

**Tesis**

**Evaluación de capacidades de Calidad de Servicio  
de proveedores de computación en la nube.**

Qué para obtener el grado de  
**Maestro en ingeniería electrónica.**

Presenta:

**Ing. José Alberto López Cázarez.**

DIRECTOR DE TESIS:

**M.C. Juan Francisco Ibáñez Salas.**

CO - DIRECTOR DE TESIS:

**Dr. Arnoldo Díaz Ramírez.**

## Prólogo

Este trabajo se realizó con el fin de conocer más de los nuevos paradigmas referente al área de sistemas computacionales. Con el gran avance de la tecnología y la ciencia se desarrollan nuevas herramientas de trabajo por tal motivo se inicia este trabajo. En particular, el objetivo de este trabajo es el de conocer y evaluar el desempeño de proveedores de computación en la nube, específicamente con base en sus capacidades de calidad de servicio (QoS).

Un aspecto interesante dentro del trabajo de tesis fue el realizar la investigación y selección de proveedores de plataforma como servicio, para llegar al objetivo general, ya que en el mercado existen muchas empresas que brindan este tipo de servicios.

Al desarrollar la aplicación se presentaron problemas ya que las herramientas que se utilizan para cada proveedor son diferentes a pesar que se manejó el mismo lenguaje de programación. Pero una vez que se empezó a trabajar con las herramientas del proveedor y se entendió el funcionamiento, la implementación se tornaba cada vez más sencilla.

## Agradecimientos.

Quiero expresar mis agradecimientos a mis padres que me brindaron su apoyo incondicional. También agradecer a mi novia que fue una pieza importante para la realización de este trabajo. No puedo perder el momento para agradecer a mi asesor Francisco Ibáñez por la ayuda y la tutoría ofrecidas sin olvidar mencionar a los maestros integrantes del departamento de posgrado por el apoyo y consejos brindados en la etapa del desarrollo del trabajo. Muchas gracias.

## Índice General

### Capítulo 1

1.1 Introducción .....	6
1.2 Objetivo General .....	7
1.3 Objetivos Específicos .....	7

### Capítulo 2

2.1 Servicios web .....	8
2.2 Computación en la nube .....	8
2.3 Características de la nube .....	9
2.4 Entidades de la nube .....	11
2.5 Modelos de entrega de la nube .....	12
2.6 Modelos de despliegue de la nube .....	14
2.7 Propuesta .....	16

### Capítulo 3

3.1 Marco conceptual .....	21
3.1.1 Marco conceptual de OpenShift .....	28
3.1.1.1 Desarrollo de un web service en OpenShift .....	29
3.1.2 Marco conceptual de Azure .....	33
3.1.2.1 Desarrollo de un web service en Azure .....	34
3.1.3 Marco conceptual de Koding .....	36
3.1.3.1 Desarrollo de un web service en Koding .....	38
3.2 Diseño de aplicación .....	39
3.3 Implementación .....	40
3.3.1 Implementación en la plataforma OpenShift .....	41
3.3.2 Implementación en la plataforma Azure .....	44
3.3.3 Implementación en la plataforma Koding .....	45
3.4 Evaluación del desempeño de los “Enterprise Service Bus” de acuerdo a parámetros de orquestación .....	45

### Capítulo 4

4.1 Conclusiones .....	51
4.2 Bibliografía .....	52
4.3 Apéndice A .....	56

## Índice de figuras

Figura 1. Modelos de entrega de la nube .....	13
Figura 2. Proveedores de Servicio .....	16
Figura 3. Porcentaje de uso de usuarios de la nube .....	17
Tabla 1. Proveedores de servicio PaaS .....	17
Tabla 2. Proveedores open source de plataforma como servicio en la nube .....	19
Figura 4. Aplicación cliente de servicio web .....	22
Figura 5. Aplicación que consume el servicio web creado en Java .....	26
Figura 6. Pantalla de registro .....	28
Figura 7. Herramientas ofrecidas por OpenShift. ....	29
Figura 8. Pagina de bienvenida .....	29
Figura 9. Publicación de servicio web .....	32
Figura 10. Pantalla de registro .....	33
Figura 11. Editor de texto online .....	34
Figura 12. Pantalla de bienvenida .....	34
Figura 13. Creación del servicio web .....	35
Figura 14. Publicación del servicio web .....	35
Figura 15. Página de inicio de Koding .....	36
Figura 16. Terminal de Koding .....	37
Figura 17. Editor de texto Ace en Koding .....	37
Figura 18. Pagina por defecto que Koding da como ejemplo .....	38
Figura 19. Creación del servicio web .....	38
Figura 20. Publicación del servicio web .....	39
Figura 21. Diseño de la aplicación .....	40
Tabla 3. Latencia en el servicio de kardex .....	46
Tabla 4. Latencia en el servicio de extraescolares .....	46
Tabla 5. Latencia en el servicio de beca .....	46
Figura 22-A. Tiempo de procesamiento para el servicio kardex .....	47
Figura 22-B. Tiempo de procesamiento para el servicio kardex .....	48
Figura 23-A. Tiempo de procesamiento para el servicio extra .....	48
Figura 23-B. Tiempo de procesamiento para el servicio extra .....	49
Figura 24-A. Tiempo de procesamiento para el servicio beca .....	49
Figura 24-B. Tiempo de procesamiento para el servicio beca .....	50

# Capítulo 1

## 1.1 Introducción

Debido al gran desarrollo tecnológico, el acceso a la web se ha utilizado como medio de comunicación, medio de entretenimiento e indispensable para el trabajo, esto ha orillado a la ciencia a buscar nuevas formas de ofrecer servicios con mejor desempeño, facilidad de acceso y al momento de desarrollar e implementar una aplicación se necesita agilidad y rapidez ya que las empresas no obtienen ganancias cuando se invierte tiempo a estas acciones. También se investiga la manera de poder explotar la amplia gama de oportunidades que nos ofrece el internet.

Anteriormente se ofrecían servicios en la web para poder comunicar sistemas que muchas veces estaban desarrollados en diferentes lenguajes y plataformas. Estos servicios se basaban básicamente en el intercambio de archivos XML para la comunicación de los sistemas.

Al buscar otras formas de cómo poder ofrecer nuevos servicios, nos lleva a la evolución de los servicios web dando como resultado la Computación en la nube (Cloud Computing). Este paradigma ya no solo ofrece la comunicación entre sistemas, sino que también se puede brindar entre muchas otras cosas como lo es la infraestructura, la cual inicialmente las empresas tenían que adquirir, armar y poner en funcionamiento, impactando fuertemente en el presupuesto de las empresas.

