PONTIFICA UNIVERSIDAD CATÓLICA DEL ECUADOR SEDE ESMERALDAS

CARRERA:

INGENIERÍA DE SISTEMAS Y COMPUTACIÓN

PREVIO AL GRADO ACADÉMICO DE INGENIERÍA DE SISTEMAS Y COMPUTACIÓN

TEMA DE INVESTIGACIÓN:

INFRAESTRUCTURA O PLATAFORMA COMO SERVICIOS PARA DESPLIEGUES CONTINUOS DE APLICACIONES WEB EN BLAZOR

LÍNEA DE INVESTIGACIÓN:

AUTOMATISMOS Y APLICACIONES INTELIGENTES

PREVIO A LA OBTENCIÓN DE TITULO DE:

INGENIERÍA DE SISTEMAS Y COMPUTACIÓN

AUTOR

GRACIA GALLÓN CESAR STEVEN

ASESOR

WILSON CHANGO SAILEMA (Mgt.)

ESMERALDAS, 2021

TRIBUNAL DE GRADUACIÓN

Título: Infraestructura o Plataforma como Servicio para Despliegues Continuos de Aplicaciones web en Blazor

Autor(a): Cesar Steven Gracia Gallón

Mgt. Wilson Chango Sailema **Asesor**	f._____
Mgt. Evelin Lorena Flores **Lector #1**	f._____
Mgt. José Luis Carvajal **Lector #2**	f._____
Mgt. Susana Patiño Rosado **Coordinadora de carrera**	f._____

AUTORÍA

Yo, **Gracia Gallón Cesar Steeven** con número de cédula de identidad **0803262252** manifiesto que mediante la presente investigación sobre el tema **"INFRAESTRUCTURA O PLATAFORMA COMO SERVICIOS PARA DESPLIEGUES CONTINUOS DE APLICACIONES WEB EN BLAZOR"** los resultados obtenidos como tesis de grado, previo a la obtención del título de **"INGENIERO EN SISTEMAS Y COMPUTACIÓN"** son de total responsabilidad del autor, y que se ha respetado las fuentes de información consultadas, realizando las citas correspondientes y los resultados alcanzados son totalmente personales, únicos y legítimos. Al mismo tiempo declaro que todo el contenido incluyendo resultados, discusión, conclusiones, recomendaciones y otros efectos legales y académicos que se desglosan, son y serán exclusiva responsabilidad legal y académica del autor y de la PUCESE.

Gracia Gallón Cesar Steeven
0803262252

DEDICATORIA

Este estudio de investigación va dedicado para mi familia y a las personas que durante estos 6 años me han acompañado y me han ayudado en mi formación personal y profesional, por ayudarme a superar todos los retos, problemas y dificultades que se interpusieron en mi camino.

A mi madre Marjorie Janeth Gallón Banguera por ser una de las personas que siempre me apoyaron desde el inicio hasta el final de mi carrera, por todos compartir siempre esa calidez, por impartirme tus enseñanzas, por siempre aconsejarme en momentos claves o por ser una de las personas claves que me ayudaron a crecer como persona de bien, todo eso y mucho más.

A mi padrastro Jairo Fernando Diaz Góngora por haber actuado como un padre para mí y como mi hermano mayor en ocasiones, por siempre apoyarme y ayudarme en todo lo que necesitaba y darme las fuerzas de seguir adelante en cada momento, aun cuando había problemas entre los 2, él nunca me dejó solo, junto con mi madre siempre me apoyaron y creyeron en mi más que nadie.

A Jessica Daniela Betancourt Estrada por siempre estar presente en cada momento ya sea bueno o malo, por brindarme los valores, motivaciones y consejos que me permitieron convertirme en la persona que soy ahora. A pesar de ya llevar más de 9 años de tu triste partida, tus ideales, sueños y deseos, tu legado siempre vivirá dentro de mi corazón, sé que estarás conmigo en cada etapa de mi vida y tu recuerdo e ideales siempre estará ahí cuando más lo necesite, en donde sea que estés.

A mis hermanos, familiares y amigos con quienes he compartido todos estos años de dura travesía, por todos esos momentos buenos y malos, de alegría y tristeza, por ser parte de mi historia y mis motivos de seguir adelante

Cesar Gracia.

AGRADECIMIENTO

Le agradezco de todo corazón a las personas que siempre me apoyaron y que estuvieron a mi lado, que nunca me abandonaron, y que a pesar de todo siempre estuvieron dándome fuerzas a mi espíritu de avanzar cada vez más. A mis padres, mis hermanos, mis amigos y mejores amigos, a mis tías que siempre que podían me daban su cariño y apoyo y a las personas que fueron parte de mi vida.

Le agradezco de todo corazón al profesor Pablo Pico por ser una de las personas que, en el momento más crítico de mi vida, me apoyó y me ayudo de la forma más desinteresada, de no haber sido por él, no hubiera llegado hasta aquí, le debo mucho. Le agradezco a la vida y al destino por darme el placer de haber conocido a tan gran persona.

A todas aquellas personas que he conocido durante estos años, y que ahora tengo el placer de llamarlas mis amigos, profesores y docentes, empezando desde la escuela hasta la universidad que me inculcaron todos sus conocimientos tanto académicos como valores.

A mis mejores amigos José Carlos Muñoz y Alejandra Montenegro, que como si se tratara de hinchas apoyando al equipo de sus amores, siempre estuvieron ahí apoyándome, alentándome, ayudándome en lo que más fuera, sea lo más grande o lo más mínimo. Gracias por todas las experiencias buenas y malas que viví junto a ustedes, y por siempre brindarme de su calor en los momentos en el que el universo se volvía más frio e indiferente.

Índice de contenidos

Índice de Figuras

Índice de tablas

RESUMEN

La vida humana al igual que la computación y el desarrollo software han ido evolucionando constantemente con respecto al tiempo, servicios que ahora son ofertados por empresas como Google, YouTube, Spotify, Netflix entre otros, que antes eran desarrollados en entornos locales y físicos, eran muy costosos de mantener para las empresas que los desarrollaban. Ahora, la evolución de la computación en la nube, han migrado a entornos virtuales, en donde es más sencillo trabajar, los costes son menores y la conexión entre los servicios a los consumidores es más satisfactoria, cosa que se ha convertido ahora en una tendencia.

Gracias a esto, muchas personas y empresas que buscan incursionar en el mundo del desarrollo han optado por usar servicios de computación en la nube, pero no muchos saben elegir correctamente el modelo de servicio adecuado, o desconoce por completo qué modelo es el que mejor se adapta para su necesidad. Por tal motivo se planteó desarrollar una comparativa entre los modelos infraestructura y plataforma como servicio, con el objetivo de desplegar aplicaciones web y analizar su rendimiento en ambos modelos.

La investigación realizada es de tipo documental y experimental, se indagó y recolectó investigaciones y fichas técnicas para su análisis, además se diseñó una prueba experimental basada en pasos establecidos en las normativas estandarizadas ISO 25000.

Se evaluaron los resultados del análisis de rendimiento de las aplicaciones en los distintos modelos de servicio aplicando normativas ISO. Se pudo verificar la incidencia de los modelos en cada una de las aplicaciones en su despliegue, cuyos valores se verán reflejadas en tablas de resultados que serán útiles para la toma de decisiones de una persona al momento de optar por alguno de los servicios.

Palabras Claves: Computación en la nube, Infraestructura como servicio, Plataforma como Servicio, Normativas ISO, C#, Blazor Web Assembly, Desarrollo Web, Comparativa de Modelos.

ABSTRACT

Human life as well as computing and software development have been constantly evolving with respect to time, services that are now known as Google, YouTube, Spotify, Netflix among others, which were previously developed in local and physical environments, were very expensive to maintain for the companies that developed them. Now, the evolution of cloud computing has migrated to virtual environments, where it is easier to work, costs are lower and the connection between services to consumers is more satisfactory, which has now become a trend. .

Thanks to this, many people and companies seeking to enter the world of development have chosen to use cloud computing services, but not many know how to choose the right service model correctly or are completely unaware of which model is the best one. adapts for your need. For this reason, it was proposed to develop a comparison between the infrastructure and platform-as-a-service models, with the aim of deploying web applications and analyzing their performance in both models.

The research carried out is documentary and experimental, research and technical data sheets were investigated and collected for analysis, in addition to designing an experimental test based on steps established in the standardized ISO 25000 regulations.

The results of the application performance analysis in the different service models were evaluated by applying ISO standards. It was possible to verify the incidence of the models in each of the applications in its deployment, whose values will be reflected in tables of results that will be useful for a person's decision-making when choosing any of the services.

Keywords: Cloud Computing, Infrastructure as a Service, Platform as a Service, ISO Standards, C #, Blazor Web Assembly, Web Development, Model Comparison.

CAPITULO I: INTRODUCCIÓN

1.1 Presentación del problema

La humanidad al igual que la computación ha avanzado a pasos agigantados, en las últimas décadas la computación en la nube (o más conocido como Cloud Computing) se ha vuelto una tendencia para desarrolladores de aplicaciones en muchos ámbitos, ya sean para aplicaciones móviles, escritorio o web. Este tipo de computación ha permitido a muchos desarrolladores crear productos o servicios de manera más económica, y fácil, ofreciendo trabajar remotamente y ahorrarse sumas de dinero enormes en armar equipos para desarrollar proyectos.

Los servicios de computación en la nube son usados a diario por una gran cantidad de usuarios, cada vez más empresas de desarrollo utilizan estos servicios para contruir proyectos futuros e implementarlos en infraestructuras de tecnologías de la información. Es ahí en donde dos tipos de servicios de computación en la nube aparecen para el desarrollo de aplicaciones ya sean de escritorio, web o móviles, infraestructuras como servicio y plataformas como servicio.

Según Management Solutions [1], estos son modelos de servicios en la nube que se vienen ofreciendo desde hace tiempo atrás, una tendencia que de poco en poco se está volviendo realidad, con un crecimiento significativo tanto en la calidad de sus servicios, como la de sus consumidores, provocando el nacimiento de empresas dedicadas a la venta y alquiler de modelos de servicios en la nube.

Él presente trabajo de investigación propone comparar ambos modelos de servicio, resaltando las características de cada modelo para luego que luego se despliegue una aplicacion programada en el entorno de desarrollo Blazor para ver su rendimiento.

1.2 Planteamiento del problema

Gracias a la computación en la nube, muchos de los servicios tecnológicos ofrecidos en la actualidad como Google, Facebook, Twitter, YouTube, entre otros, han migrado a plataformas en la nube.

Con el pasar del tiempo estos modelos de servicio han conseguido una gran aceptación tanto por el público desarrollador como por múltiples empresas, siendo

incorporadas en ellas y teniendo un rol fundamental en las operaciones de estas, permitiéndoles abaratar costos y reducir los costes de compra de hardware y equipo especializado.

Según investigaciones realizadas por Management Solutions [1], son variadas las empresas a nivel mundial que ofrecen estos servicios, entre ellos los más importantes como son Microsoft Azure, Google Cloud, Amazon Web Services, International Business Machines y Salesforce.com, tal es el caso de Amazon WS [2], es considerado el tercer mayor proveedor de Infraestructura como Servicio con un 44.2%, mientras que con el modelo Plataforma como Servicio, Microsoft Azure y Amazon Web Services encabezan el mercado de este modelo a nivel mundial.

A pesar de la cantidad de proveedores de servicio en la nube las empresas desconocen cuál debe ser el modelo indicado para desarrollar proyectos de infraestructura o plataforma para sus clientes, ellos pueden o no saber elegir el modelo que se adapte a su realidad, desenvocando en una mala o buena inversion rentable a futuro.

1.3 Justificación

Los modelos de servicio en la nube son muy beneficiosos para el público y las empresas que necesiten de una infraestructura equipada, una de ellas es el trabajo remoto, pero algo que no comunican los proveedores de estos modelos de servicio es si el modelo que se está comprando, se adapta realmente a la necesidad que se busca cubrir.

Esto ocurre muy amenudo en el mercado de la nube, en donde se invierten grandes sumas de dinero, el público que alquila el servicio no esta muy informado sobre cuál de los 3 modelos de servicio es el mejor o cuál es el que mejor se adapta para una situación en específico.

Es por eso que la presente investigacion tiene como fin realizar una comparativa entre los dos modelos de servicio de computacion en la nube, infraestructuras y plataformas como servicio, en este caso para despliegues continuos de aplicaciones web desarrollados con el entorno de desarrollo Blazor.

1.4 Objetivos

1.4.1 Objetivos Generales

Evaluar el rendimiento de aplicaciones web en Blazor desplegadas en modelos de servicio Infraestructura o Plataforma, aplicando normativas ISO 25000 para medir y evaluar el rendimiento de las aplicaciones desplegadas en los modelos de servicio, para posteriormente comparar sus resultados.

1.4.2 Objetivos Específicos

- Investigar sobre los modelos de servicios de Infraestructura y Plataforma más famosos.
- Realizar una tabla comparativa entre los modelos de Infraestructura y Plataforma como servicio.
- Desarrollar y levantar aplicaciones web en Blazor en los modelos de servicio.
- Evaluar el rendimiento de las aplicaciones ya desplegadas.

CAPITULO II: MARCO TEÓRICO

2.1 Bases Conceptuales

2.1.1 Computación en la Nube

La computación en la nube es una abstracción de usuario con mira a las infraestructuras tecnológicas de la información que se dispongan, mientras estas estén conectadas a un proveedor de internet. Este modelo cumple cualquier necesidad de rendimiento, soporte y capacidad de tal forma que pueda ser totalmente modulable y escalable reduciendo el costo del uso.

Según Management Solutions [1], la Computación en la Nube (*Cloud Computing*, su traducción en inglés) es una tendencia que está cambiando el modo en el que las organizaciones diseñaban, armaban e incorporaban sus infraestructuras de Tecnologías de la Información (*TI*, por sus siglas en ingles). Este cambio ofrece un mundo de posibilidades para todas las personas involucradas en lo que es el desarrollo web, proveedores de aplicaciones y servicios, entre otros, sus trabajos impulsan a esta tendencia a seguir expandiéndose más y más. Para tener una idea un poco más clara véase la Figura 1.

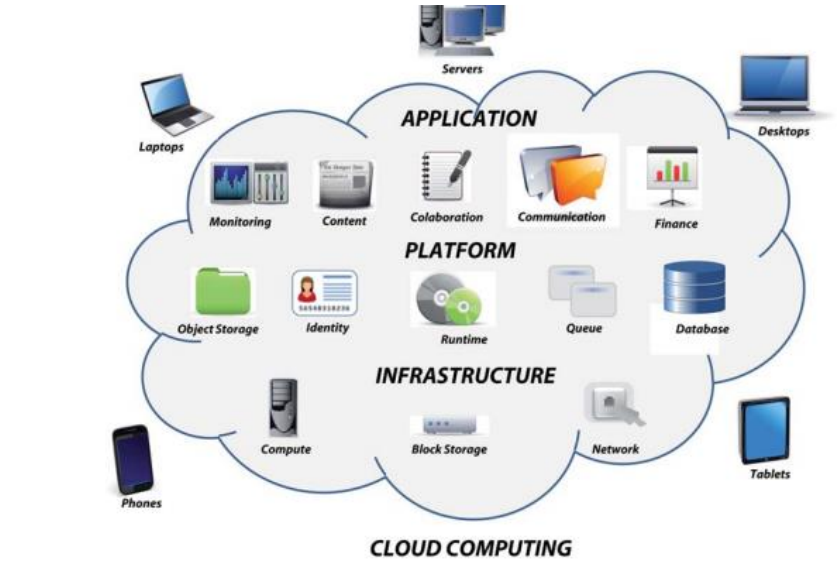

Figura 1 Computación en la Nube.

La tendencia de la computación en la nube en todo el tiempo que ha estado en auge ha dejado instalado tres modelos de servicios famoso, ahora son altamente

comercializados y bien vistos en la comunidad en general, esos son: Plataformas como Servicio (*PAAS*), Infraestructura como Servicio (*IAAS*) y Software como Servicio (*SAAS*). En la Figura 2 se visualiza los componentes que forman cada uno de estos modelos de servicio.

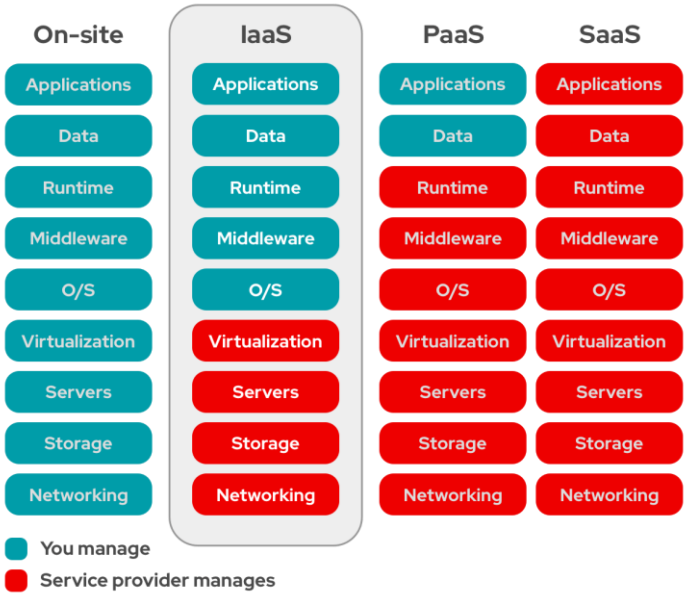

Figura 2 Modelos de Servicio de Computación en la Nube.

2.1.1.1 Modelos de Servicio de la Computación en la Nube

2.1.1.1.1 Plataformas como Servicio

Según en la investigación de Yasrab y Gu [4], informan que las Plataformas como Servicio (PaaS) son un segmento evolutivo y de aumento exponencial de la computación en la nube. Está revolucionando los futuros mercados de desarrollo de aplicaciones. Da recursos tecnológicos heterogéneos, con un trabajo de configuración mínimo que conduce a un veloz desarrollo de aplicaciones. No obstante, el mercado de la tecnología Plataforma como Servicio está bastante fragmentado en diferentes ecosistemas. En la actualidad, diferentes proveedores de servicios Plataforma como Servicios ofrecen recursos tecnológicos semejantes de forma distinto. Las Plataforma como Servicios adolece de una falta de estandarización que conduce a la pluralidad en las infraestructuras subyacentes, los instrumentos de programación y las habilidades accesibles.

Figura 3Composicion de PAAS.

Según El Instituto Nacional de Estándares y Tecnología (*NIST*, por sus siglas en inglés) [6] menciona que el modelo de servicio Plataforma como Servicio representa una solución de cliente final a grado de creador que tiene como primordial objetivo dar herramientas primordiales para hacer viable la creación de aplicaciones. además, da facilidades para que el inventor no deba instalar, configurar, conservar y gestionar recursos de infraestructura computacional como base de datos, servidores de aplicación, servidores de administración de Código fuente, entre diversos otros. Con esto, se pretende situar en producción diversos tipos de aplicaciones con estabilidad, escalabilidad, enorme volumen de almacenamiento y copias de resguardo.

