación académica que sus semejantes en el mercado laboral. Se quiere, con ello, significar que la mano de obra calificada que existe dentro de estas entidades ayuda a que puedan definir varios aspectos de sus operaciones con mayor precisión, siendo uno de ellos, los roles y sus tareas específicas para lograr sus objetivos como organización.

4.2.4. Modelos de entrega de *software.*

En relación con la problemática principal expuesta, no se podía escapar el haber medido el modelo a utilizar por parte de las micro y pequeñas empresas. Dentro de este orden de ideas, es necesario, ya que se pretendía conocer la forma como se crean y distribuyen sus productos de *software*, y a su vez, este aspecto iba a proveer información valiosa acerca del conocimiento que cada organización participante tiene actualmente.

Antes que nada, a modo de aclaración, este ítem se midió a nivel de empresa y no por colaborador. Lo anterior, debido a que la metodología utilizada es generalmente aplicada a toda la organización. En los casos expuestos tampoco fue la excepción. También esto se refleja en la tabulación y presentación de las respuestas a esta pregunta, cuya visualización se puede ver en la figura 4.4:

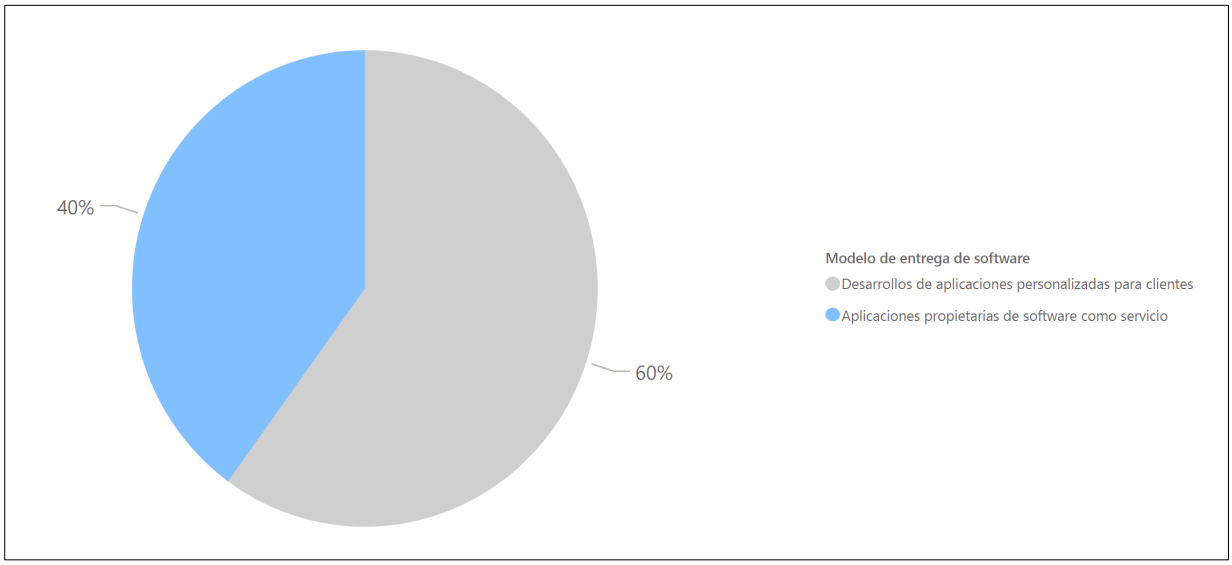


Figura 4.4. Proporción del uso del modelo de entrega de *software* utilizado por empresa. Fuente: investigación propia.

En este trabajo se propuso evaluar el modelo de *Software* como Servicio contra los productos entregados bajo la modalidad de personalización. Se examinan porque las aplicaciones desarrolladas a la medida son abundantes dentro del mundo del desarrollo de *software*, especialmente en Costa Rica. Bajo esta evidencia, se creó este ítem para obtener claramente la utilización de un modelo sobre el otro.

En conjunto, mayor porcentaje de participación se traduce en que 9 de 16 empresas siguen haciendo uso del modelo de entrega tradicional de *software* personalizado a sus clientes; es decir, que el producto se desarrolla y entrega en su totalidad para ser parte de la propiedad del cliente, incluyendo el diseño, el código fuente, datos, etc. Por consiguiente, se muestra que las premisas consideradas al inicio del estudio son reales.

No obstante, no deja de ser sorprendente la proporción de empresas que dicen apostar al modelo de *software* como servicio para ofrecer sus productos. Cabe indicar que el 40% de participación es mayor a la expectativa que se asumía en este trabajo de investigación. En tal sentido, constituye un hallazgo sumamente interesante y, del mismo modo, da validez a la afirmación mencionada en la primera parte de este estudio, la cual atribuye un nivel de conocimiento sobre el *software* como servicio aceptable para estas organizaciones.

Sucede pues, que al tener estos datos se podría considerar que, en poco tiempo, especialmente durante la siguiente década, sería un modelo dominante en la mayoría de los productos de *software* que se llegasen a concebir. Es por ello, que recolectar esta información es relevante para saber qué capacidad de aprendizaje y adopción tienen las micro y pequeñas empresas (PYME) de tecnología de introducir nuevos programas con esta característica.

Por eso, además de centrarse en el modelo principal en el que se basan sus productos de *software*, también se decidió medir los tipos de modelos de entrega de *software* de la

computación en la nube. En la figura 4.5 se visualiza la utilización de una combinación de estos para llegar a un resultado final:

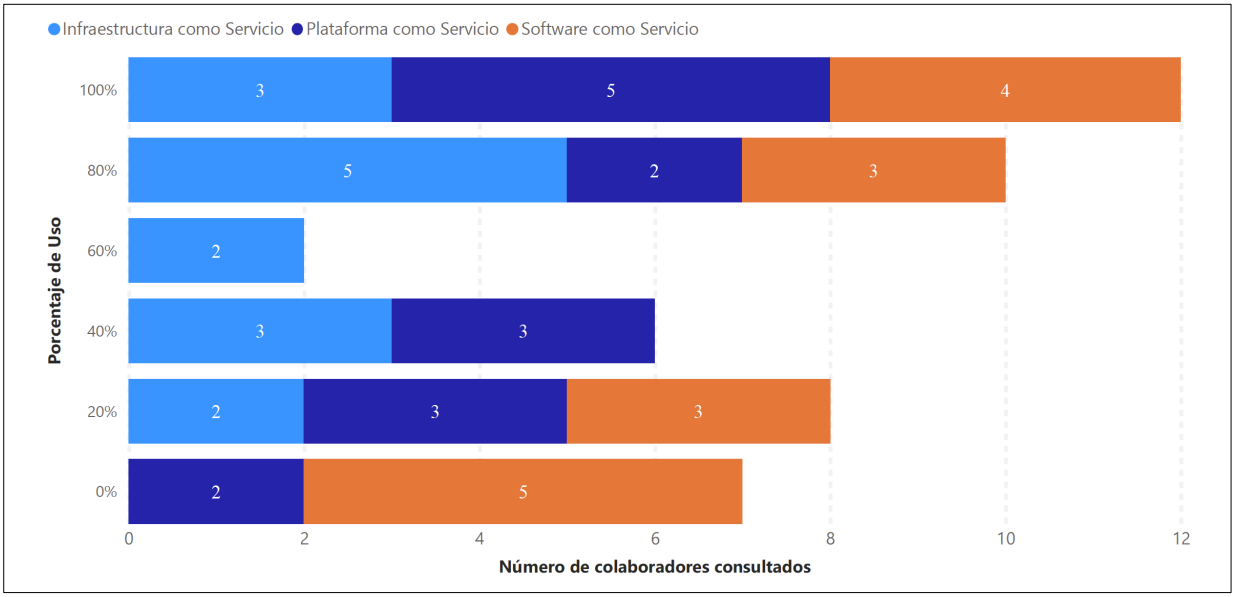


Figura 4.5. Representación del empleo de los modelos de entrega de computación en la nube. El porcentaje de utilización está separado en tres categorías y el respectivo número participantes dado en cada empresa. Fuente: investigación propia.

Brevemente, sale a relucir un jugador clave en el éxito de la computación en la nube: la Infraestructura como Servicio, el cual tiene, prácticamente, una participación total. Ninguno de los participantes consultados dijo no tener que ver con el modelo de IaaS. Sin embargo, esta no es la realidad con respecto al *Software* como Servicio y a la Plataforma como Servicio, de los cuales al menos un 46,66% de los entrevistados dijo no utilizar del todo estos modelos.

Por su parte, de los participantes que respondieron sobre el empleo de estos otros modelos para entregar sus propios programas de *software*, afirman hacer uso de ellos en menor porcentaje. En relación con este tema, se propicia una respuesta, entonces, con la que se puede

enlazar la operación de entregar *software* personalizado y como servicio utilizando entornos completamente virtuales en la nube.

Sin duda, el beneficio que estas empresas han obtenido a través del auge de la computación en la nube es tremendo. Estos datos revelan la magnitud de la frecuencia con la cual más micro y pequeñas empresas se insertan en el mercado rápidamente con solo subscribirse a un servicio de algún proveedor de computación en la nube.

4.2.5 Madurez de los procesos operativos.

A continuación, se presentan los datos recopilados acerca de los procesos operativos que forman parte del flujo de trabajo de una empresa de desarrollo de *software*. La lista de procedimientos está pensada en relación con los problemas de negocio más esenciales en los cuales las micro y pequeñas empresas se enfrentan en su cotidianidad; es decir, el número de tareas necesario para concebir un producto de *software* en todo su ciclo de desarrollo. Para ello, se toman aspectos técnicos (como es la administración de la infraestructura y aspectos del desarrollo mismo de aplicaciones), así también como aspectos administrativos (como la gerencia de proyectos, prácticas de metodologías ágiles, etc.)

En la figura 4.6 se presenta el nivel de madurez alcanzado por cada empresa en cuestión de sus procesos operativos. Cada uno tienen una evaluación por separado, inclusive un color distintivo para su rápida clasificación, en el cual los colaboradores ayudaron a identificar mediante el autodiagnóstico durante la encuesta realizada.

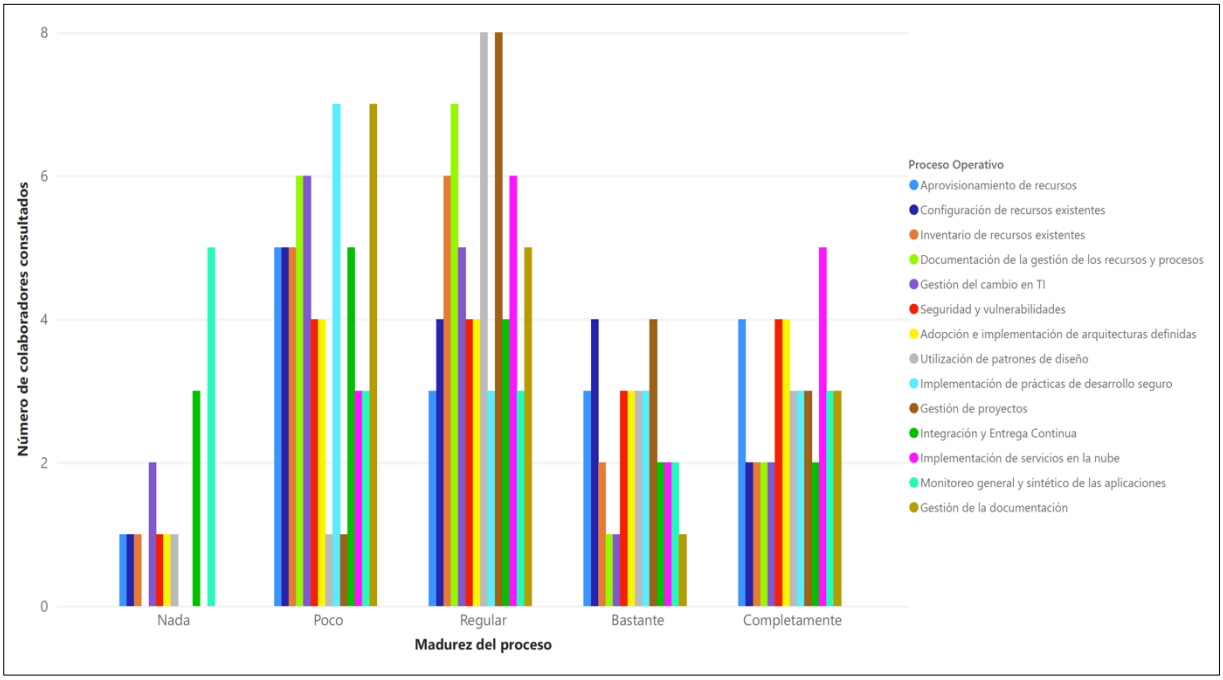


Figura 4.6. Representación del nivel de madurez de los procesos operativos. Fuente: investigación propia.

En primer lugar, se puede observar que un gran número de empresas posee un nivel aceptable de madurez en sus procesos. Al menos, esto se señala en el gráfico anterior donde una gran cantidad de respuestas se situaron entre los niveles: regular, bastante y completamente, lo que puede indicar que dentro de sus operaciones diarias pueden haber encontrado un punto de equilibrio para cumplir con sus obligaciones contractuales con clientes.

Asimismo, cuando se analiza y traduce a números la cantidad de respuestas por proceso operativo: el 31,3% de estas indican como “regular” el estado de madurez en sus procedimientos; mientras que el 15,2% indican que es “bastante”; y el 18,8% se inclinaron por el nivel “completamente”. Por otro lado, un porcentaje del 27,7% calificó como “poco”, o al menos cierto nivel de principiante en la definición de sus procesos de negocio, y solamente el 7,1% dijo no tener madurez en sus procesos del todo.

De esta manera, se reflexiona que las micro y pequeñas empresas son candidatas que tienen un potencial significativo para buscar la estandarización en su forma de producir *software*. Cabe añadir que las varias formas en que se puede producir un producto de *software* aumentan la complejidad con que las entidades autoras acuerden una forma de trabajar. Siempre que existan numerosas opciones, puede ser confuso para nuevos jugadores y, por ende, es contraproducente desde el punto de vista técnico y de competencia en el mercado.

Por ejemplo, uno de los procesos operativos con porcentaje de madurez mayor o igual a "regular" es la implementación de servicios en la nube. Aproximadamente, un 81,25% de los participantes respondió positivamente en este ítem. De este subgrupo, alrededor de un 46,15% evaluó con un nivel "regular", un 38,46%; con "completamente", y el restante 15,38%; con "bastante". Esto no evita reiterar que este tipo de organizaciones acrecienta sus posibilidades de entrega y operación haciendo uso de servicios de la computación en la nube, tal y como lo hacen empresas semejantes de otros sectores comerciales.

Es importante agregar que otro proceso operativo que estuvo muy bien evaluado fue la gestión de los proyectos. Por tal motivo, se podría considerar contradictorio, ya que anteriormente se mencionó que el rol de gerente de proyectos representó apenas el 10% de los encuestados. En concordancia, se puede inferir que, aunque el puesto no se muestre definido en la micro y pequeña empresa (PYME) de desarrollo de *software*, existe una enorme probabilidad de que la jefatura y supervisión de la entidad, así como los desarrolladores, estén ejecutando las labores correspondientes a este rol de trabajo.

Si se apela a un ejemplo se puede encontrar en la figura 4.7, la cual demuestra la predominancia que tiene las respuestas de los participantes en la gestión y administración de proyectos con respecto a que el nivel de madurez sea "regular":

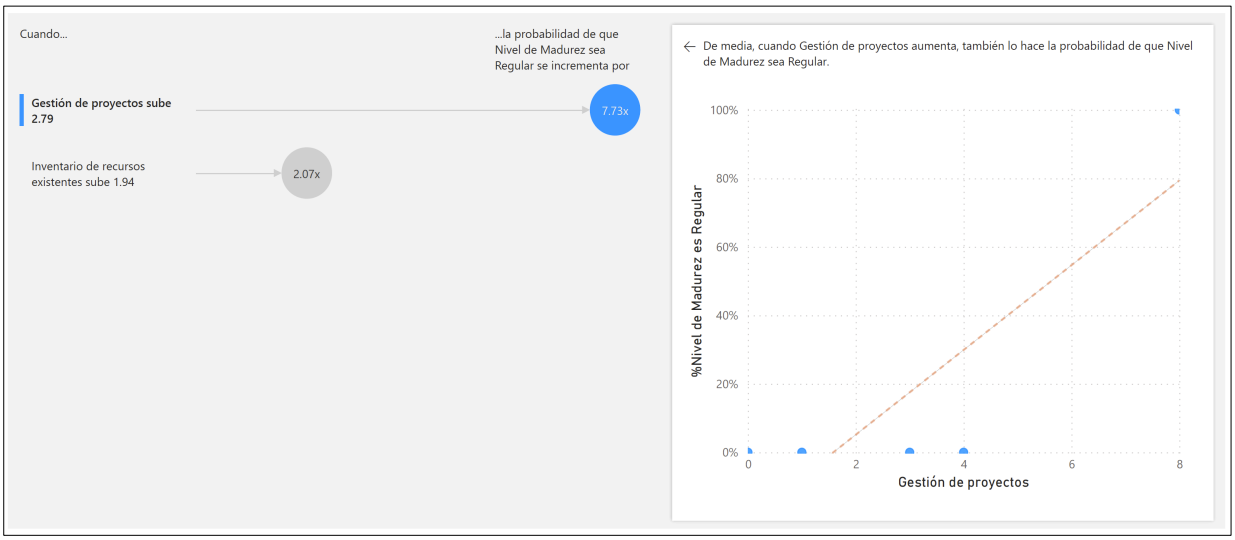


Figura 4.7. Tendencia del dominio del nivel de madurez de la gestión de proyectos. Fuente: investigación propia.

Así pues, como dato interesante, del análisis se produjo otro incremento que siguió el mismo movimiento. En este caso, el proceso de negocio impactado es el de inventario de recursos existentes, llámese recurso a cualquier bien tangible o intangible que la organización tenga en su poder para el desenvolvimiento de la actividad de programación informática.

De modo similar, se consigue examinar los demás procesos y cómo estos afectan la percepción de los colaboradores de una empresa que se desenvuelve en este terreno. Otro ejemplo que se puede incluir es el del uso de patrones de diseño. Aunque no es sorprendente, por el hecho de que el campo de especialización es el desarrollo de *software*, un 87,50 % de los participantes colocó el nivel de madurez arriba o igual al de “regular”.

Al mismo tiempo, cuando se empieza a ver procesos clave, como el aprovisionamiento de recursos, inventario de recursos y administración de estos, se puede ver un declive en el nivel de madurez. Respectivamente, alrededor del 56,25%, 62,50%, y 75% de los participantes valora tener una madurez de sus procesos de nada, poco o regular, lo que sugiere que la computación en

la nube, aun siendo un aliado muy importante, continúa siendo un reto mayor para estas empresas y sus recursos.

En relación con este tema, otro par de procesos cuya proporción de respuestas es alta con respecto a niveles inferiores de madurez son: integración y entrega continua, y monitoreo sintético de las aplicaciones, ambos con un 50% de participación entre poco o nada de madurez, lo cual supone un reto adicional para las micro y pequeñas empresas, ya que la entrega de *software* debe ser ágil, segura, robusta y rápida; más aún cuando se trata de dar soporte a las aplicaciones en un ambiente de producción. Por ende, este nivel evaluado es un candidato perfecto para la mejora de proceso.

Como ejemplo, en la figura 4.8 se muestra cómo la probabilidad de que la respuesta haya sido un nivel poco de madurez en el proceso, aumenta cuando se trata de evaluar la "Integración y Entrega Continua":

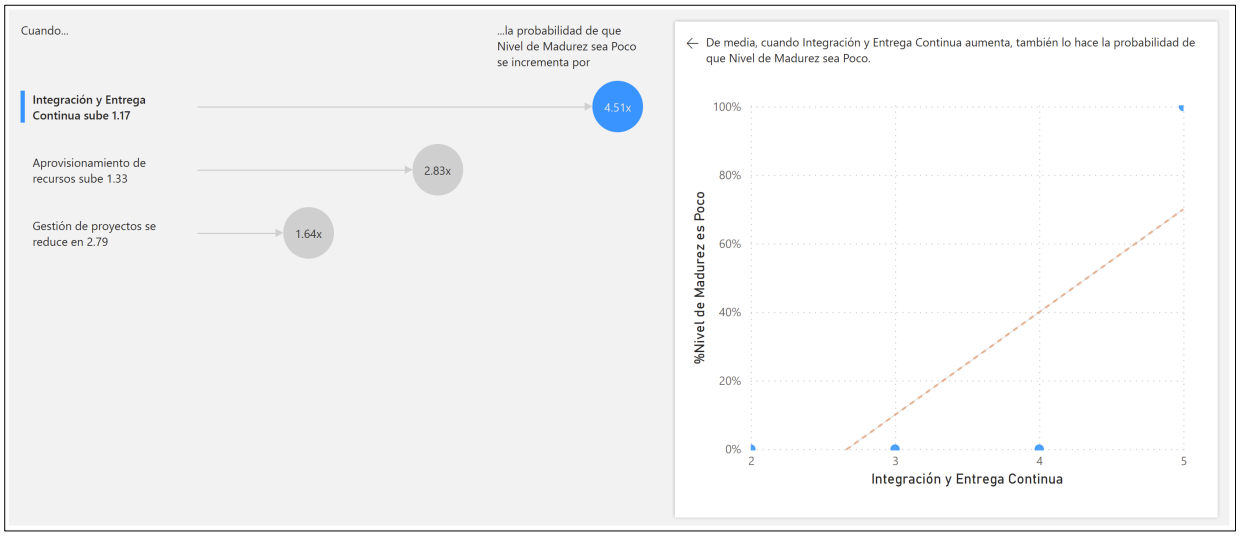


Figura 4.8. Tendencia de nivel bajo de madurez en los procesos. Fuente: investigación propia.

Es interesante ver cómo este patrón de tendencia a la alta y baja se presenta en este tipo de empresas de desarrollo, lo cual, en una entidad con más experiencia y años en el mercado,

supondría prácticas cotidianas e incluso, con un grado mayor de complejidad en la implementación, el cual es todavía un gran desafío para las organizaciones de menor tamaño. En parte, factores como el poco personal y la falta de tiempo afectan grandemente el desarrollo interno de estas.

4.2.6 Nivel de confianza.

En lo concerniente al nivel de fiabilidad que tiene o perciben los colaboradores de las micro y pequeñas empresas (PYME) de desarrollo de *software*, se convierte en indispensable la medición de esta. Obtener la lectura correcta de cómo está ejecutando la organización sus procesos contra lo que sus trabajadores señalan que no funciona óptimamente. Visto de esta forma, se pretende demostrar el comportamiento reflejado en datos acerca de la confianza de los participantes en los procesos operativos de sus respectivas compañías.

Dentro de este marco, primero debe entenderse el término confianza. Este se refiere a "la esperanza firme que se tiene de alguien o algo" (Real Academia Española, s. f., definición 1). Otra interpretación que posee este diccionario es "la familiaridad o libertad excesiva", lo cual sugiere una seguridad y credibilidad en los pasos a seguir para cumplir una tarea(s) específica(s).

Habitualmente, tanto la firmeza como la familiaridad son términos indispensables para que un producto marche bien en una compañía. Al mismo tiempo, es parte de los factores que influyen en la madurez de un proceso. Naturalmente, si no se confía en el procedimiento (llámese aprovisionamiento de los recursos, configuración de los recursos, gestión de proyectos, etc.), entonces, se es propenso en incurrir en errores que, eventualmente, le cuestan dinero y reputación a la organización.

Así pues, en la figura 4.9 se representa el nivel de confiabilidad que existe por parte de los colaboradores de estas empresas en los procesos operativos de cada una. Además, como nota aclaratoria, la lista de procesos sigue siendo la misma que se utilizó en el rubro de madurez de estos:

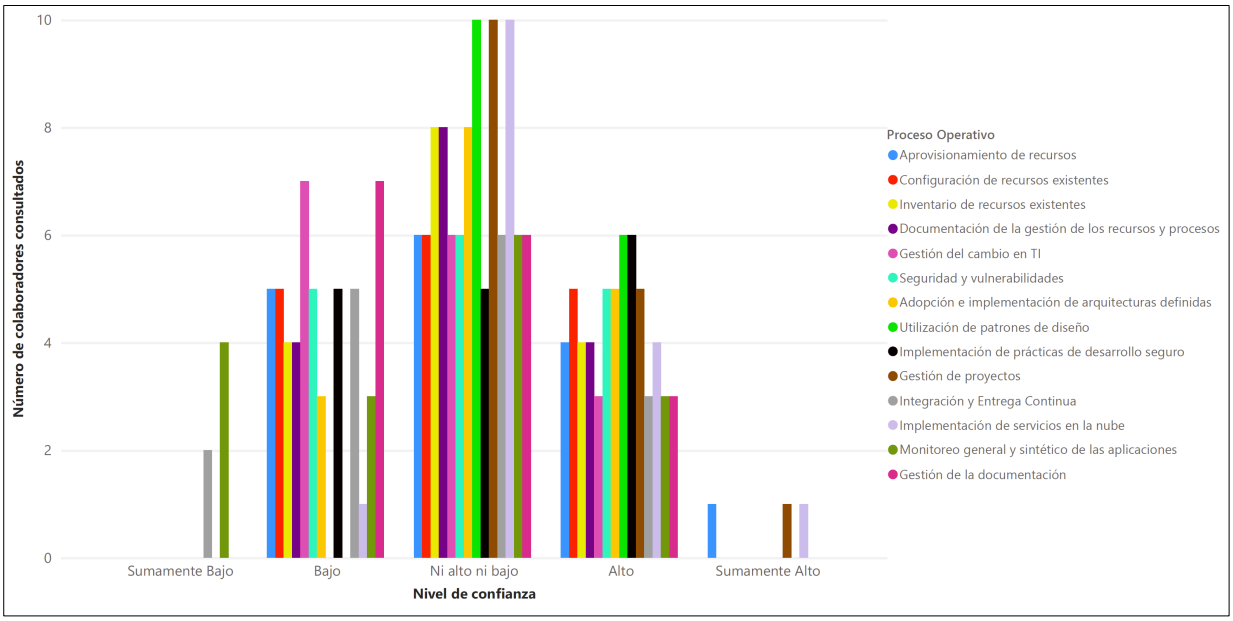


Figura 4.9. Interpretación del grado de confianza en los procedimientos operativos. Fuente: investigación propia.