## 1.2 Objetivo General

Identificar la mejor opción de proveedores de plataforma como servicio (PaaS) en la nube evaluando plataformas entre distribución comercial y distribución Open Source, en base al desempeño de calidad de servicios (QoS).

## 1.3 Objetivos Específicos

Buscar proveedores de plataformas para ser evaluadas.

Definir plataformas para hacer la evaluación.

Definir características y métricas a evaluar.

Diseñar un sistema para implementarlo en diferentes plataformas en la nube.

Implementar el sistema en plataforma en la nube de distribución Open Source.

Implementar el sistema en plataforma en la nube de distribución comercial.

Hacer comparaciones y tablas estadísticas para visualizar los resultados y obtener conclusiones.

## Capítulo 2

### 2.1 Servicios web

Un servicio web es un módulo de software autónomo disponible a través de una red, tal como Internet, que completa tareas, resuelve problemas, o lleva a cabo transacciones en nombre de un usuario o aplicación. Los servicios Web constituyen una infraestructura informática distribuida formada por diferentes módulos de aplicación que interactúan tratando de comunicarse a través de redes públicas o privadas (incluyendo Internet) para formar prácticamente un solo sistema lógico. Los servicios Web pueden variar en función de las solicitudes simples (por ejemplo, la comprobación de crédito y autorización, consultas de precios, comprobar el estado de un inventario, o un informe del tiempo) o combinar información de varias fuentes, tales como un sistema de intermediación de seguros, un cálculo de seguro de responsabilidad civil, un planificador de viajes automatizado o un sistema de seguimiento de paquetes [16].

Existen muchas definiciones sobre los Servicios Web, lo que muestra un gran problema al querer dar un concepto adecuado que englobe todo lo que son e implican. Una buena definición sería hablar de ellos como un conjunto de aplicaciones o de tecnologías con capacidad para comunicarse y trabajar entre sí en la Web. Estas aplicaciones o tecnologías intercambian datos entre ellas con el objetivo de ofrecer servicios. Los proveedores del servicio los ofrecen como procedimientos remotos y los usuarios solicitan un servicio llamando a estos procedimientos a través de la Web.

En los años 2008 y 2009 hubo una popularización y consolidación de un nuevo paradigma tecnológico “Computación en Nube” (Cloud Computing). En Agosto de 2008 la revista “Bloomberg Business Week” y en octubre de 2008 la revista “The Economist”, dos de las grandes cabeceras mundiales de revistas económicas ya miraban el pronto advenimiento de este paradigma publicando artículos importantes en sus ediciones. Desde ese periodo a la fecha este paradigma ha tomado demasiada relevancia, ya que muchas empresas han cambiado la manera de manejar sus sistemas decidiendo adaptarlos hacia este nuevo paradigma [8].

### 2.2 Computación en la nube (Cloud Computing)

No existe una definición estándar aceptada universalmente; sin embargo, existen muchos autores que hacen sus propios conceptos:



*“Un estilo de computación en el que las capacidades de IT son habilitadas, escalables, elásticas y se entregan como un servicio a los clientes externos utilizando las tecnologías de Internet”. Gartner (2008)*

*“Capacidades de TI estandarizadas (servicio, software e infraestructura) que se entregan a través de las tecnologías de Internet en un pago por uso, modo autoservicio”. Forrester.*

*“Computación en la nube es un modelo ubicuo, conveniente, para habilitar el acceso a la red bajo demanda para compartir un grupo de recursos informáticos configurables (red, servidores, almacenamiento, aplicaciones y servicios) que pueden ser rápidamente provisionados y puesto en marcha con un esfuerzo mínimo de gestión o interacción del proveedor de servicios. Este modelo se compone de cinco características esenciales, tres modelos de servicio y cuatro modelos de implementación”. National Institute of Standards and Technology (2011).*

La computación en la nube es una forma especializada de computación distribuida que introduce modelos de utilización para la provisión de forma remota de recursos escalables y medidos [3].

La computación en la nube es un paradigma el cual nos ayuda a ofrecer servicios de computación a través de internet. Ya no simplemente es ofrecer servicios web, como anteriormente se hacía, ahora se ofrece todo tipo de servicios enfocados a recursos de IT (tecnologías de la información). Estos servicios pueden brindar una amplia gama de recursos como lo son: a) Infraestructura: este servicio brinda al cliente un conjunto de dispositivos mediante máquinas virtuales, b) Plataforma: esta opción brinda al cliente una estación de trabajo para desarrollar e implementar sus aplicaciones donde no se necesita la instalación de ningún tipo de aditamento ya que el proveedor brinda todas las herramientas necesarias para este fin, c) Software: en este servicio el proveedor pone a disposición un sistema o programa para el cliente lo utilice a través de internet.

## 2.3 Características de la nube.

En el libro de Erl, Mahmood y Puttini [3], se dice que en el paradigma de la nube existen 6 características las cuales son las más comunes en este entorno:

- Uso bajo demanda.
- Acceso ubicuo.
- Múltiples inquilinos.
- Uso medido.
- Flexibilidad.
- Elasticidad.

Uso bajo demanda. El consumidor de la nube puede acceder a los recursos de la nube dando al consumidor la libertad de elegir y configurar estos recursos. Una vez configurado, el uso de estos recursos es automatizado, no requiere de la intervención humana adicional por el consumidor o el proveedor de la nube. El consumidor no necesita realizar algún tipo de conexión o instalación de hardware ya que el proveedor ofrece la opción de adquirirlos mediante una máquina virtual. Dando como resultado un entorno de uso bajo demanda.

Acceso ubicuo. El acceso ubicuo representa la habilidad del servicio de la nube sea altamente accesible. El establecimiento de un acceso ubicuo a un servicio en la nube puede requerir el apoyo a una amplia gama de dispositivos, protocolos de transporte, interfaces y tecnologías de seguridad. El consumidor del servicio tiene la facilidad de acceder a él en cualquier parte del mundo mediante la conexión a internet.

Múltiples inquilinos. Es la característica de un programa informático que permite a una instancia del sistema servirle a diferentes consumidores (inquilinos), por lo cual cada uno está aislado del otro. El proveedor de la nube junta sus recursos de IT para atender a múltiples consumidores de servicios de la nube mediante el uso de esta característica (múltiples inquilinos) que con frecuencia se basa en el uso de tecnologías de virtualización. Esto es el proveedor de la nube le ofrece a cada consumidor una Máquina virtual teniendo la posibilidad de brindarle todos sus recursos a través de esta.

Uso medido. El uso medido es la característica que representa la habilidad de una plataforma en la nube de poder realizar un seguimiento del uso de sus recursos. De esta manera el proveedor de la nube puede cobrarle al consumidor por el tiempo que ha utilizado sus recursos. Esta característica ayuda a las empresas a controlar su presupuesto y pagar solamente por lo que se ha consumido. Por esta característica hace que el paradigma de computación en la nube se asemeje a los servicios públicos brindados por el estado, ya que solo se facturará el uso que se le haya dado en el lapso de tiempo acordado por el consumidor y el proveedor de la nube.