Las plataformas como servicio en la piramide de la computacion en la nube se ubica a la mitad de esta por el simple hecho de que esta comparte cosas de ambas, se la podria considerar como un servicio híbrido, posee control de infraestructura hardware como una Interfaces como Servicio y software desempeñado como un Software como Servicio. Existen muchos ejemplos de este modelo de servicio, algunos de estos ejemplos se mostrarán en la Tabla 1:

Tabla 1 Ejemplos Plataformas como Servicio

TIPO DE SERVICIO	EJEMPLO
Plataformasde desarrollo	• Amazon Simple Queue Service (Amazon SQS) (Amazon Web Services, Amazon Simple Queue Service (Amazon SQS)), Amazon Simple Storage Service (Amazon S3) (Amazon Web Services, LLC) • Google App Engine (Google) • GRIDS Lab Aneka (Vecchiol, Chu, & Buyya, 2009).
Bases de datos	• Amazon SimpleDB (Amazon Web Services, Amazon SimpleDB) • Big Table (Chang, y otros, noviembre de 2006) • Microsoft SQL Azure Database (Microsoft).
Cola de mensajes	• Amazon Simple Queue Service (Amazon SQS) (Amazon Web Services, Amazon Simple Queue Service (Amazon SQS)).
Servidores de aplicaciones	• NetSuite Business Operating System (NS-BOS) (NetSuite, Inc.)

.

2.1.1.1.2 Infraestructura como Servicio

Según la información que brinda Management Solutions [1], menciona que las Infraestructuras como Servicios (*IAAS*) son un modelo de servicios el cual el comprador se le da un tamaño de almacenamiento específico fundamental como una secuencia de habilidades de cómputo en la red. Todo ello realizando uso de sistemas operativos que ya se hallan virtualizados o de servidores localizados en la nube a eso que los consumidores entran por medio de la red. La estructura del modelo de servicio se encuentra detallado en la Figura 4.

Figura 4 Composición IAAS.

Según el Instituto Nacional de Estándares y Tecnología [6] la capacidad que se otorga al consumidor es proveer computación, almacenamiento, redes y otros donde el cliente puede llevar a cabo un programa arbitrario, que puede integrar sistemas operativos y

aplicaciones. El consumidor no administra ni controla la infraestructura de nube subyacente.

Las infraestructuras como servicio en la computación en la nube serian lo que para nosotros son las bases de un edificio, de esta se sostienen las Plataformas como Servicio y los Software como Servicio, en las Tabla 2 y **¡Error! No se encuentra el origen de la referencia.** se mostrarán algunos ejemplos de Infraestructuras como Servicio:

Tabla 2 Ejemplos de Infraestructuras como Servicio.

TIPO DE SERVICIO	EJEMPLO
Procesamiento	• Amazon Elastic Compute Cloud (Amazon EC2) (Amazon Web Services, LLC) • Sun Network.com (Sun Grid) (SUN Microsystems, Inc.) • ElasticHost (ElasticHosts Ltd,) • Eucalyptus (Nurmi, y otros, 2009) • Nimbus (Alliance) • OpenNebula (Grupo de Arquitectura Distribuida), • Enomaly (Enomaly, Inc.).
Distribuciónde contenido a través de servidores virtuales	• Akamai (Technologies) • Amazon CloudFront Beta (Amazon Web Services,LLC).
Almacenamiento	• Amazon Simple Storage Service (Amazon S3) (Amazon Web Services, LLC) • Amazon SimpleDB (Amazon Web Services, Amazon SimpleDB) • Amazon Elastic Block Store (Amazon Web Services, Amazon Elastic Block Store (EBS)) • Microsoft SkyDrive (Microsoft Corporation) • Flicker (Flickr, LLC) • Youtube (YouTube, LLC) • Nirvanix Storage Delivery Network (Nirvanix) • Microsoft Live Mesh Beta (Microsoft Corporation, 2009)
Administraciónde sistemas	• Elastra (Elastra Corporation) • Engine Yard (Engine Yard, Inc.) • FlexiScalable (XCalibre Communications) • Grid Layer (Layered Technologies, Inc.) • Joyent (Joyent, Inc.) • Mosso (Rackspace, US Inc.) • Savvis Virtual Intelligent Hosting (Savvis, Inc.).
Administraciónde alojamiento	• Digital RealtyTrust (DigitalRealtyTrust, Inc.) • GoDaddy.com (GoDaddy.com, Inc.) • Layered Technology (Layered Technologies, Inc.).
Alojamiento autónomo	• Rackspace (Rackspace, US Inc.) • Savvis Virtual Intelligent Hosting (Savvis, Inc.) • Terremark Worldwide (Terremark Worldwide) • FlexiScalable (XCalibre Communications) • 1&1 Internet (1&1 Internet, Inc.).
VLAN (Virtual Local Area Network)	• CohesiveFT (Cohesive Flexible Technologies, Corp.).

2.1.1.1.3 Software como Servicio

Management Solutions [1] menciona que el software como servicio es un modelo de computación en la nube en el que el comprador se le da un programa al cliente donde solo puede manipular la parte final (el programa) y no su plataforma base o la infraestructura

que lo sostiene, dichos quedan alojados en el servidor del distribuidor del servicio en la nube teniendo acceso al comprador a ellos por medio de un navegador.

Figura 5 Modelo SAAS.

Según el Instituto Nacional de Estándares y Tecnología [6] la capacidad concedida al consumidor es usar las aplicaciones del abastecedor que se ejecutan en una infraestructura en la nube. El cliente no administra, no controla, no opera la infraestructura de la nube, incluida la red, los servidores, los sistemas operativos, el almacenamiento o inclusive el software propio del servicio, pero sí es capaz de utilizar las aplicaciones finales.

En la actualidad, el software como servicio es del tipo más utilizado por ser el más común, más barato y sencillo, no es tan complejo trabajar con ellos ya que la curva de dificultad que presentan no es elevada. En la actualidad hay una cantidad gigante de proveedores y de productos de software como servicio y en la Tabla 3 se les presentarán algunos ejemplos:

Tabla 3 Ejemplos de SAAS.

TIPO DE SERVICIO	EJEMPLO
Aplicacionescomo sitios Web	• Box.net (Box.net) • Microsoft Office Live (Microsoft) • Facebook (Facebook, Inc.) • LinkedIn (LinkedIn Corporation) • Twitter (Twitter, Inc.) • MySpace (MySpace.com) • Zillow (Zillow.com) • Google Maps (Google).
Colaboración y aplicaciones de oficina	• Cisco WebEx Weboffice (Cisco Systems, Inc.) • Google Docs (Google) • Google Talk (Google) • IBM BlueHouse (IBM, Corp.) • Microsoft Exchange Online (Microsoft) • RightNow (RightNow Technologies, Inc.) • Gmail (Google) • Microsoft Hotmail (Microsoft Hotmail) • Yahoo! Mail (Yahoo! Inc.).
Servicios de pago	• Amazon Flexible Payments Service (Amazon FPS) • (Amazon Web Services, LLC) • Amazon DevPay (Amazon Web Services, LLC).
Software basado en Web integrable a otras aplicaciones	• Flickr Application Programming Interface (API) (Flickr, LLC) • Google Calendar API (Google) • Saleforce.com´s AppExchange (Salesforce.com, Inc.) • Yahoo! Maps API (Yahoo! Inc.) • Zembly (Sun Microsystems, Inc.).

2.1.1.2 Ventajas de la Computación en la Nube

- Acceso a datos con suma facilidad desde cualquier parte del planeta.
- Alivianar la carga de datos almacenados en el disco duro de nuestro ordenador.
- Económico, se puede ahorrar gastos de mantenimiento y servicio.
- Permite adquirir más servicios o expandir la capacidad de almacenamiento.
- Mantiene los datos en su versión más pura y disponible a toda hora.
- Puede cancelar en función del servicio o tiempo que se requiera consumir.
- No requiere de consumo de recursos informáticos del cliente.
- No requiere de espacio físico amplio para poder funcionar.
- Promueve la creación y aprovechamiento al máximo de nuevas tecnologías, a si también como las impulsa.

2.1.1.3 Desventajas de la Computación en la Nube

- Privacidad de datos a medias.
- Dependencia de la infraestructura y de las plataformas que lo mantiene.

- Sin conexión a internet no se puede acceder a los sitios donde se almacena la información.
- Genera dependencia hacia los proveedores del servicio.
- Existe la posibilidad de pérdida de información valiosa a causa del volumen de información que se comparte en la nube.
- La latencia de los servidores o de tu proveedor de internet puede ser un factor molesto.
- En la mayoría de los países del mundo no existe leyes de protección de datos, privacidad y responsabilidad por violación de derecho informáticos, contenidos y licencias.

2.1.2 Frameworks de desarrollo de software

Para Dinarle [9] un Framework (en español entorno de desarrollo) de desarrollo es un esquema de reutilización del programa formado por elementos e interrelaciones en medio de éstos, ejemplificando: la abstracción de clases, objetos o elementos que la componen; además, provee diferentes elementos de conexión a base de datos, como controladores para conexión directa (*MySQL, SQL Server, Oracle*) o de forma general, por medio del estándar ODBC (*Open DataBase Connectivity* por sus siglas en ingles).

Las razones por las que en la actualidad se usan más los entornos de desarrollo a la hora de programar son variadas, una de ellas es que se evita la repetición de código, algo que es muy común en el desarrollo de software. Otra razón más es la utilización de buenas prácticas, los entornos de desarrollo al basarse en patrones (comúnmente MVC) ayudan a separar y organizar los datos, la lógica de negocio y la interfaz de usuario (*UI* por sus siglas en inglés) y la velocidad con la que se puede desarrollar proyectos de formas limpia y segura.

Existen múltiples entornos de desarrollo, pero se tienen que tomar en cuenta algunos aspectos al momento de elegir uno, estos aspectos son los siguientes:

- **Documentación**: La documentación de un entorno de desarrollo es una variable muy importante, tiene que ser actualizada, amplia, legible, sencilla y que pueda resolver los problemas que puedan ocurrir en el transcurso del desarrollo. La comunidad y documentación es algo que va de la mano.

- **Soporte de la comunidad y la desarrolladora**: Un factor importante al elegir un entorno de desarrollo es la comunidad, si la comunidad es participativa, amigable y activa, será más sencillo conseguir ayuda o soporte en momentos de dificultad en el desarrollo de proyectos.

- **Simplicidad y potencia:** Los entornos de desarrollo deben ser sencillos de utilizar, la mayoría de ellos son potentes, pero difíciles de entender. El código que se genere por el Framework debe ser limpio y claro, pero que no tengas la preocupación por detalles internos.
- **Arquitectura:** La arquitectura en un entorno de desarrollo es muy importante, ayuda a tener separado la parte lógica con la de negocios, datos y la de interfaz gráfica. De esta forma, el código estará más ordenado y se hará más sencillo darle mantenimiento, entenderlo o actualizarlo.
- **Reutilización:** Es vital que los elementos que se desarrollen puedan ser reutilizados en proyectos futuros, esto con el fin de evitar repetir la generación de código.
- **Estilos de Patrón:** Es importante poseer una clase que se encargue de realizar operaciones que se modifiquen y se consulten a una determinada tabla de una base de datos en específico. De esta forma el proyecto quedará aislado del sector de SQL.
- **Posicionamiento:** Esto ya depende del proyecto y su visión, puede que no sea de interés, pero si no es así, es de mucha importancia que el entorno de desarrollo ofrezca algunas funciones amigables como las de URL.
- **Seguridad:** Es el factor más importante, uno debe adaptar el entorno de desarrollo a las necesidades que se tenga del proyecto.

Para el desarrollo de este proyecto se terminó eligiendo Blazor.

2.1.2.1 Blazor

Blazor es una implementación propia Microsoft.NET, es un entorno de desarrollo para diseñar páginas web bajo Aplicación de Pagina Única (SPA, según sus siglas en inglés) utilizando C# como lenguaje base, Razor page y Lenguaje de Marcas de Hipertexto que se ejecutan en el navegador. Esto habilita una nueva generación de programadores Full

Stack bajo una plataforma estable y de alta productividad como es NET. Blazor, provee beneficios de un entorno de desarrollo de UI en el lado de cliente y servidor que se describirán a continuación:

- Modelo de componentes UI.
- Routing.
- Layout.
- Formularios y validación.
- Inyección de dependencias.
- Interoperabilidad con JavaScript.
- Renderizado de UI en servidor.
- Live-reloading durante el desarrollo.

2.1.2.1.1 Componentes

Como menciona Microsoft [10] en su manual las aplicaciones de Blazor se fundamentan en elementos. En Blazor, un elemento es un componente de la interfaz de cliente, como una página, un cuadro de diálogo o un formulario de ingreso de datos. Los componentes trabajan de la siguiente forma:

- Definen la lógica de representación de la interfaz de usuario.
- Controlan acciones del usuario.
- Pueden anidar y reutilizar secciones específicas de código fuente.
- Pueden compartir y distribuir bibliotecas de clases de Razor o paquetes NuGet.

La clase del componente comúnmente se redacta a modo de página de marcado de Razor con la expansión de nombre de documento ".razor". Formalmente se refiere a los elementos de Blazor como elementos de Razor. Razor es una sintaxis para combinar marcado HTML con código de C# diseñada para incrementar la productividad del programador.

2.1.2.1.2 Blazor WebAssembly

Este es un marco de software desarrollado por Microsoft para páginas exclusivas de Aplicaciones de Pagina Única orientado para Blazor con el fin de ejecutar software web de tipo interactiva con el cliente apuntando hacia .NET. Este mismo usa estándares web

de tipo abierto, pero sin la necesidad de usar accesorios u otras compilaciones de código en otros lenguajes. WebAssembly opera en todos los navegadores web actuales, integrados a sus contrapartes móviles.

Microsoft [10] explica que WebAssembly es un estándar web abierto y se admite en navegadores web sin accesorios, por lo que la ejecución de .NET en navegadores web se consigue por medio de WebAssembly; es un formato de código de bytes compacto y optimizado para descargas rápidas y ligeras en su ejecución máxima, en la Figura 6 se puede apreciar el funcionamiento estructural de WebAssembly.

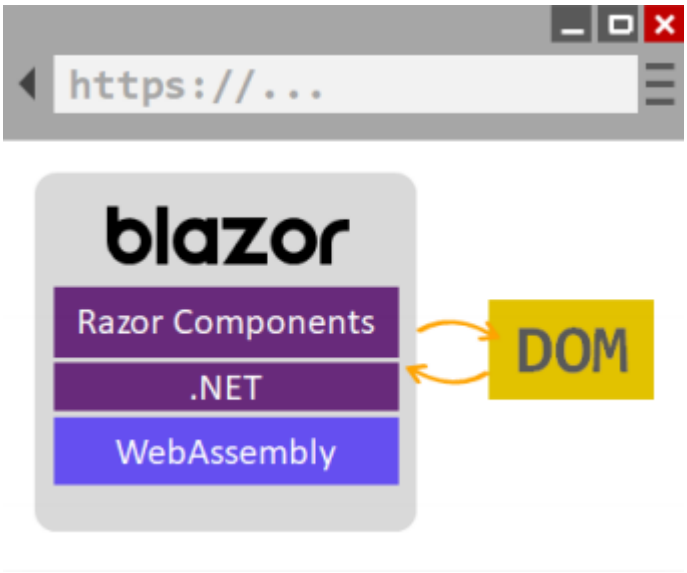

Figura 6 Blazor WebAssembly.

2.1.2.1.3 BlazorServer

Microsoft [10] explica que Blazor separa la lógica de representación de elementos del modo que se utilizan las actualizaciones de la interfaz de cliente. Blazor Server da compatibilidad con el hospedaje de elementos de Razor en el servidor de una aplicación ASP.NET Core. Blazor Server opera de la siguiente forma:

- Ejecución del código C# de la aplicación.
- El envío de eventos de UI desde el explorador al servidor.
- La aplicación de actualizaciones de la UI al componente representado que retorna al servidor.

En la Figura 7 se muestra un ejemplo de cómo es la comunicación entre ASP.NET Core con el Modelo de objetos de documento (DOM, según sus siglas en inglés) del Navegador.

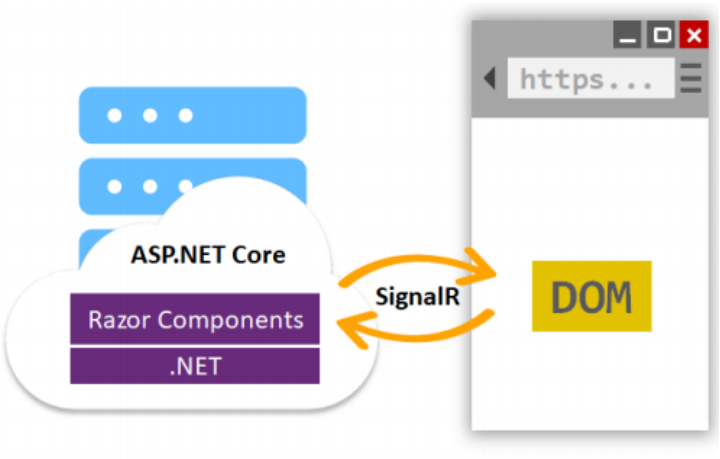

Figura 7 BlazorServer.

2.1.2.1.4 Interoperabilidad de JavaScript

Microsoft [10] explica que en el caso de aplicaciones que necesiten bibliotecas JavaScript de terceros y acceso a una Interfaz de programación de aplicaciones (API, según sus siglas en inglés) de explorador, los componentes interoperan con JavaScript. Los componentes podrán usar cualquier biblioteca, API que pueda utilizar JavaScript. El código de C# podrá inicializar el código JavaScript y, a su vez el código de JavaScript inicializará al código de C#.

2.1.2.1.5 Uso compartido de código y .NETStandard

Microsoft [10] explica que Blazor implementa .NETStandard, que permite a proyectos de Blazor hacer referencias a las bibliotecas que cumplen las especificaciones de ese estándar. .NETStandard es una especificación formal de Apis de .NET comunes entre implementaciones de .NET. Las bibliotecas de clase .NETStandard podrán compartir sea través de diferentes plataformas .NET, como Blazor, .NET Framework, .NET Core, Xamarin, Mono y Unity.

2.1.2.1.6 Plataformas admitidas de Blazor de ASP.NET Core

Blazor WebAssembly y BlazorServer admiten los siguientes exploradores que se mostraran en la siguientes Tabla 4, Tabla 5 y Tabla 6:

Tabla 4 Navegadores Admitidos por los Servicios de Blazor.

EXPLORADOR	VERSIÓN
Apple Safari, incluido iOS	Current†
Google Chrome, incluido Android	Current†
Microsoft Edge	Current†
Mozilla Firefox	Current†

Tabla 5 Navegadores con Blazor WebAssembly.

EXPLORADOR	VERSIÓN
Apple Safari, incluido iOS	Current†
Google Chrome, incluido Android	Current†
Microsoft Edge	Current†
Microsoft Internet Explorer	No admitidas‡
Mozilla Firefox	Current†

Tabla 6 Navegadores con BlazorServer.

EXPLORADOR	VERSIÓN
Apple Safari, incluido iOS	Current†
Google Chrome, incluido Android	Current†
Microsoft Edge	Current†
Microsoft Internet Explorer	11‡
Mozilla Firefox	Current†

2.1.2.2 ReactJS

React.JS es una librería de JavaScript de código abierto (*open-source* por su traducción en inglés) moldeada para desarrollar entornos de usuario con el fin de agilitar la programación y diseño de aplicaciones en una sola ventana.

React.JS ayuda a desarrolladores a elaborar aplicaciones que usan datos que cambien de forma constante. Su objetivo es ser sencillo y legible, es declarativo y homologable. React.JS es la Vista en ambiente que puede usar el patrón Modelo-Vista-Controlador o Modelo-vista-Modelo de Vista.