Resulta claro ver la relación que tiene la confianza con la madurez de los procesos operativos. Ahora bien, dentro del porcentaje de respuestas obtenidas se encontró que el 45,1% de los participantes (al menos 7 de ellos) prefieren no pronunciar un criterio certero con respecto a la credibilidad que disponen de las operaciones de desarrollo de *software* de las empresas para las cuales laboran. Esto es lo mismo que poseer un índice promedio de confianza en los pasos que se realizan dentro de la empresa para obtener algún objetivo o producto.

Por otro lado, el 26,8% de los encuestados dicen poseer un nivel alto de confianza, mientras que otra proporción similar del 24,1% contempla un reducido grado de fiabilidad en este rubro. Al mismo tiempo, en ningún caso se logró obtener una cifra que topara el máximo

porcentaje en la confidencia de los trabajadores con respecto a sus empleadores/socios de negocios. En este sentido, se obtuvo proporciones inferiores al 3% de contestaciones. Con exactitud, se examina un 1,3% para el ítem “Sumamente Alto”; y un 2,7 %, para el “Sumamente Bajo”.

Como resultado de estas generalidades, también se puede leer cuáles fueron los procesos operativos en los cuales más votaron durante la encuesta. Un ejemplo específico de esto se visualiza en la figura 4.10, la cual muestra la propensión de tener una confianza “ni alta ni baja” y el procedimiento en cuestión:

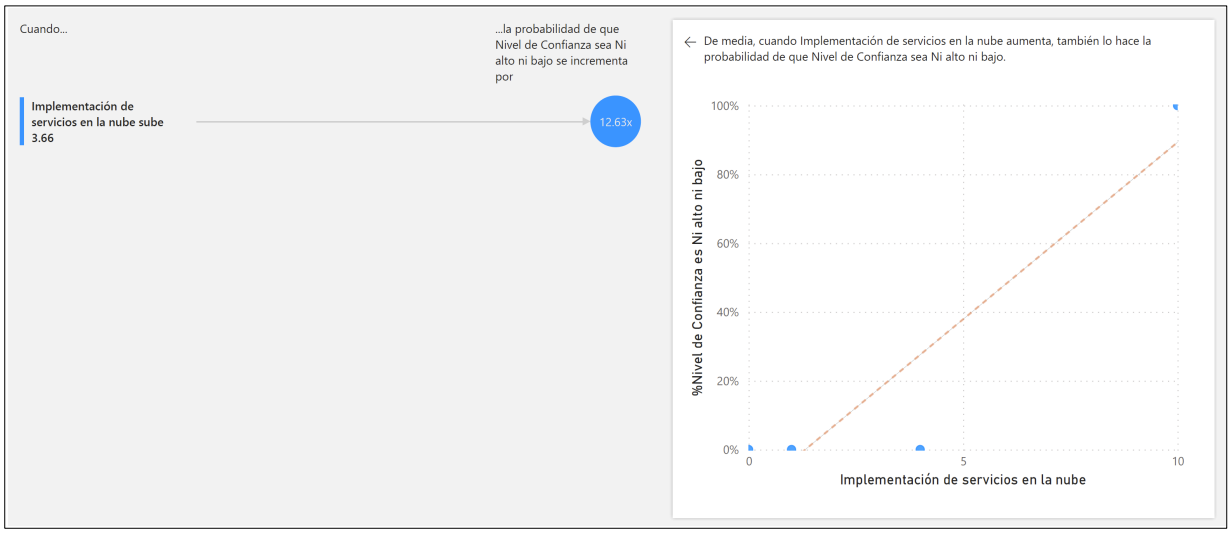


Figura 4.10. Desviación del nivel de confianza promedio. Fuente: investigación propia.

Debido a esto, de acuerdo con los datos anteriormente mostrados, se visualiza que el procedimiento de *implementación de servicios en la nube* obtuvo uno de los porcentajes mayores en cuanto a nivel de madurez superior o igual a “regular”, y curiosamente, en este apartado también es un sólido conductor de la variable de confiabilidad, la cual en su punto más alto (más del 80%), posee una cantidad de respuestas considerable para crear una tendencia. De la misma

manera, otros procesos cuyo comportamiento es similar son: gestión de proyectos, utilización de patrones de diseño y documentación de la gestión de los recursos y procesos.

En función de lo planteado, se infiere que el ambiente ágil y cambiante en el cual se mueven las micro y pequeñas empresas (PYME) de desarrollo de *software* impone un nivel moderado de incertidumbre en sus colaboradores en aspectos técnicos y administrativos de su cotidianidad. La flexibilidad con la cual se adaptan a nuevas tecnologías y la rapidez con la que pretenden competir en el mercado, influyen tanto en la madurez de sus procesos como en la seguridad de su personal para garantizar un cumplimiento óptimo de sus obligaciones.

No obstante, se entiende que la utilización de servicios de computación en la nube es un pilar fundamental para desarrollar productos nuevos, innovadores y robustos. Como se pudo visualizar en la figura 4.9, varios de los procesos operativos están inmersos en el funcionamiento dentro de un ambiente totalmente en la nube. Ejemplo de esto recae en ítems relacionados, como el aprovisionamiento de recursos, configuración de recursos, inventario de recursos, monitoreo sintético de aplicaciones, integración y entrega continua, entre otros servicios que dan los proveedores de computación en la nube, los cuales tienen una participación discreta (tendencia a la baja) en el gráfico de nivel de confianza dentro del funcionamiento de este tipo de entidades.

4.2.7 Frecuencia de mejora continua.

Se plantea, entonces, el considerar un aspecto tan importante como es la frecuencia con la cual se realizan cambios en procesos existentes. De este modo, se pretende mostrar que es un indicador importante y que está relacionado directamente con la confianza y madurez de las operaciones de desarrollo de *software*.

De esta forma, la mejora continua se basa en los principios de LEAN de toma de decisiones con base en datos y eliminación de desperdicio, que lleva a realizar pequeñas e incrementales mejoras de calidad. Esto trae beneficios drásticos y son difíciles de emular para los competidores. (Chen *et al*., 2007; Fowler, 1999; Jarvinen *et al*., 1999 y Krasner, 1992, citados en Fitzgerald y Stol, 2014, p. 4)

Ante todo, el estado de las organizaciones con respecto a este tema se evaluó debido a la constancia con la cual se ejecuta un autodiagnóstico sobre cómo están realizando las tareas. Además, se tomó en cuenta rubros sencillos en escala ordinal para colocar los valores desde una frecuencia baja o nula hasta una mayor; esto con el objetivo de examinar rápidamente, tanto la presencia de este acto como el peso e influencia que tiene dentro de los procesos operativos.

A continuación, en la figura 4.11 se presenta la visualización de cómo las micro y pequeñas empresas (PYME) de desarrollo de *software* se comportan con respecto a la mejora continua de sus procesos de producción de *software.*

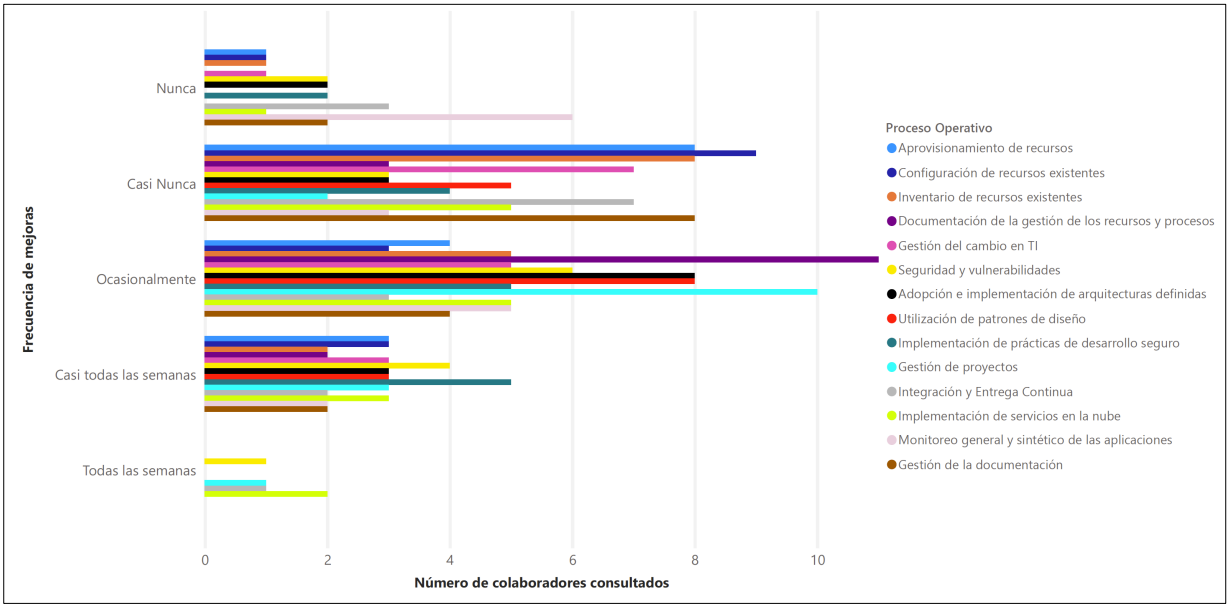


Figura 4.11. Representación gráfica de la frecuencia de mejora continua. Fuente: investigación propia.

En relación con estos datos, solo basta con dar un simple vistazo para concluir que la mejora continua no es un fuerte claro de este tipo de compañías. Incluso, se podrían debatir muchos posibles factores que podrían influir en la falta de mejora continua más constante; sin embargo, no es del interés de la investigación por el momento.

Ahora bien, prestando atención a los números concretamente, un manifiesto 36,6% de los participantes expresa implementar “ocasionalmente” algún tipo de perfeccionamiento a sus procesos operativos, entiéndase este rubro como una acción que se presenta irregularmente dentro de lapsos de tiempos extendidos. Por otro lado, como competidor directo está un nivel menor de frecuencia: “casi nunca”, el cual llega a un 33,5% de intervención. Este último tiene un porcentaje alto para un ítem que significa que las mejoras pasan a lo mucho una vez por año aproximadamente.

Recapitulando, si se suma la proporción de respuestas que indican que “nunca” ponen en marcha procesos de mejora continua, la cual es del 9,8%, junto con estos grupos anteriores, se

llega a un total del 79,9% de participantes que expresan tener una frecuencia de mejora continua deficiente. Por el contrario, el contraste es absoluto con aquellos que manifiestan una positividad en este ámbito, que actualmente llega al 20,1% de la cuota.

Por consiguiente, la realidad que envuelve a la micro y pequeña empresa (PYME) de desarrollo de *software* con respecto a la mejora continua es escaso. Con preocupación, se puede inferir que puede llegar a ser un factor negativo determinante al escalar su negocio a un mayor nivel. El crecimiento siempre lleva a cabo cambios necesarios a la operación, y uno de ellos es la excelencia en los procedimientos para obtener productos de alta calidad.

Ahora bien, considerando este escenario, también se reflexiona cómo algunos procesos operativos específicos tienen una influencia marcada en la frecuencia de mejora continua. Por ejemplo, en la figura 4.12 se observa la tendencia a bajar cuando disminuyen las respuestas del proceso de implementación de servicios en la nube:

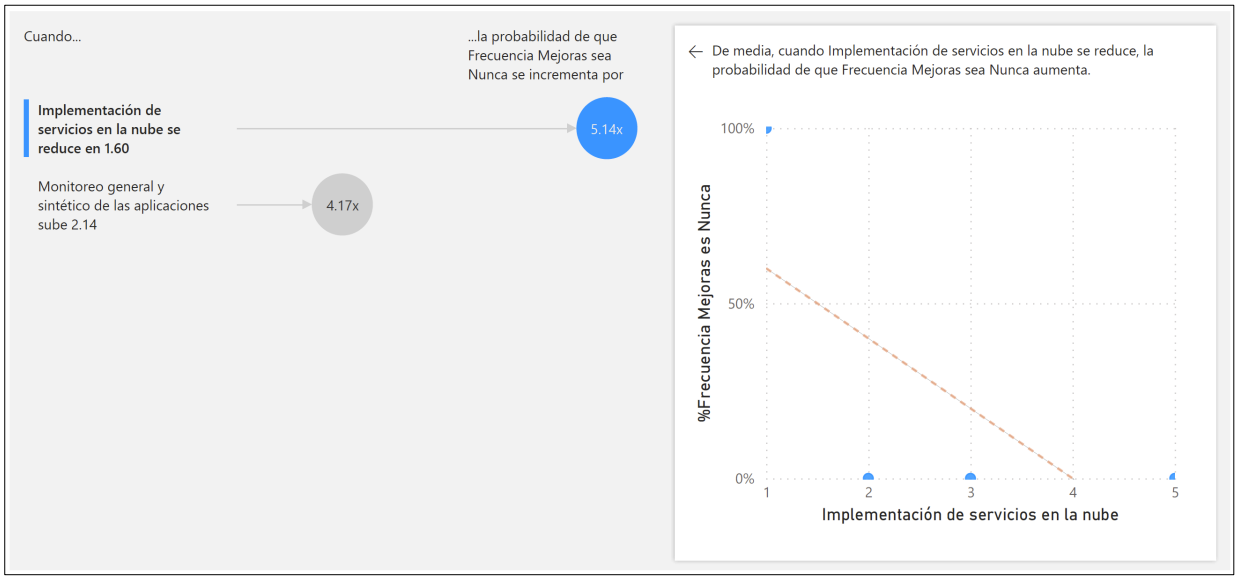


Figura 4.12. La probabilidad de que la frecuencia de mejoras sea nula. Fuente: investigación propia.

En resumen, lo que esto significa es que el proceso operativo de implementación de servicios de la nube es uno de los principales contribuyentes para el proceso de mejora continua. Desde luego, esta afirmación concuerda con la confianza y madurez que tiene este procedimiento por parte de la micro y pequeña empresa, y que juega un rol protagónico en el ciclo de desarrollo de productos de *software* que producen estas compañías.

Al comparar de forma similar este rubro con la frecuencia "ocasionalmente", se puede conseguir otro proceso operativo que influye bastante la presencia de esta respuesta. En la figura 4.13 se visualiza la posibilidad que existe de que este tipo de frecuencia de mejora continua esté presente en estas organizaciones:

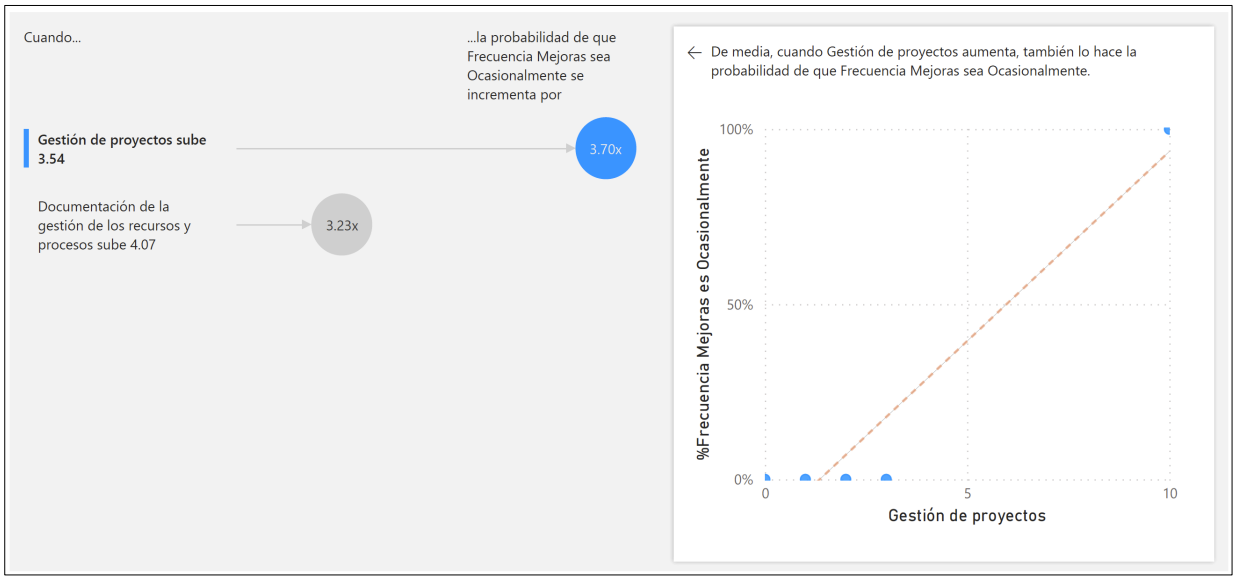


Figura 4.13. Influencia de la frecuencia de mejora continua para ser ocasionalmente. Fuente: investigación propia.

Primeramente, esto se traduce en una gestión de proyectos constante en su metodología; es decir, deja poco camino a la prueba de nuevos conceptos o mecanismos que eventualmente podrían tener un impacto positivo en la administración de estos. Aunque sea constante en la

forma en que ejecuta sus proyectos, no necesariamente expresa un crecimiento en la optimización del trabajo en esta área. Ocasionalmente, evaluar este procedimiento y decidir implementar un cambio para mejorarlo es un progreso muy lento.

A pesar de que la correcta gestión de proyectos conlleva una gran responsabilidad para entrega satisfactoria de productos de *software* a sus clientes, no solo para las micros y pequeñas, sino para compañías más grandes, representa un obstáculo enorme el no reinventarse para administrar proyectos de diversos tamaños. A su vez, se reitera otra afirmación mencionada anteriormente, que es el problema de detener el progreso de una organización micro y pequeña a niveles superiores.

En conjunto, todos estos indicadores impactan de forma negativa a las micro y pequeñas empresas de desarrollo de *software*. Tanto el nivel de confianza, como el de madurez de sus procesos de desarrollo no poseen una buena definición para sostener proyectos mucho más demandantes. Simultáneamente, se convierte en una oportunidad de progreso para todos estos tipos de empresa, especialmente en el mundo del desarrollo de *software*.

4.2.8 Razones para tener procesos indefinidos.

Como complemento a los datos ya recopilados, se estableció una pregunta un tanto abierta con la finalidad de obtener información general acerca de si las micro y pequeñas empresas tienen razones específicas para no poseer procesos definidos; es decir, se suministró una serie de opciones de respuesta que se consideró que los participantes podían seleccionar. Sin embargo, se dejó un espacio para que se explique o provea alguna otra respuesta que sea relevante al conocimiento de este rubro.

Por lo tanto, es evidente que el análisis de este ítem se pensó como tema informativo y no precisamente como indicador o variable que se esté midiendo para el estudio. Una razón que existe acerca de este tema es la subjetividad con la cual se puede responder esta pregunta y el enfoque de la investigación. Inclusive, se puede abrir una sola tesis acerca de las razones para no tener definidos procesos operativos dentro del flujo de trabajo de la organización.

Se explica en la figura 4.14 la representación proporcional de las opciones que los participantes seleccionaron durante la investigación. Adicional a esto, se despliegan también brevemente las respuestas que los colaboradores sugirieron que impactan sus organizaciones:

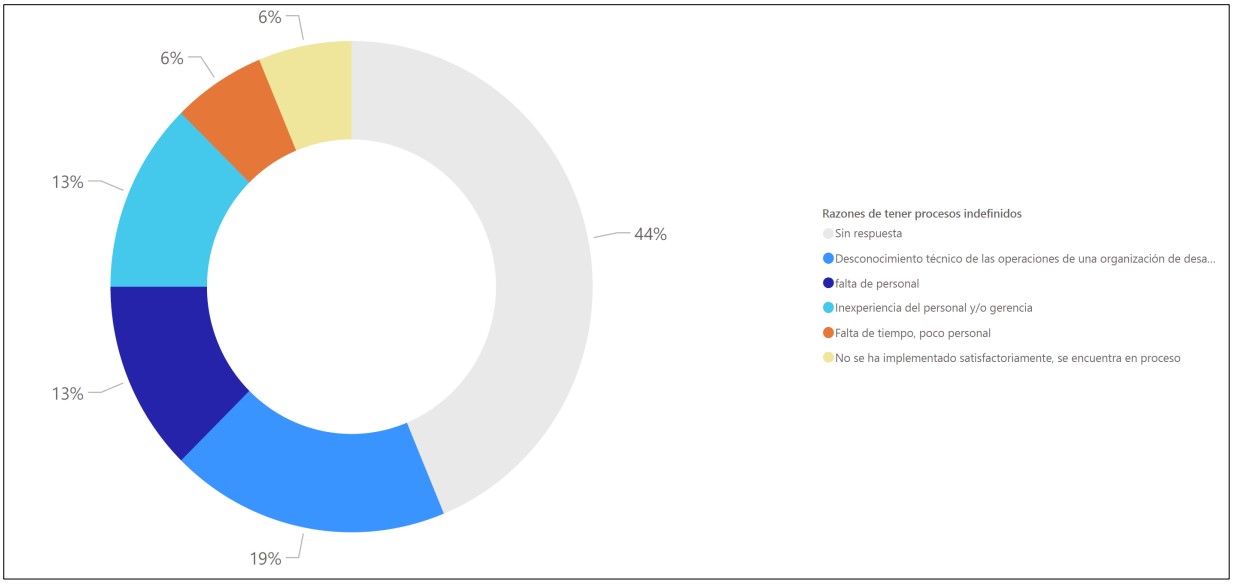


Figura 4.14. Gráfica del porcentaje de razones para no tener procesos definidos por colaborador de cada compañía. Fuente: investigación propia.

Visto de esta forma, es interesante resaltar que casi la mitad de los participantes decidieron no responder esta pregunta. Se adopta, entonces, una posición de neutralidad que concuerda con los indicadores anteriores (madurez, confianza y frecuencia mejora continua), ya

que existe un grupo grande de personas que opinan igual. Por otro lado, evidentemente, para el resto de los participantes acontecen otras razones más específicas para no tener procesos operativos definidos.

Dentro de este orden de ideas, es curioso saber que un 19% de los participantes abogan por la falta de personal y tiempo. Aunque no es sorpresa el deducir este tipo de información, especialmente por el tamaño de organización en estudio, es intrigante el reflexionar si esta situación más bien se expande a las otras compañías de las cuales no se obtuvo respuesta en esta pregunta. En cambio, alrededor del 32% dice tener problemas con la inexperiencia del personal y/o desconocimiento técnico de algunos procedimientos de la organización, es decir, existe un enfoque alto en la calidad de la mano de obra calificada que participa en este tipo de empresas, aunque sea un poco contradictorio con el nivel de educación promedio de la mayoría de los participantes, los cuales tienen estudios de educación superior.

Por ende, se puede conjeturar que el factor humano juega un papel importante en la capacidad de las micro y pequeñas empresas (PYME) de desarrollo de *software* de mantener procesos operativos definidos y maduros para el diseño, creación y mantenimiento de productos de *software* a nivel general, pero más específicamente del modelo de *Software* como Servicio.

4.2.9 Evaluación de los acuerdos de nivel de servicio.

4.2.9.1 Tiempo de respuesta en relación con la discontinuidad del servicio.

La disponibilidad del servicio brindado por los productos de *software* son un tema crítico. Actualmente, una gran cantidad de transacciones diarias confían plenamente en el funcionamiento correcto de los sistemas de información y de cómo estos interactúan con otras

plataformas para mantenerse en sincronía y proveer, con la mayor precisión posible, los datos a los usuarios que lo requieren.

En este sentido, la urgencia de mantener el servicio disponible al menos un 99,99999% del tiempo dice mucho de la reputación de los proveedores de productos de computación en la nube, siendo el *software* como servicio el enfoque de este estudio. Resulta claro que es un buen indicador para medir la eficiencia de este tipo de empresas para verificar si están preparadas para asumir retos mayores en este campo.