Flexibilidad. Computación flexible es una forma de tolerancia de error que distribuye implementaciones redundantes de recursos a través de la ubicación física. Los recursos de IT pueden ser preconfigurados, y por eso se convierte en defectuoso y propensos a cometer fallos. Dentro de la computación en la nube la característica de la flexibilidad puede referirse a los recursos redundantes dentro de la misma nube o en diferentes nubes. Esto ayuda a que el consumidor no pierda el acceso al servicio, ya que si existe una catástrofe donde se encuentra físicamente almacenado el servicio exista una opción de poder utilizar el servicio desde otra parte donde se encuentre una réplica del servicio. Ayudando así a evitar la caída del servicio.

Elasticidad. Es la habilidad automatizada de la nube de escalar de forma transparente los recursos de IT, según sea necesario, en respuesta a las condiciones de tiempo de ejecución o predeterminado por el consumidor o el proveedor de la nube. La elasticidad es a menudo considerada por las empresas una justificación fundamental para la adopción de computación en la nube, principalmente debido al hecho de que está estrechamente ligado con la reducción de la inversión y proporcional al costo beneficio. Ya que la empresa no necesita invertir mucho en la adquisición de equipo (hardware), porque esto lo brinda el proveedor de la nube. De aquí se desprende la escalabilidad [22].

La escalabilidad representa la habilidad de incrementar o disminuir el manejo de recursos de IT dependo la demanda. Existen dos tipos de escalabilidad: horizontal y vertical [3].

La escalabilidad horizontal se refiere asignación o liberación de los recursos que son del mismo tipo. Por ejemplo la asignación de un servidor virtual, puede ser asignado una o varias veces.

La escalabilidad vertical ocurre cuando un recurso existente es reemplazado por otro con mayor o menor capacidad. Cuando el recurso es reemplazado por uno con mayor capacidad se le conoce como escalabilidad hacia arriba y cuando es reemplazado por uno menor se le conoce como escalabilidad hacia abajo.

## 2.4 Entidades de la nube.

En toda organización y sistema existen personas o programas que intervienen en el ciclo de vida de un sistema en este paradigma existen dos entidades muy importantes que interactúan con la nube [3].

Proveedor de la nube (Cloud Provider).

La organización que proporciona recursos de IT basados en la nube. Cuando se asume el rol de proveedor de servicios, La organización es responsable de hacer servicios en la nube disponibles para los consumidores de la nube según lo acordado en las garantías del acuerdo de nivel de servicio (SLA). El proveedor de la nube tiene además la tarea de cualquier gestión necesaria y las tareas administrativas para garantizar el funcionamiento continuo de la infraestructura global de la nube [3].

El proveedor de la nube debe asumir la responsabilidad dependiendo del servicio ofrecido ya que la configuración de muchas opciones no las hace el que provee el servicio siendo así responsabilidad del consumidor.

Consumidor de la nube (Cloud Consumer).

El consumidor de la nube es una organización o humano que tiene un contrato formal o un arreglo con el proveedor de la nube para usar los recursos de IT puestos a disposición por el proveedor de la nube [3]. El consumidor juega un papel muy

importante porque es quien va a evaluar los servicios adquiridos, puede existir la posibilidad de que el consumidor configure mal las opciones que le brinda el proveedor, cayendo totalmente la responsabilidad del mal funcionamiento de la nube al consumidor.

Una persona u organización que posee legalmente un servicio en la nube es llamado propietario del servicio en la nube. El propietario de un servicio en la nube puede ser el consumidor de la nube ya que es este quien lo construye y pone en funcionamiento, el proveedor de la nube que es dueño de la nube donde reside todos los datos del servicio también puede ser propietario del servicio, ya que es este quien lo ofrece a un consumidor.

El administrador de recursos de la nube es la persona u organización responsable de administrar los recursos de IT basados en la nube (incluyendo servicios). El administrador de recursos de la nube puede ser el consumidor de la nube o el proveedor de la nube dentro del cual reside el servicio. También puede ser una organización externa contratada para administrar los recursos.

## 2.5 Modelos de entrega de la nube

Los modelos de entrega de servicios en la nube representan una combinación de recursos pre empaquetados ofrecidos por un proveedor. Los modelos más usuales que se ofrecen a los clientes y usuarios de la nube (organizaciones, empresas y usuarios) son: SaaS (Software as a Service), software como servicio, PaaS (Platform as a Service), plataforma como servicio e IaaS (Infraestructura as a Service), infraestructura como servicio [3].

**Software como Servicio:** El modelo software como servicio se refiere esencialmente al software residente (instalado) en la nube puesto a disposición del consumidor a través de un navegador web, el cliente puede acceder a este servicio a través de cualquier dispositivo con conexión a internet. En este modelo el usuario no sabe, ni necesita conocer donde está alojado el software, en que sistema operativo está instalado o que lenguaje se utilizó para desarrollarlo (PHP, Java o .Net) y además el usuario no tiene que instalar ningún programa de software como se hace actualmente. El consumidor no gestiona ni controla la infraestructura adyacente de la nube como pueden ser: sistemas operativos, servidores ni incluso las características o funcionalidades de las aplicaciones individuales con la excepción de la posible configuración que pueda requerir en el lado del usuario.

**Plataforma como Servicio:** el proveedor ofrece un entorno de desarrollo al consumidor del modelo, esencialmente para desarrolladores de sistemas, quienes crean

sus aplicaciones y ofrecen sus servicios a través de la plataforma del proveedor. El proveedor normalmente ofrece para el desarrollo kits de herramientas (toolkits); lenguajes de programación, herramientas, estándares de desarrollo y canales de distribución. El proveedor recibe un pago por proporcionar la plataforma y los servicios de distribución y ventas. Este modelo permite el desarrollo y programación de aplicaciones de software.

El modelo PaaS es muy útil ya que facilitan a los desarrolladores y empresas pequeñas a desplegar aplicaciones basadas en la web sin el coste y complejidad que supondría la compra de servidores y sus correspondientes configuraciones y puestos en funcionamiento. En este modelo el consumidor se salva de la carga administrativa de la creación y el mantenimiento de los recursos de IT de la infraestructura. El usuario solo es responsable de la creación y mantenimiento del servicio que desarrolle en la nube.