Por consiguiente, React.JS reproduce un control sólido donde se puede edificar y ejecutar cualquier proyecto con JavaScript. Además, ofrece muchas variantes al desarrollo, debido a que da muchas cosas ya listas, en las que no se requiere invertir tiempo de trabajo. Sirve para desarrollar aplicaciones web de una forma más ordenada, con menos código reutilizado, usando JavaScript puro o librerías como jQuery centradas en la manipulación del DOM. Facilita a que las vistas se asocien con los datos, por lo cual, si cambian los datos, además cambian las vistas. En la Figura 8 se muestran los servicios y ventajas que ofrece ReactJS.

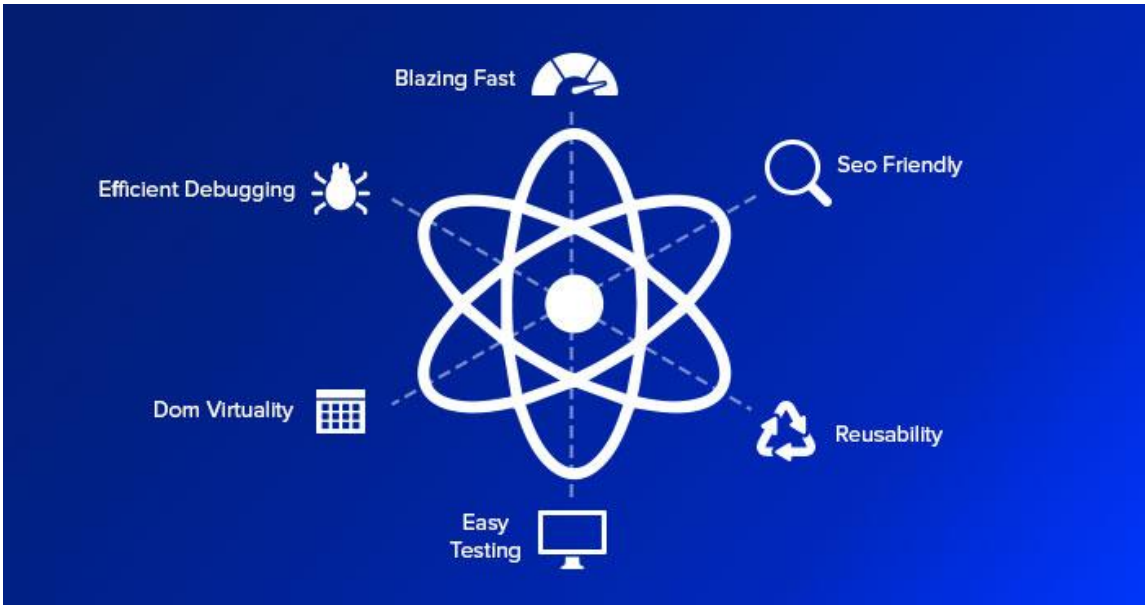

Figura 8 React.JS y sus funcionalidades.

2.1.2.2.1 ¿Qué es el Virtual DOM?

Tiene como función desarrollar software para navegadores web de una manera más jerárquica y limpia, con menos código reutilizado, utilizando el lenguaje de programación JavaScript en un estado muy puro y con librerías como jQuery que apuntan a la alteración

y control del DOM. Es una de las primordiales propiedades de React.JS. En la actualidad, se puede comentar que el virtual DOM es una emulación del DOM.

El Virtual DOM está desarrollado en una idea bastante simple, pero hábil. Cuando una vista se actualiza, React.JS actualiza DOM Virtual, tarda muy poquito tiempo para actualizar DOM de navegador. Es algo muy vital y transparente, cabe recalcar que no es necesaria la intervención del programador para que la ejecución de la aplicación alcance su máximo rendimiento, para ver cómo actúa Virtual Dom revise la Figura 9.

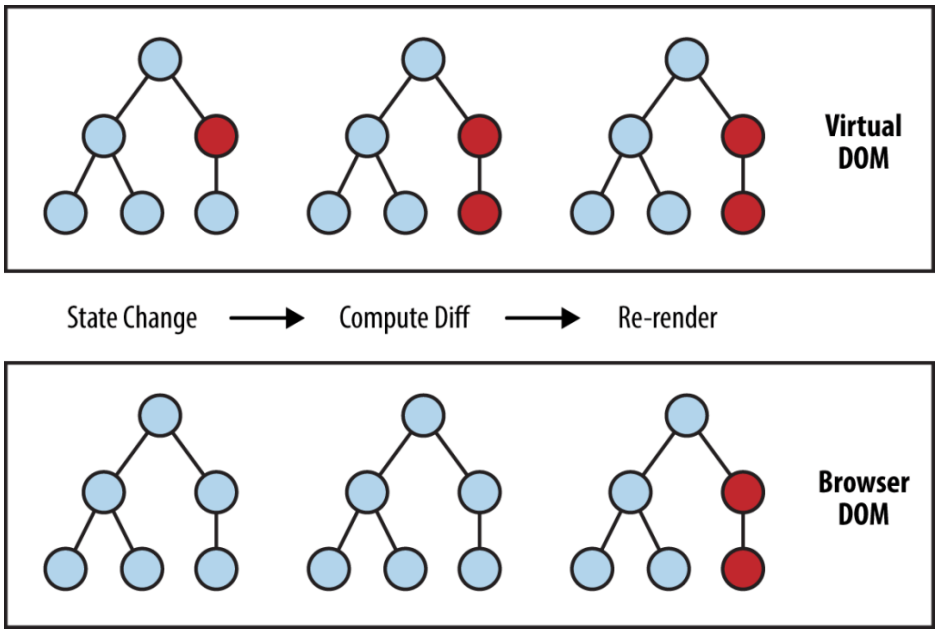

Figura 9 Ejecución de Virtual Dom.

2.1.2.2.2 React es isomórfico

Con el mismo código que trabaja, renderiza a HTML tanto en el servidor como en el cliente, rebajando la carga de trabajo necesaria para realizar aplicaciones web para navegadores.

La inquietud de las aplicaciones desarrolladas en el lenguaje JavaScript es que en muchas ocasiones reciben datos en estado puro de parte del proveedor, o de alguna API que este enlazada a un archivo de formato de Notación de objetos JavaScript (JSON, según sus siglas en inglés). Las librerías de JavaScript y entornos de desarrollo toman estos datos para realizarlo en HTML que debe desempeñarse al navegador. Esta arquitectura

representa la solución más idónea para el proceso de programa web, debido a que permite desarmar el desarrollo apuntando al servidor y el cambio del lado del cliente.

2.1.2.2.3 Ecosistema de React.JS

Como desarrolladores, programadores y diseñadores se puede elegir entre una gama variada de entornos de desarrollo encargados del transporte de datos. Por tanto, antes de programar algo en React.JS es conveniente ver si otro programador ya ha publicado un componente que realice su misma función o en la mayoría de la función.

2.1.2.3 VueJS

En 2017, Grandes [12] en su investigación explica que Vue.JS está diseñado con una incorporación muy ligera del patrón Vista-Modelo que mejora la relación entre la capa de presentación con la capa de negocio de forma eficiente. Vue.JS es un entorno de trabajo muy fácil de incorporar a cualquier proyecto existente y aprovecha al máximo sus funcionalidades casi sin problemas. Al contrario que otros entornos de desarrollo como Angular que son más famosos y que apuntan iniciar proyectos desde cero, y con una arquitectura por defecto que va dirigido por el entorno de desarrollo que eligió.

2.1.2.3.1 Introducción al patrón Vista-Modelo

En 2017, Grandes [12], menciona que el patrón Vista-Modelo o Modelo-Vista-Vista-Modelo tiene como función agregar una capa de lógica entre nuestra capa de negocio y nuestra capa de interfaz de usuario, para evitar que una tenga que comunicarse directamente con otra: cuando la vista o el modelo se modifiquen, el Vista-Modelo se encargará de actualizar su contrapartida en la otra capa de forma automática. Facilita el proceso de pruebas y permite crear, diseñar y desarrollar software más moldeables y reutilizables, además evita una gran cantidad de código 'boilerplate' que se actualiza a mano.

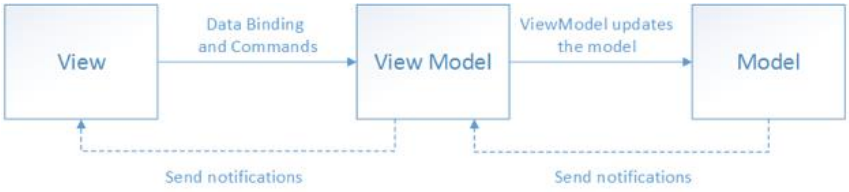

Figura 10 Patrón MVVM.

2.1.2.3.2 Propiedades de una instancia de Vue.JS

- **El:**

  Determinan el tag HTML sobre los que funcionan nuestra instancia Vue.JS. Es un selector CSS como los que se utilizan en las jQuery.

- **Template:**

  Indica una cadena con HTML que reemplazará el HTML que se ha fijado como El.

- **Data:**

  Son objetos con propiedades que se pueden utilizar en una o varias instancias. Tanto la directiva v-bind como en el método área () refieren estas propiedades. También se utiliza entre corchetes en el template.

- **Computed:**

  Son objetos que contienen una serie de métodos que retornan valores ya calculados a partir de las propiedades del objeto.

- **Methods:**

  Son objetos con funciones que pueden llamar instancias, y cuyo contexto está enlazado con el objeto.

- **Hooks:**

  Funciones que se inicializan en momentos específicos del ciclo de vida de una instancia, se puede visualizar la estructura de código en la Figura 11.

```
1   new Vue({
2       el : '#app',
3       template : `<div v-bind:style="estilos" v-on:click="log()"`,
4       data : { titulo : 'Rectángulo', largo : 100, alto : 40 },
5       computed : {
6           area : function(){ return this.largo * this.alto; } ,
7           estilos : function(){
8           return {
9           width : this.largo + 'px',
10          height : this.alto + 'px'
11        }
12
13       },
14      methods : { log : function(){ console.log('Soy un rectángulo con un area de ' + this.area + ' pixels' ); } },
        created : function(){ console.log('se crea la instancia: todavía no se ha reemplazado "#app" por el template de esta
15  instancia'); },
        mounted : function(){ console.log('ya se ha renderizado el template en el DOM'); }
16
17  });
18
```

Figura 11 Código en VueJS con algunas instancias.

2.1.2.3.3 Componentes

Los componentes para Grandes [12] son HTML configurables, adaptables y reutilizables, permiten definir etiquetas que se pueden usar en markup dotados de una API y

comportamiento especifico. La estructura de los componentes no es similar a la de los custom elements, es posible transformarVue.JS “components” a “custom elements” con simpleza mediante “vue-custom-elements”.

La API más rápida de Vue.JS es un constructor, que usando instancias simples de Vue.JS no se podrá levantar con facilidad las instancias dentro de otras instancias, ni volver a usar código para casos similares. En la investigación Grandes [12] explica que los componentes son básicamente instancia Vue.JS que se usará dentro de otra, y que será llamada por medio de una etiqueta específica. Los componentes son compatibles, pueden agregar otros componentes dentro de componentes, y tienen orden jerárquico: los padres pueden modificar los datos de los hijos, pero no de forma al revés. Los hijos pueden únicamente notificar eventos a los padres, pero no modificar su estado. Esto puede ser una verdadera molestia, pero mejora el workflow y arregla varios bugs, además que tiene que asegurar cuando se modifiquen los modelos se tenga un punto que gestione la entrada.

2.1.2.3.4 Ecosistema

En 2017, Grandes [12] menciona que el núcleo de Vue.JS está diseñado para software medianos a pequeños, y específicamente para software web, donde cada página funciona por separado desde el back-end. Para software de aplicaciones únicas de página, es necesario usar una herramienta para gestionar rutas y la situación de estado de la aplicación. También, el ecosistema de Vue.JS existen sin número de herramientas variadas para utilizar webpack o browserify al momento de armar un software basado en vue.js.

En 2017, Grandes [12] explica que Vue.JS en un inicio presentaba una librería oficial para realizar peticiones http, vue-resource, pero fue quitada por generar bugs con el núcleo de Vue.JS, por tanto, tiene sentido poseer un banco de herramientas para Vue.JS, ese no es el caso de un cliente HTTP.

2.2 Antecedentes

Para la recopilación de la información de la presente investigación, se utilizaron distintas herramientas bibliográficas como bibliotecas virtuales, repositorios universitarios, revistas científicas, documentación técnica e información certificada por el Instituto Nacional de Estándares y Tecnología estadounidense (por sus siglas en inglés NIST), referentes al tema de la computación en la nube y a sus múltiples infraestructuras.

Se emplearon 2 métodos de búsqueda de la información, su utilización fue escalonada, una metodología de búsqueda y otro de selección y filtrado de la misma. La primera modalidad de búsqueda se basa en un protocolo de búsqueda científica utilizando una cadena de búsqueda con palabras claves, en este caso como la investigación trató sobre la computación en la nube y sobre las infraestructuras IAAS y PAAS, las cadenas utilizadas fueron "(("Document Title": "PAAS") AND ("Document Title": "PLATFORM AS A SERVICES" OR "Document Title": "CLOUD COMPUTING"))" y "(("Document Title": "IAAS") AND ("Document Title":" INFRASTRUCTURE AS A SERVICE" OR "Document Title": "CLOUD COMPUTING"))".

En la recopilación de la información del presente estudio se aplicó una restricción en el tiempo de publicación, con el fin de obtener información de los últimos 7 años (2014 hasta el presente año 2021), esto con el fin de identificar y obtener antecedentes actuales. De la totalidad de todos los artículos recabados en esta búsqueda y recolección, se seleccionaron 8 artículos que cumplen los siguientes 2 criterios, modelo de servicio, y el alcance de la investigación.

Como primer estudio propuesto se tiene el desarrollado por Mannella Lemos [7], "*Diseño de una guía para la implementación del uso de computación en la nube como mecanismo de recuperación ante desastres tecnológicos en PYMES en un DMQ*". Gracias a los resultados obtenidos de este estudio es posible la simulación de un entorno de pruebas de este tipo para ver el rendimiento de servicios y su comportamiento con información de alto valor. La autora de este estudio propone que la información almacenada en el modelo de preferencia descentralizada, en caso de un ataque esta puede estar en otro lugar que no es el que atacaron y como último aporte, los respaldos de información deben ser masivos y semanales en el mejor de los casos.

A continuación, el segundo estudio propuesto elaborado por Xia *et al.*[15], esta investigación apunta a un "*Sistema de predicción de estado de una máquina física en un entorno de nube IaaS basado en el proceso Hidden Markov*", predice el estado y el tiempo en que la máquina física pasa por una sobrecarga, servirá como guía para la programación de recursos en la nube Infraestructura como Servicio. Los autores de este estudio proponen utilizar el proceso de Hidden Markov para poder predecir el estado y el tiempo de las máquinas físicas de los servicios Infraestructura como Servicio.

Las 2 investigaciones mencionadas tratan temáticas de ambientación y predicción de rendimiento, temas de interés para la investigación pero que se podrían complementar muy bien, al integrarle la teoría de Hidrkov que plantea en la investigación de Sistema de predicción.

A continuación, el tercer estudio propuesto elaborado por Li *et al.* [16], donde plantean una interesante investigación sobre los costos de "*Competencia de precios en un mercado de la nube de Duopolio IaaS*", en donde el precio de los servicios pueden ser muy competitivos pero a la vez muy caros. Los valores se pueden predecir gracias a una teoría que plantean llamada "teoría del juego del mercado", en donde alterando los precios de los CSP se puede controlar el precio de estos servicios, y en caso de no ser controlarlos se puede predecir su precio. Se ha decidido tomar esta investigación con el fin de poder comprender esta teoría y así saber elegir el momento indicado para comprar estos servicios, con el fin de optimizar recursos económicos.

El cuarto estudio propuesto fue elaborado e investigado por Kiwon y Kwangseob [17], ellos implementaron un "*Sistema de análisis de imagen basado en GEO que soporta el estándar OGC-WPS en la plataforma Open PAAS Cloud*", esta investigación plantea incorporar un Servicio de Procesamiento de Web (WPS) con un estándar de un Sistema de Información Georreferencial abierto compatible con un procesamiento de información asincrónico en línea donde por medio de capturas satelitales de grandes volúmenes puede localizar la posición geográfica de lo que se desee, todo esto almacenado en una nube de Plataforma como Servicio y utilizando los recursos propios de WPS. Sus autores recomiendan utilizar geoprocesamiento para capturas nítidas y de alta calidad de capturas satelitales además de proteger el sistema con múltiples capas de seguridad, lo cual es significativo a tener un mayor nivel de seguridad informática ante ataques, viendo que en la documentación comentan que al momento de accionarse este proyecto ocurrieron varios ataques al servidor padre haciendo caer el sistema.

El quinto estudio propuesto fue diseñado por Akinbi y Pereira [18], esta investigación se enfoca en el "*Mapeo de requisitos de seguridad para identificar áreas críticas de seguridad de enfoque en modelos de nube PaaS*", mencionan que el objetivo fue permitir a los clientes la capacidad de cuantificar los requisitos mínimos de seguridad de su servicio para identificar áreas críticas en las arquitecturas de nube Plataforma como

Servicio ofrecidos por los CSP. Con el uso de una seguridad adaptativa de matriz de mapeo, el estudio utilizó un enfoque cuantitativo para presentar hallazgos de datos numéricos que muestran las áreas críticas de una arquitectura dentro del entorno Plataforma como Servicio. La recomendación que dan los autores de este artículo es aplicar este método de mapeo para poder verificar el nivel de seguridad final que posee los modelos de servicio.

En 2016, Gesvindr y Buhnova [19], presentaron su investigación "*Tácticas arquitectónicas para el diseño de aplicaciones eficientes de PaaS en la nube*", por medio de sus experiencia con el diseño e implementación de aplicaciones en la nube Plataforma como Servicio altamente escalables, resultó en una comprensión profunda de las debilidades y beneficios de este modelo de servicio, ayuda a identificar y evaluar una serie de tácticas arquitectónicas para aplicaciones en la nube Plataforma como Servicio.

Un trabajo similar al anterior realizado por Gesvindr *et al*. [20], diseñan un "*Controlador de calidad de aplicaciones PaaS en la nube usando prototipos generados*" con el objetivo de mitigar el riesgo de cambios arquitectónicos posteriores debido a una infracción de los requisitos de calidad (como rendimiento, tiempo de respuesta y escalabilidad).

El último estudio de investigación propuesto por Gesvindr y Buhnova [21], diseñan una "*Herramienta de evaluación de calidad de aplicaciones usando PaaSArch*", tiene como objetivo analizar la calidad y rendimiento de la aplicación usando prototipos propios generados por PaaSArch (por sus siglas en ingles Plataforma como Servicio de búsqueda) para luego ser sintetizados y vueltos a analizar por un analizador léxico propio de Plataforma como Servicio incorporado a PaaSArch. Lo que aporta esta investigación es la herramienta llamada PaaSArch que permite realizar control de calidad a aplicaciones cargadas a la plataforma.