En la figura 4.15 se puede observar el tiempo de respuesta promedio dado por colaborador con respecto a los eventos de*: degradación del servicio e interrupción del servicio.* Para ello se tomó una escala simple ordinal que permitiera visualizar intervalos de tiempo de forma sencilla:

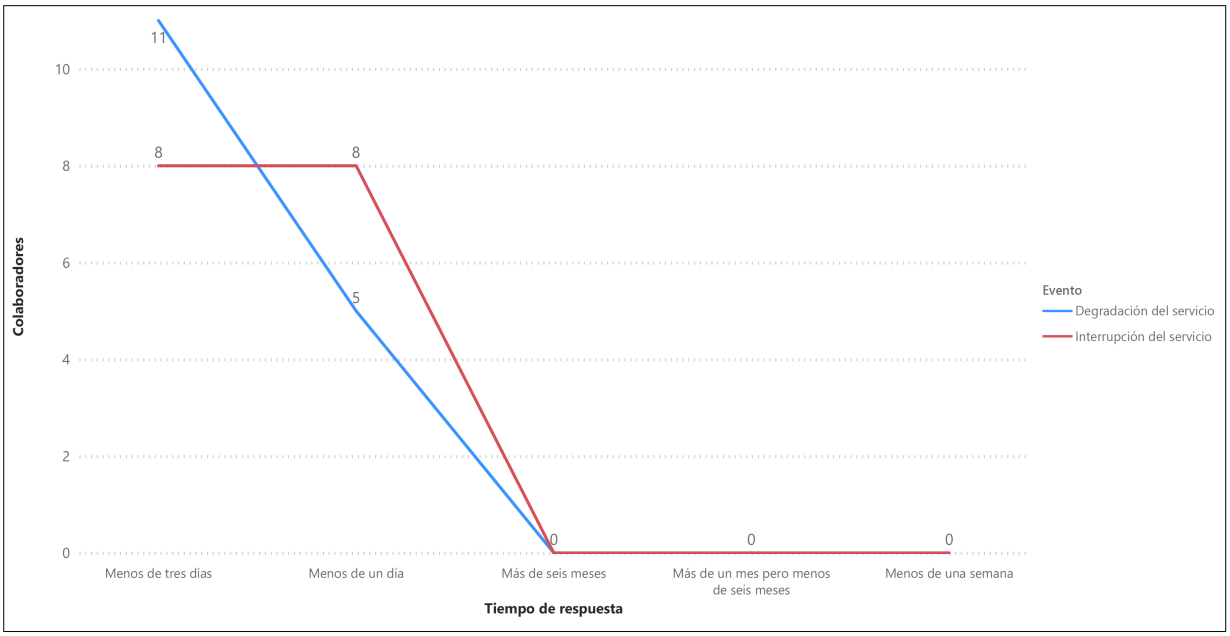


Figura 4.15. Tiempo de respuesta con respecto a la discontinuidad del servicio. Fuente. investigación propia.

Es conveniente acotar que el 100% de los participantes expresan tener un tiempo de respuesta de menos de tres días ante una interrupción total del servicio de sus productos de *software*. Aceptable o no, es un tope máximo que parece no ser la normalidad para todas las empresas encuestadas, ya que solo el 50% maneja un acuerdo de nivel de servicio de menos de un día, lo cual es mejor todavía. Ahora bien, tampoco se habla de tiempos específicos, pero al menos se observa una aproximación de los tiempos reales.

Asimismo, el promedio depende del tipo de servicio que se esté brindando; sin embargo, si se considera que son tiempos bastante extendidos que se tienen que reducir cuando se habla de productos de *software* como servicio, y mucho más en una era en la cual se vuelve una demanda esperada por los usuarios más que un aspecto que resalta.

En segundo lugar, cuando se visualiza el tiempo de respuesta para degradaciones de sus servicios, se observa una leve inclinación por un intervalo de tiempo mayor que el aspecto anterior. En este caso, el 68,75% indica un tiempo de respuesta de menos de tres días, lo que significa un incremento del 37,50% con respecto a los participantes que optaron por este ítem en la interrupción total del servicio. Al contrastar ambos eventos, se examina que la diferencia la puede dictaminar la naturaleza de cada uno. Por ejemplo, no es lo mismo que el servicio siga funcionando con los recursos mínimos a que se encuentre totalmente detenido. Aunque los números promedio tampoco reflejan un tiempo óptimo de respuesta para los clientes de estas empresas.

4.2.9.2 Tiempo de entrega en relación con la mejora del producto o servicio.

Ahora bien, otro aspecto a considerar en gran manera es la conducta que tienen las micro y pequeñas empresas para: corregir o modificar sus productos de *software* y/o implementar

mejoras a los procesos operativos de tecnología. Teniendo en cuenta sus limitaciones mostradas anteriormente, se trata de buscar un estado general que refleje la capacidad de estas empresas con respecto a este rubro.

En la figura 4.16 se representa gráficamente los tiempos de entrega evaluados por los colaboradores de las micro y pequeñas empresas (PYME) de desarrollo de *software* del distrito económico central de Alajuela.

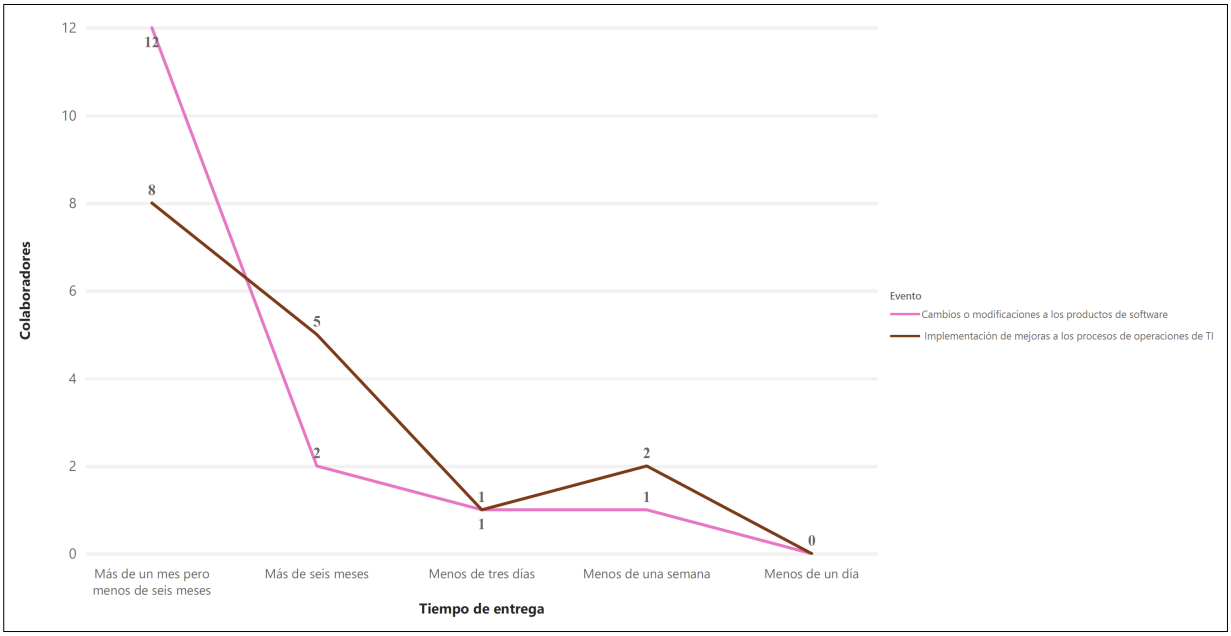


Figura 4.16. Gráfico del tiempo de entrega de mejoras a los procesos operativos y modificaciones a los productos de software. Fuente: investigación propia.

En primer lugar, al referirse al evento de cambios o modificaciones de los productos de *software*, evidentemente existe un grupo que representa el 75% de los colaboradores, los cuales decidieron evaluar un tiempo de más de un mes, pero menos de seis meses; contra un 12,50% que considera más de seis meses un tiempo aceptable. Por otro lado, un 6,25% considera que este tipo de cambios se pueden realizar en menos de una semana, y otro 6,25% dice tener un tiempo

de entrega de menos de un día; claro, tomando en cuenta aspectos generales que conlleva cambios de código y mejoras a los programas informáticos que posean, pues se evidencia un poco más real la respuesta de la mayoría y menos realista la de la minoría. Esto es porque cuando se sopesa y trae a colación otros factores, pues hay ciertos pasos de seguridad y cumplimiento que toman siempre tiempo e influyen en la entrega.

De manera similar, cuando se presta atención al evento de implementación de mejoras a los procesos de operación de TI, el 81,25% de la población coloca este acontecimiento entre un mes y seis meses o más de periodo de tiempo. Prácticamente, esto representa un incremento del 8,33% con respecto al evento anterior; aunque, de igual forma, se puede ver como la mayoría de los participantes se colocan dentro de este grupo. Está claro que periodos tan largos para validar mejoras no es el escenario ideal para un mercado tan activo como el del desarrollo de *software*.

En síntesis, los resultados obtenidos a través de este trabajo de investigación proveen una realidad esperada para las micro y pequeñas empresas (PYME) de desarrollo de *software*. Resulta razonable argumentar que este tipo de compañías poseen en sus filas muchos puntos a favor que les hacen acreedores de poder diseñar, crear y mantener productos de *software* como servicio. En este sentido, la hipótesis inicial se valida con respecto al ámbito actual en que se desenvuelven estas organizaciones. Sin embargo, tienen retos y dificultades para escalar a niveles superiores, así también como la metodología que usan para fabricar *software* de esta gama. Por ende, es necesario mayor involucramiento y estandarización en sus procesos internos.

CAPÍTULO V. CONCLUSIONES Y RECOMENDACIONES

5.1 Conclusiones

El objetivo principal de esta sección es la recapitulación de lo estudiado durante la investigación. Ante todo, se pretende dar una visión amplia al lector de los aspectos más interesantes que se desarrollaron en este estudio. Asimismo, como el foco está en el modelo de *Software* como Servicio, este abarcará una gran parte de las afirmaciones aquí descritas.

En primer lugar, se reflexionó sobre la oportunidad que existe sobre el establecimiento de una guía que esté orientada a la micro y pequeña empresa (PYME) de desarrollo de *software* utilizando un entorno virtual de infraestructura en la nube; esto a partir de la estandarización de conocimientos, métodos y procesos que permitieran a estas compañías no solamente crear y diseñar *software* con mayor facilidad y agilidad, sino también para que puedan crecer de una forma constante y firme.

En segundo lugar, logró identificar los elementos, definiciones, pautas y prácticas recomendadas por el mercado del desarrollo del *software* dentro y fuera del ámbito nacional. Gracias a la exhaustiva cantidad de publicaciones de textos, libros, investigaciones, documentación técnica, entre otros, se encontró un hilo conductor que sirvió como base para esta investigación. Comenzando desde el estado del arte con conceptos como la computación en la nube, su influencia en las últimas tres décadas sobre el comercio del *software* actualmente, hasta aspectos más específicos sobre el *Software* como Servicio y su concepción, tanto en empresas establecidas como en otras en proceso de desarrollo y expansión.

Ahora bien, en este sentido, fue necesaria una indagación y aclaración sobre los diferentes modelos de computación de la nube que existen: infraestructura como servicio, software como servicio y plataforma como servicio. Esto con el fin de entender que el *software* a

crear y mantener por estas empresas, en la mayoría de los casos, tenía una combinación de los tres para optimizar al máximo el uso y facilitar el proceso del ciclo de vida de un producto de *software.*

Simultáneamente, como se mencionó al principio de este apartado, se analizaron todos los componentes y conceptos que forman parte del *Software* como Servicio, desde sus ventajas con respecto a otros modelos, así como la subscripción de sus clientes; ya que esto los libera completamente de tareas complejas de mantenimiento y soporte que normalmente requerirían de personal especializado, el aislamiento de los datos de cada uno a través del uso el mecanismo multi-instancia para que el acceso sea exclusivo y cada ambiente único para cada cliente usando el mismo *software*, la fácil actualización y mantenimiento del *software* por parte del proveedor para con miles o millones de clientes, entre otros. Por consiguiente, se logró identificar este candidato para ser el principal actor de los productos que desarrollen las micro y pequeñas empresas (PYME) de desarrollo de *software*.

De forma similar, se cubrió otros aspectos técnicos como la arquitectura adoptada por sistemas de información que operan bajo la modalidad de *software* como servicio; evidentemente, se vio un dominio del modelo de microservicios en este ámbito, ya que permite operar y escalar aplicaciones de forma ágil y rápida, algo que a las empresas les es beneficioso para su incrementar su capacidad de respuesta y soporte de sus propias soluciones. Además, se pudo explorar otra ventaja que tiene, que es el riesgo reducido que se produce al corregir o modificar el *software* para cualquier fin. Esto porque la afectación e impacto de cambios en servicios autónomos es significativamente menor en una arquitectura de microservicios que en una aplicación analítica, lo que convierte esta opción en la favorita para la construcción de *software* de esta índole.

Por otro lado, la seguridad en las aplicaciones se exploró con detenimiento debido al impacto y riesgo que se infringen cuando se implementan de manera equívoca. Por ejemplo, se reflexionó sobre la importancia de un modelo de seguridad compartido en la nube, la necesidad de tener una segregación de roles con permisos únicos siguiendo el principio de menos privilegio, entre otras prácticas y productos que los proveedores de computación en la nube ofrecen para mantener los ambientes seguros y certificados con respecto a las regulaciones internaciones y/o locales, dependiendo de cada caso.

Del mismo modo, se encontró necesario también indagar en aspectos de integración y entrega continua. Con esto se explicó que estas técnicas son cruciales para el ciclo de vida no solo del *software* como servicio, sino que para muchos otros tipos de *software*. La automatización de esta etapa es importante para incrementar la capacidad de entrega con calidad de las compañías. Herramientas como Jenkins, GitHub, GitLab, entre otras, también propietarios de cada proveedor de computación en la nube ofrecen una amplia gama de utilidades que facilitan el trabajo de revisión de código, ejecución de pruebas unitarias, de integración y promoción de código entre ambientes desarrollo, pruebas y producción. Por ende, se consideró que las micro y pequeñas empresas (PYME) de desarrollo de *software* deberían entender bien estos conceptos y aplicarlos para tener una entrega más ágil y robusta de sus productos.

Ahora bien, más adelante se hizo énfasis en la gestión de operaciones y muchos procesos que deben estar bien definidos para el éxito de la organización. Por ejemplo, la documentación es un elemento importante de discusión y se ve, en la computación en la nube, una oportunidad de mejora para empresas de menor tamaño de poseer amplios repositorios de conocimiento que permitan la fácil administración de sus propios procesos operativos.

Cabe resaltar que el siguiente pasó consistió en el pleno descubrimiento de las operaciones de tecnologías de la información del sector PYME del distrito económico central de Alajuela, en las que su actividad, o alguna de sus actividades comerciales, se relacionara con el desarrollo de *software*. Dentro de este ítem se logró consultar, mediante un cuestionario de encuesta, la capacidad en recursos y habilidades que poseen estas empresas para diseñar, construir, entregar y soportar *software* teniendo en cuenta varios aspectos operacionales con respecto a sus actividades diarias.

De esta forma, del análisis de los datos recopilados fue posible concluir que las micro y pequeñas empresas poseen múltiples puntos a favor que les hacen acreedores de poder diseñar, crear y mantener productos de *software* como servicio. En este sentido, las premisas iniciales se validaron con respecto al ámbito actual en que se desenvuelven estas organizaciones. No obstante, se descubrió que tienen retos y dificultades para escalar a niveles superiores, así también como la metodología que usan para fabricar *software* de esta gama.

En cuanto a los puntos a favor, se encontró que la madurez y confianza de los procesos operativos con mayor apoyo por parte de los colaboradores son la implementación de los servicios de computación en la nube, la gestión de proyectos, uso de arquitecturas definidas y prácticas de desarrollo seguro. Mientras que los menos apoyados, y en los cuales este tipo de empresas tienen más deficiencias, fueron la implementación completa de soluciones de integración y entrega continua, monitoreo sintético de aplicaciones, seguridad informática y gestión de la documentación.

Por su parte, se encontró que la mayoría de los colaboradores considera que la frecuencia de mejora continua con respecto a los procesos operativos de sus compañías no es suficiente para llegar al máximo rendimiento y, como se ha expresado anteriormente, esto supone una amenaza

significativa al crecimiento de las micro y pequeñas empresas (PYME) de desarrollo de *software*. A su vez, esto afecta o impacta también sus esfuerzos por tener acuerdos de nivel de servicios decentes, aspecto en el cual se pudo identificar una debilidad, ya que los tiempos que ofrecen con normalidad, actualmente, no son óptimamente sostenibles.

En conclusión, se identificaron puntos estratégicos que ayudaron a comparar la actualidad de las micro y pequeñas empresas con las recomendaciones de la industria. Por consiguiente, este material apoyó el diseño de la guía de implementación de productos de origen de *software* como servicio para incentivar la estandarización de métodos para traer beneficios operativos en los emprendimientos costarricenses del sector PYME tecnológico como proveedores de productos novedosos.

5.2 Recomendaciones

En relación con la problemática expuesta, se debe considerar, primero, la autoevaluación de las micro y pequeñas empresas (PYME) de desarrollo de *software*, con ello se invita a reflexionar sobre aspectos internos y externos que impacten directamente, en primer lugar, sus objetivos principales como organización y, consecuentemente, la forma como sirven para llevar a cabo esa misión. En la medida en que la gerencia, en conjunto con los colaboradores, intente identificar sus debilidades y fallas internas, será un primer paso para el crecimiento y mejor calidad de sus servicios y productos.

Se plantea, entonces, la atención del diseño y creación de planes de acción específicos que hagan énfasis en la mejora continua no solo de sus procesos, sino, también, de su capacidad de innovación; por ejemplo: la búsqueda de nuevas tecnologías, la tercerización de servicios para

disminuir costos, etc. Al enfocar los esfuerzos en este punto se tendrá mejor madurez y confianza en sus procesos existentes.

Ahora bien, en cuanto a la implementación de *software* como servicio, se pretende amoldar los procedimientos existentes, en especial el de implementación de servicios de computación en la nube, para aprovechar y explotar sus capacidades al máximo. Esto significa que el foco de atención deberá tratar de utilizar la mayoría de los servicios de la nube para construir sus productos de *software*, es decir, aplicaciones nativas en la nube. Esto no solamente ayudará con el desempeño de las transacciones, sino que incrementará la oportunidad de reducir costos de operaciones, como el mantenimiento de la infraestructura que soporta la aplicación.

En función de lo planteado, la búsqueda de la normalización de muchos procedimientos que se posea como parte de su flujo de trabajo o que se desee incorporar, se deben explorar para incentivar la creación un ambiente ágil mediante sus versiones versátiles, especialmente para este grupo de compañías más pequeñas y que se puedan beneficiar ellos y, además, muchos emprendedores que estén iniciando en el mundo del desarrollo de *software*.

CAPÍTULO VI. PROPUESTA

6.1 Introducción

En esta sección se plantea una serie de recomendaciones, cuyo objetivo es la orientación de las micro y pequeñas empresas (PYME) de desarrollo de *software* en la creación, implementación y operación de aplicaciones bajo el modelo de *software* como servicio. Por ende, se deja ver una estructura muy específica que abarca cuatro áreas en las cuales este tipo de organizaciones debe concentrar sus esfuerzos de trabajo para, de manera ágil y efectiva, implementar satisfactoriamente soluciones de esta índole en sus negocios.

Naturalmente, en primer lugar, se abarcará temas relacionados con la administración general del recurso humano, lo cual considera aspectos de roles sugeridos siguiendo principios de seguridad que aseguren una buena gestión de los recursos de estas compañías, y también de cómo se plantea cubrir los proyectos que generen una aplicación de *software* como servicio; con ello se pretende reforzar, de manera generalizada, un tema que las empresas actuales tienen muy presente: la explotación de metodologías ágiles.

En segundo lugar, se suscita a atacar las fases técnicas desde tres de los pilares cubiertos anteriormente durante este estudio: la infraestructura, el ciclo de vida del desarrollo y las operaciones de tecnología. Lo anterior es fundamental para poner en práctica la mayor parte de conocimientos adquiridos a través de los conceptos teóricos y buenos hábitos que han sido ampliamente estudiados por otros profesionales en informática, así como la implementación de estos en organizaciones de mayor tamaño en el mundo de las tecnologías de información.

Con respecto a la infraestructura, se trata de comprender aspectos de diseño que tomen ventaja completa de los servicios dados por proveedores de la nube. En este aspecto, se aboga por construir una estructura que soporte la aplicación totalmente nativa de la computación en la

nube, es decir, que se utilicen al máximo recursos que ya existen para acelerar el desarrollo, además de dejar espacio para que las organizaciones se concentren en la lógica y reglas del negocio para crear valor agregado a sus clientes.

Por otro lado, en relación con la parte de creación y soporte del ciclo de desarrollo de las aplicaciones de *software* como servicio, se hace un especial énfasis en la arquitectura del *software*, así como la estrategia de integración y entrega continua. Es conveniente comentar que la importancia que tiene la arquitectura en la base de un buen *software* como servicio es crítica. La eventual implementación, por ejemplo: si es una estructura con una sola instancia o varias, si es multi-inquilino o no, si poseerá datos compartidos en una sola base de datos o no, o inclusive un modelo híbrido, etc., estos temas no son secundarios y de ellos depende el primer éxito de una aplicación robusta, confiable y escalable a cualquier ámbito. También, la forma como se contribuye con el repositorio central del código para realizar despliegues correctos y eficientes a producción es un tema poco enfático que se logró identificar durante la evaluación de estas organizaciones.

Finalmente, dado que la entrega en un modelo de *software* como servicio es tan demandante, es por ello por lo que se pretende realizar un cúmulo de pasos expeditos para proporcionar un buen soporte al *software* y cliente. La capacidad de cualquier proveedor de satisfacer las necesidades de sus clientes y de brindarles un servicio de primera es una responsabilidad grande para este negocio. También, la habilidad de respuesta rápida ante la demanda e innovación que solvente problemas antes que aparezcan, introduce un nivel complejo de competitividad en el mercado. Por ende, las operaciones postproducción son la carta de presentación ante la oportunidad de crecimiento del modelo o el declive total de la iniciativa de la organización.

6.2 Consideraciones Generales

Debe señalarse que una buena implementación y operación de un producto de *software* como servicio comienza con la eficiente administración de sus recursos humanos. Evidentemente, también se coloca la gestión en segundo nivel de importancia para el éxito de plataformas bajo este modelo. Esto quiere decir que tener un plan de acción y con tareas definidas, se puede influir grandemente en el desarrollo correcto de soluciones de alta calidad.

En este sentido, se comprende que un aspecto primario que predominará este tema será las metodologías ágiles dentro del trabajo de la micro y pequeña empresa. Sin duda, estas organizaciones necesitan una versión simplificada, versátil y eficaz para realizar sus labores; asimismo, se elimina mayoritariamente el exceso de burocracia en los procesos operativos de estas, dándoles la oportunidad de convertirse en entidades más productivas.

6.2.1 Gestión de roles definidos y sus responsabilidades.

Como una de las actividades iniciales, se debe tomar en cuenta la presencia y distribución de roles esenciales en la organización. Esto es fundamental para cubrir, de forma efectiva, las responsabilidades que existen dentro del ámbito de la administración de una aplicación de *software* como servicio. Además, otro beneficio que se obtiene es la facilidad de soporte a procesos internos y externos de auditoría de sistemas de tecnologías de la información.

6.2.1.1 Desde la perspectiva funcional.

De esta manera, a continuación se presentan los papeles que se consideran deberían estar presentes, como mínimo, para llevar a cabo los proyectos que generen algún tipo de *software* como servicio. Sin embargo, también se aclara que aquí se define roles basados en puestos de

trabajo calificados y no tomando en cuenta, por el momento, metodologías ágiles como Scrum. Este último, se detalla más adelante para combinar los roles de este marco de trabajo con los propuestos durante la primera fase de este apartado.

Analista Técnico del Negocio. Como mayor responsabilidad, la principal característica de este rol es la comunicación con el personal del negocio, es decir, todos aquellos usuarios que se convierten en clientes y solicitan características o funcionalidades para que estén presentes dentro del producto final para analizar y traducir todas esas preguntas en requerimientos definidos que sean entendibles para el equipo técnico.

De esta forma, se encuentra interesante una definición de un sinónimo llamado "ingeniero de requerimientos". De acuerdo con el *International Requirements Engineering Board* (la mesa internacional de ingeniería de requerimientos), el ingeniero de requerimientos es una persona que colabora con los clientes para obtener, revisar, documentar, validar y administrar requerimientos (Franch, Palomares y Gorschek, 2021, p. 70). A pesar de que poseen nombres diferentes, el objetivo principal es el mismo, por eso se está de acuerdo con que son roles idénticos, aunque, desafortunadamente, los límites varían según la compañía para la cual se desempeñe esta función.

Con la finalidad de proveer un conjunto de tareas recomendadas para este papel, a continuación se desglosan en una lista simple de responsabilidades que debería poseer el analista técnico de negocio:

a. Reunirse con los clientes o interesados que tengan solicitudes para nuevas funcionalidades o mejoras que tengan los productos de *software* que patrocinen.
b. Abstraer, validar y documentar solicitudes de requerimientos de clientes.

c. Analizar, filtrar y refinar los requerimientos en colaboración con el equipo técnico de desarrollo y/o el arquitecto de *software*.

d. Verificar el impacto y riesgo de implementación de cada requerimiento en conjunto con el equipo técnico de desarrollo y/o arquitecto de *software*.

e. Coordinar con el administrador de proyectos y el equipo de técnico de desarrollo, la implementación de los requerimientos.

f. Comunicar a los clientes toda información concerniente al progreso del trabajo de los requerimientos planteados en la etapa inicial.