Infraestructura como Servicio: Este modelo proporciona la infraestructura necesaria para crear, ejecutar y desplegar aplicaciones. El modelo IaaS ofrece espacio de almacenamiento, capacidad de procesamiento, servidores, sistemas operativos, esto lo hace a través de una máquina virtual. El consumidor no gestiona ni controla la infraestructura fundamental de la nube, pero tiene el control sobre sistemas operativos, almacenamiento y aplicaciones desplegadas y es posible un control limitado de recursos. En la práctica el cliente IaaS “alquila” (paga por uso y prestaciones) recursos informáticos, en lugar de comprarlos e instalarlos.

En la figura 1 muestra hasta donde el proveedor del servicio es responsable del mantenimiento y funcionamiento.

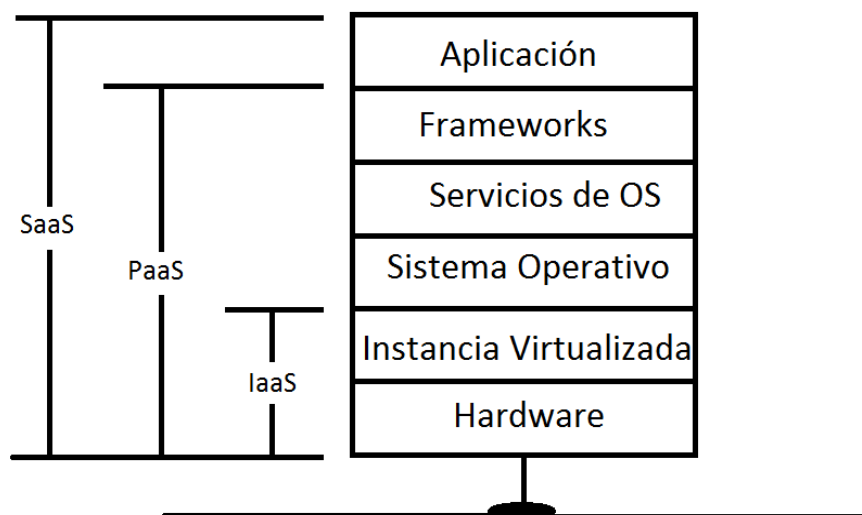


Figura 1. Modelos de entrega de la nube.

## 2.6 Modelos de despliegue de la nube

Los modelos de implementación representan un tipo específico de entorno en la nube distinguido principalmente por la propiedad, tamaño y acceso.

De acuerdo con el libro de Erl, Mahmood y Puttini [3] existen 4 modelos comunes de despliegue de la nube:

- Nube pública.
- Nube comunitaria.
- Nube privada.
- Nube híbrida.

Nube pública: Es un entorno de nube de acceso público que es propiedad de un proveedor de la nube. Los recursos de IT son puestos a disposición del usuario, dependiendo del modelo de entrega de la nube.

Una nube pública está alojada, operada y gestionada por un proveedor en el cual cae la responsabilidad de la creación y el mantenimiento continuo de la nube y de sus recursos. El servicio se ofrece a múltiples clientes mediante una infraestructura común. En una nube pública, la gestión de la seguridad y las operaciones son controladas por un proveedor que es responsable de la oferta de servicios de la nube. Por estas razones se tiene un control muy bajo de la seguridad física y lógica.

Nube comunitaria: Este modelo es parecido al de la nube pública con la excepción que el acceso es limitado a una específica comunidad de consumidores. La nube comunitaria puede ser propiedad conjunta por miembros de la comunidad o un proveedor externo que provee una nube pública con accesos limitados. Los miembros de la comunidad consumidores de la nube comparten la definición y evolución de la nube comunitaria.

Nube privada: Las nubes privadas o nubes internas se refieren al funcionamiento de las nubes de un modo similar a una red o centro de datos privado. La infraestructura de la nube es propiedad y gestionada por una única organización, bien directamente o por terceras partes y puede existir en la organización (on-premise) o bien fuera de la organización (off-premise).

En una nube privada, la organización cliente establece un entorno de virtualización en sus propios servidores, en cualquiera de sus propios centros de datos o en los de un proveedor de servicios. Las organizaciones deben comprar, construir y gestionar la nube a cambio de tener el control de la misma, sus costes y condiciones de gestión serán más altos. Los clientes organizacionales y empresariales de una nube privada son los responsables del funcionamiento de la misma.

La estructura de nube privada es útil para empresas que o bien tienen inversiones o costes significativos de sus tecnologías de la información o consideran que deben tener un control total sobre los diferentes aspectos de infraestructura. La ventaja principal de las nubes privadas es el control. Se tiene el control sobre su infraestructura y se ganan todas las ventajas de la virtualización.

En general, en un modelo de funcionamiento de nube privada, la gestión de la seguridad y las operaciones diarias de los servicios alojados son responsabilidad de la organización o una empresa externa a la que se ha subcontratado. En consecuencia, en este modelo de gobierno directo, un cliente de una nube privada debe tener un alto grado de control sobre los aspectos físicos y lógicos de la seguridad de la infraestructura de la nube y en consecuencia será más fácil para el cliente cumplir los estándares, políticas y regulación de la seguridad.

Nube híbrida: Este modelo de nube híbrida es una composición de dos o más tipos de modelos de despliegue. Con una nube híbrida las organizaciones pueden ejecutar aplicaciones no fundamentales en una nube pública, mientras mantienen las aplicaciones fundamentales y los datos sensibles internos en una nube privada.

## 2.7 Propuesta

Uno de los objetivos asentados en este trabajo es el de buscar proveedores de plataforma como servicio en la nube, por tal motivo encontramos que los distribuidores más utilizados con fines de educación e investigación en lugar de asuntos administrativos y de negocios de TI son: Amazon, FutureGrid, Azure, Red Cloud, Google cloud, Globus Online, Grid (XSEDE Cloud Survey Report [12]). La figura 2 muestra el porcentaje de usuarios que utilizan estos proveedores.

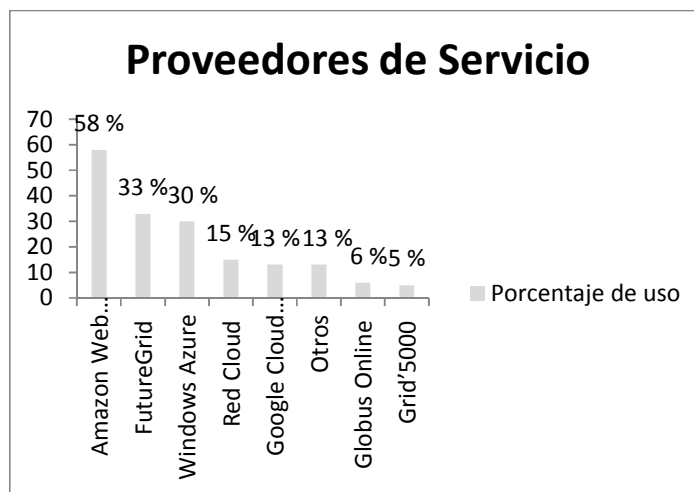


Figura 2. Proveedores de Servicio. Muestra el porcentaje de uso de los usuarios de la nube.