Las tres últimas investigaciones presentan un tema en común muy interesante, control de calidad usando Plataforma como Servicio o herramientas propias en Plataforma como Servicio, algo que será de suma importancia en la investigación por el hecho que se probará aplicaciones de entornos de desarrollo en los modelos Infraestructura como Servicio y Plataforma como Servicio. Las investigaciones desde un punto de vista crítico son las mismas, pero con ciertos cambios, mientras que la octava investigación basa su

control usando PaaSArch, la séptima investigación usando prototipos generados por el mismo controlador que usa y la diferencia entre estas dos con la sexta es la utilización de una técnica de arquitectura.

En base a los antecedentes ya presentados, tres de las ocho investigaciones basan sus investigaciones en el rendimiento de las Infraestructuras como Servicios y sobre su operabilidad en algunos entornos de trabajo y funciones asignadas, una investigación de las ocho se enfoca en la predicción del valor de costo de los servicios de Infraestructura como Servicio, esta investigación tendrá mucho interés en el presente trabajo investigativo por la razón de predecir el precio futuro de los modelos de servicio Infraestructura como Servicio. Mientras que las últimas 4 investigaciones se enfocan mucho en el control de calidad de aplicaciones usando algunos servicios y componentes Plataforma como Servicio, tendrán también algo de atención, ya que al igual que la investigación de predicción de valor de costo de Infraestructura como Servicio, el control de calidad en Plataforma como Servicio podría ser una variable para tomar en cuenta al momento de decidir por uno de los servicios. La factibilidad de implementar algún estudio de la investigación es alta, todos cumplen las rúbricas ya mencionadas y aportan más información, técnicas y modelos al trabajo como tal, especialmente aquellas que hablan sobre el control de calidad en aplicaciones y el control predictivo de estado físico de las Infraestructura como Servicio.

2.3 Fundamentación Legal

Dentro de las bases legales, normativas y legislaciones vigentes hasta la fecha en el Ecuador, se deben tomar muy en cuenta las siguientes leyes y códigos penales para el desarrollo del presente proyecto de investigación: Ley Orgánica de Protección de Datos Personales [22], Código Orgánico de la Economía Social de los Conocimientos, Creatividad e Innovación [23] y Ley Orgánica de Emprendimiento e Innovación[24].

En la Ley Orgánica de Protección de Datos Personales por medio del capítulo dos titulado “Principios”, basado en el Art.9 determina que: Los datos personales deben tratarse con estricto apego y cumplimiento a los principios, derechos y obligaciones establecidas en la Constitución, los instrumentos internacionales, la presente Ley, su reglamento y la demás normativa y jurisprudencia aplicable. En ningún caso los datos personales podrán ser tratados a través de medios o para fines ilícitos o desleales. Las

relaciones derivadas del tratamiento de datos personales deben ser trasparentes y se rigen en función de las disposiciones contenidas en la presente Ley, su reglamento y demás normativa atinente a la materia[22]. La ley indica que los datos de usuarios que se vayan a almacenar en el desarrollo de este proyecto deben respetarse y mantenerse bajo estrictas normativas de seguridad.

En el Código Orgánico de la Economía Social de los Conocimientos, Creatividad e Innovación del libro 3 titulado "De la Gestión de los Conocimientos" en el capítulo número 2 del Apartado Segundo cuyo nombre "De las tecnologías libres y formatos abiertos", Art.142 menciona que: "Se entiende por tecnologías libres al software de código abierto, los estándares abiertos, los contenidos y el hardware libres. Los tres primeros son considerados como Tecnologías Digitales Libres" [23]. La ley plantea sobre la utilización de Software libre para el desarrollo de proyectos ya sean privados o independientes en distintos entornos de desarrollo como ReactJS, VeuJS Blazor, entre otros para cumplir con los objetivos propuestos anteriormente mencionados y mejorar los resultados.

En la Ley Orgánica de Emprendimiento e Innovación correspondiente al capítulo 2 titulado "Políticas Públicas e Institucionalidad del Emprendimiento", mediante el Art.6 menciona que "créase el Consejo Nacional para el Emprendimiento e Innovación – CONEIN, como organismo permanente estratégico para promover y fomentar el emprendimiento, la innovación y la competitividad sistémica del país" [24]. La ley en si favorece la innovación y el desarrollo de proyectos de investigación al fomentar el emprendimiento y al mismo tiempo beneficiaria el presente proyecto en cierta parte, como el proyecto se basa en una comparación de funciones de 2 modelos de servicio de Computación en la Nube, la ley impulsaría el desarrollo del proyecto agilitando su desarrollo y comercialización, mientras se realiza las pruebas comparativas.

CAPITULO III: MATERIALES Y MÉTODOS

3.1 Delimitación

El presente proyecto de investigación se ejecutó en un ambiente de pruebas en la nube utilizando los servicios de distintos proveedores de los modelos de servicio que se investigó para su análisis y comparación. La investigación se realizó en un plazo de 4 meses, iniciando en abril y terminando en julio presente año 2021.

3.2 Tipo de investigación

Para la ejecución de este estudio comparativo se consideró en realizar la investigación con un enfoque mixto, dicho de otro modo, una investigación según sus datos y su naturaleza cualitativa, cuantitativa, y experimental.

Del Canto y Silva Silva [25] mencionan que una investigación con enfoque cuantitativo se basa en identificar y formular una problemática científica, para luego ser revisada de forma literal afín a su tema. En su investigación con el que se formará un marco teórico-referencial, para posteriormente formular una hipótesis de dicha investigación donde se precisan las variables de alta importancia, las cuales serán definidas operacional y conceptualmente hablando. El presente enfoque de investigación es de carácter técnico y fue de mucha importancia en la investigación, después que los servicios fueron habilitados y las aplicaciones levantadas, fue necesario medir, analizar y comparar todos los resultados de estabilidad y funcionamiento de los servicios al momento de su ejecución según la normativa ISO 25010.

En el estudio presentado por Salas [26], una investigación con enfoque cualitativo parte de una problemática científica pero que claramente no sigue un proceso definido, además que las preguntas que se planteen en la investigación no siempre serán conceptualizadas por completo. Por lo general estas no se prueban bajo hipótesis, ya que se forman durante el proceso de investigación y se mejoran conforme se recolecta más información o son resultados del estudio. Este enfoque de investigación fue necesario porque permitía conocer variables que tuvieron incidencia durante el desarrollo e implementación de los servicios, por dar un ejemplo la escala de dificultad de uso y aprendizaje de los modelos de servicio.

En el estudio presentado por Marradi [27], una investigación con enfoque experimental es una tendencia de investigación en donde se manipula múltiples variables de estudio a la vez, para controlar su aumento o la disminución de estas variables y poder observar la conducta de sus efectos, dicho en otras palabras, consiste en realizar un cambio de valor en las variables independientes y observar sus efectos en otras variables, esta metodología se basa en la teoría de "Causa y Efecto". La implicación clave que se extrae de esto es que se llevan a cabo en ambiente controlados o de pruebas, con el objetivo de describir todos los eventos que se suscitan en el entorno de pruebas. Se planteó esto porque después del levantamiento de los servicios, se realizaron las pruebas de rendimiento donde se comprobaron y compararon todas sus funciones, en consecuencia, se manipularon sus variables. Las pruebas se realizaron de forma remota utilizando servicios de proveedores de Infraestructura y Plataforma como Servicio.

3.3 Métodos de investigación

Con relación a los métodos de investigación científica, se utilizaron métodos de estudio experimental por el hecho que se plantearon condiciones para establecer relación de causa y efecto, para la obtención de conclusiones en entornos de prueba, métodos de estudio descriptivo por el nivel y escala de profundidad, y método de estudio documental por el hecho que al ser un estudio comparativo la información debió tener como base documentos de otros estudios ya realizados en el pasado.

En la investigación planteada por Barnet-Lopez *et al.* [28], se explica que el método descriptivo describe el comportamiento de una serie de variables orientando a la investigación en el proceso de búsqueda de respuestas a preguntas que se planteen en el camino. El método fue necesario para describir las variables con sus indicadores.

En la investigación presentada por Acevedo Borrego *et al.* [29], fundamenta que el método experimental se presenta mediante la manipulación ya sea directa o indirecta de múltiples variables, en condiciones controladas (ambientes de prueba), con el objetivo de describir las causas y razones que se susciten al momento de realizar pruebas. El método fue necesario porque en el desarrollo de la investigación se planteó trabajar en un entorno de pruebas durante la ejecución final, se pudo descubrir las variables que tuvieron una importancia mayor en el proceso de análisis, evaluación y comparación al momento del desarrollo del estudio y la tabla comparativa durante el proceso de evaluación.

El método documental según Dulzaides Iglesias [30] en su artículo menciona que el método de investigación se basa en la recolección de documentos, artículos, entre otras bases de información para luego ser analizados de forma rigurosa con respecto al tema de investigación para ser utilizada como soporte o descubrir soluciones al tema de investigación central. La importancia del método fue vital, al tratarse de una comparativa teórica, se necesitó de documentación sobre proyectos con modelos de servicios, como por ejemplos fichas técnicas entre modelos Infraestructura y Plataforma como Servicio.

3.4 Población y muestra

El estudio de investigación presente se llevó a cabo en un entorno virtual, específicamente en la nube, utilizando herramientas de servicios como computación en la nube (*cloud computing*) para su desarrollo, por temas de latencia y cercanía geográfica, el lugar de alojamiento de nuestros servicios terminó siendo Miami en Estados Unidos. A través de análisis de proveedores se escogieron las empresas Microsoft y Google para el alquiler de los modelos de servicio infraestructura (Microsoft Azure, Google Cloud/Compute Engine, y Amazon Web Services) y plataforma (Microsoft Azure, Google Cloud/App Engine y Digital Ocean).

La muestra que se obtuvo del estudio fueron las aplicaciones ya programadas y desplegadas en los modelos. La muestra al ser muy pequeña, esta se la aplicó a toda la población en general.

3.5 Técnicas e instrumentos de recolección de datos

Para el estudio se planteó poner en marcha la incorporación de varias técnicas de investigación, con el que se abarcaron los puntos más importantes del estudio:

La primera técnica que se utilizó fue la investigación documental, al tratarse de una comparativa entre distintos modelos de servicio y sus proveedores, se tuvo que recabar información sobre fichas técnicas, trabajos investigativos, o revistas de desarrollo relacionados con el tema del proyecto investigativo, toda la información que se recolectó en el análisis de todos los documentos sirvió para realizar la comparativa teórica de ambos modelos y poder definir sus usos.

La segunda técnica de recolección de datos que se utilizó fue el diseño de una prueba experimental, basado en las normativas ISO 25010, la prueba experimental bajo esta

normativa se la denomina “evaluación de la calidad de software”, también conocido como “Proyecto de Requisitos y Evaluación de la calidad del software y del sistema” (SQARE, por sus siglas en ingles). Con esta herramienta se pudo obtener los resultados de las características que se eligió evaluar de las aplicaciones en los modelos. Este proceso se llevó a cabo después de ya preparados los entornos y levantadas las aplicaciones, para luego realizar las respectivas instalaciones de los entornos de trabajo y sus posteriores pruebas de estabilidad, seguridad y rendimiento. Todo fue documentado de forma secuencial con el fin de dejar detallado todos los pasos de elaboración de la evaluación de calidad de software dentro del modelo.

3.6 Técnicas de procesamiento y análisis de datos

Para el análisis de información del estudio de investigación se utilizó técnicas de estadística descriptiva que según explica Faraldo [31], es un grupo de técnicas numéricas fijas y gráficas que tienen como objetivo describir y analizar grupos de datos de cualquier tamaño, sin necesidad de llegar a conclusiones sobre a su población de origen. Estas técnicas se las utilizó para la gestión de resultados cuantificables o que se puedan medir (entorno de pruebas, control de seguridad, estabilidad de sistemas, homogeneidad de la arquitectura, adaptabilidad) que estuvieron en el estudio y que se obtuvieron a partir de un diseño experimental.

3.7 Operacionalización de variables

En la presente investigación se estudió 3 variables: Infraestructura, plataforma como servicio como servicio de cloud computing y Blazor como entorno de desarrollo web; cada una de estas variables es descrita en la Tabla 7.

Tabla 7 Tabla de Variables.

VARIABLE	DESCRIPCIÓN	DIMENSIONES	INDICADORES	TIPO DE VARIABLE	UNIDAD DE MEDIDA
Tema de estudio: IAAS como servicio de cloud computing	Modelo de servicio de Cloud Computing orientado a la infraestructura informática.	Infraestructura	Proveedor	Cualitativa	
			Costo del Servicio	Cuantitativa	Dólar Estadounidense
			Compatibilidad con la Nube	Cualitativa	Privada, Publica o Mixta
			Soporte	Cualitativa	Buena, Regular, Mala
			Arquitectura	Cualitativa	
			Latencia	Cuantitativa	Mbps
			Estabilidad	Cualitativa	
			Seguridad	Cualitativa	Buena, Mala, Regular
			Rendimiento	Cualitativo	
Tema de estudio: PAAS como servicio de cloud computing.	Modelo de servicio de Cloud Computing orientado a un entorno mixto entre infraestructura y	Infraestructura	Proveedor	Cualitativa	
			Costo del Servicio	Cuantitativa	Dólar Estadounidense
			Compatibilidad con la Nube	Cualitativa	Privada, Publica o Mixta
			Soporte	Cualitativa	Buena, Regular, Mala

	administración software.		Arquitectura	Cualitativa	
			Latencia	Cuantitativa	Mbps
			Estabilidad	Cualitativa	
			Seguridad	Cualitativa	Buena, Mala, Regular
			Rendimiento	Cualitativa	
Tema de estudio: Blazor como entorno de desarrollo web	Marco web desarrollado por Microsoft gratuito opensource, permite a los programadores crear software usando lenguaje C# y HTML	Seguridad	Nivel de cifrado de código	Cualitativa	Criptografía Simétrica, Criptografía Asimétrica
			Niveles de seguridad	Cualitativa	Parcial, Riesgo Informado, Repetible y Adaptativo
		Arquitectura	Homogeneidad	Cualitativa	Si, No
			Escalabilidad	Cuantitativa	Totalmente escalable, Parcialmente escalable
		Adaptabilidad	Multiplataforma	Cualitativa	Multiplataforma, Uniplataforma
		Aprendizaje	Tiempo de Aprendizaje	Cuantitativa	Horas
			Nivel de dificultad de aprendizaje	Cualitativa	

3.7.1 Indicadores

Modelos de Servicio.

- **Proveedor:** Empresa que vende, presta o alquila el servicio.
- **Costo del Servicio:** Valor a pagar por el servicio prestado o comprado.
- **Compatibilidad con la Nube:** tipo de seguridad del servicio en la nube.
- **Soporte:** Apoyo del proveedor a sus clientes.
- **Arquitectura:** Modelo estructural del modelo de servicio.
- **Latencia:** Tiempo de demora de respuesta en envió de información entre el cliente y el servicio.
- **Estabilidad:** Tiempo en línea del servicio.
- **Seguridad:** Protección otorgada por el proveedor.
- **Rendimiento:** calidad del servicio en relación con el trabajo que desempeñe.

Entornos de desarrollo

- **Nivel de cifrado de código:** Nivel de seguridad de encriptación del código.
- **Niveles de seguridad:** El nivel de seguridad que el entorno de desarrollo otorgue a la aplicación.
- **Homogeneidad:** Que pueda mezclarse o hibridarse con otros lenguajes.
- **Escalabilidad:** Capacidad de respuesta y adaptación de un sistema con respecto a los cambios de rendimiento a medida que este incrementan de forma significativa.
- **Multiplataforma:** Capacidad de adaptación a diferentes navegadores.
- **Tiempo de Aprendizaje:** Tiempo de demora en aprender y manejar el entorno.
- **Nivel de dificultad de aprendizaje:** Escala de facilidad o dificultad al momento de aprender a manejar el entorno.

3.8 Normas éticas

El objetivo del estudio fue comparativo, para mostrar a la comunidad las ventajas y desventajas que tienen las Infraestructuras y Plataformas como servicio, para influir en el resultado de las personas al momento de elegir uno de ellos. Después de levantado los servicios, ejecutadas las aplicaciones y ya obtenido los datos del análisis, la información será celosamente protegida, con el fin de ofrecer seguridad y fiabilidad al público que fungirá de población de muestreo.

La información, códigos de programación, configuraciones, base de datos, seguridad y arquitectura serán realizadas bajo las reglas, leyes, normativas de seguridad de la Pontificia Universidad Católica del Ecuador Sede Esmeraldas, el código penal del Ecuador y las leyes de ciberseguridad de los Estados Unidos (ya que la ubicación de los servidores de que almacenará esta información se encontrará en dicho país), lo que asegura que el desarrollo de la presente investigación estará al margen de la ley.

CAPITULO IV: RESULTADOS

Análisis e Interpretación de los resultados.

Para el desarrollo de este proyecto de investigación se recabaron estudios e investigaciones concernientes al tema de desarrollo de proyectos de aplicaciones web en entornos de computación en la nube en los modelos de infraestructura o plataforma como servicio, la mayoría de la información se la extrajo de la base de datos documental IEEExplore, esta información se verá reflejada primero en la Figura 12, en esta se mostrará el porcentaje de información referente al tema que se buscó, en la Figura 13 se visualizará un flujograma de búsqueda de información documental desarrollada, y en la Tabla 8 se verán reflejadas las ecuaciones que se utilizaron para la búsqueda y recopilación de la información (cabe recalcar que toda esta información data del año 2012 en adelante, teniendo en cuenta que el auge de la computación en la nube tuvo su primer repunte en ese año [32]).