Gerente de Administración de Proyectos. Encargado de liderar todas las iniciativas de gran escala en las que está envuelta la compañía. Se centra en proyectos de cualquier índole, pero en este sentido, estrictamente relacionados con tecnologías de la información.

A continuación, se desglosa una lista simple de responsabilidades que debería poseer el gerente de administración de proyectos:

a. Trazar un plan maestro y calendario definido con las tareas y fases necesarias para completar un proyecto.

b. Comunicar la ruta a tomar con los clientes, liderazgo y otros interesados en el proyecto.

c. Liderar iniciativas globales que involucran, en la mayoría de ocasiones, varios equipos técnicos y no técnicos.

d. Darle seguimiento a las tareas que forman parte de la línea de un proyecto y asegurarse de que las fechas de los entregables se cumplan.

e. Ser un facilitador con respecto a los demás equipos y personas con las que se colabora, siendo capaz de remover impedimentos o manejar negociaciones de forma consensuada.

Arquitecto de Software-Soluciones. Generalmente, es el rol de quien se encarga de todo el análisis y diseño que conlleva todo producto de software. La arquitectura en todos los aspectos del software es el corazón de este papel. Lidera la gestación del producto desde la estructura primaria, definiendo todos los aspectos técnicos que debe llevar la aplicación para que se pueda construir. Inclusive, muchos aspectos de diseño se expanden a otras áreas, como la integración de sistemas de información, lo que permite al arquitecto darse cuenta del flujo de información en un ambiente complejo en el cual posiblemente la aplicación vaya a estar interactuando con otros servicios y aplicaciones.

A continuación, se desglosa una lista simple de responsabilidades que debería poseer el arquitecto de *software*-soluciones:

a. Comprender a profundidad los casos de negocio que se busca resolver; asimismo, el impacto que generan en la organización desde el punto de vista técnico y de negocio.

b. Analiza el diseño de la aplicación a un nivel superior y define las pautas generales y específicas de las tecnologías a utilizar dentro de la solución.

c. Analiza el diseño del entorno en el cual se va a desenvolver la aplicación y define las pautas generales de integración y específicas del flujo de la información desde y hacia la aplicación.

d. Genera documentación y diagramas que ayudan a soportar el diseño estructural de la solución a implementar.

e. Colabora con otros arquitectos, ingenieros, analistas y/o jefaturas en las soluciones planteadas para implementación de mejoras y refinación del diseño inicial.

f. Supervisa la implementación del proyecto desde un punto de vista técnico en el cual se vela que se cumplan los acuerdos iniciales.

Desarrolladores de Software. Son el núcleo de la programación informática, los autores principales de escribir todo el código que forma parte de una solución de *software*. Todo el código que escriben o configuración realizada debe seguir estándares y líneas de diseño que se acordaron inicialmente en etapas tempranas de proyectos.

A continuación, se desglosa una lista simple de responsabilidades que debería poseer el desarrollador de *software*:

a. Realiza actividades generales de desarrollo de *software* como codificación, ejecución de pruebas unitarias, despliegue del código a ambientes de prueba y/o producción.

b. Analiza, implementa y defiende aspectos específicos de diseño ante otros compañeros, arquitecto de *software* del proyecto y clientes de aplicaciones con impactos varios: pequeña, mediana o a gran escala.

c. Participa activamente en la implementación de la solución de principio a fin en colaboración con los demás jugadores clave de los proyectos.

d. Dependiendo del nivel de experiencia y antigüedad, puede participar como mentor de otros desarrolladores menos experimentados.

e. Genera documentación técnica de todo el trabajo que se realiza, tanto a nivel de código como superior de la(s) aplicación(es) en las que participa.

Ingeniero de Calidad del Software. Este rol, al igual que un desarrollador de *software*, define pruebas, casos de prueba, lleva a cabo automatización de pruebas en cierto grado, etc. Se encarga de probar el *software* para que se entregue y logre funcionar con un margen de error mínimo.

A continuación, se desglosa una lista simple de responsabilidades que debería poseer el ingeniero de calidad del *software*:

a. Realiza actividades generales de desarrollo de *software* como codificación, ejecución de pruebas unitarias, despliegue del código a ambientes de prueba y/o producción.

b. Analiza requerimientos y genera un plan de pruebas adecuado para la validación y verificación de la aceptación de estos.

c. Utiliza herramientas y marcos de trabajo de automatización para las pruebas de *software* en varios ambientes.

d. Participa y colabora activamente en todas las etapas de implementación del proyecto desde el punto de vista de calidad del *software*, para asegurarse que ha sido probado en todas las fases.

e. Acciona y documenta los resultados obtenidos al ejecutar el plan de pruebas para posteriores mediciones de desempeño.

f. Se encarga de supervisar las actividades de desarrollo de *software* para asegurarse que las mejores prácticas se sigan y que la calidad del trabajo esté a un nivel profesional.

Ingeniero de Confiabilidad de Sitios. Un papel fundamental alrededor de las operaciones de tecnologías y sistemas de información. La responsabilidad primaria de este rol comprende el asegurarse de que las aplicaciones de software estén operando a su máxima capacidad de manera

confiable, sustentable y eficiente. Dentro de este ámbito se encuentra la administración de infraestructura, configuración, de integración y entrega continua, etc.

En términos generales, un equipo de ingeniería de confiabilidad de sitios es responsable de la "disponibilidad, latencia, desempeño, eficiencia, administración del cambio, monitoreo, respuesta a emergencia y capacidad de planeamiento de sus servicios" (Sloss, 2016, p. 22). Por ende, se entiende que la misión de este papel va de la mano con el compromiso de la organización de mantener un servicio de calidad para sus clientes; asimismo, libera de esta carga al equipo de desarrollo para que se enfoque en las funciones del sistema o aplicación de *software* como tal, en lugar de su operación.

En otras palabras, se puede inferir que un ingeniero de confiabilidad de sitios no es más que un desarrollador de *software* que usa sus habilidades de codificación para automatizar tareas de soporte y mantenimiento de sistemas e infraestructura en producción y otros ambientes de pruebas, aceptación de usuario y desarrollo.

Dado que muchas de las tareas que normalmente realizaría este papel ya fueron descritas anteriormente, a continuación se desglosa y recapitula una lista simple de responsabilidades, pero no limitada, que debería poseer el ingeniero de confiabilidad de sitios:

a. Realiza actividades generales de desarrollo de *software* como codificación, ejecución de pruebas unitarias, despliegue del código a ambientes de prueba y/o producción.

b. Utiliza herramientas y marcos de trabajo de automatización para mecanizar la configuración y mantenimiento de los sistemas e infraestructura durante sus operaciones en ambientes de producción y pruebas.

c. Encargado de configurar, supervisar, monitorear y auditar los servicios y flujos encargados de la integración y entrega continua para controlar los despliegues de código a producción.

d. Participa y colabora activamente en todas las etapas de implementación de un proyecto desde la perspectiva general de diseño de sistemas y posterior trabajo de monitoreo y soporte en operaciones.

e. Colabora generalmente en actividades de supervisión de eventos (*On-call*, como se conoce por su término en inglés).

f. Analiza y asiste en proyectos de mejora en aspectos de diseño de la infraestructura de los servicios de las aplicaciones para que no sobreutilicen recursos, ni tampoco los desperdicien.

g. Genera documentación técnica de todo el trabajo que se realiza, tanto a nivel de código como superior de la(s) aplicación(es) en las que participa.

Por último, es conveniente acotar que los roles acá descritos están presentes en la mayoría de las organizaciones, inclusive aquellas cuyo modelo de negocios principal no es la tecnología de información. Sin embargo, se considera que la existencia de roles con tareas definidas no es sinónimo de equidad de personal que está a cargo de asumirlos. Claramente, esto último se ha demostrado en capítulos anteriores durante este trabajo de investigación.

Por consiguiente, se propone esta lista inicial con responsabilidades principales. Por supuesto, cada organización deberá tener estos papeles dentro de sus arcas como recomendación para obtener un resultado altamente eficiente al operar dentro del marco de las tecnologías de la nube y mucho más al tratar de implementar una aplicación de tipo *software* como servicio.

6.2.1.2 Desde la perspectiva de metodología ágil.

Con respecto a este tema, es mandatorio no sacar las metodologías Agile del marco de funcionamiento de las micro y pequeñas empresas (PYME) de desarrollo de *software*, con la finalidad de convertirse en una guía liviana para este sector, el tener este concepto presente les permitirá operar de forma robusta y eficiente con los recursos limitados que poseen. Es por ello, que los roles mencionados tienen que asociarse de alguna forma con los que menciona en este caso la metodología Scrum, escogida como principal modelo a seguir.

Por su parte, la razón de la escogencia de esta metodología no es sorpresa; sin embargo, se amplía en el punto 6.2.2. Para efectos de esta parte, se limita a mencionar su relación con los papeles de responsabilidades antes mencionados.

De esta manera, en la tabla 6.1 se muestra la asociación correspondiente de los actores de Scrum con los roles técnicos. Esto quiere decir que cualquiera de estos roles puede cumplir la función dentro del marco Agile.

Tabla 6.1

Relación de los papeles existentes en Scrum con respecto de los técnicos desde un punto de vista funcional.

Dueño del Producto	**Facilitador del proceso Scrum**	**Equipo de Desarrollo**
Analista del Negocio Administrador de Proyectos	Arquitecto de *Software*-Soluciones Desarrollador de *Software* Analista del negocio Administrador de Proyectos	Arquitecto de *Software*-Soluciones Desarrollador de *Software* Ingeniero de Calidad del *Software* Ingeniero de Confiabilidad de Sitios

Nota. Fuente: elaboración propia.

En función de lo anterior, se plantea entonces que, bajo la premisa de los recursos limitados, una sola persona pueda asumir dos roles, como el de analista del negocio, administrador de proyectos y, así, actúe como el dueño del producto. Aunque se sabe que un dueño del producto por sí mismo no es un gerente de proyectos, el tener esta responsabilidad a cargo es mucho más sencillo para esta persona por el alto grado de influencia que ejerce sobre la pila de trabajo (historias de usuario). Además, los tres roles tienen responsabilidades que se comparten, esto implica más familiarización con la gestación de un producto, su acompañamiento en la creación, implementación y entrega.

En esta línea, desde la posición de Shastri, Hoda y Amor (2017), un hallazgo clave de su estudio fue la presencia del papel de un gerente en equipos Agile y se identificó que este, a su vez, tenía al menos cuatro funciones diferentes: mentor, coordinador, negociador y adaptador de procesos (p. 54). Por consiguiente, se concluye que es una tarea que no se puede dejar de lado, mucho menos en equipos pequeños que son los que conforman en su mayoría a estas micro y pequeñas empresas (PYME).

Sin embargo, se está un poco en desacuerdo en que la función de Scrum Máster (facilitador del proceso) esté presente como un papel único, es decir, que lo ejecute una persona exclusivamente. Esto porque la principal característica está en el propiciar las condiciones favorables para que el personal técnico realice su trabajo y cumpla con su objetivo final del periodo de carrera durante una iteración. Ahora bien, se considera que cada miembro del equipo es lo suficiente autónomo para convertirse en un facilitador para su equipo cuando sea posible. En este sentido, se comprende que la rotación de esta responsabilidad puede darse sin problemas y, a su vez, da paso a la conjetura de que el papel de facilitador no exista oficialmente y, más bien, cada miembro del equipo asuma este trabajo como parte de sus competencias.

Finalmente, el equipo de desarrollo debe estar conformado por los papeles técnicos, esto comprende desde los arquitectos de *software*-soluciones hasta los ingenieros de confiabilidad de sitios. El mantener a todo el equipo trabajando bajo este mismo modelo les permitirá estar mayormente involucrados con los progresos de cada proyecto. En este sentido, se sobreentiende que cada miembro es responsable de aportar valor con su parte para el correcto desarrollo, implementación y entrega de cada producto de *software* que tenga a cargo la organización.

6.2.2 Administración de proyectos y planes de acción.

Dentro del ámbito de trabajo de las micro y pequeñas empresas (PYME), se define entonces la implementación de una adaptación de la metodología Scrum para el manejo de proyectos. No se puede asegurar con certeza que es 100% Scrum debido a las sugerencias antes mencionadas con respecto a los papeles de los miembros del equipo. Como resultado, nace esta ligera variante que busca apoyar el proceso de gestión de un proyecto de inicio a fin, especialmente en un ambiente limitado como el que está en estudio.

En primer lugar, Scrum es uno de los métodos Agile más populares que pueden manejar la producción de sistemas de *software* complejos. Es un proceso de desarrollo de *software* creado por Jeff Sutherland y Ken Schwaber, que ha sido ampliamente adoptado (Schwaber y Sutherland, 2017, citados en Wang, 2020, p. 121). Por ende, debido a su uso extendido y comprobado por varias organizaciones alrededor del mundo, es que se considera como recomendación principal para la gestión de creación de productos de *software* como servicio.

Bajo este contexto es que se expone, a continuación, los eventos que deberían adoptar este tipo de empresas basándose en modelo de Scrum:

Primordialmente, el refinamiento del banco de historias de usuario es una actividad que no se puede dejar de lado; inclusive, debe tratarse con un grado alto de prioridad, ya que el tener los tiquetes ordenados en conjunto con una buena definición de estos comprende un avance significativo del trabajo. No hay peor ocasión que aquella en la cual algún requerimiento no esté completo, sea ambiguo o le falte información. Eso genera atrasos y esfuerzos extra para definir las historias hasta que lleguen a trabajarse en su totalidad.

Asimismo, el tener listas las historias de usuario provee al equipo de un alcance definido; es decir, se sabe con anticipación todos los elementos que tienen que formar parte del producto o función. Por consiguiente, será menos complejo estimar la duración completa del proyecto, inclusive el seguimiento se convierte en una tarea definida con etapas de implementación. Esto permite a la organización decidir si se compromete a cumplir con las fechas sugeridas o si se deben modificar.

Por ende, es tarea de la persona que se encarga de la gestión de los proyectos el crear y diseñar una ruta del producto desde el inicio. Esto no significa que será un documento enorme que contenga todas las consideraciones, implicaciones y acuerdos desde un inicio, si no que comience como un diagrama simple de las etapas más significativas que tenga el producto en su camino a la meta. Como resultado, esto ayudará a guiar todos los procesos que le siguen: el análisis, diseño, producción, implementación y entrega; inclusive, algunos más administrativos, como es el análisis de riesgos y financiero, como de inversión, gastos, rentabilidad y, por último, de factibilidad.

En esta perspectiva, este tipo de responsabilidad tiende a representarse como un papel aparte. En la mayoría de las compañías grandes y reconocidas de la industria de tecnologías de la información existe lo que se conoce como *Software Product Management* (Gestión del producto

de *software*). Como lo hace notar Ebert (2014), es la disciplina y el proceso de negocio que gobierna un producto desde su concepción hasta su entrega. El gerente de producto exitoso no solo domina los procesos del ciclo de vida, sino que se adueña de estos (p. 22). Sin embargo, en este caso, se recomienda al dueño del producto y gerente de proyectos realizar este papel, ya que es importante para el triunfo de la solución que se vaya a ofrecer a los clientes externos de la organización.

De esta manera, como actividad consecuente, se propone agendar un espacio de al menos una hora por cada iteración para refinar la lista de historias de usuario; aunque la recomendación es realizarlo dos veces por iteración, esto debido a las limitaciones de los recursos de la micro y pequeña empresa (PYME). Ahora bien, durante la sesión se debe incluir la mayoría de los miembros del equipo técnico (en este sentido, se procura cambiar el vocablo "desarrollo" por "técnico" para obtener una generalización de quienes pueden participar. Esto comprende roles que no necesariamente son programadores de *software*). Lo anterior, es para aclarar cualquier duda que surja sobre alguna historia de usuario, analizarla, discutirla, que se inicie una tarea de seguimiento sobre ello y se culmine con la conversación y/o negociación con el interesado que hizo la solicitud original, de ser estrictamente necesario.

Por otro lado, las reuniones de seguimiento diarias son otro ejemplo de prácticas del modelo de Scrum que tampoco se pueden eliminar, esto debido a la trazabilidad que estas reuniones permiten obtener para el trabajo de campo que se realiza durante una iteración. Durante estas sesiones se suelen capturar errores, dificultades u obstáculos que amenazan la entrega satisfactoria del entregable de una iteración. Así pues, si se tiene la posibilidad de detectar estas situaciones temprano, es posible que el impacto negativo disminuya a niveles muy bajos.

También, durante estas sesiones diarias se debe ser lo más productivo posible; es decir, es un lugar de alzar la mano y brindar detalles de lo que está pasando en torno a la(s) historia(s) de usuario que se están desarrollando. Si el progreso es bueno, entonces se proveen detalles de las acciones a tomar como siguientes pasos; por el contrario, cuando surgen problemas, no es un tiempo en el cual los colaboradores deban inundar y acaparar con problemas, sino que definan acciones tomadas y cuáles serán los movimientos que harán después para remover el obstáculo.

Se incluye, entonces, una forma de dirigir las sesiones diarias de seguimiento que se orienten en acciones. Esto brinda seguridad al equipo y a los colaboradores, también hace que las dependencias entre miembros sean muy pocas. Se puede señalar que darle la capacidad de resolver sus propios bloqueos a los colaboradores eventualmente añadirá madurez y autonomía al equipo. Por ejemplo, durante los 15 minutos, según la teoría, que debería durar la sesión, se pueden enfocar en entender el progreso de cada miembro; en esencia, lo que esto representa para la meta de la iteración. No obstante, cualquier impedimento se puede conversar después del tiempo oficial, o agendarse un espacio por separado para tratar temas más específicos.

Después, en relación con las revisiones de la iteración, se provee dos perspectivas desde las cuales conviene trabajar: primero, entregables que son orientados al cliente final; convierten en importantes solamente las historias o entregables que dependan de la aprobación de funcionalidades grandes y que se presenten en el producto final; y segundo, entregables que no son orientados al cliente final, es decir, aquellos que están presentes para mejorar el producto de *software* desde una perspectiva técnica. Entonces, se recomienda realizar revisiones de iteración; en otras palabras, demonstraciones, para que se acepte o rechace el trabajo realizado solo en aquellos casos en los cuales la funcionalidad impacte la forma de operar el producto del cliente

final. Esto reducirá el tiempo invertido en estas reuniones y, eventualmente, enfocará el tiempo del equipo técnico y el del cliente también.

Algo semejante ocurre con la ceremonia de retrospectiva. Si se toma en cuenta la realidad expuesta en el capítulo cuatro de este trabajo de investigación de las micro y pequeñas empresas (PYME), se pensaría que esta ceremonia puede ser crucial, ya que es un precedente para la mejora continua. Sin embargo, debe existir balance en el tiempo invertido en el trabajo de campo con respecto al administrativo. En función de esto, se propone realizar las retrospectivas basados en hitos u objetivos más grandes que una simple iteración. Esto puede ser variable, ya que la duración de cumplimiento de un objetivo específico o subetapas de proyectos no está sujeta a un determinado tiempo.

Si se apela a un ejemplo, se podría pensar en la necesidad de migrar un grupo de usuarios de una aplicación a otra. Entonces, de forma hipotética, se considera que esto es un objetivo general que está conformado de varias etapas u objetivos específicos. Basado en este escenario, se podría considerar una de las etapas el cargar datos de usuarios de un país o región geográfica primero y después los demás. Al realizar esta etapa primera, la cual puede estar conformada por una o dos iteraciones, se debe acudir a una retrospectiva para discutir situaciones de mejora con respecto al objetivo de la etapa. Resulta lógico que las acciones que se tomen en esa retrospectiva ayuden a ejecutar las etapas restantes de forma más rápida y eficaz.

En síntesis, la necesidad de implementar una versión propia de Scrum significa mucho en el contexto de la micro y pequeña empresa (PYME) de desarrollo de *software*. El poseer una forma clara de trabajo y administración de los proyectos que dará vida a sus productos de *software*, sin duda alguna se transforma en una implementación exitosa. Esto lo deben tener claro

todos los miembros partícipes de la organización que son responsables del logro de las aplicaciones de *software* que lleguen a desarrollar.

6.3 Guía de Implementación de una Aplicación de *Software* Como Servicio

En esta sección se abarca los pasos y trayectos necesarios para llevar a cabo el diseño, creación, implementación y entrega de un producto de *software* como servicio. Como ya se ha mencionado, el camino hacia la meta final de esta propuesta está compuesto por tres grandes fases: la infraestructura, diseño y arquitectura, y operaciones. Estas requieren una atención significativa por parte de las organizaciones para ser completadas con éxito. Son dependientes una de la otra, es por ello que se agregaron estratégicamente en forma secuencial.

Ahora bien, algunos de los métodos, opiniones, sugerencias o recomendaciones que se describen a continuación, se abordan de una forma muy generalizada. Esto debido a que desarrollar cada uno implicaría una cantidad significativa de esfuerzo y tiempo; por ende, el realizarlo así permite mantener la línea constante y enfocada de este trabajo de investigación con respecto a su objetivo principal.

6.3.1 Gestión de la infraestructura.

La correcta administración de la infraestructura que soporta los productos de *software* aporta considerablemente a la fiabilidad del servicio. Naturalmente, poseer un servicio confiable atrae más clientes y, a su vez, se traduce en una mayor aceptación del producto. El incremento de clientes trae consigo un sinnúmero de retos para cualquier aplicación de *software*; es por ello que muchos sistemas de información necesitan tener la habilidad de adaptarse a su demanda, ya que de eso depende la confianza que tengan sus usuarios. Por ejemplo, el poder escalar y expandirse

para servir a millones de solicitudes, así también como la capacidad de dejar de consumir recursos que no se necesitan cuando estas solicitudes bajan.

De esta forma, se propone cubrir puntos como la escogencia de un(os) proveedor(es) de servicios de computación en la nube, estrategias de administración de estos recursos de la nube y, sobre todo, orientar la infraestructura con el diseño del sistema en función de su objetivo, además del uso de servicios para integración y entrega continua, monitoreo y auditoría.

6.3.1.1 Selección de los proveedores de computación en la nube.

El objetivo principal de este punto es dar a conocer unas consideraciones generales que se deben tomar en cuenta cuando se escoge el proveedor de servicio de computación en la nube. Sucede, pues, que conocer las fortalezas y debilidades de cada proveedor le dará una perspectiva más amplia a la organización para saber si se alinea con sus objetivos generales y específicos del producto de *software* que vayan a soportar. Además, esto permite conocer las limitaciones técnicas de cada proveedor, lo cual se traduce en retos en la creación y operación de dicho producto.

Brevemente, se considera que se debe incluir al menos los servicios de dos proveedores de la nube. En primer lugar, tener un ambiente multinube permite obtener una seguridad mayor en el sentido de disponibilidad del servicio, es decir, si el primer proveedor es impactado por cualquier eventualidad, entonces se tendrá el servicio intacto del segundo proveedor; en otras palabras, actúa como una estrategia de *failover* (conmutación por error).

No obstante, algunos profesionales pensarán que esta medida puede ser contraproducente en el ámbito de una organización de tamaño micro o pequeña. La cantidad de proveedores a los que se pueda afiliar suele ser costoso si no se administra de una manera cautelosa. Generalmente,

no todos los proveedores tienen precios cómodos, ni tampoco es en todos los servicios. Unos pueden ser más caros al rentar instancias de servidores virtuales, mientras que otros lo son en servicios de administración de obtención de dominios, gestión de aprendizaje profundo e inteligencia artificial.

Con la finalidad de encontrar un balance, se aconseja tomar como punto de comparación los servicios de un proveedor sobre otro, de este modo se incentiva la optimización del uso de los mejores representantes de cada fabricante, eso sí, manteniendo un límite de uno o dos proveedores, inclusive, se podría agregar un tercero o cuarto. Todo depende de lo ordenado y efectivo que sea gestionar esos proveedores en cuestión del gasto en que incurra la compañía. En la figura 6.1 se muestra un ejemplo de cómo combinar servicios en un ambiente multinube:

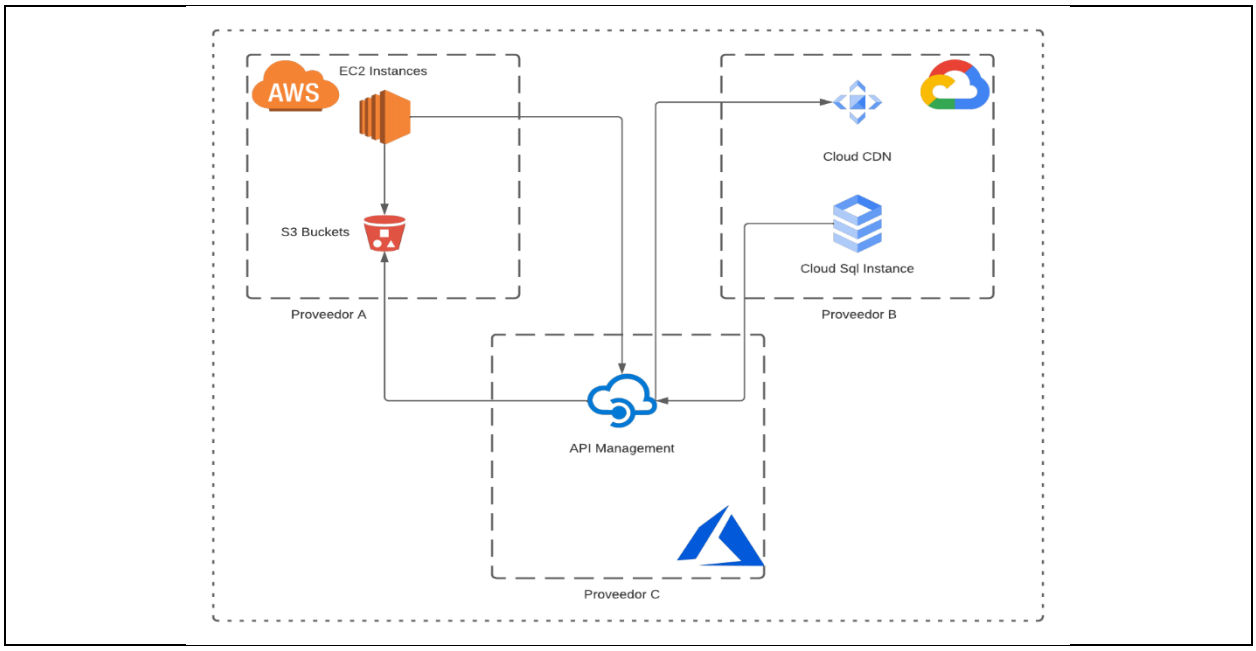


Figura 6.1. Ejemplificación de un ambiente multinube con varios servicios. Fuente: elaboración propia.