Otros autores toman como referencian el total de ingresos por servicios en la nube, crecimiento porcentual de los ingresos en servicios de la nube, servicios en la nube ofrecido (IaaS, PaaS, SaaS, etc) para determinar cuáles son los mejores proveedores (Panettieri, J. and Katz, A. [15]). En este estudio se dividen los proveedores por categorías dependiendo del servicio que ofrece al cliente, pero para nuestra tesis solo se tomaron los proveedores de PaaS. En la figura 3 se muestra el uso de los proveedores de la nube según el estudio antes mencionado.



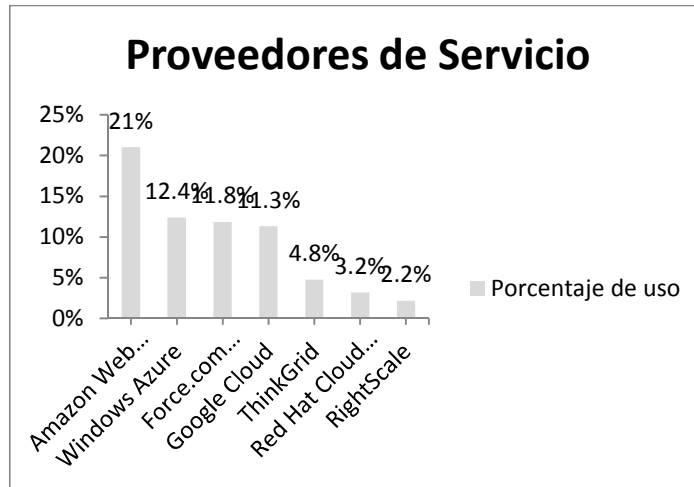


Figura 3. Porcentaje de uso de usuarios de la nube.

Después de la popularidad de la computación en la nube en 2007, varias empresas se convirtieron en proveedores de computación en la nube: Amazon y su Elastic Compute Cloud (EC2), Google con Google App Engine, Microsoft con su Windows Azure Platform, Salesforce con Force.com (A Survey on Open-source Cloud Computing Solutions [23]).

De acuerdo con su disposición los servicios podrían dividirse en seis categorías: Infrastructure as a Service (IaaS), Network as a Service (NaaS), Platform as a Service (PaaS), Identity and Policy Management as a Service (IPMaaS), Data as a Service (Daas) y Software as a Service (SaaS). Para fines de nuestro trabajo solo tomaremos en cuenta los proveedores de plataformas, nombrando así a los distribuidores líderes de esta categoría: Microsoft, Google, Salesforce y Enging Yard (Services in the Cloud Computing era: A survey [24]). Los mismos autores señalan a Amazon como proveedor de Infraestructura como servicio.

Amazon es un proveedor de infraestructura ya que da acceso completo a la máquina virtual en ejecución (root): de este modo permite a los usuarios iniciar sesión en la instancia en ejecución, instalar los programas necesarios, copiar los datos, y ejecutar la aplicación. Google app engine es un proveedor el cual brinda un servicio gratuito siempre y cuando se respeten los límites de uso de los recursos (How to Use Google App Engine for Free Computing [13]).

En la tabla 1 se muestra los proveedores que fueron mencionados por cada autor para darnos una perspectiva grafica de los proveedores que fueron mayormente mencionados.

Proveedores Autores	Amazon	Google	Microsoft	Salesforce	Enging Yard
Takako	X	X	X	X	
Zhou		X	X	X	X

Tabla 1. Proveedores de servicio PaaS.

Microsoft con su plataforma Azure es un proveedor clave de computación en la nube en la capa PaaS. Algunos de los criterios en que se basaron para la definición de esta clasificación fueron las ideas centrales de los sistemas distribuidos para el procesamiento masivo de datos. Estos criterios se centran en la arquitectura de nube, la gestión de la virtualización, los servicios, la tolerancia a fallos mediante el análisis de

mecanismos como el balanceo de carga, la interoperabilidad y el almacenamiento de datos escalable (A Taxonomy and Survey of Cloud Computing Systems [19]). En esta publicación también se mencionan otros proveedores, pero para nuestro estudio solo prestamos mayor atención a los proveedores de PaaS.

Se observó que Microsoft es uno de los proveedores con mayor relevancia en el mercado de proveedores PaaS y se posiciona como uno de los mejores en su rubro. De tal manera que se propone a este proveedor para que sea evaluado como el distribuidor comercial de PaaS en este trabajo.

Con el auge de la computación en la nube también ha crecido el desarrollo de proveedores de soluciones en la nube Open Source (Takako, P. and Estacio, G. [23]), provocando esto muchas opciones para el usuario.

Cloud Foundry, Cloudfify, OpenShift, Stackato y WSO2 Stratus son mencionados como proveedores open sources de PaaS en la página perteneciente a la empresa PURCH la cual cuenta con una cartera de marcas y servicios digitales centradas en la toma de decisiones de compras complejas, fácil para los consumidores y las empresas, a través de una serie de productos relacionados con la tecnología y las categorías de servicios (Bryant, C. [2]).

La comunidad de open source está participando en la carrera de la nube de dos maneras claves; la primera es que mucho software open source, en particular software para empresas y pequeños negocios, están disponible en una base SaaS. Esto proporciona a los clientes calidad, aplicaciones de bajo costo y elimina las molestias de la implementación de software en sus propios servidores. Por otro lado muchos proyectos open source se están centrando en el desarrollo de la infraestructura de software y la gestión que hace posible la computación en la nube. Se recopilieron algunos de los mejores de ambos tipos de aplicaciones de nube de open Source. Para fines de nuestro estudio nos enfocamos más en el servicio PaaS, en donde se mencionaron: AppScale, OpenShift y Cloud Foundry como proveedores de plataforma como servicio (Harvey, C. [7]).

CloudFoundry y OpenShift son proyectos de plataforma como servicios y se asegura que estos productos se lanzan bajo una licencia open source que garantiza el acceso pleno y sin restricciones al código base del proyecto. Todos los componentes, desde el software hasta las APIs utilizadas por los desarrolladores de las aplicaciones, están abiertos a los vendedores, desarrolladores y clientes por igual. Cada proyecto tiene también una comunidad de colaboradores y usuarios que trabajan horas extras para el soporte (Williams, A. [25]).