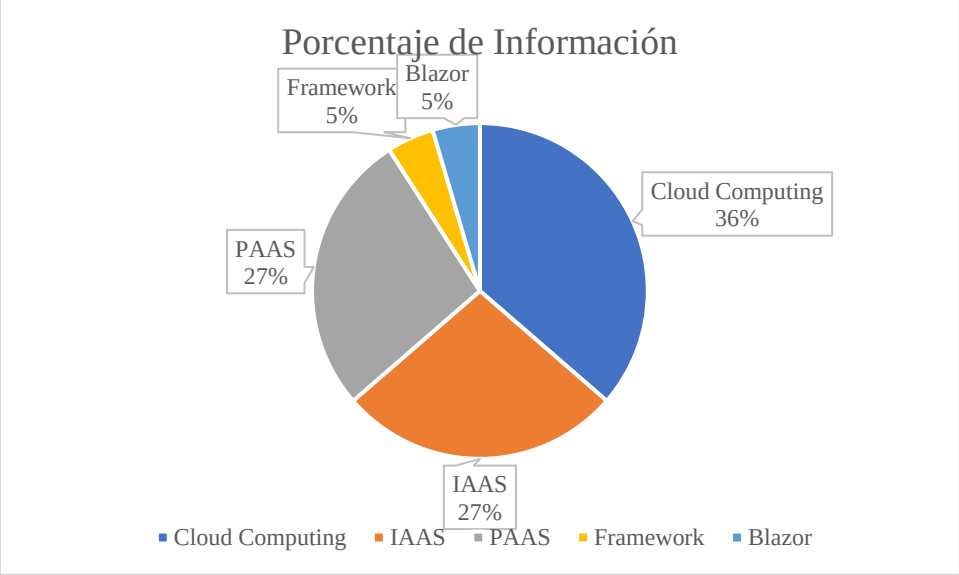

Figura 12 Base de Información Obtenida

Tabla 8 Ecuaciones de Búsqueda

Fuente de información	Ecuación de búsqueda	Documentos obtenidos
IEEExplore	(("Document Title”: “PAAS") AND ("Document Title”: “PLATFORM AS A SERVICES" OR "Document Title”: “CLOUD COMPUTING"))>2012	6

	(("Document Title”: “IAAS") AND ("Document Title":" INFRASTRUCTURE AS A SERVICE" OR "Document Title”: “CLOUD COMPUTING"))>2012	6
	("Document Title”: “CLOUD COMPUTING" OR "CLOUD")>2012	8
	("Document Title”: “FRAMEWORK")>2012	1

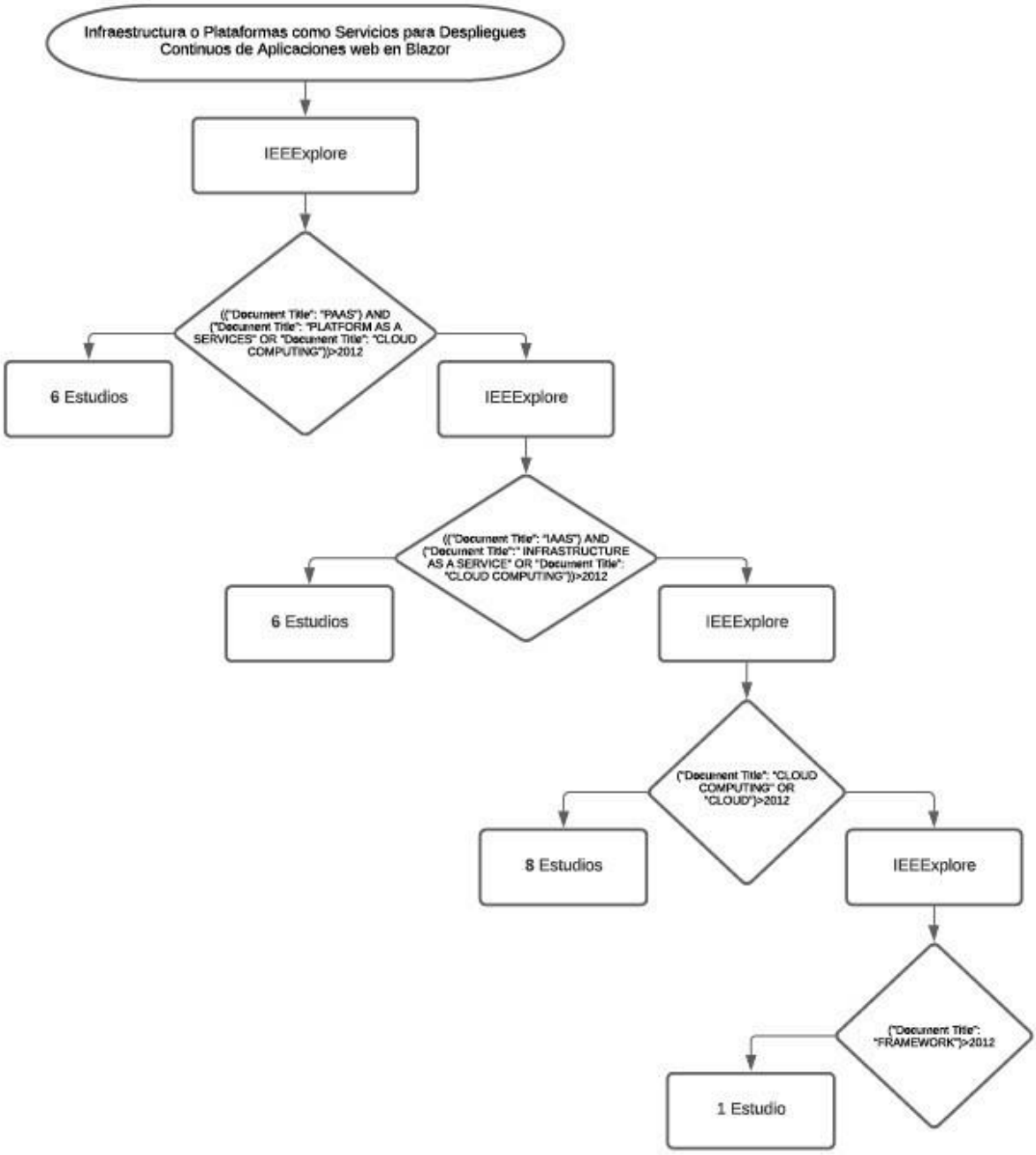

Figura 13 Flujograma de búsqueda

4.1 Investigación sobre los modelos de servicios de Infraestructura y Plataforma más famosos.

Para cumplir con el objetivo se tuvo que ejecutar una búsqueda e investigación documental, sobre los proveedores de servicios de computación en la nube, esto basándose en las tablas clasificatorias de cuota de mercado de la nube publicadas por el periódico de negocios Forbes [32], y por el foro de tecnología especializado ZDNet, estas fuentes ranquean a los proveedores basándose no solo en la tasa consumo y uso de los clientes, también en la facilidad de uso de sus servicios y seguridad.

De dichas investigaciones ya citadas, en la Tabla 9 se encuentra todas las empresas con mayor cuota de consumo de modelos de servicio en la nube, de entre todos los proveedores de ambos servicios se optó por contratar los servicios de Microsoft y Google para los modelos de Infraestructura (Microsoft Azure, Google Cloud con extensiones de Compute Engine y Amazon Web Services) y Plataforma (Microsoft Azure, Google Cloud con extensiones de App Engine y Digital Ocean) como servicio.

Tabla 9 Ranking Cloud Computing.

1. **Microsoft** – *Nadella on $20.4B run rate w/end-to-end customer-centric cloud*
2. **Amazon** -- *AWS needs software! 10 software companies Amazon might look at*
3. **IBM** – *Rometty strikes gold helping customers convert legacy IT to private cloud*
4. **Salesforce** – *Benioff must extend SFDC impact from SaaS deeply into PaaS*
5. **SAP** – *McDermott accelerating major product-line overhaul to HANA and cloud*
6. **Oracle** –*Ellison on cybercrime: 'Make no mistake, this is a war—and we're losing'*
7. **Google** – *Tons of potential but still unclear if/how it wants to play in enterprise*
8. **ServiceNow** – *Jumps ahead of Workday: revenue up 40%, new products boom*
9. **Workday** – *Q2 revenue surges 41% as Bhusri jumps into PaaS marketplace*
10. **VMware** – *revenue & stock jump on deals w/AMZN MSFT IBM GOOG for hybrid*

4.2 Comparativa entre los modelos de Infraestructura y Plataforma como servicio.

De igual forma que en el objetivo anterior, se realizó una investigación documental exhaustiva sobre Infraestructura y Plataforma como servicio para luego ejecutar una comparativa de las ventajas y desventajas que estos servicios pueden proveer. Los resultados de esta investigación se pueden ver plasmados en la Tabla 10.

Tabla 10 Comparativa entre IAAS y PAAS

Modelos Aspectos	**Infraestructura como Servicio**	**Plataformas como Servicio**

Ventajas	**1.** El proveedor ofrece administración y personalización total del hardware, instalaciones, servidores de bases de datos, manipulación del control energético de la sección asignada, entre otros. **2.** Puedes tener uso de Backups en caso de emergencia. **5.** IAAS al ser la base de la cloud computing, pueden desplegar los otros modelos encima de él.	**1.** Se entrega un entorno armado y completo. **2.** El nivel de dificultad de comprensión de uso es muy sencillo. **3.** Puedes tener uso de Backups en caso de emergencia. **4.** Puedes desplegar SAAS desde una PAAS de forma sencilla.
Desventajas	**1.** Se tiene que armar y configurar el entorno desde 0. **2.** Algunos proveedores no son muy intuitivos. **3.** Levantar otros modelos encima de IAAS es complicado y prohibido por los proveedores (algunos). **4.** Los proveedores te pueden dar soporte de seguridad externa, pero la interna ya corre por cuenta propia	**1.** La manipulación del entorno no es 100% total. **2.** Solo administras la parte Software y algunos (no todos) aspectos del hardware. **3.** No puedes usar algunas herramientas de tu entorno (como servidores de BD o interactuar con la infraestructura de red o secciones de otros clientes o la propia)

4.3 Desarrollo y levantamiento de aplicaciones web en Blazor en los modelos de servicio.

Para la cumplir con el presente objetivo se diseñó y programó una aplicación web en el entorno de desarrollo Blazor, un aplicativo web chat llamado Blazor Chating. Cabe recalcar que se programaron estas APPS usando Blazor Web Assembly.

4.3.1 Blazor Chating

4.3.1.1 Shared

Shared contiene archivos .cs que fueron programados con el objetivo de realizar una conexión entre el servidor y la base de datos, en su programación se incorporó elementos para registro e inicio de sesión usando redes sociales, ya sea Facebook, Twitter, Google o un registro propio de la aplicación. En la Figura 14 se puede visualizar cómo se estructuró Shared en el desarrollo de la aplicación. En los ficheros de nombre Facebook y Twitter se programaron constructores correspondientes al registro, inicio de sesión, acceso de usuario y datos principales relacionados con ambas redes sociales.

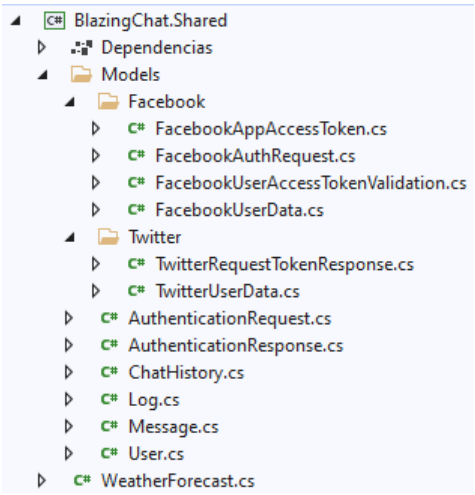

Figura 14 Componentes de .Blazor Chating shared

La parte con mayor importancia que fue programada en Shared corresponde a todos los archivos que fueron almacenados dentro de la carpeta llamada “Models” que no estuvieron almacenados en las carpetas Twitter y Facebook. Todos estos elementos que fueron programados como constructores, clases de enlace, almacenamiento de registro o de autentificación y verificación, realizan una conexión con el archivo de contexto de base de datos (de tipo SQLite) al Shared.

4.3.1.2 Server

Para la programación de Server se realizó una configuración en la plantilla básica de Blazor, al tratarse una aplicación que se alimenta sola se configuró a una Web API convencional, de esta forma en el desarrollo se modificó su modelo de estructura, el cual se dividió en 6 partes como se muestra en la Figura 15. Cabe recalcar que el desarrollo del Server fue crucial, ya que permitió forzar el rendimiento de los entornos que lo contenía.

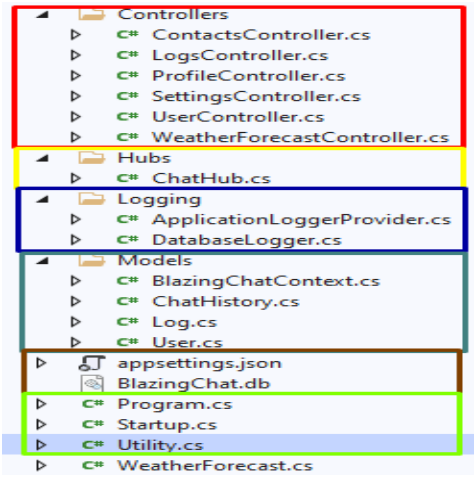

Figura 15 Componentes Blazor Chating Web API

En la carpeta "Controllers" se programaron los controladores API de todas las funciones lógicas de una aplicación de chat, ya sean para un inicio de sesión, perfiles (Algoritmo 1), configuraciones de cuenta (Algoritmo 2), entre otras funciones. Si un usuario realiza alguna petición, este apartado las recibía y ejecutaba al momento.

En la carpeta "Hubs" se programó una clase que permitió mandar cadenas de información temporal a la parte gráfica del cliente, además sirvió como una pequeña memoria caché (Algoritmo 3).

La carpeta llamada "Logging", al igual que en las carpetas anteriores fue programada para almacenar y que cumpla funciones de ejecución de métodos de inicio de sesión directos a la base de datos, además se agregó una clase más llamada "DatabaseLogger.cs" que validaba por medio de consultas si la cuenta que iniciara sesión se encontraba en existencia o no (Algoritmo 4).

La carpeta "Models" (al igual que su gemelo de Shared) se lo programó como un constructor, pero como un constructor que unió el otro constructor en shared y que formaba una cadena de conexión directa al contexto de base de datos.

Cuando se generó la plantilla base para la programación del aplicativo, se creó un archivo llamado "appsettings. json", este archivo fue modificado para que arranque la base de datos SQLite y de dirección del puerto por el cual la aplicación debió salir (Algoritmo 5).

Por último, tenemos las clases sin carpetas específicas, que fueron generadas propiamente por la plantilla de Blazor. Cabe recalcar que se realizó una integración de

librerías y complementos SQLite para que Web API pueda conectarse sin problemas a la base de datos, esta integración se la hizo en la sección de componentes y se la llamó en "Startup.cs".

En la parte inferior podrá visualizarse el código programado en la Web API.

```
[HttpPut("updateprofile/{userId}")]
        public async Task<User> UpdateProfile(int userId, [FromBody] User user)
        {
            User userToUpdate = await _context.Users.Where(u => u.UserId ==
userId).FirstOrDefaultAsync();

            userToUpdate.FirstName = user.FirstName;
            userToUpdate.LastName = user.LastName;
            userToUpdate.EmailAddress = user.EmailAddress;
            userToUpdate.AboutMe = user.AboutMe;
            userToUpdate.ProfilePicDataUrl = user.ProfilePicDataUrl;

            await _context.SaveChangesAsync();

            return await Task.FromResult(user);
        }

        [HttpGet("getprofile/{userId}")]
        public async Task<User> GetProfile(int userId)
        {
            return await _context.Users.Where(u => u.UserId == userId).FirstOrDefaultAsync();
        }
```

Algoritmo 1 ProfileController.cs

```
       [HttpPut("updatetheme/{userId}")]
        public async Task<User> UpdateTheme(string userId, User user)
        {
            User userToUpdate = _context.Users.Where(u => u.UserId ==
Convert.ToInt32(userId)).FirstOrDefault();
            userToUpdate.DarkTheme = user.DarkTheme; //== "True" ? 1 : 0;

            await _context.SaveChangesAsync();

            return await Task.FromResult(user);
        }
```

Algoritmo 2 SettingsController.cs

```
namespace BlazingChat.WebAPI.Hubs
{
    public class ChatHub : Hub
    {
        public async Task SendMessage(Message message)
        {
            var users = new string[] { message.ToUserId, message.FromUserId };
            //await Clients.Users(users).SendAsync("ReceiveMessage", message);
            await Clients.All.SendAsync("ReceiveMessage", message);
        }
    }
}
```

Algoritmo 3 ChatHubs

```
        public void Log<TState>(LogLevel logLevel, EventId eventId, TState state, Exception
exception, Func<TState, Exception, string> formatter)
        {
            long userId = 0;

            Log log = new();
            log.LogLevel = logLevel.ToString();
            log.UserId = Convert.ToInt64(userId);
            log.ExceptionMessage = exception?.Message;
            log.StackTrace = exception?.StackTrace;
```

```
            log.Source = "Server";
            log.CreatedDate = DateTime.Now.ToString();

            _context.Logs.Add(log);
            _context.SaveChanges();
        }
```

Algoritmo 4 DatabaseLogger.cs

```
  "AllowedHosts": "*",
  "ConnectionStrings": {
    "BlazingChat": "Data source=BlazingChat.db"
  },
```

Algoritmo 5 Appsettings.json

4.3.1.3 Client

Para el desarrollo del Client se dividieron y agruparon los componentes en secciones, cada sección se programó con el fin de estandarizar sus funciones, se compone de “Handlers y Logging”, “Pages y Shared” y los “ViewModels”.

La sección “Handlers y Logging” se armó con las carpetas del mismo nombre, estas carpetas fueron programadas con el objetivo de recibir los datos que le da el cliente al momento de iniciar sesión o de registrarse, y mandarlas por medio de métodos y funciones a la Web API para almacenar información en caso de que realice un registro de sesión. Handlers fue programado para cumplir funciones de consulta y respuesta entre el servidor y el cliente. La estructura se la puede visualizar en la Figura 16.

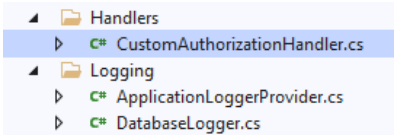

Figura 16 Handlers y Logging

La sección “Pages y Shared” se conforma por carpetas del mismo nombre, en ellas se almacenaron archivos razor, como ya se especificó en la parte teórica los archivos de razor son archivos de visualización que también contienen funcionen lógicas. En esta sección se programaron todas las vistas como por ejemplo el menú inicial, el menú de registro e inicio de sesión, el perfil, los contactos, la ventana de error, y el scrollBar en donde se muestran todos los elementos para el chat. La estructura resultante se puede visualizar en la Figura 17.

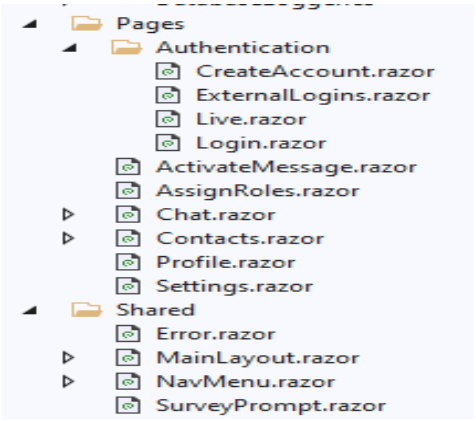

Figura 17 Pages y Shared

La sección “ViewModels” fue armada y programada para contener archivos de clases que puedan almacenar datos temporales de vista de todas las funciones gráficas y logísticas que se ejecutan al momento de interactuar con el usuario. Los elementos que están dentro de esta sección también sirven como conexión entre las vistas principales con los conectores de la base de datos que se encuentran en la Web API. La estructura resultante se visualiza en la Figura 18.

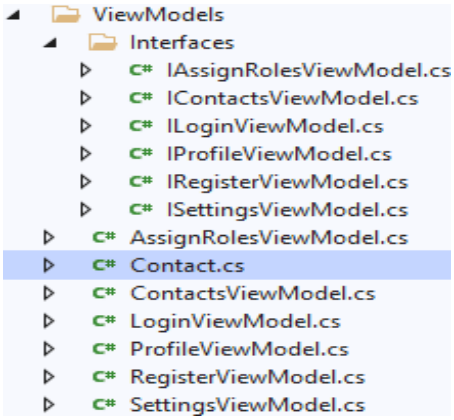

Figura 18 ViewModels

4.4 Evaluación del rendimiento de las aplicaciones ya desplegada.

Para el desarrollo de este objetivo se optó por realizar una evaluación de la calidad del software usando la ISO 25000, también conocida como proyecto Square. Esta evaluación se desarrolló en 5 pasos, cada paso correspondió a la delimitación y determinación de la evaluación, diseño de la evaluación, obtención y visualización de datos con su respectiva conclusión de informe de forma estandarizada.

4.4.1 Determinar los Requisitos de la Evaluación

En el presente apartado se obtuvieron los siguientes resultados según los aspectos que se muestran en la Tabla 11.