En función de lo planteado, se debe proseguir con el siguiente paso que es la elección del proveedor(es) de los servicios que se van a utilizar. Por ende, se necesita de algún proceso

sistemático que apoye la decisión. Se estima utilizar los métodos multicriterio de toma de decisiones (MCDM, por sus siglas en inglés), porque este proporciona una herramienta ágil, sencilla y eficaz para la selección del proveedor de computación en la nube para micro y pequeñas empresas (PYME) en estudio.

Por consiguiente, la idea general detrás de esta metodología es seleccionar características significativas que se requieran en la organización y así darles la importancia, o bien el peso, que dentro de la compañía se considera que vale cada una de ellas.

> Se puede decir que el uso de MCDM es una forma de lidiar con problemas complejos al separarlos en componentes más pequeños. Después de pesar algunas de las consideraciones y aplicar juicio sobre estos componentes, las piezas son reensambladas para presentar los términos generales a los que toman las decisiones. La mayoría de los métodos MCDM trabajan con alternativas discretas, que son descritas por un conjunto de criterios. Los valores de los criterios pueden ser determinados como información cardinal u ordinal. (Mardani, Jusoh, Nor, Khalifah, Zakwan y Valipour, 2015, p. 517)

De esta forma, en la tabla 6.2 se muestra una propuesta de activación de este método para la distinción de los servicios de los proveedores de computación en la nube. Comprende aspectos generales que pueden ser significativos para la micro y pequeña empresa; sin embargo, como cada empresa tiene necesidades y prioridades diferentes, dependerá de cada caso el adicionar más escenarios en la tabla de evaluación.

Tabla 6.2

Aplicación de método MCDM de ponderación aditiva simple

Criterio de evaluación	**Puntaje en Peso**	**Proveedor $A_1 \ldots A_n$**	**Total $A_1 \ldots A_n$**
Precio por servicio.	0.3	[a]3…5	0.9…1.5
Implementación de seguridad por medio de acceso por identidad.	0.4	3…7	1.2…2.8
Soporte para micro y pequeñas empresas.	0.1	6…7	0.6…0.7
Oferta de catálogo de servicios varios.	0.02	5…8	0.1…0.16
Administración de infraestructura por medio de código.	0.075	6…6	0.45…0.45
Compatibilidad con otras tecnologías.	0.025	7…9	0.175…0.225
Trabaja con modelos de BYOB (utilice su propia licencia).	0.025	4…4	0.1…0.1
Limitaciones en el servicio.	0.055	3…4	0.165…0.22
Total	**100%**	**n/a**	**3.69…6.155**

Nota: El puntaje dado por proveedor se multiplica por cada peso y se suma como un total. Este total luego se compara para validar la mejor opción posible. En este ejemplo, el proveedor A_n es la mejor opción con un puntaje de 6.155 sobre las demás opciones. Fuente: elaboración propia.

[a]La escala de medición va desde 1=Sumamente deficiente hasta 10 = Sumamente eficiente.

En este sentido, se tiene una herramienta versátil y fácil de implementar para los colaboradores de las micro y pequeñas empresas (PYME) de desarrollo de *software.* Al poseer la manera de justificar sus elecciones, se podrán solventar casos de negocio con los servicios de los proveedores que más se adecuen a las necesidades de cada problema. De esta forma, este paso de

selección de proveedores de la nube tiene un peso enorme en toda la solución que tome parte del producto de *software* como servicio.

Por otro lado, se puede conjeturar que una deficiencia que posee este método es la subjetividad; es decir, a pesar del uso de escalas y valores para evaluar una opción sobre la otra, realmente hasta cierto punto estos dependen del juicio propio de cada evaluador por empresa. Es por ello que se necesita comparar varias opciones conforme a criterios significativos y medibles. Entre más se trabaje en la definición de las variables de medición, mejores resultados se obtendrán al ejecutar el proceso de selección.

6.3.1.2 Orientar la infraestructura en función del objetivo del producto de software.

La clave de una infraestructura robusta es la relación que tiene con el problema de negocio que resuelve; es decir, debe adecuarse a las necesidades de la aplicación, servicios y demás que soporte. La sincronía y compatibilidad entre las capas de servicios y los recursos de la infraestructura debe complementarse para proveer del máximo rendimiento. Por ejemplo, si la aplicación manipula muchos mensajes, entonces una mejor forma de apoyar este proceso sería usar algún servicio gestionado para este propósito, como Amazon SNS, SQS o Google Pub/Sub. En otras palabras, no todo significa tener servidores virtuales y componentes de red configurados en la infraestructura que se elija, ni tampoco soluciones personalizadas que vendrían a realizar la misma tarea.

En función de lo planteado, primeramente, la selección de los servicios se concreta con respecto a los requerimientos que tenga la aplicación o capa de servicio a desplegar, por lo cual se debe llevar a cabo una revisión y refinamiento de los requerimientos para obtener un mejor criterio de decisión cuando se seleccionen los servicios de los proveedores de computación en la

nube. Levantar una lista con los requerimientos es una forma simple de realizar esta tarea, para posteriormente ingresar al siguiente paso, que se menciona a continuación.

En segundo lugar, debe "externarse" la responsabilidad de ejecutar algún proceso cuando sea necesario. La capacidad de realizar estas dos acciones les dará mayor fiabilidad a las operaciones de una aplicación usando un modelo de *software* como servicio. En particular, se puede mencionar que si la necesidad es grande en lectura y escritura de datos, entonces se subscribe a algún servicio gestionado como Amazon DynamoDB, Google Datastore o Cloud Spanner en lugar de desplegar una instancia virtual en la cual se instale el sistema gestor de base de datos. Otro ejemplo puede ser el uso de Funciones como Servicio, en la cual eventos ejecutan pequeñas secciones de código que realizan alguna acción, como llamar algún API, guardar información, llamar a otro servicio, etc. Servicios como AWS Lambda y Google Cloud Functions pueden resolver este problema en lugar de invertir tiempo y dinero en código que haga lo mismo dentro de la aplicación.

En tercer lugar, teniendo en cuenta estos aspectos anteriores, se propone cubrir los aspectos de la infraestructura en las siguientes fases:

6.3.1.2.1 Componentes de red.

En relación con la red, se debe tener claro el diseño que contendrá la red privada virtual en la cual la empresa colocará los recursos a los cuales se suscriba. Esta parte es crucial debido al nivel de seguridad y aislamiento que necesita asegurar para que el ambiente de infraestructura se comunique de forma efectiva sin abrir puertas a vulnerabilidades de acceso. Ningún recurso que no sea destinado a la parte frontal y pública de la aplicación debe permanecer con acceso abierto de tráfico desde Internet.

La mayoría de los proveedores de servicios en la nube de bajo nivel proporcionan habilidades de administración y mantenimiento de redes y subredes dentro de cada cuenta de subscriptor. Estas redes privadas virtuales se comportan como las redes comunes y corrientes que se suelen conocer como: LAN´s, y WAN´s. En este sentido, se debe configurar la comunicación entre esas redes y subredes que existan entre diferentes zonas y regiones. Además, es requerido que estén presentes algunas listas de acceso con reglas de enrutamiento de tráfico por IP, ya que agrega una capa de seguridad a la red.

De esta forma, es relevante que sin importar el número de proveedores de computación en la nube que se haya escogido previamente, es responsabilidad del equipo de arquitectura asegurarse de que existe un buen diseño de red en cada cuenta para la correcta colocación de recursos de computación, almacenamiento, bases de datos, entre otros.

6.3.1.2.2 Componentes de recursos de computación.

Los recursos de computación son los más comunes dentro del ámbito de computación en la nube. Simplemente se trata de servidores virtuales (VM´s, por sus siglas en inglés) en los cuales normalmente estarán alojadas las aplicaciones de *software* que se desplieguen al ambiente de la nube por parte de los consumidores.

Además, aquí también se suelen encontrar otros tipos de servicios que combinan los recursos de computación para generar valor y escalabilidad a los clientes. Por ejemplo, AWS Beanstalk o Google Cloud Kubernetes son utilizados para orquestar fácilmente miles de instancias virtuales para escalar fácilmente sus aplicaciones dependiendo de la carga de tráfico que llegue hacia la aplicación.

Como resultado, una buena práctica es hacer uso de estos servicios para manipular las instancias virtuales de forma dinámica. Si empieza la aplicación con diez nodos dentro de un contenedor, entonces que el servicio apoye en crear o remover nodos del ambiente según corresponda. Esto reduce significativamente la complejidad de la gestión de las máquinas virtuales, lo cual conlleva a un aumento en el tiempo del equipo para enfocarse en la lógica del producto de *software.*

Por otro lado, se considera apropiado revisar las imágenes creadas previamente para las instancias virtuales de servidores. Normalmente, algunas proveerán de *software* predefinido y agentes preinstalados que incrementen la rapidez de configuración de cada máquina. No obstante, si se piensa en crear una imagen desde cero, se recomienda guardarla y replicarla en todas las cuentas en las cuales se tenga pensado configurar el ambiente de la nube.

6.3.1.2.3 Componentes de almacenamiento de datos.

Habitualmente, el almacenamiento de datos no se puede dejar de lado. Cada organización define la importancia en la información que guarda. Asimismo, los tipos de datos son variados, al igual que la forma como están persistidos; desde bases de datos relacionales, hasta no relacionales, pasando por almacenamiento de archivos y objetos.

Con ello se quiere significar que la revisión del listado de requerimientos, en conjunto con el tipo de datos que se requieren archivar para la aplicación, estarían definidos en otro documento aparte, correspondiente a la gestión de los datos. De la misma forma, se puede señalar la clasificación de la información que se vaya a almacenar, ya que esto brindará a la organización visibilidad sobre qué áreas de la infraestructura y aplicación de *software* deben resguardar más con respecto a la seguridad de la información.

En tal sentido, existen servicios gestionados que se encargan de aliviar mucha de la carga que significa mantener sistemas gestores de bases de datos y servidores de archivos. Algunos ejemplos son: Amazon RDS, Amazon DynamoDB, Google Cloud SQL y Cloud Spanner, para guardar datos transaccionales, y Simple Storage Service y Cloud Storage, para archivos u objetos (información no transaccional).

6.3.1.2.4 Servicios gestionados.

Según lo expresado en párrafos anteriores, el foco puesto en los servicios gestionados por los proveedores de la nube necesita ser uno de los puntos críticos a revisar en la lista de las micro y pequeñas empresas (PYME) de desarrollo de *software*. Primero, la cantidad de trabajo en mantenimiento y auditoría se reducirá a revisar reportes que sean menos complejos de generar y normalmente proveídos por cada suministrador de computación en la nube. Segundo, el desempeño de la aplicación mejorará siempre y cuando se elijan los servicios y el plan adecuados para procesar, de forma eficiente, las acciones que se externen del código base de cada aplicación.

Naturalmente, los proveedores de la nube han innovado a un ritmo sumamente acelerado en la última década. La cantidad de servicios dentro de los catálogos que se ofertan ha crecido exponencialmente y esto abre puertas a otro modelo de operación que se llama Plataforma como Servicio. No obstante, no es la intención de este trabajo afirmar si estos otros servicios se clasifican o no dentro de este modelo de entrega de la nube; más bien, se centra en incentivar la revisión de estas ofertas para reducir la complejidad de algunas funcionalidades de la aplicación que se construya.

Siguiendo esta línea, se puede exponer otro caso, que es el servicio de autenticación e identidad de usuario. Ejemplos de este son: Amazon Cognito, Google Identity Platform o Microsoft Azure AD B2C, entre otros, los cuales aportan capacidades de control de identidad y autenticación de forma fácil y segura. Es por ello, que se puede tomar ventaja de un servicio de esta naturaleza, adecuarlo en el diseño del sistema y, así, quitar una carga de construir un proceso de identidad y autenticación desde cero. En la figura 6.2 se puede apreciar cómo uno de estos servicios se puede agregar al diseño de la arquitectura de un sistema.

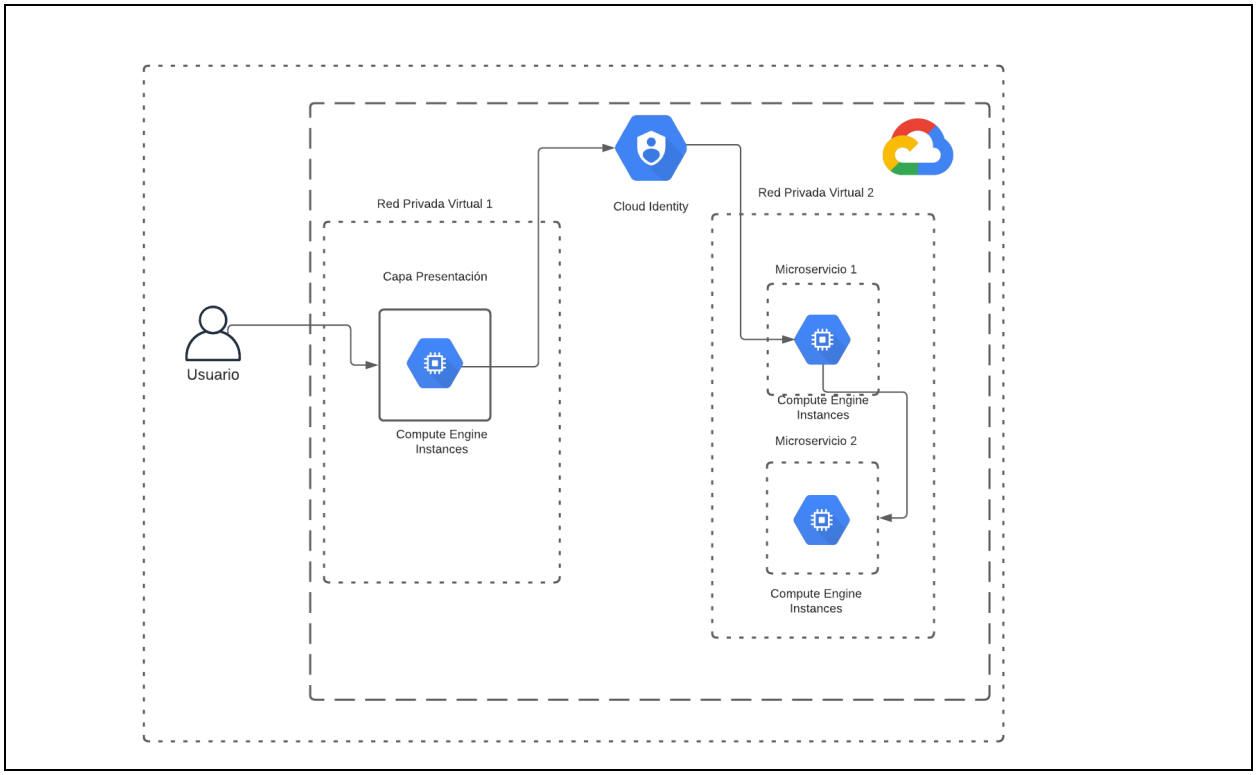


Figura 6.2. Incorporación de un servicio gestionado de identidad y autenticación a la arquitectura del sistema en diseño utilizando Google Cloud Platform. Fuente: elaboración propia.

En último término, es importante asegurar que el servicio al cual se le esté dando la responsabilidad del procesamiento tenga la capacidad de escalar al nivel que lo haga la aplicación, especialmente cuando se está gestionando una aplicación de *Software* como Servicio.

La lista de limitaciones y compatibilidad se precisa dentro del documento donde se especifica la utilización, escogencia y evaluación de los servicios seleccionados.

6.3.1.2.5 Seguridad y cumplimiento.

La seguridad de la información es un requerimiento irrevocable. Cada recurso y servicio que se contrate dentro del ámbito de la computación en la nube necesita una evaluación minuciosa en el aspecto de seguridad. La capacidad que tenga cada organización de garantizar una gestión segura de los datos e infraestructura de los sistemas de *software* es vital para la confianza de los clientes, en especial en el modelo de *software* como servicio.

Como producto de esta preocupación, se recomienda realizar una evaluación de la infraestructura a utilizar. Dentro de esta, se considera analizar aspectos como autenticación de los servicios con la aplicación de *software*, la forma como se guardan los datos en los métodos de almacenamiento (cifrado de datos, habilidad de enmascarar datos), el envío de datos de forma segura (encriptación de los canales de comunicación, por ejemplo: el uso de protocolos como TLS, SSL), la frecuencia de actualización de los servicios por parte del proveedor e, inclusive, aspectos de planes de recuperación ante desastres y pruebas de testeo de vulnerabilidades, por ejemplo, las pruebas de penetración que hayan tenido en los últimos tres, seis meses o un año.

Por otro lado, también es importante mencionar la definición de roles y niveles de acceso a lo interno, esto involucra responder preguntas como: ¿quién tiene permisos para modificar cierto recurso o servicio?, ¿quién tiene permiso sólo de lectura a ciertos servicios?, ¿cuántas cuentas de servicio se utilizarán para la ejecución de los servicios de la nube?, ¿qué grupos de seguridad o roles se necesitan para que los procesos se ejecuten de forma correcta?, etc. En la actualidad, los proveedores de la nube brindan herramientas de IAM o Identity Access

Management Control (herramientas de gestión y control de acceso e identidad), que ofrece roles, grupos de seguridad, dominios y demás para administrar los permisos de forma granular.

Ahora bien, es necesario enfatizar que la responsabilidad principal de llevar a cabo todo este análisis estará del lado del arquitecto(s) de *software*-soluciones, es el encargado de liderar la iniciativa de valorar las opciones, diseñar los servicios que formarán parte de la infraestructura de los sistemas y, más importante, de implementar la propuesta aceptada. No obstante, también se debe tomar en cuenta otros roles técnicos como desarrolladores y otros no tan técnicos, como los dueños del producto, para que se pueda determinar la mejor alternativa para la empresa para la cual colaboran.

En resumen, para el diseño de la infraestructura se debe generar documentación que indique claramente los servicios que formarán parte de la solución. Muchas empresas tienen diferentes sistemas para documentar; sin embargo, tomando como premisa que se posee un simple método de almacenamiento de archivos general, se recomienda, dentro del proyecto principal, crear una carpeta para el diseño y creación de la infraestructura, y dentro de él, generar un archivo que contenga el listado de requerimientos en conjunto con los servicios que vendrían a solventar cada necesidad; inclusive, aquellos cuya solución se está considerando adjuntar como parte de la funcionalidad del *software* a construir.

Igualmente, al lado de este archivo se deberán agregar los diagramas que posteriormente especifican qué servicios de los proveedores de la nube se tienen en juego. Dentro de estos, se debe considerar los siguientes: diagrama de red, diagrama de flujo de datos, diagrama de actividades, etc.; todo ello con la finalidad de comunicar el propósito general y específico de la infraestructura de la aplicación de *software*.

6.3.1.3 Estrategias de administración de los recursos en la nube.

Los procesos de gobernanza que existan dentro de la organización, especialmente para la micro y pequeña empresa (PYME), deben estar también documentados dentro del mismo espacio donde está la evidencia de selección de servicios junto con los requerimientos. Esto obedece a que la gestión de estos recursos necesita inversión de esfuerzo y tiempo por parte del personal de la organización; de otra forma, sería difícil escalar la aplicación horizontal y verticalmente.

Ahora bien, en primera instancia, es preferible que se tenga un ambiente de pruebas y uno de producción en cuanto a infraestructura se refiere. Esto puede cumplirse tan sencillamente como es tener dos cuentas por separado o una cuenta maestra con dos o más cuentas hijas asociadas a ella. Inclusive, puede darse el caso de que los recursos de un ambiente sobre otro estén en la misma cuenta, separados lógicamente por redes privadas virtuales independientes; es decir, la organización establecerá la forma más adecuada para trabajar en ello dependiendo del presupuesto que posean.

Dentro de este marco, variadas son las razones por las cuales es necesario poseer un ambiente de pruebas, pero una de ellas que se puede recalcar, es la habilidad de probar la adición de servicios gestionados, recursos de computación, almacenamiento o red de forma provisional para monitorear el rendimiento de posibles cambios a la infraestructura de la aplicación. Evidentemente, otra ventaja de esto es el costo menor de poseer varios ambientes de prueba, lo cual sería demasiado elevado si se estuviera pensando en infraestructura tradicional en sitio.

En segunda instancia, se debe establecer una estrategia y proceso definido para la implementación de infraestructura como código, ya que el mayor propósito de utilizar esta técnica es tratar la infraestructura como un problema de *software*; entonces, es mandatorio

establecer primero un proceso de SDLC para el correcto manejo de las fases de creación, mantenimiento y pruebas de toda la base de código. Dentro de los responsables primarios se encuentran los ingenieros de confiabilidad de sitios; sin embargo, se considera un esquema de rotación de recursos de desarrolladores para ayudar con el proceso de SDLC de la infraestructura.

De esta forma, tanto el proceso de desarrollo de *software* de la aplicación como la gestión de la infraestructura se tratan de la misma forma, administrados en proyectos ágiles cuyo propósito es la generación de valor al proceso rápido y de manera constante. Adicional a esto, se puede utilizar el mismo canal de integración y entrega continua para tener la capacidad ágil de responder ante los cambios demandantes del mundo comercial. Este canal necesita ser definido a nivel de la cuenta del proveedor o proveedores donde se posean recursos que se estén utilizando.

Por ejemplo, servicios como GitHub, Google Cloud Source Repositories, Cloud Build, Artifact Registry y Google Kubernetes Engine pueden aportar gran capacidad de automatización dentro de la infraestructura de Google. Por otro lado, si se habla de Amazon, existen servicios como AWS Code Commit, S3, Code Deploy, AWS EC2 y AWS Code Pipeline que perfectamente soportan flujos de cualquier índole, desde el más simple hasta el que tenga mayor complejidad. Ahora bien, si se aboga a otro ejemplo, Microsoft Azure posee el servicio conocido como DevOs Starter para el establecimiento de canales de integración y entrega continua, así como la configuración y soporte del proceso completo.

En otras palabras, la existencia de este proceso debe darse de la mano con las herramientas y servicios gestionados que proporcionan los proveedores de computación en la nube. Inclusive, muchos de ellos brindan pautas y patrones de diseño que las organizaciones pueden utilizar como base de sus propios canales de integración y entrega continua. El propósito

principal se cumple mientras exista una forma de controlar las versiones de código, las revisiones de código, los cambios en la base del código y los despliegues entre ambientes de pruebas y producción.

En relación con este punto, se aconseja hacer uso de proyectos de código abierto como Terraform de forma primaria para el manejo de los recursos de la nube, especialmente en un mundo con múltiples proveedores de la nube. No obstante, cuando sea posible también se recomienda el uso de servicios gestionados como AWS CloudFormation, Azure Resource Manager o Google Cloud Deployment Manager, por ejemplo. Antagónicamente, esta es la única ocasión en la cual este artículo hace una excepción a la regla impuesta anteriormente para traer como opción primaria un servicio que no sea autogestionado de algún proveedor de la nube.

Por otro lado, además de la infraestructura como código a nivel de recursos de la nube, también es necesario utilizar herramientas de configuración. En este caso, se recomienda utilizar servicios como Ansible, Puppet o Chef, aunque, de preferencia, Ansible para la gestión de la configuración de los recursos como instancias de servidores virtuales que requieran atención especial en los atributos y valores que posean. Otro punto a favor de esta herramienta es la simpleza que tiene al emplear YAML, lenguaje que es altamente conocido por muchos desarrolladores e ingenieros en la nube que están familiarizados con las herramientas más utilizadas para la gestión de servicios de la nube.

Se puede adicionar otro aspecto importante a tomar en cuenta: el inventario de recursos de la nube. Para ello es crucial que se cuente con una política simple pero directa que gobierne la administración de recursos, ya que esto le ahorra tiempo y dinero a la compañía. Eventualmente, si la estrategia no es apropiada, se puede llegar a incurrir en gastos operativos que sobrepasan la capacidad o presupuesto de la organización. El complementar esta política con acciones de

limitar los permisos que se le asignan a ciertos roles dentro de la organización, disminuye el riesgo de errores de cálculo y desperdicio de recursos.