Se nombra a OpenShift como líder dentro de los proveedores open source y también se asegura que las soluciones open source de plataforma como servicio (OPaaS) tienen 4 características: la solución open source está disponible como código abierto y

el apoyo de una gran comunidad open Source. Portable, Los desarrolladores pueden implementar aplicaciones OPaaS en múltiples infraestructuras de nube, incluyendo las nubes públicas y privadas, Plataforma de servidor abierto. Todos los desarrolladores serán capaces de utilizar lenguajes estándar y código existente dentro de las OPaaS y plataforma del cliente flexible. Los desarrolladores y usuarios finales deben tener las herramientas estándares y fáciles de usar para configurar la interfaz del usuario (Nadir, K. and Zang, T. [21]).

En la tabla 2 se muestra gráficamente los proveedores open source señalados por los autores de los artículos antes citados.

Proveedores Autores	CloudFoundry	OpenShift	Stackato	WSO2 Stratus	AppScale
Christian Bryant	X	X	X	X	
Cynthia Harvey	X	X			X
Alexander Williams	X	X			

Tabla 2. Proveedores open source de plataforma como servicio en la nube.

OpenShift tiene una gran influencia en los artículos y publicaciones encontrados se decide proponer a este proveedor como el contendiente open source a evaluar por su relevancia y también porque es nombrado uno de los líderes en los proveedores open source.

Una vez seleccionado los proveedores a evaluar el siguiente objetivo para este estudio es el indicar como serán evaluados, para esto se necesitan asignar métricas y/o atributos que se utilizaran para la evaluación.

Las métricas de calidad de un servicio son: rendimiento, fiabilidad, integridad, accesibilidad, disponibilidad, interoperabilidad y seguridad. El rendimiento del servicio mide la velocidad de finalización de las solicitudes del servicio. El rendimiento se puede medir en términos de tiempo de respuesta, latencia, tiempo de ejecución, y algunos otros indicadores bajo esta clase (Ladan, M. I. [10]).

Otros autores anuncian que los atributos de la calidad de los servicios web son: Escalabilidad, latencia, rendimiento, tiempo de respuesta, capacidad, disponibilidad, fiabilidad, precisión, robustez, estabilidad, costo, seguridad, mensajería confiable, integridad e interoperabilidad (Geetha, E. and Suresh, T. [4]).

Existen atributos objetivos, los cuales se pueden medir directamente y los atributos subjetivos, los cuales no se pueden medir directamente (Qu, L. and Wang, Y. [18]). Para nuestro estudio necesitamos atributos objetivos, para poder medir y así

conseguir datos significativos, analizarlos detalladamente para obtener una conclusión en base a los resultados arrojados por las evaluaciones.

Uno de los atributos objetivos del artículo “Comparing Public-Cloud Providers” es la latencia en solicitudes [11]. También mencionado en el artículo “Qos of web service: survey on performance and scalability” como un atributo de la calidad de los web service [4]. La latencia es el tiempo transcurrido que hay entre el envío de la solicitud y la recepción de la respuesta (Ladan, M. I. [10]).

Se tomara en cuenta este atributo para medir esta característica en los proveedores que se van a evaluar ya que ayuda a poder obtener el tiempo de respuesta por parte del proveedor para atender las peticiones de los servicios web.

Otro atributo objetivo es el tiempo de procesamiento o tiempo de ejecución. Este es una de las métricas de la calidad de los servicios (“Web Services Metrics: A Survey and A Classification” [10]). Por otro lado el artículo que también hace uso de esta métrica es “QoS based trust management model for Cloud IaaS” al utilizar este atributo para la creación del modelo [5]. Una manera de evaluar la eficiencia de un proveedor con este atributo es implementando algoritmos de multiplicación de matrices como se realizó en el artículo “Superlinear speedup in Windows Azure cloud” [6].

Se decidió tomar en cuenta este atributo para la evaluación de los proveedores ya que ayudará a observar y detectar el procesamiento que brindan a sus usuarios, y así poder indicar cual tiene mejor desempeño en la calidad de servicios.

Una vez obtenidos estos atributos serán utilizados en conjunto para medir el desempeño de los proveedores antes señalados mediante el consumo de servicios web desplegados en las diferentes plataformas.

Se trabajara paralelamente la aplicación con el proveedor libre Koding que no se encontró en las bibliografías pero está teniendo un gran auge en los blogs de desarrolladores ya que ofrece una plataforma como servicio gratuitamente montada en Amazon. Una vez que se empezó a trabajar con OpenShift se pudo observar que al trabajar con una computadora diseñada para la nube (Samsung Google Chrome) no es posible el desarrollo de la aplicación con este proveedor, ya que para trabajar con OpenShift se necesita instalar unos programas en la computadora del usuario, cosa que no se puede hacer en la máquina de Google. Al percatarse esto se puso en contacto con el personal de OpenShift mediante el correo de soporte que está en la página del proveedor. Contesto Nick Harvey confirmando lo que ya se había encontrado. Definitivamente no se puede trabajar en este proveedor con la computadora de Google Chrome.

## Capítulo 3

### 3.1 Marco conceptual

Se construyó un servicio web para conocer su ambiente de desarrollo y su funcionalidad más a fondo y así poder medir su desempeño. Utilizando las herramientas “Visual Studio 2010”, ya que este es un editor de texto para el lenguaje C#, el servidor que trae Windows por defecto “IIS” y el servidor de base de datos “SQLServer”.

Se inició por crear un proyecto nuevo en Visual Studio. En base al artículo “Análisis Comparativo entre .Net y Mono” [14], se tomó en consideración solo la parte de insertar datos en la base de datos, esto para medir solamente el desempeño del servicio web en la acción de incrustar los datos.

El método es llamado “Insertar” obtiene como parámetros el nombre, apellido paterno, apellido materno y numero de control. Esta función devuelve el lapso de tiempo que tarda el servicio en realizar la acción, script 1.

```
public string Insertar(String nombre, String appaterno, String apmaterno, String
    nocontrol)
{
    ...
    Return lapso
}
```

Script 1 Método insertar.

Antes de que el servicio realice cualquier acción se toma el instante de tiempo en el que se iniciará el trabajo:

```
Inicio = DateTime.Now.Ticks;
```

Script 2. Función para tomar el tiempo.

Una vez obtenido este dato se establece la conexión a la base de datos "ABC" en el servidor de base de datos "BetO\_PC\SQLEXPRESS", script 3.

```
System.Data.SqlClient.SqlConnection con;
con = new System.Data.SqlClient.SqlConnection();
con.ConnectionString = ("Data Source=BetO_PC\SQLEXPRESS;Initial Catalog=ABC;Integrated
Security=True;Pooling=False");
```

Script 3. Función para conectase a la basa de datos.

Establecida la conexión se procedió a insertar la información en la base de datos, script 4.

```
sql1 = "INSERT INTO alumno VALUES('" + this.noControl + "','" + this.nombre +
    "','" + this.apellidoP + "','" + this.apellidoM + "')";
SqlCommand cmd1 = new SqlCommand(sql1, con);
cmd1.ExecuteNonQuery();
```

Script 4. Método para insertar a la base de datos.