Tabla 11 Determinar los Requisitos de la Evaluación

Aspectos	Resultados
Establecer el propósito de evaluación	Medir el rendimiento de las 2 aplicaciones web en los modelos de servicio IAAS y PAAS
Obtener los requisitos de calidad a evaluar	1. Eficiencia de Desempeño 2. Seguridad
Identificar las partes del producto que se va a evaluar	1. Eficiencia de Desempeño • Comportamiento temporal. • Utilización de recursos. • Capacidad. 2. Seguridad • Confidencialidad.

Para una mejor comprensión de los resultados de la tabla se procede a explicar los aspectos de los requisitos de la evaluación, "establecer el propósito de la evaluación" hace referencia al objetivo de realizar la evaluación de calidad.

"Obtener los requisitos de la calidad a evaluar" hace referencia a las características a tomar en cuenta de la ISO 25010, estas características son las que se van a tomar como criterios de evaluación de calidad de las aplicaciones analizadas.

"Identificar las partes del producto que se van a evaluar" hace referencia a los elementos de las características que se seleccionaron como criterios de medición, cabe aclarar que no se eligieron todas las características solo las más importantes.

4.4.2 Especificar la Evaluación

En el presente apartado se obtuvieron los siguientes resultados según los aspectos que se muestran en la Tabla 12.

Tabla 12 Especificar la Evaluación

Resultado Aspectos	Resultados
Seleccionar métricas	• Comportamiento temporal: Tiempo que el aplicativo tarda en responder. • Utilización de recursos: Porcentaje de utilización de recursos. • Capacidad: Cumplimiento de requisitos para el aplicativo en ejecución • Confidencialidad: seguridad de Datos.

Definir Criterios de evaluación	• Comportamiento temporal: Tiempo (segundos) • Utilización de recursos: Porcentaje en MB/s • Capacidad: Cuantificación porcentual. • Confidencialidad: Cuantificación de las vulnerabilidades cubiertas.
Definir Criterios de decisión	• Comportamiento temporal: Medir el tiempo que demora la aplicación en responder ante alguna petición dentro de la plataforma (tiempo de ejecución) y fuera de ella (tiempo de respuesta). • Utilización de recursos: Cuantificar el porcentaje de recursos consumidos de la aplicación. • Capacidad: Medir el nivel de adaptación con el que se maneja las aplicaciones • Confidencialidad: Medir el nivel de seguridad de los modelos de servicio con esta aplicación.

Para una mejor comprensión de los resultados de la tabla se procede a explicar los aspectos de los requisitos de la evaluación, "Seleccionar métricas" hace referencia a las métricas y su definición.

"Definir Criterios de evaluación" hace referencia o a la forma de cómo medirán o cuantificarán los datos que se obtuvieron en el análisis.

"Definir Criterios de evaluación" hace referencia definir preguntas que tengan relación con los 2 aspectos anteriores, pero con un enfoque de porque se seleccionaron. La pregunta que se seleccionaron fue acorde a los objetivos dados de la investigación en la Tabla 11.

4.4.3 Diseñar la evaluación

En el presente apartado se obtuvieron los siguientes resultados según los aspectos que se muestran en la Tabla 13. Los resultados que se obtuvieron fueron los pasos a seguir para el proceso de evaluación de rendimiento en cada uno de los modelos.

Tabla 13 Diseñar la evaluación

Pasos para Evaluación de Rendimiento y Calidad.	
Plataforma como Servicio	Infraestructura como servicio
1. Medir el tiempo de ejecución de las aplicaciones con la herramienta Google PageSpeed. 2. Medir la velocidad de respuesta de las aplicaciones con la herramienta GTmetrix.com. 3. Verificar y Analizar las estadísticas de uso de recursos de los proveedores al momento de ejecutar las aplicaciones. 4. Analizar el porcentaje de utilización de recursos de las aplicaciones en los modelos. 5. Medir el nivel de seguridad de las aplicaciones en el entorno en la herramienta observatory de Mozilla.	1. Medir el tiempo de ejecución de las aplicaciones con la herramienta Google PageSpeed. 2. Medir la velocidad de respuesta de las aplicaciones con la herramienta GTmetrix.com. 3. Verificar y Analizar las estadísticas de uso de recursos de los proveedores al momento de ejecutar las aplicaciones. 4. Analizar el porcentaje de utilización de recursos de las aplicaciones en los modelos. 5. Medir el nivel de seguridad de las aplicaciones en el entorno en la herramienta observatory de Mozilla.

4.4.4 Ejecutar evaluación

A. PAAS

En este paso se realizaron todas las evaluaciones según el paso anteriormente mencionado, aquí se obtuvieron los resultados de la evaluación de rendimiento y calidad en cada una de las características ya seleccionadas con anterioridad. Los procesos que se hicieron, junto con las herramientas que se utilizaron para evaluar cada aspecto del modelo (PAAS), se las menciona en la Tabla 13.

Los resultados que se consiguieron están en una escala de 0 a 10, siendo cero lo mínimo (pésimo rendimiento) y 10 el máximo (excelente rendimiento), pero cabe aclarar algo importante, el aspecto "Tiempo de ejecución", la escala que se le dio fue distinta, al tratarse de tiempo, su escala fue de 0 segundos (muy veloz) a 2 segundos (muy lento). Los resultados se pueden verse reflejados en la Tabla 14, en la Figura 19 y Figura 20.

Tabla 14 Eficiencia del Desempeño (PAAS)

Eficiencia del Desempeño (PAAS)			
	Microsoft Azure	G. C. / G. A. E.	Digital Ocean
Utilización de Recursos	8,1	9,3	9,5
Capacidad	9,5	8,93	9,1
Tiempo de ejecución	0,5 segundos	0,2 segundos	0,11 segundos

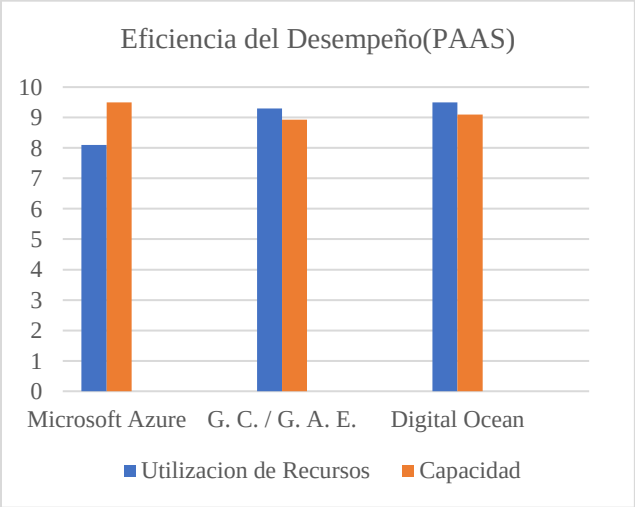

Figura 19 Eficiencia del Desempeño (PAAS)

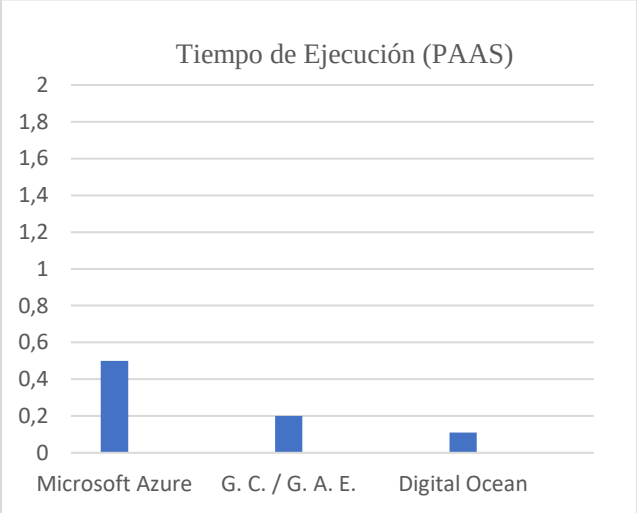

Figura 20 Tiempo de Ejecución (PAAS)

Para este paso se realizó la evaluación de seguridad de las plataformas como servicio, estos datos se los obtuvieron por medio del uso de las herramientas mencionadas en la Tabla 13, se cargaron las conexiones del servidor a las herramientas y se realizaron las evaluaciones de seguridad. Los resultados de esta evaluación estuvieron dentro de una escala siendo 0 (lo más bajo) seguridad de nivel crítico, 5 como nivel de seguridad mediano, 7 como nivel de seguridad aceptable y 10 como nivel de seguridad perfecto. Los resultados de la evaluación de seguridad se verán plasmados en la Tabla 15 y en la Figura 21.

Tabla 15 Seguridad (PAAS)

Seguridad (PAAS)			
	Microsoft Azure	G. C. / G. A. E.	Digital Ocean
Confidencialidad.	10	10	8,5

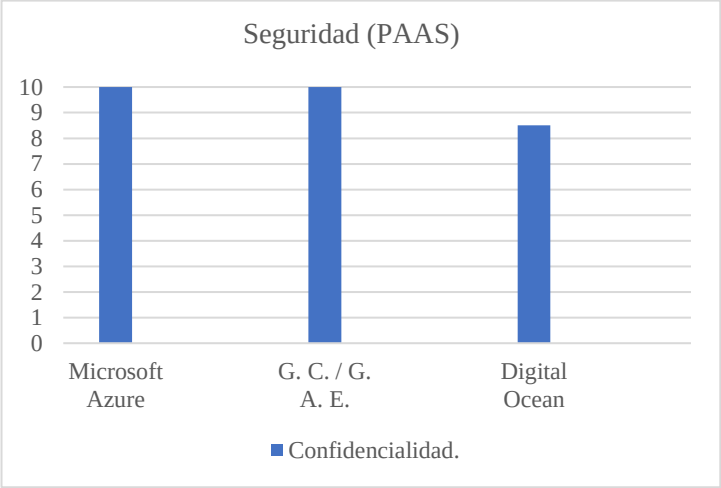

Figura 21 Seguridad (PAAS)

Para este paso se realizó la evaluación de rendimiento del tiempo de respuesta, cabe recalcar que no se debe confundir los conceptos de tiempo de respuesta con comportamiento temporal, comportamiento temporal es el tiempo que demoró en ejecutar la aplicación en la plataforma y el tiempo de respuesta es el tiempo que se demora el servidor en responder una petición del usuario. Para la obtención de los datos se implementaron 2 herramientas más, JMeter para la evaluación del tiempo de respuesta y SSTP para generar una VPN para enviar una petición desde cualquier parte del mundo a la aplicación alojada en un servidor en Estados Unidos. Los datos que se obtuvieron están en una escala de 2 como el tiempo de espera límite máximo y 0 como tiempo de espera límite mínimo. Los tiempos de respuesta de origen a destino varían acorde a algunos aspectos como la distancia que tiene que recorrer, los puntos de cruce entre servidores entre el origen y destino, la velocidad de los operadores, entre otros. Los resultados se pueden ver plasmados en la Tabla 16 y en la Figura 22.

Tabla 16 Tiempo de Respuesta (PAAS).

Tiempo de Respuesta (PAAS)			
	Microsoft Azure	G. C /G. A. E	Digital Ocean
EE. UU. - Ecuador	0,7	0,3	0,2
EE. UU. - Chile	0,8	0,73	0,2
EE. UU. - Argentina	0,8	0,87	0,28
EE. UU. - Inglaterra	0,03	0,05	0,11
EE. UU. - Alemania	0,02	0,02	0,1
EE. UU. - China	0,06	0,05	0,07
EE. UU. - Australia	0,1	0,21	0,22

EE. UU. - Taiwán	0,3	0,27	0,29
EE. UU. - Hong Kong	0,22	0,31	0,17
EE. UU. - Rusia	0,4	0,39	0,15
EE. UU. - Japón	0,28	0,25	0,13

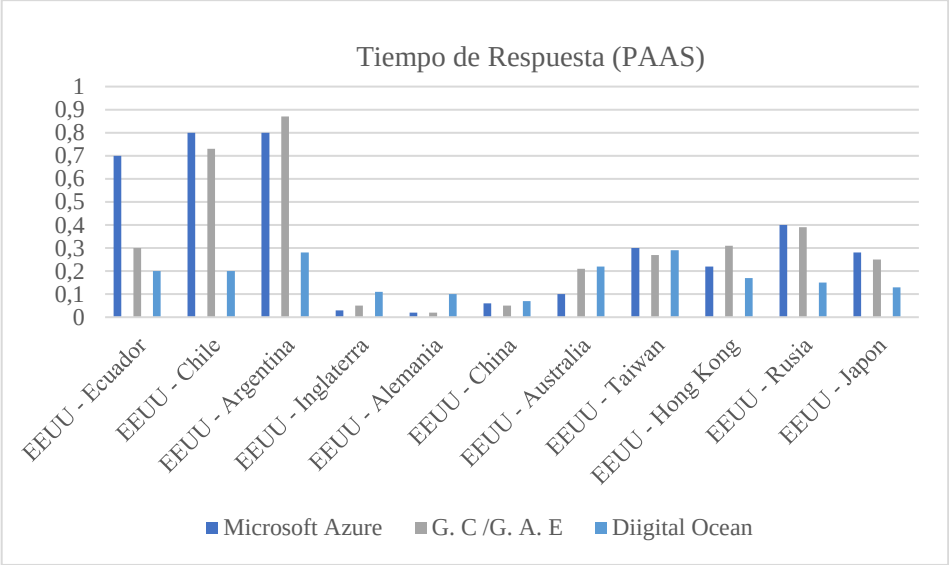

Figura 22 Tiempo de Respuesta (PAAS)

B. IAAS

Los resultados que se consiguieron están en una escala de 0 a 10, siendo cero lo mínimo (pésimo rendimiento) y 10 el máximo (excelente rendimiento), pero cabe aclarar algo importante, el aspecto “Tiempo de ejecución”, la escala que se le dio fue distinta, al tratarse de tiempo su escala fue de 0 segundos (muy veloz) a 2 segundos (muy lento). Los resultados se pueden ver reflejados en la Tabla 17, en la Figura 23 y en la Figura 24.

Tabla 17 Eficiencia del Desempeño (IAAS)

Eficiencia del Desempeño (IAAS)			
	Microsoft Azure	G. C. / G. C. E.	A. W. S.
Utilización de Recursos	8,1	8,75	9
Capacidad	9,5	9,5	8,9
Tiempo de ejecución	0,7	1,2	0,76

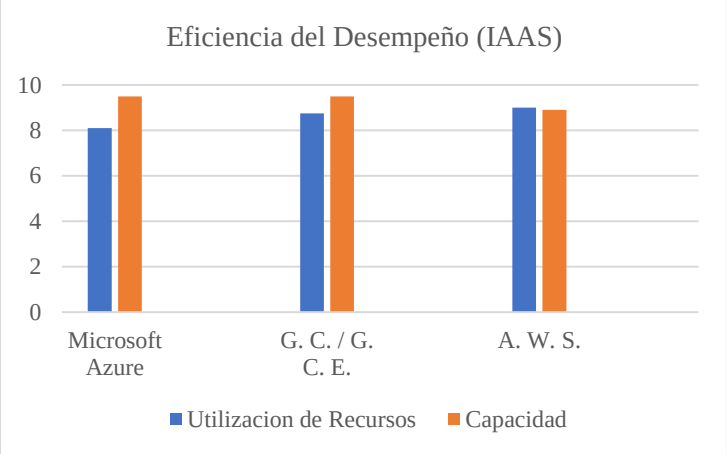

Figura 23 Eficiencia del Desempeño (IAAS)

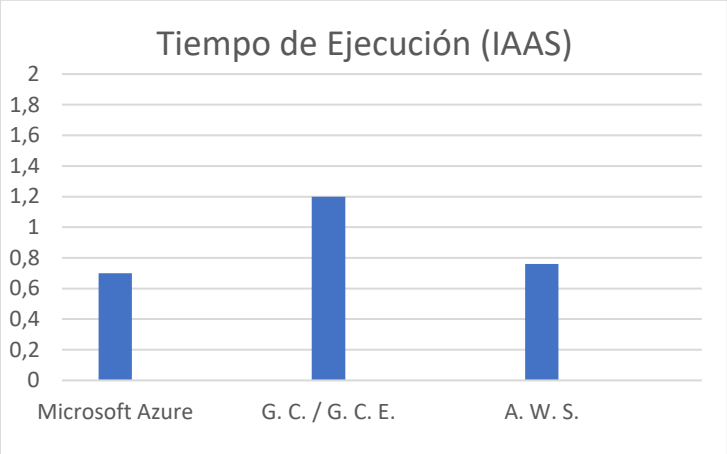

Figura 24 Tiempo de Ejecución (IAAS)

Para este paso se realizó la evaluación de seguridad de las plataformas como servicio, estos datos se los obtuvieron por medio del uso de las herramientas mencionadas en la Tabla 13, se cargaron las conexiones del servidor a las herramientas y se realizaron las evaluaciones de seguridad. Los resultados de esta evaluación estuvieron dentro de una escala siendo cero (lo más bajo) seguridad de nivel crítico, cinco como nivel de seguridad median, siete como nivel de seguridad aceptable y 10 como nivel de seguridad perfecto. Los resultados de la evaluación de seguridad se verán plasmados en la Tabla 18 y en la Figura 25.

Tabla 18 Seguridad (IAAS)

Seguridad (IAAS)			
	Microsoft Azure	G. C. / G. C. E.	A. W. S.
Confidencialidad	10	10	10

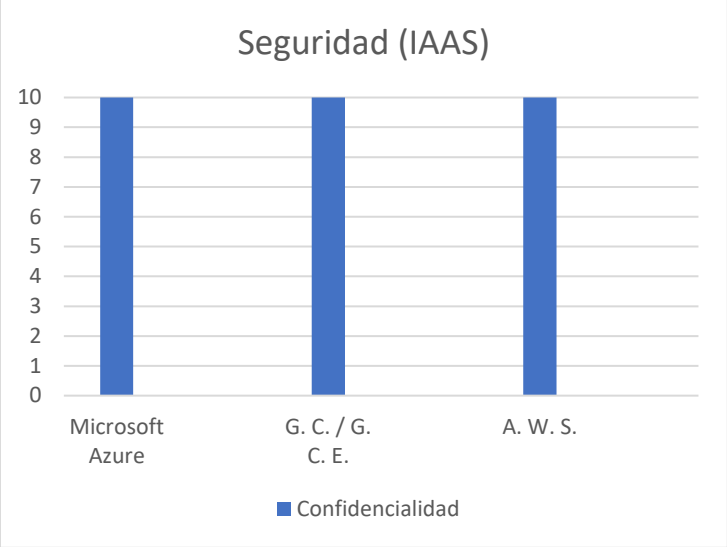

Figura 25 Seguridad (IAAS)

Para este paso se realizó la evaluación de rendimiento del tiempo de respuesta, cabe recalcar que no se debe confundir los conceptos de tiempo de respuesta con comportamiento temporal, comportamiento temporal es el tiempo que demoró en ejecutar la aplicación en la plataforma y el tiempo de respuesta es el tiempo que se demora el servidor en responder una petición del usuario. Para la obtención de los datos se implementó 2 herramientas más, JMeter para la evaluación del tiempo de respuesta y SSTP para generar una VPN para hacer la petición desde cualquier parte del mundo a la aplicación sé que ubicó en un servidor en Estados Unidos. Los datos que se obtuvieron están en una escala de dos como el tiempo de espera límite máximo y cero como tiempo de espera límite mínimo. Los resultados se verán plasmados en la Tabla 19 y en el Figura 26.