Además, esta política debe ser clara en tres aspectos: aprovisionamiento de activos, eliminación de activos y modificaciones a activos existentes. Dentro de cada uno de estos puntos, se necesita indicar los responsables de aprobación de cada flujo. Por ejemplo, un diagrama de actividades ejemplificaría los actores y el desarrollo de cada transacción que se ejecute desde el iniciador hasta el sujeto que aprueba y termina la acción. El número de pasos dependerá del tamaño de cada organización, además de la complejidad de la solicitud. En el contexto de micro y pequeñas empresas, pueden ser versiones ligeras con cadenas de aprobación simples, es decir, varía de una a dos personas máximo.

En la tabla 6.3 se muestra un ejemplo de un flujo de aprovisionamiento de algún activo de computación en la nube (máquinas virtuales, alguna subred dentro de una red privada virtual, algún cubo de almacenamiento de objetos, etc.). Es simplemente una representación de varias acciones siguiendo una secuencia definida para lograr un objetivo final. La ventaja de realizar este mapeo es la claridad de las actividades con respecto a los responsables que deben llevar a cabo el flujo.

Tabla 6.3

Flujo de aprobación para la obtención de un activo de computación en la nube

Orden	Acción	Actor	Requerido	Precondiciones
a	Inicia solicitud	Solicitante	Sí	Solo pueden ser solicitantes roles como desarrollador o ingeniero de confiabilidad de sitios.
b	Aprobación	Aprobador de primera línea	Sí	Solo puede ser aprobador de primera línea el arquitecto de *software* o ingeniero de confiabilidad de sitios.
c	Aprobación y ejecución del aprovisionamiento	Administrador	Sí	Solo puede ser administrador roles como el ingeniero de confiabilidad de sitios

Nota. Fuente: elaboración propia.

Seguidamente, cada solicitud debe realizarse por medio de un tiquete, el cual contendrá la información básica del solicitante, una descripción y además, una sección de comentarios más un estado para que la parte administradora pueda actualizar el tiquete con la información de la aprobación o denegación del requerimiento. Ahora bien, el sistema de tiquetes depende mucho de la organización y de cómo estén operando; sin embargo, si se poseen sistemas como JIRA, ServiceNow, Remedy, AzureDevOps, etc., se facilitará la administración de tiquetes, así como procesos de auditoría a los cuales se someta la empresa después para la revisión de documentación y evidencia que soporte el trabajo realizado.

Finalmente, cabe destacar que se necesita todavía una forma eficiente de rastrear todos los recursos que se están consumiendo dentro de las cuentas pertenecientes a cada organización. En relación con la gestión, existen servicios autogestionados que ya la mayoría de los proveedores soportan, por lo cual sería la opción primaria para implementar. Servicios como Google Cloud Asset Inventory, AWS Systems Manager Inventory y Microsoft Defender for Cloud ofrecen características de administración de activos de forma nativa que se integran de forma excepcional con los proveedores para los cuales sirven, respectivamente.

Asimismo, como lo menciona Loggle (2021) en uno de sus *blogs* de tecnología, la gestión de los activos de la nube proporciona visibilidad y control de todos los activos en la infraestructura de la nube. Es un proceso mejor optimizado de administración seguro de la nube. (Loggle, 2021, párr. 2)

La mayoría de estos servicios utilizan los metadatos de cada activo y la información relacionada con ella para la categorización, rastreo y administración de los recursos de la cuenta de computación en la nube de cada empresa. También ofrecen roles de seguridad y archivos de registro para mayor control y visibilidad de los ambientes de la nube que se requieren gestionar. Por último, son servicios gratuitos, en su mayoría, pero se recomienda revisar cuidadosamente cada servicio por separado para evitar incurrir en cargos inesperados.

No obstante, si no se está satisfecho con los servicios autogestionados de cada proveedor, entonces se puede buscar otra solución de terceros. Un ejemplo es IBM Cloud Cost and Asset Management, el cual es un servicio de IBM que realiza la función de administrar el inventario y costo asociados con los servicios que contrate cada organización en un ambiente multinube. Otros ejemplos son Certero Cloud Asset Management, Cloud Sploit de Aqua y Better Cloud. Claramente, unos son más costosos que otros; sin embargo, solamente son opciones.

Evidentemente, no hace falta un *software* sofisticado para llevar el inventario de activos, hasta en una hoja de Excel o un espacio de documentos compartidos se puede hacer. La recomendación no cambia, es invertir esfuerzo y dedicación en una buena gestión de los recursos que se consumen de los proveedores de computación en la nube.

6.3.2 Gestión del diseño y arquitectura del *software*.

La etapa de concepción de la arquitectura del *software* comprende un proceso que toma su tiempo y esfuerzo. Resulta lógico pensar que se deben tener definidos los pasos a realizar con respecto a la escogencia de una arquitectura que satisfaga de forma robusta las necesidades del problema que se esté resolviendo.

Se permite, entonces, considerar la implementación de una arquitectura de microservicios para aplicaciones de tipo *Software* como Servicio. La sugerencia viene de la segregación de responsabilidades por microservicio que forme parte del sistema como un todo. Algo semejante ocurre con los servicios gestionados a los cuales la organización se subscriba de cada proveedor de computación en la nube.

Como se menciona en el punto 6.3.1.2, el orientar la infraestructura en función de la problemática que se resuelve también influencia la forma como los servicios formarán parte de la aplicación. Naturalmente, la cantidad de microservicios dependerá de las entidades y procesos que el sistema vaya a tener, excepto por aquellos que son constantes en muchas plataformas, como es el caso de la autenticación de usuarios, el servicio de cobros, de pagos, registro de datos auditables, servicio de inventario, por nombrar algunos. En todo caso, con esto se recomienda realizar otro documento tipo lista en el cual se enumere todos los servicios y su propósito que

formarán parte del flujo del sistema. Esto será clave para la generación de los diagramas de actividades, UML, de arquitectura, que deberán estar adjuntos en la misma documentación.

Como segunda gran etapa, está el elegir el tipo de estructura de inquilinos que se desea tener. Como es bien sabido, la forma como se administren los datos de los clientes que se registren en el sistema es crucial para un producto de *Software* como Servicio. Para ello existen diferentes estrategias, como: un solo inquilino o varios inquilinos, cada uno con sus ventajas y desventajas.

No obstante, en este apartado se inclina por la estructura de inquilinos múltiples. En el campo de la micro y pequeña empresa se necesita incurrir en el mínimo de gastos administrativos posibles, es decir, tanto de esfuerzo y recursos de investigación, humanos y monetarios. En este aspecto, una arquitectura de inquilino individual no suele ser tan ágil para la rápida adopción de este tipo de aplicaciones en el mercado.

De esta forma, en el ámbito de la arquitectura de inquilinos múltiples se necesita pensar en formas eficientes de gestionar los datos de los clientes, para no mezclarlos y evitar latencia a la hora de responder a las acciones y demandas de estos. Se entiende, entonces, que estos dos retos consumen una gran parte del diseño y las estrategias de operación que se piensen por el equipo de trabajo en la creación de la aplicación de *Software* como Servicio.

Dado que los microservicios suelen utilizar mucha banda ancha de conexiones en la red debido a su comunicación por servicios web, entonces nacen un par de sugerencias que suelen ayudar al proceso de integración entre servicios y lectura de datos de una manera eficaz: los mensajes y también los sistemas de memoria caché.

En relación con la comunicación por medio de la red, se determina que la opción de usar servicios que hagan utilización de mensajes asíncronos es la solución que trae mayores beneficios a una estructura de servicios que ocupan hablar entre sí y sincronizar el procesamiento de datos. La comunicación asíncrona, como su nombre lo indica, radica en el desacoplamiento de los actores, eliminando la dependencia de esperar por una respuesta haciendo uso de suscriptores y emisores de mensajes que escuchan solamente aquellos que les interesa y que están disponibles en la cola de avisos. En este sentido, se está de acuerdo con explotar servicios autogestionados de la nube que, de forma conveniente, administran y procesan estos mensajes para que sean consumidos correctamente.

Desde la posición de IBM Cloud Education (2020), se define la mensajería asíncrona de la siguiente manera:

> Se refiere al tipo de comunicación interaplicación que los agentes de mensajes hacen posible. Previene la pérdida de datos importantes y permite a los sistemas continuar funcionando aun cuando enfrenten problemas de intermitencia y latencia, que son comunes en las redes de computadores. La mensajería asíncrona garantizará que los mensajes serán entregados solamente una vez (y solamente una vez) en el orden correcto relativo a otros mensajes. (¿"*What is a message broker*? [*¿Qué es un agente de mensajes?*]", párr. 6)

Asimismo, cada microservicio que necesite enviar datos a otro podrá hacerlo a través de la ejecución de un evento que posteará un mensaje a una cola que, eventualmente, avisará a los servicios que sean receptores de los datos que han sido enviados para su consumo posterior.

Por ende, servicios propietarios, como el de Amazon SQS, SNS, Google Pub/Sub, Microsoft Azure Bus, IBM MQ, Oracle Messaging Cloud Service, así también como servicios de

proyecto de código abierto como Apache Kafka (https://kafka.apache.org/), Eclipse Mosquitto MQTT (https://mosquitto.org/) o Rabbit MQ Service (https://www.rabbitmq.com/), se convierten en herramientas a tomar en cuenta cuando se trata de manipular la comunicación a través de mensajes que tendrá la aplicación de *Software* como Servicio. La diferencia radica en que los servicios autogestionados no ocupan mayor esfuerzo de configuración o instalación, mientras que los proyectos de código abierto requieren ser instalados en máquinas virtuales para ser servidores dedicados de estos servicios.

En la figura 6.3, con base en las sugerencias anteriores, se ejemplifica el modelo base de arquitectura propuesto para la implementación de una aplicación de microservicios. Claramente, cada componente es intercambiable por un servicio autogestionado o por uno personalizado.

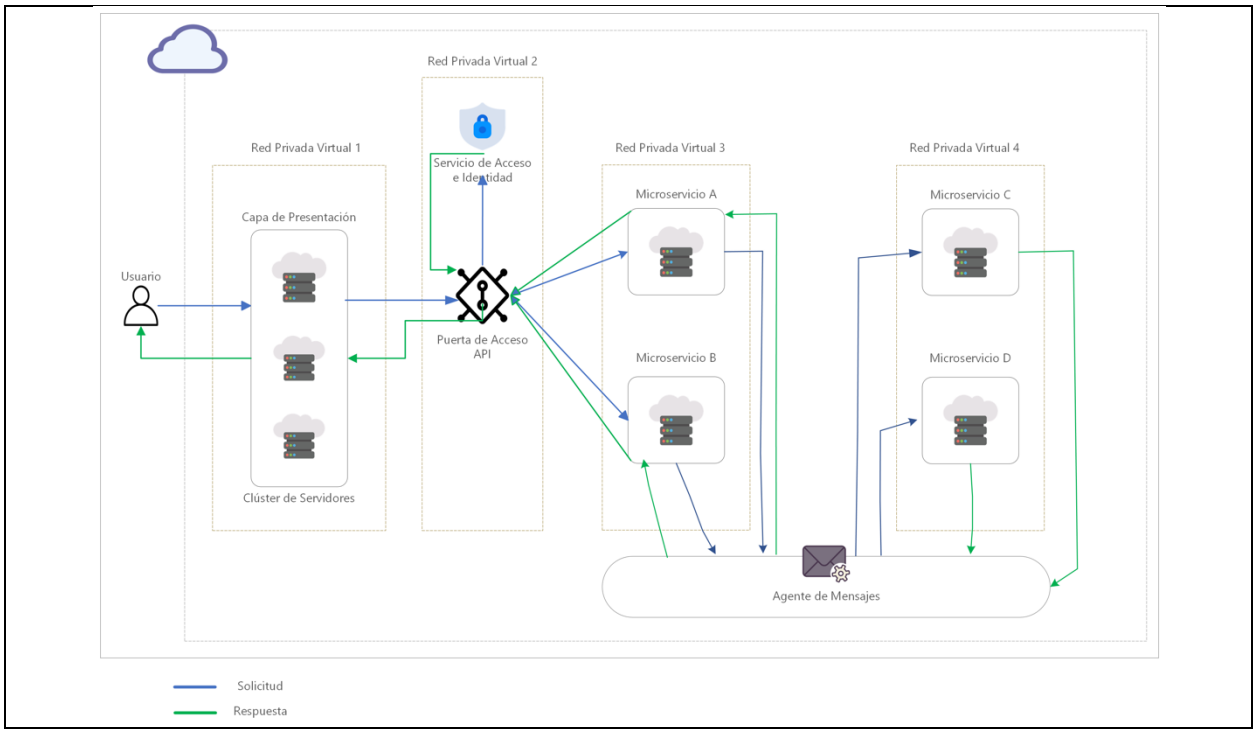


Figura 6.3. Modelo base de arquitectura de microservicios utilizando un agente de mensajería para la comunicación. Fuente: elaboración propia.

En relación con la figura 6.3, también se denota que no toda la comunicación pasa por el servicio de colas de mensajes, esto se debe a que, naturalmente, no todo tipo de interacción en el sistema requiere ser administrado de forma asíncrona. Resulta lógico, entonces, pensar que los arquitectos de *software* deberán considerar cuándo utilizar comunicación asíncrona o síncrona. No hay una respuesta definitiva, por ejemplo, la comunicación interna entre servicios mayoritariamente es asíncrona; depende mucho de la tarea que se está resolviendo, cuánto tiempo dure ejecutándose y si es tolerable el esperar por una respuesta.

Lo anterior lo afirma Anil *et al*. (2021), en un artículo publicado en la documentación de Microsoft sobre *Arquitectura de Microservicios en .NET*, otra regla que debería seguirse es el uso exclusivo de mensajes asíncronos entre servicios internos, y comunicación síncrona (como HTTP) desde la aplicación cliente hacia los servicios de presentación (puertas de enlace de API´s más el primer nivel de microservicios). (párr. 5)

El punto clave al decidir qué tipo de comunicación se debe establecer y los puntos donde se establecerán es la experiencia de usuario. El impacto que puede generar unos segundos de más a la espera de una respuesta por parte de una aplicación son devastadores en términos de calidad del servicio; por ende, el realizar este trabajo antes de comenzar con la creación de la estructura del sistema es crucial para no tener que cambiar y reparar defectos que salgan en el futuro.

En conjunto, las consideraciones anteriores generan un documento en el cual se establece la relación inicial de servicios con su tipo de comunicación. Esto puede ser una tabla sencilla que demuestre la asociación entre componentes; a su vez, esto propiciará una ayuda enorme cuando se esté creando o refinando el diagrama de red de la aplicación. Especialmente, se necesita tener una acotación simple en la cual demuestre el tipo de comunicación e interacción requerido entre ambos nodos (llámese nodos a cualquier componente: servidor, clúster, microservicio, etc.).

Ahora bien, en relación con la capa de datos, como ya es bastante estándar, se recomienda poseer una estrategia de almacenamiento en caché que sea de alto desempeño y escalable. Esto es indispensable, ya que en una arquitectura de inquilinos múltiple, varios clientes estarán accediendo a los datos de manera simultánea. Este reto es bastante conocido por los desarrolladores y, de hecho, se podría argumentar que una gran cantidad de sistemas cumplen con este servicio de almacenamiento de memoria en caché debido a su alta eficacia para evitar leer directamente desde las bases de datos una y otra vez.

Por consiguiente, dentro del caso de uso de las micro y pequeñas empresas (PYME) de desarrollo de *software*, se encuentra la recomendación principal de servicios de código abierto, como son: Redis y Memcached. En síntesis, estos servicios se pueden instalar de forma independiente en instancias virtuales de cualquier proveedor de la nube; sin embargo, también existen otros servicios autogestionados que presentan una forma más rápida y sencilla de implementar estas funcionalidades; algunos ejemplos son: Amazon ElastiCache, Amazon MemoryDB for Redis, Microsoft Azure Cache for Redis, Google Cloud Memorystore y IBM Databases for Redis.

En la figura 6.4 se puede apreciar la arquitectura de la aplicación que se mostró en la figura 6.3 con el ingrediente adicional de almacenamiento (incluyendo ambos tipos, persistente y volátil).

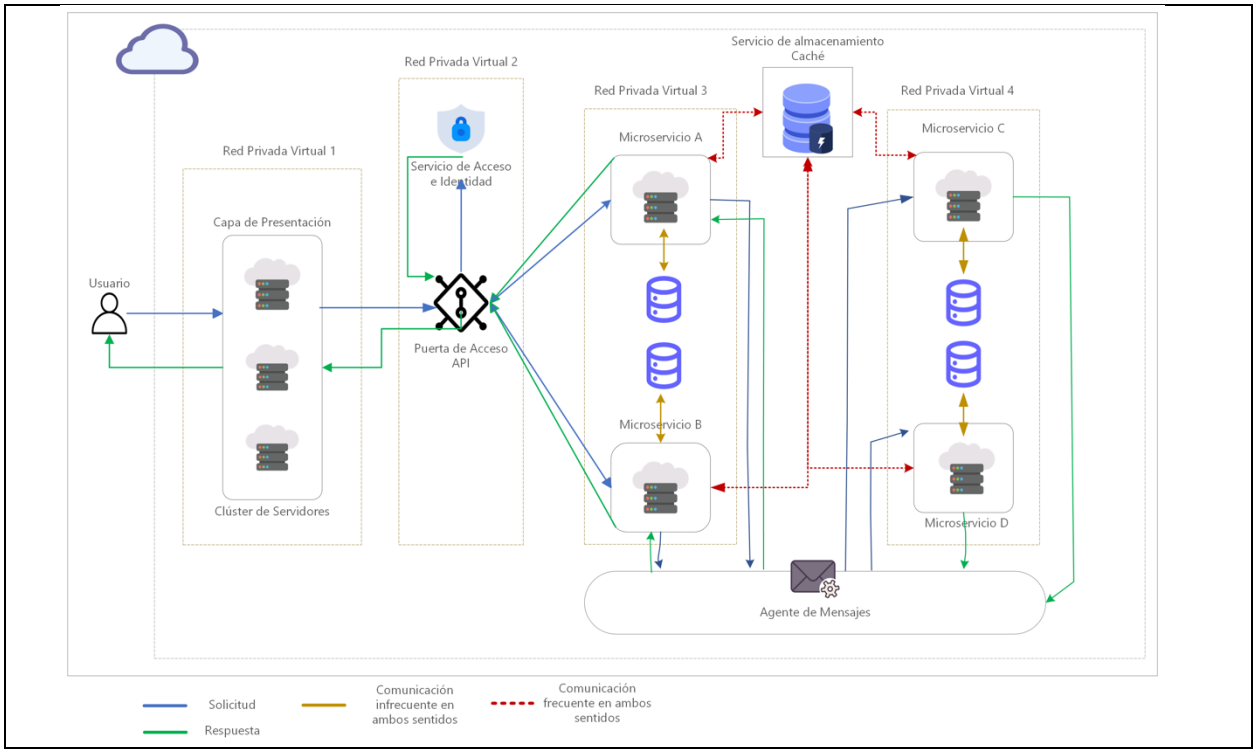


Figura 6.4. Modelo base de arquitectura de microservicios utilizando un agente de mensajería para la comunicación y almacenamiento en memoria caché. Fuente: elaboración propia.

De forma general, adicionar el servicio de almacenamiento en caché enfrente de las bases de datos y/o cualquier otro servicio o llamada a un servicio Web externo, siempre será una opción viable para mejorar el desempeño del flujo de datos y procesos dentro del sistema. Este patrón, en general, se utiliza para el acceso a datos que son de uso frecuente. Considerablemente, para el caso de negocio de una micro y pequeña empresa, este patrón es la recomendación por implementar; principalmente porque al inicio los volúmenes de datos no son enormes, ni tampoco el alcance del *software* escala a millones de usuarios; sin embargo, la viabilidad de escalar esta solución a niveles más complejos tampoco está fuera del alcance, especialmente en la era de los servicios de computación en la nube.

En síntesis, ambos diagramas mostrados en las figuras 6.3 y 6.4 son una base generalizada de cómo construir una arquitectura confiable y robusta para aplicaciones de *Software* como Servicio. Se insta a las micro y pequeñas empresas (PYME) de desarrollo de *software* a seguir este patrón para la construcción de sus productos de *software*, no solamente poseerán una estructura sólida y escalable, sino también una flexibilidad enorme de modificar su arquitectura original para ajustarla a las necesidades de negocio que más necesiten solventar, siempre y cuando sigan las buenas prácticas aquí descritas.

6.3.3 Consideraciones de la administración de las operaciones postproducción.

El objetivo principal de la gestión de la operación postproducción es asegurar el funcionamiento correcto de un conjunto de elementos que conforman el sistema y así, impactar de forma positiva los clientes del producto de *Software* como Servicio. Un sistema necesita de un seguimiento y monitoreo continuo por parte del proveedor, en este caso las micro y pequeñas empresas (PYME) de desarrollo de *software*, para garantizar la ejecución satisfactoria de su producto de *software*.

Dentro de este contexto, en este apartado se presentan dos consideraciones importantes que los proveedores de un *Software* como Servicio deben tener en cuenta.

6.3.3.1 Monitoreo sintético de la aplicación.

Una forma bastante eficiente de vigilar proactivamente la disponibilidad de la aplicación es por medio del monitoreo sintético del *software.* Este proporciona al proveedor la habilidad de simular tráfico de usuarios hacia la aplicación y así, verificar si está funcionando apropiadamente. Generalmente, se utiliza para validar los URL´s y recursos a través de los

protocolos HTTP y HTTP´s que devuelvan una respuesta correcta por medio de un código automatizado que se ejecuta cada cierto tiempo.

En este punto, en primer lugar, se necesita describir casos de prueba y escenarios a probar, esto puede compilarse y guardarse en un documento oficial del plan. Tal y como se realiza con un plan de pruebas corriente para las historias de usuario, se necesita tener los escenarios descritos con la siguiente información: identificador, datos de prueba, recurso a probar, resultado esperado, resultado final y estado. En la mayoría de los escenarios, la respuesta que se espera depende del recurso a consultar: podría ser una cadena de caracteres, una imagen, un logo, un número, etc.; el punto es asegurar que cualquiera que sea el recurso por consultar, esté disponible.

En la figura 6.5 se proyecta un diagrama de flujo sencillo donde ejemplifica un flujo de revisión que realiza la herramienta de monitoreo, generalmente a través de un proceso automatizado que se ejecuta según la frecuencia con la cual el administrador está a gusto. Esta es una de las tareas que realiza entre otras acciones que son importantes para la aplicación y auditoría, como son registros y manipulación de métricas.

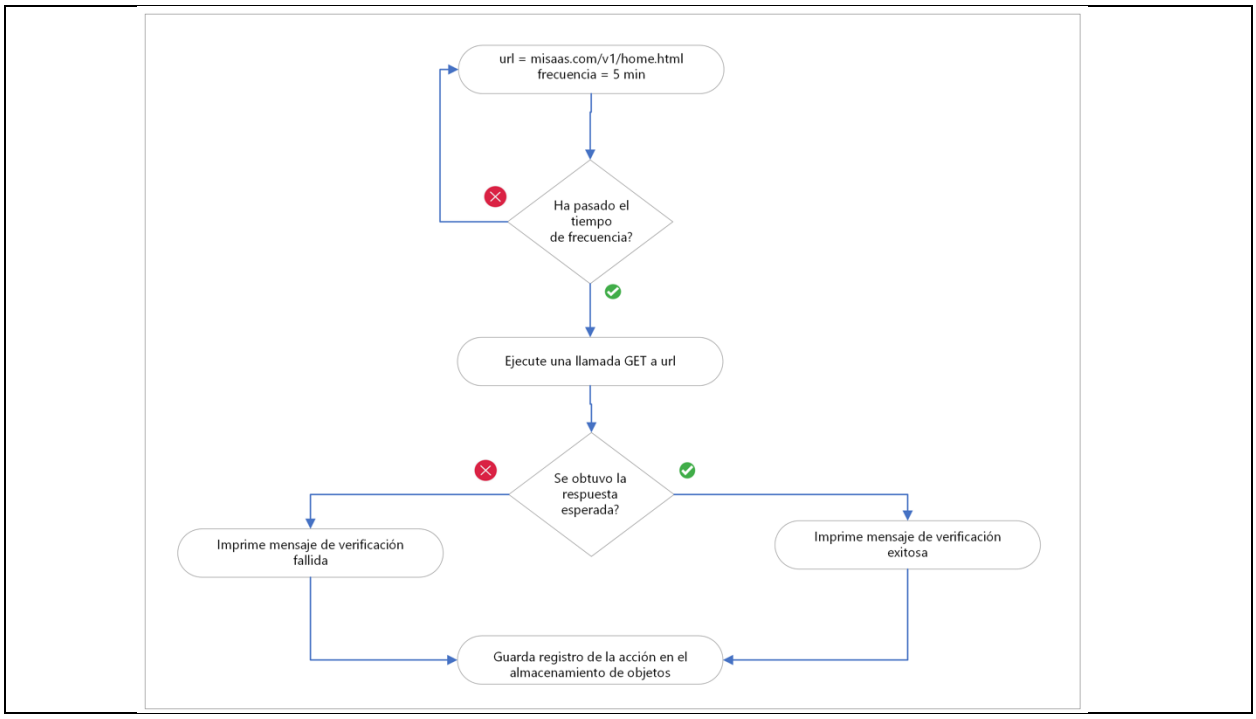


Figura 6.5. Ejemplo de revisión de un recurso vía monitoreo sintético. Fuente: elaboración propia.