Una vez terminado el proceso de inserción se toma otra vez el instante de tiempo pero esta vez se almacena en la variable "Final", script 2, para después hacer una comparación y observar el lapso de tiempo que tomo el proceso.

Teniendo los dos instantes de tiempo (inicio y final) se proceder a realizar una operación para conocer el lapso de tiempo que transcurrió en realizar la operación, script 5.

```
Lapso = Final - Inicio;
```

Script 5. Operación matemática para conocer el lapso de tiempo.

Este "lapso" es el que se entregaría como respuesta a la petición del servicio web. Terminado el desarrollo del servicio web se procede a publicarlo en el servidor IIS para después proceder a consumirlo.

Para consumirlo se crea una aplicación (cliente) como se muestra en la figura 4, la cual solicita al usuario el nombre, apellido paterno, apellido materno, Numero de control, script 6. Una vez obtenido los datos al dar clic en el botón aceptar se envía la petición al servicio web con los datos como parámetros.

```
servc.Insertar(nombre, ApePa, Apema, NoCont);
```

Script 6. Función insertar solicitada desde el cliente.

El servicio web responde la petición con el lapso de tiempo. El valor obtenido es recibido por la aplicación y lo muestra debajo de lo datos.

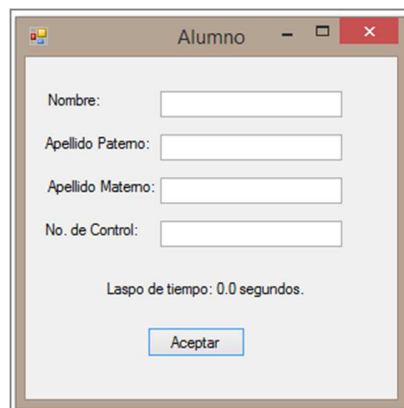
The image shows a web browser window with the title "Alumno". Inside the window, there is a form with four text input fields. The first field is labeled "Nombre:", the second "Apellido Paterno:", the third "Apellido Materno:", and the fourth "No. de Control:". Below these fields, the text "Lapso de tiempo: 0.0 segundos." is displayed. At the bottom of the form, there is a button labeled "Aceptar".

Figura 4. Aplicación cliente de servicio web.

Se realizo otro experimento para conocer cómo ejecutar servicios web en otro lenguaje de programación, esta vez JAVA. Para ello se necesitaron otras herramientas; eclipse como editor de texto, ApacheTomcat a fin de servidor de aplicaciones y el servidor de base de datos utilizado anteriormente SQLServer.

Este ejercicio contaba de buscar un usuario en una base de datos y verificar si coincidía el usuario y contraseña, una validación de credenciales simple. El servicio

devuelve “TRUE” si el usuario y contraseña coinciden, de lo contrario responde con un “FALSE” si la búsqueda no fue exitosa.

El método se llamó validaUsuario y recibe como parámetros el usuario y contraseña, script 7.

```
validaUsuario(String nom,String pass)
```

Script 7. Función validaUsuario.

El servicio se conecta con la base de datos “Login” en el servidor SQLServer, script 8.

```
Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");  
String url = "jdbc:sqlserver://localhost:1433;databasename=Login";  
conexion= DriverManager.getConnection(url,"sa","sa");
```

Script 8. Método para conectarse a la base de datos.

Después de establecer la conexión con la base de datos se procede a buscar el usuario y la contraseña recibida como parámetros, script 9.

```
Statement sentencia =conexion.createStatement();  
ResultSet sr = sentencia.executeQuery("select * From Usuarios where nombre = '"+nom+"' and password='"+pass+"'");
```

Script 9. Método para buscar al usuario y contraseña.

El servicio valida si se encontró al usuario y regresa su respuesta, script 10.

```
if (sr .getStatement()){  
    return true;  
}  
else{  
    return false;  
}
```

Script 10. Decisión del servicio.

Terminado el servicio web se publica en el servidor tomcat y se procede a realizar una aplicación (cliente) para consumir este servicio. El cliente es sencillo ya que solo se pide al usuario el nombre y la contraseña, script 11.

```
System.out.println("Usuario: ");  
nombre = in.nextLine();  
System.out.println("Password: ");  
password = in.nextLine();
```

Script 11. Metodo para capturar usuario y contraseña.

Una vez obtenido los datos se realiza la petición al servicio, script 12.

```
if (servicio.validaUsuario(nombre, password)==true) {  
    System.out.print("Se encontro el usuario");  
}  
Else {  
    System.out.print("Usuario o contraseña invalido");  
}
```

#### Script 12. Solicitud del servicio.

La aplicación obtiene el valor que devuelve el servicio web y valida si es “TRUE”, esto significa que el usuario y contraseña fueron encontrados y se muestra el mensaje “Se encontro el usuario”, de lo contrario significa que algún dato no es correcto y se despliega el texto “Usuario o contraseña invalido”.

Una vez realizado este experimento se buscó hacer otra práctica un poco más extensa, que implicara los tres métodos que se realizaron en el artículo de Mares [14], insertar, actualizar y eliminar. Este nuevo experimento se realizó con herramientas y lenguajes diferentes para ver y tener una mejor comprensión sobre el bajo acoplamiento que tienen los servicios web. El servicio web se construyó con lenguaje JAVA mientras que la aplicación cliente se realizó en C#

Se realizó el servicio web en lenguaje JAVA con la herramienta eclipse, este servicio tendría que realizar los métodos insertar, actualizar y eliminar.

El procedimiento “inserta” recibe como parámetros un nombre y una contraseña, para después insertarlos en la base de datos, script 13.

```
inserta(String nom,String pass)
```

Script 13. Método inserta.

Este método al igual que el resto realiza la conexión a la base de dato “Login” que se encuentra en un servidor SQLServer, instalado como localhost, script 14.

```
Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");  
String url = "jdbc:sqlserver://localhost:1433;databasename=Login";  
miconexion= DriverManager.getConnection(url,"sa","sa");
```

Script 14. Conexión a la base de datos.

Una vez iniciada la conexión el servicio procede a insertar los valores en la base de datos, script 15.

```
Statement sentencia =miConexion.createStatement();  
ResultSet sr = sentencia.executeQuery("insert into Usuarios(Nombre>Password)values  
                                     ('"+nom+"','"+pass+"'");  
  
if (sr.getStatement()!=null){  
    JOptionPane.showMessageDialog(null, sr);  
    miConexion.close();  
    return false;  
}  
  
Else  
{  
    miConexion.close();  
    return true;  
}
```

Script 15. Función para insertar datos.