Tabla 19 Tiempo de Respuesta (IAAS)

Tiempo de Respuesta (IAAS)			
	Microsoft Azure	G. C /G. C. E	A. W. S.
EE. UU. - Ecuador	0,7	0,3	0,27
EE. UU. - Chile	0,8	0,73	0,2
EE. UU. - Argentina	0,8	0,87	0,31
EE. UU. - Inglaterra	0,03	0,05	0,05
EE. UU. - Alemania	0,02	0,02	0,12
EE. UU. - China	0,04	0,08	0,43
EE. UU. - Australia	0,2	0,6	0,19
EE. UU. - Taiwán	0,31	0,12	0,12

EE. UU. - Hong Kong	0,03	0,51	0,22
EE. UU. - Rusia	0,121	0,09	0,21
EE. UU. - Japón	0,12	0,19	0,31

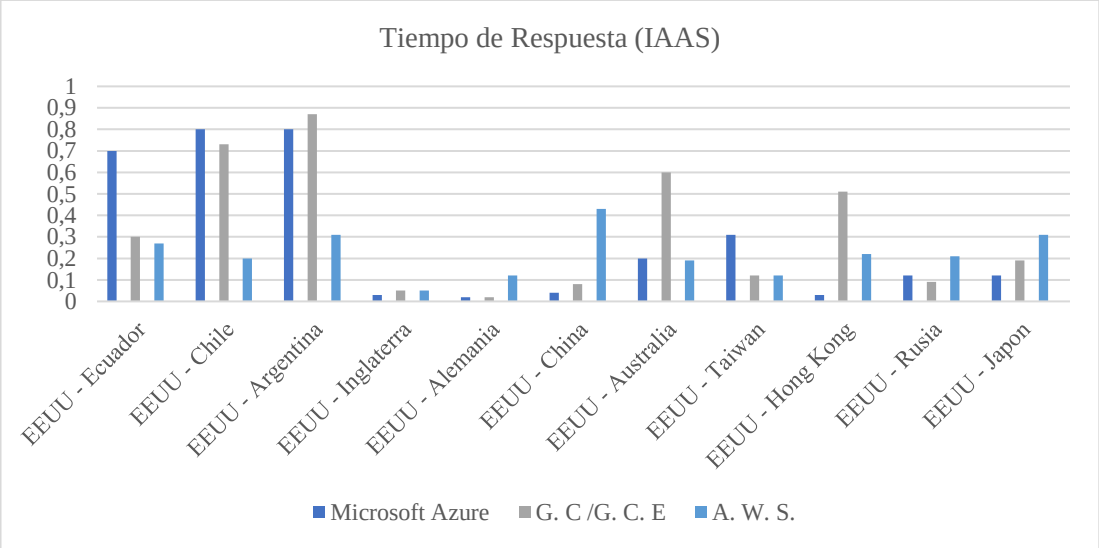

Figura 26 Tiempo de Respuesta (IAAS)

CAPITULO V: DISCUSIÓN

En la investigación realizada por Mannella Lemos [7] tuvo como objetivo armar, configurar e implementar un entorno de infraestructura como servicio para la recuperación de desastres tecnológicos de Pymes bancarias, donde realizan una infraestructura totalmente híbrida, es decir un porcentaje total de la infraestructura física como otro porcentaje de la infraestructura en la nube. En cambio, en esta investigación a pesar de que se trata de un proyecto mucho más pequeño a diferencia de la investigación anteriormente mencionada, la configuración de la infraestructura fue completamente en la nube, algo que benefició al estudio de esta investigación. Los procesos de evaluación del artículo fueron hechos para dos situaciones del mismo modelo, cosa que según la investigación lo volvió más difícil de realizar, a diferencia de la presente investigación que fue más sencilla al tener solo una situación ubicada netamente en la nube.

En la investigación propuesta por Xia *et al.*[15] tuvo como objetivo desarrollar un sistema predictivo de estado en IAAS basado en el proceso Hidden Markov, donde se realizaron estudios para poder calcular y predecir los momentos en donde una computadora o una red de infraestructuras sufre una sobrecarga. En comparación con el estudio actual, se utilizó un proceso similar para predecir en qué momento la infraestructura sufriría una sobrecarga de información, ambos estudios fueron evaluados bajo el proyecto Square, se evaluaron las mismas características (eficiencia de desempeño) pero con la diferencia que en el presente estudio se utilizó una herramienta externa para realizar la evaluación, mientras que en la investigación comparada su herramienta para evaluar esto fue su propio proyecto.

En la investigación realizada por Li *et al.* [16] plantearon que los proveedores de modelos de servicio en nube manipulan los precios de los componentes más importantes de un servidor, provocando que al encarecerse o depreciarse el precio de un componente, el costo de alquiler de algún modelo de servicio también se vea afectado. Este aporte tuvo un valor muy crucial en el primer objetivo de la investigación, enfocado en el precio de los componentes más importantes para el armado de servidores, se pudo hacer una pequeña predicción de los costos futuros, para saber el momento exacto para alquilar ambos de un proveedor específico.

En la investigación realizada por Kiwon y Kwangseob [17] plantearon incorporar un Servicio de Procesamiento de Web (WPS) con un estándar GIS abierto compatible con un procesamiento de información asincrónico en PAAS. En comparación de esta investigación se pudo tener una mejor implementación con los modelos de servicio, en la investigación anteriormente mencionada su implementación con herramientas para análisis de rendimiento fue más escueta, la incorporación de elementos fue más rústica, mientras que en la presente investigación la incorporación de herramientas internas tanto para la evaluación de rendimiento como complementos de funciones fue más completa y mejor esquematizada al ser desarrollada en entornos monitorizados por proveedores calificados. También presentó un proceso de evaluación de rendimiento con herramientas de análisis, aspectos que se asemejaron al de esta investigación, pero con la diferencia que en su desarrollo se incorporaron elementos externos del modelo alquilado, como por ejemplo bases de datos de tipo GIS.

El estudio propuesto por Akinbi y Pereira [18] desarrollaron un mapeo de requisitos de seguridad para identificar áreas críticas de seguridad de enfoque en modelos de nube PaaS con el objetivo de ofrecer un paradigma de seguridad interna. Este aporte tuvo mucha influencia en la evaluación de rendimiento y calidad, lo que otorgó fue un paradigma de medición enfocado a los puertos digitales de arranque y de red, este paradigma fue implementado en la evaluación de rendimiento de este estudio. Un elemento que hizo diferencia entre ambos estudios es que el mapeo de requisitos fue orientado a aplicaciones que tengan cadenas de conexiones externas del servidor que contiene el modelo de nube, mientras que en el estudio actual estuvo orientado a la seguridad interna y a la interacción de seguridad que tuvo la aplicación con la plataforma.

El estudio realizado por Gesvindr y Buhnova [19] tuvo como objetivo el diseño de aplicaciones por medio de arquitecturas logísticas tecnológicas incorporando técnicas de enlace a base de datos relacionales y no relacionales por medio de cadenas de conexión internas. En comparación con esta investigación, el proyecto programado tuvo una pequeña arquitectura de una base de datos relacionada integrada a la aplicación, mientras que en la aplicación de la investigación ya mencionada tuvo una arquitectura más avanzada pero que dependía del entorno, cosa que se diferenció a ambas investigaciones, la aplicación que se desarrolló no depende en su totalidad de los recursos de su entorno ya que su consumo es muy bajo por la arquitectura simple que presenta.

El trabajo realizado por Gesvindr *et al*. [20] tuvo como enfoque el diseño de un Controlador de calidad de aplicaciones PaaS en la nube usando prototipos generados, a comparación de esta investigación no se implementó ningún controlador o herramienta para controlar la calidad de la aplicación, pero se implementó una extensión de Microsoft que permitió realizar estabilización de servicios de aplicaciones y de control estándar de calidad para eventos de sobrecarga, que influyeron en las evaluaciones de rendimiento en algunas de las características.

La investigación propuesta por Gesvindr y Buhnova [21] tuvo como enfoque la utilización de herramientas de evaluación de calidad de aplicaciones usando PaaSArch para la mejora de tiempo de ejecución de la aplicación y el tiempo de respuesta de peticiones de usuario. El uso de herramientas de evaluación para una evaluación completa de rendimiento puede multiplicar el número de resultado que se puede obtener, para esta investigación se implementaron las herramientas de evaluación PaaSArch, mejoró la velocidad de ejecución de la aplicación, antes de la implementación el tiempo de ejecución era menor (lento) al promedio admitido (2 segundos como máximo), luego de su implementación el tiempo de respuesta aumentó siendo de 0,76 segundos (en algunos proveedores). Un punto que se consideró en la comparación e implementación de aspectos de ambas investigaciones fue el hecho de cómo se tomó el tiempo de respuesta, algo que influyó en la evaluación de rendimiento separando tiempo de respuesta y tiempo de ejecución y teniendo una perspectiva diferente en el aspecto de eficiencia de desempeño.

CAPITULO VI: CONCLUSIONES Y RECOMENDACIONES

Conclusiones.

Actualmente las empresas que poseen mayor taza de mercado en los servicios de computación en la nube son Google con su ya conocido Google Cloud, Microsoft con su servicio en la nube Azure y Amazon Web Services, aunque cabe recalcar que cada proveedor se diferencia de su competencia por entregar integraciones que permiten acoplar servicios externos o mejorar la experiencia de uso.

Cada modelo de servicio de computación en la nube está diseñado para un fin en específico, presentando ventajas y desventajas en sus diferentes niveles. Las infraestructuras como servicio siendo la base de la computación en la nube, son las que presentan una personalización, y administración completa de toda la interfaz hardware y software, permitiendo desde ahí el despliegue de los otros modelos de servicio (y hasta cierto punto establecer como operarán), sus únicos inconvenientes son la dificultad y el tiempo de configuración y establecimiento. Las plataformas como servicio están más orientadas a la administración y desarrollo de software donde uno puede establecer un entorno según como se desee personalizar, su desventaja radica en una ventaja del modelo anterior, el rendimiento de la plataforma se ve limitado según como se lo establezca en las configuraciones de la infraestructura (esto solo ocurre si la infraestructura es armada por un personal de recursos limitados, en un proveedor de servicios esta desventaja no existe por el hecho que uno puede seleccionar la cantidad de requerimientos que desee hasta antes del alquiler).

Las aplicaciones desarrolladas en Blazor son perfectas para proyectos para aplicaciones progresivas, que se actualicen constantemente o que se quieran adaptar a entornos digitales moldeables y personalizables. Las aplicaciones en Blazor tienen una estructura dividida en secciones que permite un trabajo directo, diseño simple y un despliegue rápido, con el aditamento que se puede trabajar en archivos similares a los Layout en los que se puede diseñar las vistas y su programación.

El rendimiento de aplicaciones desplegadas en plataforma como servicio presentan índices de rendimiento sobresalientes (claro dependiendo del del proveedor), presentaron

un comportamiento temporal alto, un mayor tiempo de respuesta, un consumo de recursos óptimo y una seguridad tanto interna como externa de alta calidad. Mientras, las aplicaciones desplegadas en infraestructura como servicio son un poco inestables, pero con algunas características. Las características de seguridad y tiempo de respuesta tienen calificaciones sobresalientes, son características manejadas propiamente por los proveedores de servicio, pero elementos como el comportamiento temporal, capacidad y utilización de recursos son más inestables por el hecho que este modelo de servicio lo otorgan sin configurar, se tiene que armar y configurar dependiendo de la necesidad para moldear el rendimiento.

Recomendaciones

Es una buena idea migrar a los servicios de computación en la nube, mientras más pasa el tiempo, más aplicaciones, servicios, páginas o elementos de software migran a entornos digitales en la nube, por razones como mayor conectividad entre servidor cliente, menores costes de mantenimiento y respaldo de la información. Es importante realizar investigaciones profundas y elegir el proveedor según el menú de precios que estos posean, las características que lo diferencian y sobre todo la necesidad y el contexto que se quiera cubrir.

Al igual que buscar el proveedor, se tiene que conocer exactamente las necesidades que se busca cubrir y conocer el proyecto que se va a realizar, si el proyecto se enfoca al desarrollo y despliegue de una aplicación sin tener que personalizar a futuro los requerimientos y componentes que la aplicación va a utilizar, se recomienda usar una plataforma, pero si se planea alterar o modificar sus recursos y entorno hardware, una infraestructura es muy recomendable, pero se tiene que tener algo de conocimiento básico para establecerlas de manera correcta.

Las aplicaciones desarrolladas con Frameworks de Microsoft desplegadas en modelos de servicio de Microsoft tienen un mejor arranque, mayor despliegue y mejor rendimiento que en otros servicios modelos de servicio.

Se recomienda tener conocimiento previo antes de comprar y utilizar una infraestructura como servicio, la manipulación y configuración de estos modelos son muy difíciles y en algunos casos muy poco intuitivos, además de que las tablas de trabajo tienen tantas características que para alguien nuevo se le puede hacer muy abrumador. Las

plataformas como servicio son perfectas para el despliegue de aplicaciones, el entorno es completo y preparado para trabajar, además que es muy sencillo implementar nuevas herramientas que expandan la funcionalidad de las aplicaciones desplegadas o del propio modelo de servicio.

REFERENCIAS

[1] Management Solutions, “La nube: oportunidades y retos para los integrantes de la cadena de valor”, *Articulo*, p. 38, 2012.

[2] Gartner, “Amazon, Microsoft y Alibaba, los tres mejores proveedores de servicios IaaS”, 2017. https://www.computerworld.es/tendencias/amazon-microsoft-y-alibaba-los-tres-mejores-proveedores-de-servicios-iaas#:~:text=Amazon%2C Microsoft y Alibaba%2C los tres mejores proveedores de servicios IaaS,-Tags%3A Cloud Computing.

[3] RedHat, “IAAS PAAS SAAS”. .

[4] R. Yasrab y N. Gu, “Multi-cloud PaaS Architecture (MCPA): A Solution to Cloud Lock-In”, *Proc. - 2016 3rd Int. Conf. Inf. Sci. Control Eng. ICISCE 2016*, pp. 473–477, 2016, doi: 10.1109/ICISCE.2016.108.

[5] J. I. Herranz, “Composicion IAAS y PAAS”. .

[6] NIST, “The NIST-National Institute of Standars and Technology- Definition of Cloud Computing”, *NIST Spec. Publ. 800-145*, p. 7, 2011.

[7] D. J. Mannella Lemos, “Diseño de una guía para la implementación del uso de computación en la nube como mecanismo de recuperación ante desastres tecnológicos en PYMES en el DMQ: Maestría en Gerencia de Sistemas. ESPE. Sede Sangolquí.”, *Sangolquí / Espe / 2012*, 2012.

[8] R. Kade, “SAAS”. .

[9] O. Dinarle, “Un Framework De Programación Web”, *Télématique*, vol. 15, nº 2, pp. 144–171, 2017.

[10] Microsoft, “Introducción a ASP.NET Core Blazor”, *Microsoft*, vol. 53, nº 9, pp. 1689–1699, 2020.

[11] N. Shahroze, “Why-ReactJS-Should-be-a-Perfect-Choice-for-Your-Next-Front-end-Application-Banner”. CouldWays, 2018.

[12] M. C. Grandes, “Introducción a Vue.js”, *Vue.JS*, p. 7, 2017.

[13] D. Britch y Olprod, “Patrones MVVM”, *Microsoft Ignite*, p. 96, 2017.

[14] D. Britch y Olprod, “El patrón MVVM”. Microsft Ingnite, 2017.

[15] Q. Xia, Y. Lan, y L. Xiao, “The Status Prediction of Physical Machine in IaaS Cloud Environment”, *Proc. - 2015 Int. Conf. Cyber-Enabled Distrib. Comput. Knowl. Discov. CyberC 2015*, pp. 302–305, 2015, doi: 10.1109/CyberC.2015.100.

[16] X. Li, B. Gu, C. Zhang, K. Yamori, y Y. Tanaka, “Price competition in a duopoly IaaS cloud market”, *APNOMS 2014 - 16th Asia-Pacific Netw. Oper. Manag. Symp.*, pp. 5–8, 2014, doi: 10.1109/APNOMS.2014.6996552.

[17] K. Kiwon, Lee. Kwangseob, “GEO-BASED IMAGE ANALYSIS SYSTEM SUPPORTING OGC-WPS STANDARD ON OPEN PAAS CLOUD PLATFORM Kiwon Lee and Kwangseob Kim”, *IGARSS 2018 - 2018 IEEE Int. Geosci. Remote Sens. Symp.*, pp. 5262–5265, 2018, doi: 10.1109/IGARSS.2018.8517646.

[18] A. Akinbi y E. Pereira, “Mapping security requirements to identify critical security areas of focus in PaaS cloud models”, *Proc. - 15th IEEE Int. Conf. Comput. Inf. Technol. CIT 2015, 14th IEEE Int. Conf. Ubiquitous Comput. Commun. IUCC 2015, 13th IEEE Int. Conf. Dependable, Auton. Se*, pp. 789–794, 2015, doi: 10.1109/CIT/IUCC/DASC/PICOM.2015.116.

[19] D. Gesvindr y B. Buhnova, “Architectural tactics for the design of efficient PaaS cloud applications”, *Proc. - 2016 13th Work. IEEE/IFIP Conf. Softw. Archit. WICSA 2016*, pp. 158–167, 2016, doi: 10.1109/WICSA.2016.42.

[20] D. Gesvindr, B. Buhnova, y O. Gasior, “Quality Evaluation of PaaS Cloud Application Design Using Generated Prototypes”, *Proc. - 2017 IEEE Int. Conf. Softw. Archit. ICSA 2017*, pp. 31–40, 2017, doi: 10.1109/ICSA.2017.43.

[21] D. Gesvindr y B. Buhnova, “PaaSArch: Quality Evaluation Tool for PaaS Cloud Applications Using Generated Prototypes”, *Proc. - 2019 IEEE Int. Conf. Softw. Archit. - Companion, ICSA-C 2019*, pp. 170–173, 2019, doi: 10.1109/ICSA-C.2019.00038.

[22] Asamblea del Ecuador, “Proyecto de ley organica de protección de datos”. p. 52, 2019.

[23] Registro oficial: Organo del Gobierno del Ecuador, “Código Orgánico De La

Economía Social De Los Conocimientos, Creatividad E Innovación", *Regist. Of.*, vol. IV, p. 113, 2016.

[24] P. Barrezueta, "Última Reforma: LEY ORGÁNICA DE EMPRENDIMIENTO E INNOVACIÓN", nº 151, pp. 1–49, 2020.

[25] E. Del Canto y A. Silva Silva, "Metodologia Cuantitativa: Abordaje Desde La Complementariedad En Ciencias Sociales", *Rev. Ciencias Soc.*, vol. 0, nº 141, 2013, doi: 10.15517/rcs.v0i141.12479.

[26] H. Salas, "Investigación Cuantitativa (Monismo Metodológico) y Cualitativa (Dualismo Metodológico): El status epistémico de los resultados de la investigación en las disciplinas sociales", *Cinta de moebio*, nº 40, pp. 1–21, 2011, doi: 10.4067/s0717-554x2011000100001.

[27] A. Marradi, "Método experimental , método de la asociación y otros", *Paradigmas*, vol. 5, nº 1, pp. 11–38, 2013.