Adicionalmente, el monitoreo sintético debe aplicarse tanto con recursos que son accesibles desde la capa de presentación de cualquier aplicación, así como también con recursos de los microservicios de forma interna. Cada escenario debe reflejar exactamente qué tipo de petición se está realizando y asignarle una prioridad e impacto si falla o si completa con éxito. De esta forma, el proveedor asegura que, si alguno de sus servicios falla en algún escenario, se evalúe su impacto y posterior plan de acción para contener y resolver el incidente.

Seguidamente, como parte del monitoreo del sistema, se requiere diseñar y construir una política de guardia por parte de los integrantes del equipo técnico a cargo del producto de software. Los miembros que formen parte de esta brigada se encargarán de accionar el plan de gestión de incidentes para resolver conflictos generados en la aplicación a partir de la alerta generada por el servicio de monitoreo. Es de suma importancia agregar documentación por medio de tiquetes a la hora de atender incidentes de esta naturaleza, especialmente si está

afectando a una gran cantidad de clientes. Dentro del plan de acción, también debe definirse a quiénes se necesita notificar y cuánto tiempo tiene que transcurrir para que estos mensajes sean enviados a los interesados.

En relación con este tema, cabe resaltar que existen dos formas de implementar monitoreo sintético: por medio de un servicio autogestionado de terceros o de algún proveedor de computación en la nube, o bien, construir un servicio personalizado nativo con otros servicios de la nube. La selección de una alternativa sobre la otra depende del presupuesto, facilidad de acceso que tenga cada empresa con respecto a las opciones disponibles y recurso humano calificado para configurar los servicios.

Por lo tanto, importan y por muchas razones, dos factores importantes: precio y fácil implementación. Si es un servicio gestionado por un tercero es más fácil de configurar y desplegar; además, tiende a ser un poco más caro que la opción de código abierto, que de todas formas hay que desplegar en la nube consumiendo recursos como máquinas virtuales y tráfico de red. Por otro lado, en el caso del código abierto, cabe adicionar que toma más tiempo, experiencia técnica y mayor involucramiento en el mantenimiento.

Basado en lo anterior, se recomienda utilizar servicios gestionados como Amazon CloudWatch Synthentics, Google Cloud Operation Suite, Google Cloud Managed Service for Prometheus, Microsoft Azure Application Insights, Azure Monitor, Amazon Managed Service for Prometheus y Amazon Managed Service for Grafana, y también servicios completamente de terceros, como DataDog Synthetic Monitoring, Sematext Synthetics Monitoring y Grafana Cloud. Todos estos servicios poseen precios que son relativamente accesibles para las micro y pequeñas empresas (PYME), ya que en su mayoría cobran por cantidad de transacciones y

tráfico generado, en un modelo de pague por lo que usa, lo cual es sumamente beneficioso para este tipo de organizaciones.

Asimismo, cabe resaltar que, si se opta por administrar completamente un servicio de estos por cuenta propia, la recomendación es realizarlo con los proyectos de código abierto de Prometheus y Grafana. El proyecto de Prometheus puede encontrarse en la página principal de https://prometheus.io/docs/prometheus/latest/getting_started/ y Grafana en https://grafana.com/docs/grafana/latest/introduction/oss-details/. Ambos poseen repositorios libres de descarga en Github con instrucciones de su uso e instalación. Adicionalmente, Grafana como tal, posee un proyecto específico para monitoreo sintético, el cual se puede encontrar en el repositorio de Github en la siguiente dirección: https://github.com/grafana/synthetic-monitoring-app.

En síntesis, no importa si es a través de un servicio gestionado por terceros o uno administrado por el proveedor de la aplicación, el foco siempre debe recaer en la proactividad que se requiere para administrar un *Software* como Servicio, en el cual los clientes dependen completamente del servicio brindado para llevar a cabo su objetivo principal. Es responsabilidad absoluta del proveedor estar alerta y responder a sus clientes ante incidentes que afecten el desempeño de la aplicación.

6.3.3.2 Servicios de mantenimiento y entrega de funcionalidades.

A continuación, se dicta una serie de apreciaciones acerca de la dinámica de entrega de funcionalidades y actualizaciones de *software*. El flujo de procesos que forme parte de esta fase conlleva mucha comunicación con los clientes del sistema, especialmente en una aplicación de *Software* como Servicio, el impacto y riesgo es algo que se entrega con cada actualización.

Primeramente, se debe contar con un plan y calendario de fechas en las cuales se comunique con anterioridad los servicios de mantenimiento y actualización de *software*. Generalmente, en estos procesos se entregan correcciones al *software* que son consecuencias de defectos encontrados en el funcionamiento de la aplicación; además, actualizaciones como parches de seguridad y mantenimientos programados de infraestructura.

De esta forma, en la fecha se encontrará la frecuencia con la cual estos servicios ocurrirán, se considera semanal o quincenal una buena opción. Se establece un tiempo máximo de duración, así como condiciones en las cuales debe suceder; por ejemplo, si durante el servicio de actualización el sistema se encontrará no disponible para los clientes. Además, también está presente un listado de defectos o mejoras que se están realizando, con su respectiva descripción y un número de identificación. En algunas ocasiones, inclusive, es buena práctica agregar el tiquete respectivo que corresponde a la evidencia de trabajo por parte del proveedor.

Asimismo, en relación con las funcionalidades nuevas, se debe tener unas fechas de entrega. No obstante, este rubro difiere de los servicios normales porque requieren tiempo de implementación, comunicación y pruebas de aceptación por parte de los clientes en los casos en los que aplique. En este último punto, es importante recalcar que es preferible tener un ambiente de pruebas especial para esta fase de aceptación de usuario en los casos en los cuales se tenga un *software* que resuelva ciertas necesidades de negocio críticas. Además, en esta fase de pruebas de aceptación, se debe invertir una cantidad de tiempo considerable antes de lanzar la nueva funcionalidad, la duración dependerá de la empresa proveedora del *software* y del problema de negocio que resuelva la aplicación.

Ambos casos deben coincidir con los canales de integración y entrega continua que se configuren a nivel de infraestructura. En este caso, los líderes de la implementación de estos

procesos son los ingenieros de confiabilidad de sitios en conjunto con los desarrolladores. Por su parte, los dueños del producto o gerentes de proyectos tienen que establecer la comunicación respectiva con los clientes de la aplicación.

Por último, la comunicación debe estar en tantos canales sea posible. Para ello se recomienda contar con un sitio Web para publicaciones masivas, en donde se detalle cada uno de los rubros del plan de actualización de *software* para mantenimientos rutinarios y para las actualizaciones mayores del sistema. A la vez, también las notificaciones por correo electrónico sirven como recordatorio de la ejecución de estos eventos. Dependiendo de la naturaleza del negocio del *software,* para aquellos que sirvan a empresas directamente, las notificaciones irán directamente al representante de la empresa cliente.

En síntesis, los pasos y recomendaciones que se han compilado en esta guía son de suma importancia para la creación, construcción y administración de un producto de *Software* como Servicio. El énfasis ha estado en la capacidad que poseen las micro y pequeñas empresas (PYME) de desarrollo de *software* para convertirse en proveedores clave de este modelo de entrega de *software*.

Por supuesto, se han hecho bastantes generalizaciones para poder abarcar varios temas, en los cuales se considera existe un alto grado de impacto para la correcta implementación del proceso que genere un producto de este tipo. Otra razón es la incapacidad de cubrir a detalle cada tema por separado. Es por ello que se insta a abrir espacio en la investigación académica a expandir a detalle cada subtema con sus propias propuestas de administración para los casos de las micro y pequeñas empresas (PYME) de Costa Rica. Existe mucho más conocimiento y métodos nuevos que se podrían aplicar a cada ítem cubierto a lo largo de esta propuesta.

REFERENCIAS BIBLIOGRÁFICAS

Ahmad, R., Hussain, A. & Baharom, F. (2015). A systematic review on characteristic and sub-characteristic for software development towards software sustainability. [Una revisión sistemática de las características y subcaracterísticas para el desarrollo de *software* hacia la sustentabilidad del *software*]. *Environment*. Retrieved from: http://wseas.us/e-library/conferences/2015/Malaysia/COMP/COMP-24.pdf

Alotaibi, M. B. (2016). Antecedents of software-as-a-service (SaaS) adoption: a structural equation model. *International Journal of Advanced Computer Research*, 6(25), 114–129. https://doi.org/10.19101/ijacr.2016.626019

Alvarado Abarca, M. (2017). *Metodología de implementación de aplicaciones de software en Amazon Web Services (AWS).* (Tesis de maestría, Universidad Nacional). Recuperado desde: http://hdl.handle.net/11056/14176

Amazon Web Services. (2021). Acerca de AWS. Recuperado el 10 de julio de 2021 desde: https://aws.amazon.com/es/about-aws/#:%7E:text=In%202006%2C%20Amazon%20Web%20Services,commonly%20known%20as%20cloud%20computing.

Anandamurugan, S., & Priyaa, T. (2014). "Service Oriented Architecture" [Arquitectura orientada a servicios]. Recuperado desde la base de datos EBSCOhost (Número de acceso: 1134313)

Anil, N., Warren, G., Coulter, D., Victor, Y., Killeen, S., Veloso, M., Wenzel, M., Parente, J. (2021). Asynchronous message-based communication [comunicación

asíncrona basada en mensajes]. Recuperado desde https://docs.microsoft.com/en-us/dotnet/architecture/microservices/architect-microservice-container-applications/asynchronous-message-based-communication

Arachchi, S. A. I. B. S. & Perera, I. (2018, Mayo). Continuous integration and continuous delivery pipeline automation for agile software project management [Automatización de un canal de integración y entrega continua para la gestión de proyectos de *software* ágiles]. *In 2018 Moratuwa Engineering Research Conference (MERCon).* (pp. 156-161). Retrieved from: https://www.academia.edu/39802146/Continuous_Integration_and_Continuous_Delivery_Pipeline_Automation_for_Agile_Software_Project_Management?from=cover_page

Arafat, M. (2018). Information security management system challenges within a cloud computing environment. [Retos en los sistemas de gestión de seguridad de la información en un ambiente de computación en la nube]. *In Proceedings of the 2nd International Conference on Future Networks and Distributed Systems (ICFNDS '18)*. (60), 1–6. doi: https://doi.org/10.1145/3231053.3231127

Asti Vera, A. (2015). *Metodología de la investigación* [E-Book]. Athenaica Ediciones Universitarias. Recuperado el 5 de abril de 2021 desde: https://elibro.net/en/ereader/biblioutn/43844?page=43

Atlassian. (s. f.). What is DevOps? Retrieved march, 6, 2021 from: https://www.atlassian.com/devops#:%7E:text=DevOps%20is%20a%20set%20of,software%20faster%20and%20more%20reliably.

Attaran, M. & Woods, J. (2018). Cloud Computing Technology: A Viable Option for Small and Medium-Sized Businesses [Tecnología de computación en la nube: Una opción viable para pequeñas y medianas empresas]. *Journal of Strategic Innovation and Sustainability*, *13*(2), 94–106. https://doi.org/10.33423/jsis.v13i2.609

Ayoobkhan, A. L. M. & Asirvatham, D. (2019). A Study on the Adoption of Software as a Service (SaaS) in Online Business SMEs in Sri Lanka. *Asian Journal of Research in Computer Science*, *1*(1), 1–13. https://doi.org/10.9734/AJRCOS/2018/45904

Babar, M. A., Brown, A. W. & Mistrik, I. (2014). Agile Software Architecture: Aligning Agile Processes and Software Architectures. [Arquitectura de *software* ágil: alineando procesos ágiles con las arquitecturas de *software*] [PDF]. Retrieved from: EBSCOhost (Access: 516109)

Baena Paz, G. (2017). *Metodología de la investigación.* (Tercera Edición) [E-book]. Grupo Editorial Patria. Recuperado el 1 de abril de 2021 desde: http://www.biblioteca.cij.gob.mx/Archivos/Materiales_de_consulta/Drogas_de_Abuso/Articulos/metodologia%20de%20la%20investigacion.pdf

BCS. (2019). History of the cloud | *BCS*. Recuperado el 7 de julio de 2021 desde: https://www.bcs.org/content-hub/history-of-the-cloud/

Briggs, B. & Kassner, E. (2017). *Enterprise Cloud Strategy* [PDF] (2nd Edition). *Microsoft Press*. Recuperado el 13 de marzo de 2021 desde:

https://info.microsoft.com/rs/157-GQE-382/images/EN-US-CNTNT-ebook-Enterprise_Cloud_Strategy_2nd_Edition_AzureInfrastructure.pdf

Bussler, C. (2002). Software as a service. *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data - SIGMOD '02*, 1. https://doi.org/10.1145/564691.564801

Cabrera, M. (2006, Agosto). *Introducción a las fuentes de información*. Universidad Politécnica de Valencia. Researchgate.Net. Recuperado el 17 de abril de 2021 desde: https://www.researchgate.net/publication/50839717_Introduccion_a_las_fuentes_de_informacion

CAMTIC. (2012, septiembre 26). *Pymes de Costa Rica adoptan cloud computing con rapidez, según estudio – Camtic*. Recuperado el 6 de marzo de 2021, desde: https://www.camtic.org/actualidad-tic/pymes-de-costa-rica-adoptan-cloud-computing-con-rapidez-segun-estudio/

Campos Ocampo, M. (2017). Métodos de investigación académica. *Estudios generales, sociedad, y cultura*, (1.1), pp. 2-84. Recuperado desde: http://www.icomoscr.org/m/investigacion/%5BMETODOS%5DFolleto_v.1.1.pdf

Castro, J. (2019, August 15). "Redes sociales son las principales aliadas para generar ventas en emprendimientos." *La República*. Recuperado el 6 de marzo de 2021 desde: https://www.larepublica.net/noticia/redes-sociales-son-las-principales-aliadas-para-generar-ventas-en-emprendimientos https://www.larepublica.net

Cegarra Sánchez, J. (2012). *Los métodos de investigación*. [E-Book]. Ediciones Díaz de Santos. Recuperado el 5 de abril de 2021 desde: https://elibro.net/en/ereader/biblioutn/62637?page=14

Cegarra Sánchez, J. (2012). *Metodología de la investigación científica y tecnológica*. Ediciones Díaz de Santos (ed.). (pp. 81-93). Recuperado de: https://elibro.net/en/ereader/biblioutn/62637?page=14

Cerny, T., Donahoo, M. J., & Pechanec, J. (2017). Disambiguation and Comparison of SOA, Microservices and Self-Contained Systems [Desambiguación y comparación de SOA, microservicios, y sistemas auto contendidos]. *RACS '17: Proceedings of the International Conference on Research in Adaptive and Convergent Systems*, 228–235. doi: https://doi.org/10.1145/3129676.3129682

Chandrasekaran, K. (2014). *Essentials of Cloud Computing* (1st ed.). [PDF] Recuperado el 10 de julio de 2021 desde: https://keyhannet.com/wp-content/uploads/2018/11/K.-Chandrasekaran-Essentials-of-Cloud-Computing-2014-Chapman-and-Hall_CRC.pdf

Cortés, M. E. C., Iglesias León, M. I. y Carmen, U. A. (2005). *Generalidades sobre metodología de la investigación*. (Primera Edición) [E-Book]. Universidad Autónoma del Carmen. Recuperado el 1 de abril de 2021: https://www.ucipfg.com/Repositorio/MIA/MIA-12/Doc/metodologia_investigacion.pdf

Cusumano, M. A. (2019). Technology Strategy and Management: The Cloud as an Innovation Platform for Software Development: How cloud computing became a

platform. [Estrategia y Administración de tecnología: La nube como una Plataforma de innovación para el Desarrollo de *software*: cómo la computación en la nube se convirtió en plataforma]. *Communications of the ACM*, 62(10), 20–22. https://doi-org.ezproxy.utn.ac.cr/10.1145/3357222

Diaby, T., & Rad, B. B. (2017). Cloud Computing: A review of the Concepts and Deployment Models [Computación en la nube: una revisión de los conceptos y modelos de despliegue]. *International Journal of Information Technology and Computer Science*, 9(6), 50–58. https://doi.org/10.5815/ijitcs.2017.06.07

Dubey, A. & Wagle, D. (2007). Delivering software as a service [Entregando *software* como servicio]. *The McKinsey Quarterly, 6(2007), 2007*. Recuperado el 5 de marzo de 2021 desde: https://abs.in/sites/default/files/Delivering%20Software%20as%20a%20Service.pdf

Ebert, C. (2014). Software product management [Gestión del producto de software]. *IEEE Software*, 31(3), pp. 21-24. Recuperado desde https://www.academia.edu/31682481/Software_Product_Management?from=cover_page

ECPI University. (2020). A Brief History of Cloud Computing. Recuperado el 7 de julio de 2021, desde: https://www.ecpi.edu/blog/a-brief-history-of-cloud-computing

Fitzgerald, B. & Stol, K. (03 de junio 2014). Continuous software engineering and beyond: trends and challenges [Ingeniería del *software* continua y más: tendencias

y retos]. *Proceedings of the 1st International Workshop on Rapid Continuous Software Engineering*, 1-9. doi: https://doi.org/10.1145/2593812.2593813

Foster, I., Vasiliadis, V. & Tuecke, S. (2013). Software-as-a-Service as a path to software sustainability. [El *software* como servicio como ruta a la sustentabilidad del s*oftware*]. *Figshare*. Recuperado de https://integration.globuscs.info/sites/default/files/saas-as-a-path-to-sustainable-software-delivery.pdf

Fox, A., & Patterson, D. (2014). Engineering Software as a Service: An Agile Approach Using Cloud Computing (First Edition) [Construyendo *Software* como servicio: una propuesta ágil utilizando computación en la nube (primera edición)]. Recuperado desde https://cin.ufpe.br/~fbma/4P/essas.pdf

Franch, X., Palomares, C. & Gorschek, T. (24 de mayo, 2021). On the requirements engineer role. [En el rol de ingeniero de requerimientos]. *Communications of the ACM*, 64(6), pp. 69-75. doi: https://doi.org/10.1145/3418292

Gallego Lorenzo, J. y Juncà Campdepadrós, M. (s. f.). Fuentes y servicios de información. *Open Access.Ouc.Edu*. Recuperado el 5 de abril de 2021 desde: http://openaccess.uoc.edu/webapps/o2/bitstream/10609/241/5/Fuentes%20de%20información%20I_Módulo%201_Fuentes%20y%20servicios%20de%20información.pdf

Garg, S. & Garg, S. (2019, Marzo). Automated cloud infrastructure, continuous integration and continuous delivery using docker with robust container security. [Infraestructura en la nube automatizada, integración y entrega continua

utilizando dócker con seguridad robusta de contenedor]. *In 2019 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, (pp. 467-470). Recuperado de https://www.researchgate.net/profile/Somya-Garg/publication/331131851_Automated_Cloud_Infrastructure_Continuous_Integration_and_Continuous_Delivery_using_Docker_with_Robust_Container_Security/links/5cd5c53ba6fdccc9dd9f54d9/Automated-Cloud-Infrastructure-Continuous-Integration-and-Continuous-Delivery-using-Docker-with-Robust-Container-Security.pdf

Goh, T. (2013). An evaluation of configuration management for high performance computing on clouds. [Tesis de Grado, Monash University]. Recuperado desde: https://bridges.monash.edu/articles/thesis/An_evaluation_of_configuration_management_for_high_performance_computing_on_clouds/4701088

Gomaa, H. (2011). Software Modeling and Design: UML, Use Cases, Patterns, and Software Architectures. [Modelado y diseño de *software*: UML, Casos de uso, Patrones, y Arquitecturas de *software*]. Recuperado desde la base de datos EBSCOhost (Número de acceso: 361602)

Google (2021). *Implementación continua en Google Kubernetes Engine mediante Jenkins.* Recuperado de: https://cloud.google.com/architecture/continuous-delivery-jenkins-kubernetes-engine

Google. (2021). *Google App Engine Documentation*. [*Documentación de Google App Engine*]. Recuperado el 20 de julio de 2021, desde: https://cloud.google.com/appengine/docs

Google. (2021b). Elección de la arquitectura correcta para la distribución de datos globales. [Figura]. Recuperado de: https://cloud.google.com/architecture/global-data-distribution?hl=es

Guerrero Dávila, G. (2015). *Metodología de la investigación*. [E-Book]. Grupo Editorial Patria. Recuperado el 5 de abril de 2021 desde: https://elibro.net/en/lc/biblioutn/titulos/40363

Haig-Smith, T. & Tanner, M. (2016). Cloud Computing as an Enabler of Agile Global Software Development. *Issues in Informing Science & Information Technology*, 13, 121–144. https://doi-org.ezproxy.utn.ac.cr/10.28945/3476

Hernández Martín, Z. (2012). *Métodos De Análisis De Datos (Apuntes)* (1.ª ed.) [Monografía]. Recuperado de: https://publicaciones.unirioja.es/catalogo/monografias/mdm06.shtml

Hernández Sampieri, R. (2014). *Metodología De La Investigación* (6th ed.). McGraw Hill. Recuperado el 1 de abril de 2021 desde: https://www.uca.ac.cr/wp-content/uploads/2017/10/Investigacion.pdf

Hettrick, S. (2016). Research software sustainability: Report on a knowledge exchange workshop. [Investigación de sustentabilidad del *software*: reporte de una sesión de trabajo de intercambio de conocimiento]. Recuperado de: https://digitalcommons.unl.edu/cgi/viewcontent.cgi?article=1005&context=scholcom

https://abs.in/sites/default/files/Delivering%20Software%20as%20a%20Service.pdf

Hustad, E. & Olsen, D. H. (2021). Creating a sustainable digital infrastructure: The role of service-oriented architecture. [Creando una infraestructura digital sustentable: El rol de la arquitectura orientada a servicios]. *Procedia Computer Science*, 181(2021), 597–604. https://doi.org/10.1016/j.procs.2021.01.210

Hsu, T. (2018). Hands-On Security in DevOps [Enfoque práctico de seguridad en DevOps]. Recuperado de: https://learning.oreilly.com/library/view/hands-on-security-in/9781788995504/

IBM Cloud Education (2020). *Message Brokers* [*Agentes de Mensajes*]. Recuperado desde https://www.ibm.com/cloud/learn/message-brokers

IETF (2022). *RFC 9110 - HTTP Semantics [RFC 9110 - Semántica de HTTP*]. Recuperado desde: https://httpwg.org/specs/rfc9110.html

Institute for Work & Health (2015). What researchers mean by… cross-sectional and longitudinal studies. [Lo que los investigadores quieren decir con… estudios transversales y longitudinales]. *At Work*, (81), pp. 1-8. Recuperado desde https://www.iwh.on.ca/sites/iwh/files/iwh/at-work/at_work_81.pdf

Ivkic, I., Wolfauer, S., Oberhofer, T. & Tauber, M. G. (2017). On the cost of cyber security in smart business. [El costo de la ciberseguridad en negocios inteligentes]. *In 2017 12th International Conference for Internet Technology and Secured Transactions (ICITST)*, 255-260. Recuperado desde https://arxiv.org/pdf/1905.06711.pdf

Jadeja, Y. & Modi, K. (2012). Cloud computing - concepts, architecture, and challenges. *In 2012 International Conference on Computing, Electronics and Electrical Technologies (ICCEET)*, 877–880. Recuperado desde: https://www.academia.edu/4693591/Cloud_computing_concepts_architecture_and_challenges?from=cover_page

Jenkins, M. (2021). Q&A: Configuration Management Tools vs Infrastructure as Code. [Registro Web] Recuperado 14 de agosto de 2021, de https://www.linkedin.com/pulse/qa-configuration-management-tools-vs-infrastructure-code-jenkins

Jiménez Aragón, L. (2018). Las matrices de evaluación: clasificación y normas de forma y fondo para su elaboración. *UMBRAL*, *XLI*. Recuperado de: http://www.colypro.com/ee_uploads/revista/UMBRAL-41.pdf

Jones, C. (1996). Software Change Management [Gestión del cambio de *software*]. *Computer*, 29, 80–82. https://doi.org/10.1109/2.485858

Kaisler, S. H. (2005). *Software Paradigms*. [*Paradigmas de software*]. Recuperado de la base de datos EBSCOhost (Número de acceso: 130950)

Kumar, R. & Jain, K. (2014). Apache CloudStack: Open-Source Infrastructure as a Service Cloud Computing Platform. *International Journal of advancement in Engineering technology, Management and Applied Science*, 1(2). doi: https://doi.org/10.13140/2.1.1290.0483

Kwan, A., Jacobsen, H., Chan, A., & Samoojh, S. (2016). Microservices in the modern software world [Microservicios en el mundo moderno del software], 297–299. doi: https://doi.org/10.5555/3049877.3049915

Lee, J. (2013). A View of Cloud Computing. *International Journal of Networked and Distributed Computing*, *1*(1), 2. https://doi.org/10.2991/ijndc.2013.1.1.2

Lee, L. S. & Brink, W. D. (2019). Trust in Cloud-Based Services: A Framework for Consumer Adoption of Software as a Service. *Journal of Information Systems*, 34(2), 65–85.doi: https://doi.org/10.2308/isys-52626