[28] S. Barnet-Lopez, M. Arbonés-Garcia, S. Pérez-Testor, y M. Guerra-Balic, "Construcción Del Registro De Observación Para El Análisis Del Movimiento Fundamentado En La Teoría De Laban", *Pensar en Mov. Rev. Ciencias del Ejerc. y la Salud*, vol. 15, nº 2, p. 27334, 2017, doi: 10.15517/pensarmov.v15i2.27334.

[29] A. Acevedo Borrego, C. Linares Barrantes, y O. Cachay Boza, "Investigación en la acción. Un ejemplo de estudio experimental en el mercadeo de servicios", *Ind. Data*, vol. 16, nº 2, p. 79, 2016, doi: 10.15381/idata.v16i2.11925.

[30] M. E. Dulzaides Iglesias y A. M. Molina Gómez, "Análisis documental y de información: Dos componentes de un mismo proceso", *Acimed*, vol. 12, nº 2, 2004.

[31] P. Faraldo, "Estadística y metodología de la investigación", *Univ. Santiago Compost.*, p. 15, 2013.

[32] B. Evans, "The Top 5 Cloud-Computing Vendors: #1 Microsoft, #2 Amazon, #3 IBM, #4 Salesforce, #5 SAP", *Forbes Contributors*, 2017. https://www-forbes-com.cdn.ampproject.org/c/s/www.forbes.com/sites/bobevans1/2017/11/07/the-top-5-cloud-computing-vendors-1-microsoft-2-amazon-3-ibm-4-salesforce-5-sap/amp/ (accedido jun. 04, 2021).

[33] Larry Dignan, “Top cloud providers in 2021: AWS, Microsoft Azure, and Google Cloud, hybrid, SaaS players”, *ZDNet*, 2021. https://www.zdnet.com/article/the-top-cloud-providers-of-2021-aws-microsoft-azure-google-cloud-hybrid-saas/ (accedido jun. 07, 2021).

ANEXOS

Anexo 1

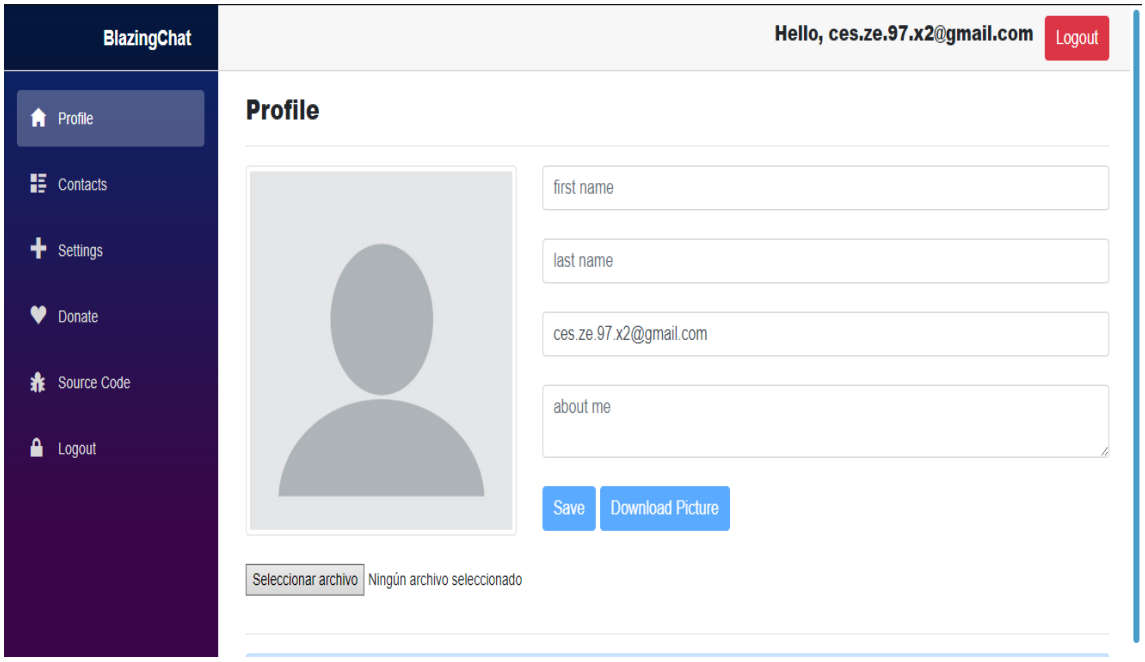

Figura 27 Blazor Chating 1

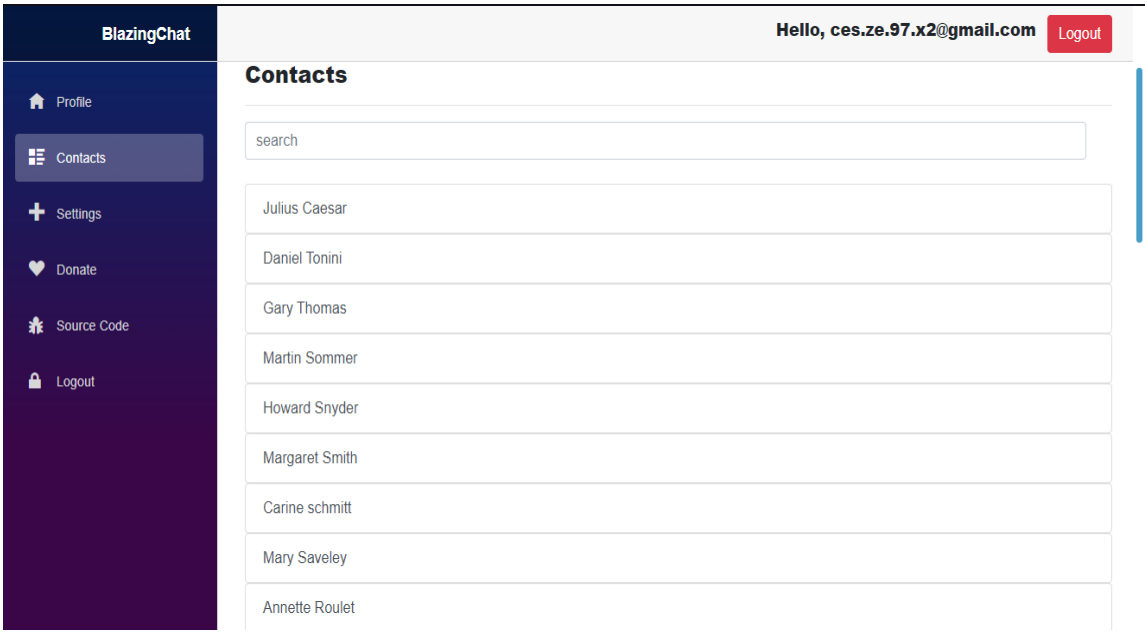

Figura 28 Blazor Chating 2

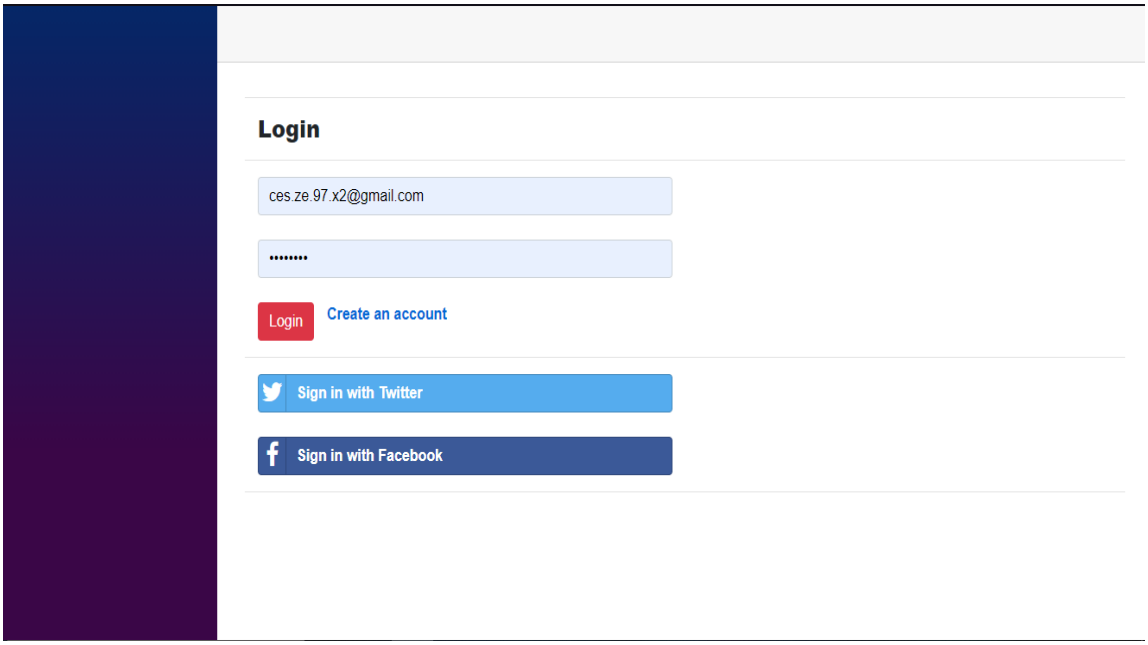

Figura 29 Blazor Chating 3

Anexo 2

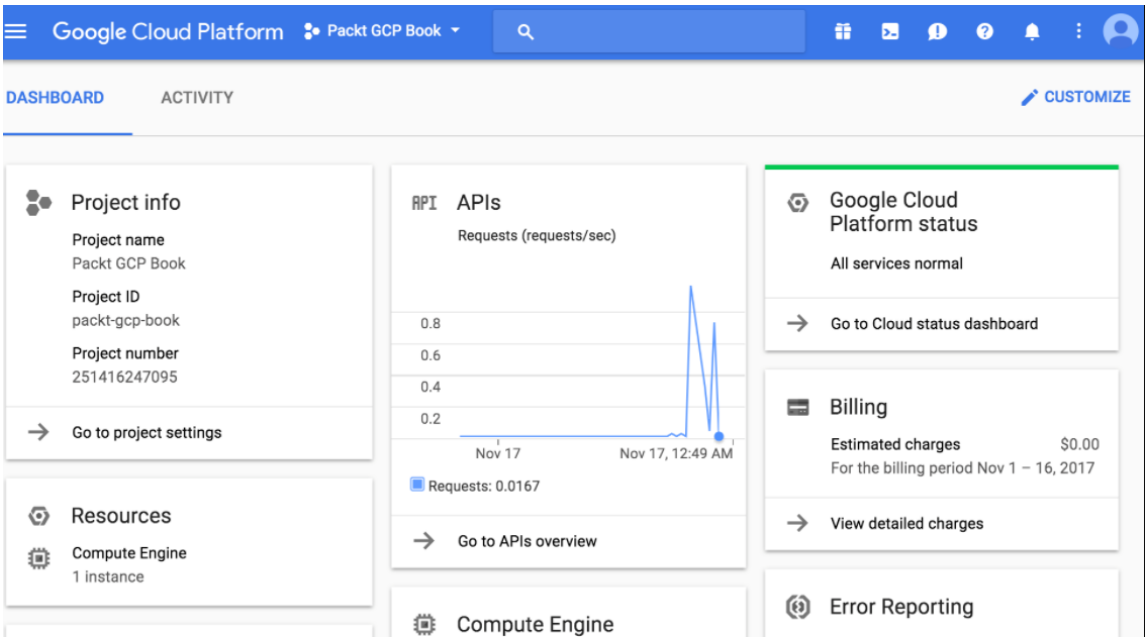

Figura 30 Dashboard Google Cloud

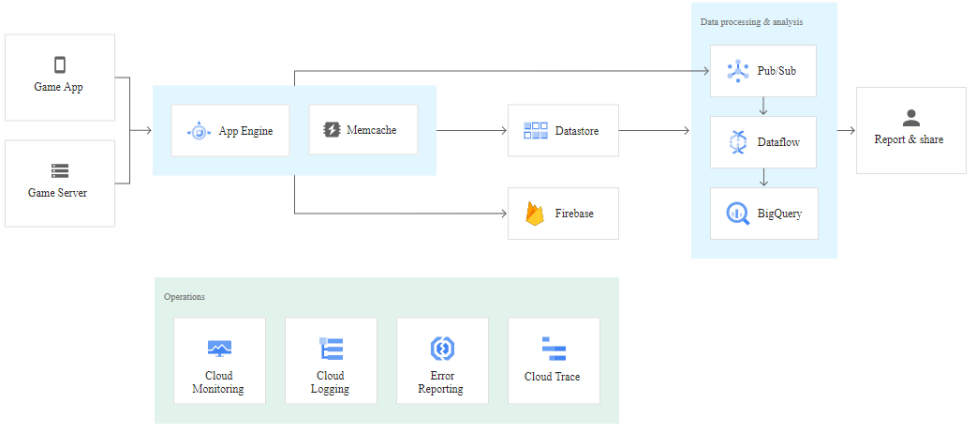

Figura 31 Estructura de Google Cloud

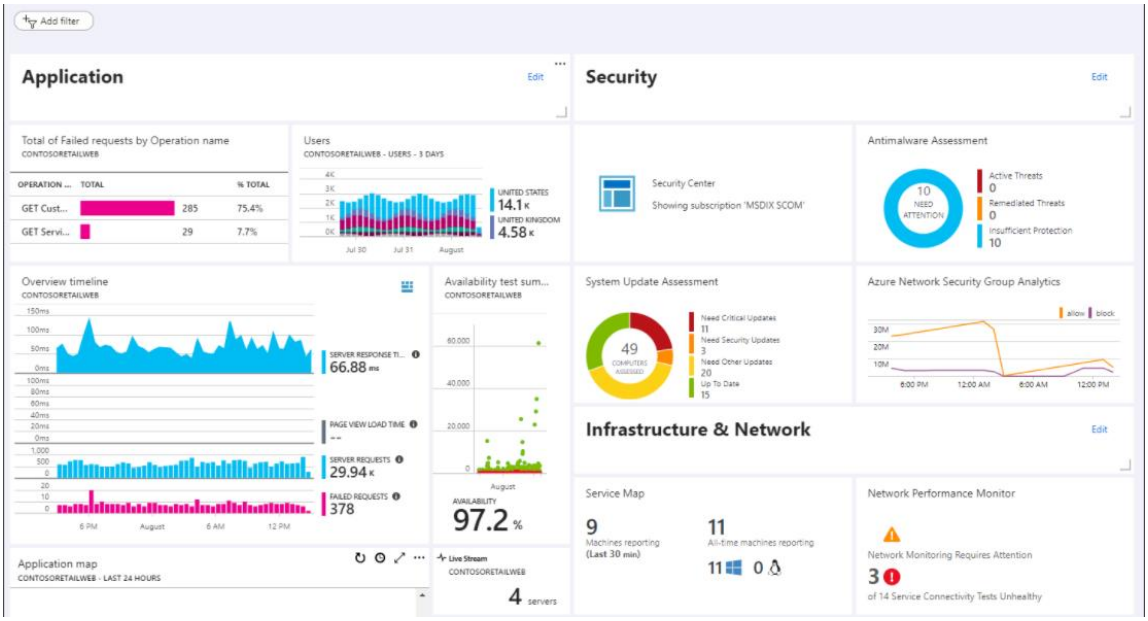

Figura 32 Dashboard de Microsoft Azure

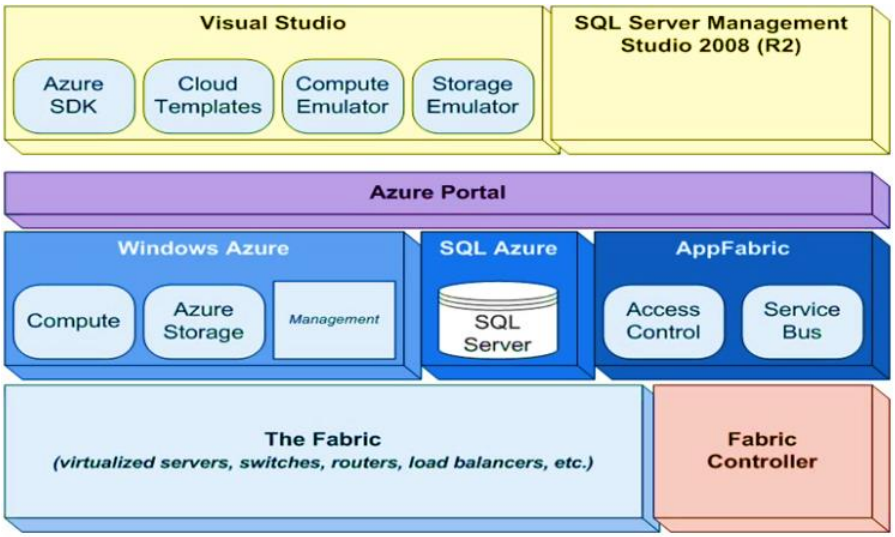

Figura 33 Arquitectura de Microsoft Azure

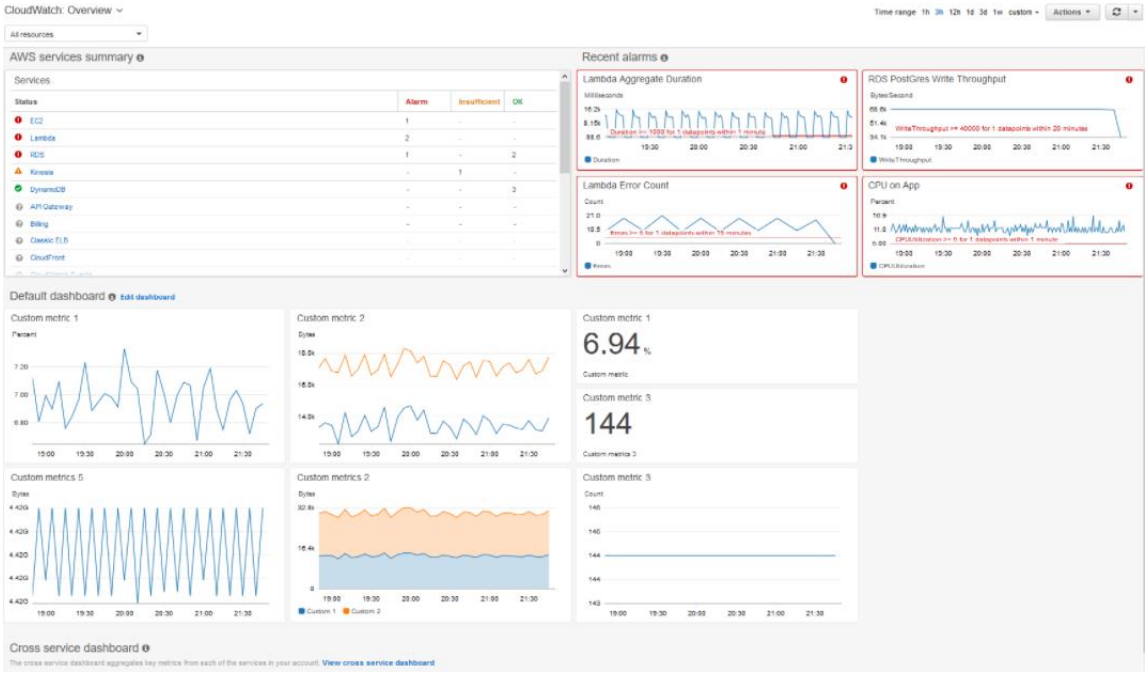

Figura 34 Dashboard de AWS

Anexo 3

Repositorio de la Aplicación: https://github.com/CesarGracia97/BlazProyect