Lee, L. S. & Brink, W. D. (2019). Trust in Cloud-Based Services: A Framework for Consumer Adoption of Software as a Service [Figura]. Recuperado desde: https://ezproxy.utn.ac.cr/login?url=https%3a%2f%2fsearch.ebscohost.com%2flogin.aspx%3fdirect%3dtrue%26db%3dbsu%26AN%3d145694400%26lang%3des%26site%3dehost-live

Li, X., & Stuart, S. E. (2015). Understanding the Dynamics of Service-Oriented Architecture Implementation [Entendiendo la dinámica de la implementación de la arquitectura orientada a servicios]. *Journal of Management Information Systems*, 32(2), 104–133. doi: https://doi.org/10.1080/07421222.2015.1063284

Loggle (2021). *What is Cloud Asset Management?* [*Qué es gestión de activos de la nube*]. Recuperado desde https://loggle.io/knowledgebase/cloud-asset-management

Lukman, H. (2020). Conceptual Framework of Cloud Computing Implementation on Start-Up Companies with Approach. *IOP Conference Series: Materials Science and Engineering*, *1007*, 012176. https://doi.org/10.1088/1757-899x/1007/1/012176

Luna Garcia, J., Langenberg, R., & Suri, N. (2012). Benchmarking cloud security level agreements using quantitative policy trees [Comparando los niveles de seguridad de la nube utilizando una política cuantitativa de árboles]. *In Proceedings of the 2012 ACM Workshop on Cloud computing security workshop (CCSW '12)*, 103–112. doi: https://doi.org/10.1145/2381913.2381932

Mahimkar, A., de Andrade, C. E., Sinha, R. & Rana, G. (2021). A Composition Framework for Change Management. [Un marco de trabajo de composición para la gestión del cambio]. *SIGCOMM '21: Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, 788–806. https://doi.org/10.1145/3452296.3472901

Mardani, A., Jushod, A., Nor, K. M., Khalifah, Z., Zakwan, N. & Valipour, A. (2015). Multiple criteria decision-making techniques and their applications – a review of the literature from 2000 to 2014. [Las técnicas de toma de decisión de criterio múltiple y sus aplicaciones - una revisión de literatura desde el 2000 al 2014]. *Ekonomska Istraživanja / Economic Research*, 28(1), pp. 516-571. Recuperado desde https://www.researchgate.net/publication/309900481_Multiple_criteria_decision-making_techniques_and_their_applications-a_review_of_the_literature_from_2000_to_2014

Martínez Ruiz, H. (2012). *Metodología de la investigación* [E-Book]. Cengage Learning. Recuperado el 5 de abril de 2021 desde: https://elibro.net/en/lc/biblioutn/titulos/39957

MEIC. (2021). *PYMES activas*. Recuperado el 17 de abril de 2021, desde: https://www.meic.go.cr/meic/web/761/datos-abiertos/pyme/registro-de-empresas.php

Mell, P. & Grance, T. (2011). The NIST Definition of Cloud Computing. [La definición de computación en la nube de NIST]. *NIST Special Publication SP 800–145*. Published. Retrieved from: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf

Microsoft. (2021a). *What is a Cloud Service Provider – Definition.* [*Qué es un proveedor del servicio de la nube - Definición*]. Recuperado el 31 de julio de 2021, desde: https://azure.microsoft.com/en-gb/overview/what-is-a-cloud-provider/

Moghaddam, F. F., Rohani, M. B., Ahmadi, M., Khodadadi, T. & Madadipouya, K. (2015). Cloud Computing: Vision, Architecture and Characteristics. *In 2015 IEEE 6th Control and System Graduate Research Colloquium (ICSGRC)*, 1–6. Retrieved from: https://www.researchgate.net/profile/Faraz-Fatemi-Moghaddam/publication/296329949_Cloud_Computing_Vision_Architecture_and_Characteristics/links/56d449a908ae2ea08cf8e971/Cloud-Computing-Vision-Architecture-and-Characteristics.pdf

Mohan, T. S. (2011). Migrating into a Cloud [Migrando hacia la nube]. En Buyya, R., Broberg, J., & Goscinski, A. M. (Eds.) *Cloud Computing: Principles and*

Paradigms (1.a ed.) (pp. 43-55) Recuperado de http://dhoto.lecturer.pens.ac.id/lecture_notes/internet_of_things/CLOUD%20COMPUTING%20Principles%20and%20Paradigms.pdf

Morrow, S. (2011). Data Security in the Cloud [Seguridad de datos en la nube]. In: Buyya, R., Broberg, J. & Goscinski, A. (Eds), *Cloud Computing: Principles and Paradigms* (pp. 573-591). [PDF]. Retrieved from: http://dhoto.lecturer.pens.ac.id/lecture_notes/internet_of_things/CLOUD%20COMPUTING%20Principles%20and%20Paradigms.pdf

Monteiro-Dias, R., Oliveira-Zacarias, R., de-Lima-Varella, JL. & Pereira-dos-Santos, R. (2022). Investigating Information Security in Systems-of-Systems [Investigando la seguridad de la información en sistemas de sistemas]. *In XVIII Brazilian Symposium on Information Systems (SBSI),* 1–8. doi: https://doi.org/10.1145/3535511.3535523

Nadareishvili, I., Mitra, R., McLarty, M., & Amundsen, M. (2016). Microservice Architecture Aligning Principles, Practices, and Culture (1.a ed.) [Arquitectura de microservicios: alineando principios, prácticas, y cultura]. Recuperado desde https://docs.broadcom.com/doc/microservice-architecture-aligning-principles-practices-and-culture

Namasudra, S. (2018). Cloud Computing: A New Era. *Journal of Fundamental and Applied Sciences*, *10*(2), 113–135. https://doi.org/10.4314/jfas.v1 Recuperado el 26 de julio de 2021 desde: http://jfas.info/psjfas/index.php/jfas/article/view/39860i2.9

Namasudra, S. (2018). Cloud Computing: A New Era. *Journal of Fundamental and Applied Sciences*, 10(2), 113–135 [Figura]. Recuperado el 26 de julio de 2021 desde: http://jfas.info/psjfas/index.php/jfas/article/view/3986

Neogy, S. (2017). Information security: course [Seguridad de la información: curso]. *In Proceedings of the 1st International Conference on Internet of Things and Machine Learning (IML '17)*, 1–8. doi: https://doi.org/10.1145/3109761.3158397

Neto, M. D. (2019). A brief history of cloud computing. Recuperado el 7 de julio de 2021, desde: https://www.ibm.com/blogs/cloud-computing/2014/03/18/a-brief-history-of-cloud-computing-3/

Observatorio de Desarrollo de la Universidad de Costa Rica (2018). *Informe de resultados III Encuesta Nacional de la micro, pequeña y mediana empresa en Costa Rica 2018*. (Informe No. III). Recuperado de http://reventazon.meic.go.cr/informacion/estudios/2018/estadopyme/informe.pdf

Odun-Ayo, I., Ananya, M., Agono, F. & Goddy-Worlu, R. (2018). Cloud computing architecture: A critical analysis. *In 2018 18th International Conference on Computational Science and Applications (ICCSA)*, 1–7. Retrieved from: https://www.researchgate.net/profile/Isaac-Odun-Ayo/publication/327125094_Cloud_Computing_Architecture_A_Critical_Analysis/links/5cc6ed0f299bf120978802bc/Cloud-Computing-Architecture-A-Critical-Analysis

Onar, S. C., Oztaysi, B. & Kahraman, C. (2018). Multicriteria Evaluation of Cloud Service Providers Using Pythagorean Fuzzy TOPSIS. [Evaluación multicriterio

de los proveedores de servicio de la nube usando el método TOPSIS difuso pitagórico]. *Journal of Multiple-Valued Logic & Soft Computing*, 30(2–3), 263–283. Retrieved from: EBSCOhost. (Access: 128016144)

Otzen, T., & Manterola, C. (2017). Técnicas de Muestreo sobre una Población a Estudio. *International Journal of Morphology*, *35*(1), 227–232. doi: https://doi.org/10.4067/s0717-95022017000100037

Pethuru, R. (2011). Enriching the "Integration as a Service" paradigm for the cloud era [Enriqueciendo el paradigma de "integración como servicio" en la era de la nube]. En Buyya, R., Broberg, J., & Goscinski, A. (Eds), *Cloud Computing: Principles and Paradigms* (pp. 57-XX). [PDF]. Retrieved from: http://dhoto.lecturer.pens.ac.id/lecture_notes/internet_of_things/CLOUD%20COMPUTING%20Principles%20and%20Paradigms.pdf

Ponsard, C. & Deprez, J.C. (Mayo, 2018). Helping SMEs to better develop software: experience report and challenges ahead [Ayudando a las pequeñas y medianas empresas a desarrollar mejor software: experimente reportes y retos anticipadamente]. *Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Practice*, pp. 213-214, doi: https://doi.org/10.1145/3183519.3183553

PROSIC. (2010, May). Computación en la nube en Costa Rica. *Informes hacia la sociedad de la información y el conocimiento*. Recuperado el 13 de marzo de 2021 desde: http://www.prosic.ucr.ac.cr/sites/default/files/recursos/informe_2010.pdf

PROSIC. (2010). Computación en la nube en Costa Rica. Informes hacia la sociedad de la información y el conocimiento [Gráfico]. Recuperado el 13 de marzo de 2021, desde: http://www.prosic.ucr.ac.cr/sites/default/files/recursos/informe_2010.pdf

RACSA. (2019a, Mayo 28). *Cuidados a la hora de elegir un gestor documental adecuado*. [Registro Web]. Recuperado el 15 de marzo de 2021 desde: https://www.racsa.go.cr/blog/cuidados-a-la-hora-de-elegir-un-gestor-documental-adecuado/

RACSA. (2019b, Junio 25). Beneficios De Elegir VIRTUAL DESKTOP de RACSA. [Registro Web]. Recuperado el 15 de marzo de 2021 desde: https://www.racsa.go.cr/blog/beneficios-de-elegir-virtual-desktop-de-racsa/

Rahman, M. R., Enck, W., & Williams, L. (Eds.). (2020). Do Configuration Management Tools Make Systems More Secure? An Empirical Research Plan [¿Hacen más seguros los sistemas las herramientas de gestión de la configuración? Un plan de investigación empírica]. *HotSoS '20: Proceedings of the 7th Symposium on Hot Topics in the Science of Security*, 2, 1-2. Obtenido en la base de datos ACM Library. DOI: https://doi.org/10.1145/3384217.3384223

Rahulamathavan, Y., Pawar, P. S., Burnap, P., Rajarajan, M., Rana, O.F., y Spanoudakis, G. (2014). Analysing Security requirements in Cloud-based Service Level Agreements [Analizando requerimientos de seguridad en los acuerdos de nivel de servicio basados en la nube]. *In Proceedings of the 7th International Conference on Security of Information and Networks (SIN '14)*, 73–76. doi: https://doi.org/10.1145/2659651.2659735

Rastogi, S. (2021). *Cloud Computing Simplified*. Recuperado el 10 de julio de 2021 desde: https://books.google.co.cr/books?id=GYQnEAAAQBAJ&pg=PT26&lpg=PT26&dq=2009+google+apps+launch&source=bl&ots=O05NVzU4tX&sig=ACfU3U3AZre3n250q6jZxGbxWO3uik_6RQ&hl=en&sa=X&ved=2ahUKEwjyxce109nxAhVO4qwKHXOABBgQ6AEwEnoECBUQAw#v=onepage&q=2009%20google%20apps%20launch&f=false

Real Academia Española. (s.f.a). Confianza. En *Diccionario de la lengua española*. Recuperado el 06 de enero de 2022 desde: https://dle.rae.es/confianza?m=form

Real Academia Española. (s.f.b). Aplicación. En *Diccionario de la lengua española*. Recuperado el 16 de julio de 2022 desde: https://dle.rae.es/aplicaci%C3%B3n

Real Academia Española. (s.f.c). Software. En *Diccionario de la lengua española*. Recuperado el 16 de julio de 2022 desde: https://dle.rae.es/software?m=form

Real Academia Española. (s.f.d). Dato. En *Diccionario de la lengua española*. Recuperado el 17 de julio de 2022 desde: https://dle.rae.es/software?m=form

Real Academia Española. (s.f.e). Digital. En *Diccionario de la lengua española*. Recuperado el 18 de julio de 2022 desde: https://dle.rae.es/digital

Real Academia Española. (s.f.f). Base de datos. En *Diccionario de la lengua española*. Recuperado el 21 de julio de 2022 desde: https://dle.rae.es/base#CiiosqO

Red Hat (2018). El concepto de las interfaces de programación de aplicaciones. Recuperado desde https://www.redhat.com/es/topics/api

Ruparelia, N. B. (2016). *Cloud Computing (The MIT Press Essential Knowledge series)* (Illustrated ed.). Recuperado el 13 de marzo de 2021 desde https://ezproxy.utn.ac.cr/login?url=https%3a%2f%2fsearch.ebscohost.com%2flogin.aspx%3fdirect%3dtrue%26db%3de000xww%26AN%3d1238001%26lang%3des%26site%3dehost-live%26ebv%3DEB%26ppid%3Dpp_ii

Santos, N. A. & Quilliam, W. (2015). An Overview of the Change Management Process and Examples of Software to Help Organizations Effectively Manage Change. [Una perspectiva general del proceso de la gestión del cambio y ejemplos de software que ayuda organizaciones a administrar efectivamente los cambios]. *GSTF Journal on Business Review (GBR)*, 4(1), 1–4. https://doi.org/10.7603/s40706-015-0001-x

Senarathna, I., Wilkin, C., Warren, M., Yeoh, W. & Salzman, S. (2018). Factors That Influence Adoption of Cloud Computing: An Empirical Study of Australian SMEs. *Australasian Journal of Information Systems*, 22. https://doi.org/10.3127/ajis.v22i0.1603

Seppänen, V. (2020, Mayo 18). JYX - Customer retention in software-as-a-service business. *University of Jyväskylä*. Recuperado el 5 de marzo de 2021 desde: https://jyx.jyu.fi/handle/123456789/69028

Sharma, S. (2017). Mastering Microservices with Java 9 - Second Edition [Dominando los microservicios con Java 9 - segunda edición]. Recuperado desde la base de datos EBSCOhost. (Número de acceso: 1652584)

Shastri, Y., Hoda, R. & Amor, R. (5 de febrero, 2017). Understanding the Roles of the Manager in Agile Project Management. [Entendiendo los roles del gerente en la administración ágil de proyectos]. *ISEC '17: Proceedings of the 10th Innovations in Software Engineering Conference*, pp. 45-55. doi: https://doi.org/10.1145/3021460.3021465

Sloss, B. T. (2016). Introduction [Introducción]. En: Beyer, B., Jones, C., Petoff, J. & Murphy, N.R. (Eds), *Site Reliability Engineering*. Retrieved from: https://learning.oreilly.com/library/view/site-reliability-engineering/9781491929117/

Stein, M., Campitelli, V. & Mezzio, S. (2020). Managing the Impact of Cloud Computing. [Administrando el impacto de la computación en la nube]. *CPA Journal*, 90(6), 20–27. Retrieved from: EBSCOhost. (Access: 144364480)

Tizzei, L. P., Nery, M., Segura, V. C. V. B. & Cerqueira, R. F. G. (2017). Using Microservices and Software Product Line Engineering to Support Reuse of Evolving Multi-tenant SaaS. [Utilizando Microservicios y una línea de ingeniería de producto de *software* para el reuso de la evolución del *software* como servicio multitenencia]. *Proceedings of the 21st International Systems and Software Product Line Conference - Volume A*, *A*, 205–214. doi: https://doi.org/10.1145/3106195.3106224

Vanbrabant, B. & Joosen, W. (2014). Configuration management as a multi-cloud enabler. *In Proceedings of the 2nd International Workshop on CrossCloud*

Systems (pp. 1-3). [Figura]. Retrieved from: ACM Library. doi: https://doi.org/10.1145/2676662.2676672

Varghese, B. & Buyya, R. (2018). Next generation cloud computing: New trends and research directions. [La siguiente generación en computación en la nube: Nuevas tendencias y direcciones de investigación]. *Future Generation Computer Systems*, *79*, 849–861. https://doi.org/10.1016/j.future.2017.09.020

Vu Viet Hoang Pham, Xiao Liu, Xi Zheng, Min Fu, Deshpande, S. V., Weidong Xia, Roger Zhou & Abdelrazek, M. (2017). PaaS - Black or White: An Investigation into Software Development Model for Building Retail Industry SaaS. *ICSE: International Conference on Software Engineering*, 285–287. doi: https://doi-org.ezproxy.utn.ac.cr/10.1109/ICSE-C.2017.57

Wang, Z. (Junio, 2020). Comparisons on Scrum Team Strategies: A multi-agent Simulation. [Comparación de las estrategias del equipo de Scrum: Una simulación multiagente]. *ICCMS '20: Proceedings of the 12th International Conference on Computer Modeling and Simulation*, pp. 120-124. doi: https://doi.org/10.1145/3408066.3408087

Yarlagadda, R. T. (2018). Understanding DevOps & bridging the gap from continuous integration to continuous delivery. [Entendiendo DevOps y enlazando la integración continua con la entrega continua]. *Journal of Emerging Technologies and Innovative Research (JETIR)*, 5(2), 1420–1424. Retrieved from: https://www.researchgate.net/profile/Ravi-Teja-Yarlagadda/publication/350157935_Understanding_DevOps_bridging_the_gap_f

rom_continuous_integration_to_continuous_delivery/links/6053df5e458515e83455a764/Understanding-DevOps-bridging-the-gap-from-continuous-integration-to-continuous-delivery.pdf

Zhang, P. & Liu, N. (2019). Longitudinal Mixed Methods Designs in Language Teaching Research. [Diseños de métodos mixtos longitudinales en investigación de enseñanza de lenguaje]. *International Journal of Multiple Research Approaches*, *11*(2), pp. 132-133. doi: https://doi.org/10.29034/ijmra.v11n2a1

GLOSARIO

A

Aplicación: "*Inform*. Programa preparado para una utilización específica, como el pago de nóminas, el tratamiento de textos, etc." (Real Academia Española, s.f.b, definición 4).

Arquitectura de *Software*: Es un conjunto de estructuras necesitadas por una razón acerca del sistema, que compromete elementos del *software*, las relaciones entre ellos y propiedades de ambos. (Bass, Clements & Kazman, 1998, citado en Babar, Brown y Mistrik, 2014, p. 4)

Arquitectura Orientada a Servicios (SOA): Es un estilo de arquitectura que soporta servicios interoperables altamente desacoplados para habilitar la flexibilidad y agilidad del negocio. (Borges, Holley y Arsanji, 2004, citado en Li y Madnick, 2015, p. 105)

B

Base de datos: "*Inform*. Conjunto de datos organizado de tal modo que permita obtener con rapidez diversos tipos de información". (Real Academia Española, s.f.f, definición 1).

C

Computación en la nube: Es un modelo para permitir acceso omnipresente, conveniente y bajo demanda a través de la red a un conjunto de recursos configurables de computación compartidos. (Mell y Grance, 2011, p. 2)

D

Dato: "*Inform.* Información dispuesta de manera adecuada para su tratamiento por una computadora". (Real Academia Española, s.f.d, definición 3)

Digital: "Dicho de un dispositivo o sistema: que crea, presenta, transporta o almacena información mediante la combinación de bits". (Real Academia Española, s.f.e, definición 3).

I

Infraestructura de la nube: Es el conjunto de *hardware* y *software* que puede verse como contenedor de ambas capas: física: necesaria para soportar los servicios proveídos, y abstracción: del *software* desplegado en la capa física. (Mell y Grance, 2011, p. 2)

Infraestructura como Servicio (IaaS): El proveedor administra y entrega como servicio la infraestructura básica de los servidores, *software*, almacenamiento y equipo de redes de computadores. (Stein, Campitelli y Mezzio, 2020, p. 23)

Interfaz de programación de aplicaciones (API): Es un conjunto de herramientas, definiciones y protocolos que se utiliza para integrar los servicios y el *software* de aplicaciones. (Red Hat, 2018, párr. 2)

M

Microservicio: Es un componente desplegable independiente de un alcance compartido que soporta interoperabilidad a través de comunicación basada en mensajes. (Nadareishvili, Mitra, McLarty y Amundsen, 2016, p. 6)

Multi-inquilino: Modelo de *software* como servicio que soporta varios clientes con la misma combinación de infraestructura y capa de *software*, y que separa sus datos respectivos de manera lógica.

N

Nube Comunitaria: La infraestructura es aprovisionada para uso exclusivo de una comunidad específica de consumidores de organizaciones que tienen intereses comunes. (Mell y Grance, 2011, p. 3)

Nube Híbrida: Es una composición de dos o más infraestructuras de la nube distintas (privada, comunitaria, pública) que se mantienen como entidades únicas, pero están juntas debido a una tecnología propietaria o estandarizada que permite la portabilidad de datos y aplicaciones. (Mell y Grance, 2011, p. 3)

Nube Privada: La infraestructura es aprovisionada para uso exclusivo de una organización comprendiendo varios clientes (por ejemplo, unidades de negocio). Puede ser propiedad de, administrada y operada por: la misma organización, tercero o una combinación de ambos. (Mell y Grance, 2011, p. 3)

Nube Pública: La infraestructura es aprovisionada para uso abierto para el público en general. Puede ser propiedad de, administrada y operada por una organización: comercial, académica, gubernamental o alguna combinación de estas. (Mell y Grance, 2011, p. 3)

P

Plataforma como Servicio (PaaS): El proveedor entrega y administra la infraestructura, sistema operativo, herramientas de desarrollo y servicios, que el cliente usa para crear aplicaciones. (Stein, Campitelli y Mezzio, 2020, p. 23)

PYME: Todas aquellas micro y pequeñas empresas afiliadas al MEIC.

Protocolo de transferencia de Hipertexto (HTTP): Es una familia de protocolos sin estado, a nivel de capa de aplicación, solicitud/respuesta que comparten una interfaz genérica,

semántica extensa y mensajes autodescriptivos que permiten una interacción flexible con otros sistemas en la red. (IETF, 2022, párr. 9)

S

***Software*:** "m. *Inform*. Conjunto de programas, instrucciones y reglas informáticas para ejecutar ciertas tareas en una computadora". (Real Academia Española, s.f.c, definición 1)

***Software* como Servicio (SaaS):** El proveedor entrega una o más aplicaciones y todos los recursos (sistemas operativos y herramientas de programación) y la infraestructura que la soporta que el cliente puede consumir bajo demanda. (Stein, Campitelli y Mezzio, 2020, p. 23)

ANEXOS

Universidad Técnica Nacional

Anexo III

CARTA DE AUTORIZACIÓN PARA USO Y MANEJO DE LOS TRABAJOS FINALES

DE GRADUACIÓN UNIVERSIDAD TÉCNICA NACIONAL

(Trabajo Individual)

Ciudad, Alajuela, Costa Rica

Fecha. 31 de agosto de 2022

Señores/as

Vicerrectoría de Investigación. Sistema Integrado de Bibliotecas y Recursos Digitales

Estimados señores/as:

Yo Dilan alberto altamirano Cerda portador (a) de la cédula de identidad número 155802398030. En mi calidad de autor (a) del trabajo de graduación titulada:

Análisis del modelo de software como Servicio orientado a la micro y pequeña empresa costarricense de desarrollo de software, utilizando un entorno virtual en la nube

El cual se presenta bajo la modalidad de, marque una opción:

_____ Proyecto de Graduación

_X_ Tesis de Graduación

Presentado en la fecha DIA/MES/AÑO 19/08/2022, autorizo a la Universidad Técnica Nacional, sede Central, para que mi trabajo pueda ser manejado de la siguiente manera:

Autorizo
Ver capítulo V, disposiciones finales, artículo 41 (O aquel que refiera a derechos patrimoniales)

Marque con una X o un ✓	
Conservación de ejemplares para préstamo y consulta física en biblioteca.	✓
Inclusión en el catálogo digital del SIBIREDI (Cita catalográfica)	✓
Comunicación y divulgación a través del Repositorio Institucional	✓
Resumen (Describe en forma breve el contenido del documento)	✓
Consulta electrónica con texto protegido	✓
Descarga electrónica del documento en texto completo protegido	✓
Inclusión en bases de datos y sitios web que se encuentren en convenio con la Universidad Técnica Nacional contando con las mismas condiciones y limitaciones aquí establecidas.	✓
Divulgación del resumen en el Repositorio UTN, con una cantidad de 200 a 500 palabras	✓

Por otra parte, declaro que el trabajo que aquí presento es de plena autoría, es un esfuerzo realizado de forma personal, académica e intelectual con plenos elementos de originalidad y creatividad. Garantizo que no contiene citas, ni transcripciones de forma indebida que puedan devenir en plagio, pues se ha utilizado la normativa vigente de la American Psychological Association (APA). Las citas y transcripciones utilizadas se realizan en el marco de respeto a las obras de terceros. La responsabilidad directa en el diseño y presentación son de competencia exclusiva, por tanto, eximo de toda responsabilidad a la Universidad Técnica Nacional.

Consciente de que las autorizaciones no reprimen mis derechos patrimoniales como autor del trabajo. Confío en la que Universidad Técnica Nacional respete y haga respetar mis derechos de propiedad intelectual.

Firma del estudiante: _____

Cédula: 155802398030

Día: 31/08/2022