Después que los datos son insertados el servicio valida si hubo algún error, si es así se muestra un mensaje con el error, se cierra la conexión con la base de datos y se



regresa un “FALSE”, indicando que la operación no se realizó exitosamente. De lo contrario cierra la conexión con la base datos pero regresa un “TRUE” señalando que la acción se realizó sin problemas.

El método “actualiza” actúa de la misma manera que “inserta” solo que la diferencia es que actualiza la contraseña del nombre indicado. Este procedimiento recibe como parámetros el nombre que se va a actualizar y la contraseña que se cambiara por el valor anterior, script 16.

```
actualiza(String nom,String pass)
```

Script 16. Método actualiza.

Establece la conexión para después actualizar el valor, script 17.

```
ResultSet sr = sentencia.executeQuery("update Usuarios set Password='"+pass+"'
                                     where Nombre='"+nom+"'");
```

Script 17. Método para actualizar el valor.

De La misma manera verifica si existió algún error y regresa los mismos valores como el método anterior, script 18.

```
if (sr.getStatement()!=null){
    JOptionPane.showMessageDialog(null, sr);
    miConexion.close();
    return false;
}
Else
{
    miConexion.close();
    return true;
}
```

Script 18. Método para revisar el proceso.

El procedimiento “elimina” realiza el mismo algoritmo que los otros métodos solo que elimina el nombre de la base de datos, script 19. Este método recibe como parámetro el nombre que se desea eliminar y procede con la conexión y los valores que regresa de igual manera que los otros procesos, script 20.

```
elimina(String nom)
```

Script 19. Método elimina.

Este método elimina el nombre que recibe como parámetro, script 20.

```
ResultSet sr = sentencia.executeQuery("delete from Usuarios where Nombre='"+nom+"'");
if (sr.getStatement()!=null){
    JOptionPane.showMessageDialog(null, sr);
    miConexion.close();
    return false;
}
Else
{
    miConexion.close();
    return true;
}
```

Script 20. Función para eliminar el usuario.

Cuando se terminó de realizar los métodos del servicio web se procedió a publicar el servicio en el servidor tomcat para poder consumirlo. Teniendo publicado el servicio web se hizo una aplicación con la cual se consumiera el servicio. Esta aplicación fue creada con lenguaje C#.

En la figura 5 se muestra la ventana de la aplicación que consume el servicio web, esta solo pide el nombre y password al usuario y contiene también 3 botones “Insertar”, “Actualizar” y “Eliminar” indicando la acción que realiza cada botón. Debajo de los botones se muestra el tiempo que transcurre en cada una de las acciones. Al momento de insertar un nombre solo se indicara el tiempo del campo Insertar y así sucesivamente.

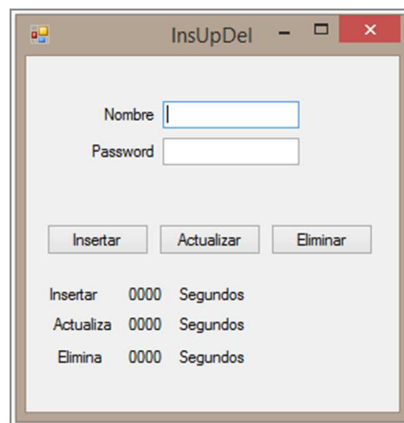


Figura 5. Aplicación que consume el servicio web creado en Java.

Al seleccionar el botón “Insertar” la aplicación obtiene los valores de nombre y password, para después hacerle la petición al servicio de “insertar”, pasando los datos obtenidos previamente, script 21.

```
pass = txtPass.Text;
inicio = DateTime.Now.Ticks;
if (servicio.inserta(nombre, pass)){
    final = DateTime.Now.Ticks;
    lapso = final - inicio;
    lapso = convertir(lapso);
    lbIns.Text = lapso.ToString();
}
else
{
    MessageBox.Show("La acción no se realizo exitosamente",
        "Notificacion", MessageBoxButtons.OK, MessageBoxIcon.Error);
}
```

Script 21. Método que realiza el cliente.

Antes de realizar la petición al servicio web se toma el instante de tiempo para saber el tiempo en el que se inició la operación. Si todo se realizó con éxito la aplicación valida si el servicio devuelve true para saber que no hubo problemas y proseguir a con

tomar el instante de tiempo en el que se terminó la acción y así poder comparar los instantes de tiempo y sacar el lapso de tiempo que necesito el servicio web en realizar la operación. Si todo corrió sin problemas se muestra el lapso de tiempo en el campo de Insertar.

Si existió algún error la aplicación mostrara un mensaje de error con el texto “La acción no se realizo exitosamente” indicando que no se pudo completar la acción”.

De la misma manera se realizaron los otros métodos, con la única diferencia que se invoca a diferentes procesos del servicio web, el botón “Actualizar” invoca al método “actualiza” del servicio web, script 22, y el botón de “Eliminar” al procedimiento “elimina”.

```
nombre = txtNom.Text;
pass = txtPass.Text;
inicio = DateTime.Now.Ticks;
if (servicio.actualiza(nombre, pass)){
    final = DateTime.Now.Ticks;
    lapso = final - inicio;
    lapso = convertir(lapso);
    lbAct.Text = lapso.ToString();
}
else
{
    MessageBox.Show("La acción no se realizo exitosamente",
    "Notificacion",MessageBoxButtons.OK, MessageBoxIcon.Error);
}
```

Script 22. Función actualiza del cliente.

A diferencia de los dos primeros métodos el procedimiento “eliminar” solo recibe el nombre como parámetro. Script 23.

```
nombre = txtNom.Text;
pass = txtPass.Text;
inicio = DateTime.Now.Ticks;
if (servicio.elimina(nombre)){
    final = DateTime.Now.Ticks;
    lapso = final - inicio;
    lapso = convertir(lapso);
    lbEli.Text = lapso.ToString();
}
else
{
    MessageBox.Show("La acción no se realizo exitosamente",
    "Notificacion",MessageBoxButtons.OK, MessageBoxIcon.Error);
}
```

Script 23. Método elimina del cliente.

### 3.1.1 Marco conceptual de OpenShift.

OpenShift es la plataforma como servicio de Red Hat que permite a los desarrolladores crear rápidamente, hospedar, y escalar aplicaciones en un entorno de nube. OpenShift ofrece una selección de lenguajes de programación como Java, Ruby, PHP, Node.js, Python y Perl y un conjunto completo de herramientas de desarrollo para aumentar la productividad del desarrollador y acelerar la entrega de aplicaciones [27]. Para tener acceso a OpenShift basta con darse de alta con una cuenta de correo electrónico. En la figura 6 se muestra la pantalla que OpenShift ofrece para registrarse.

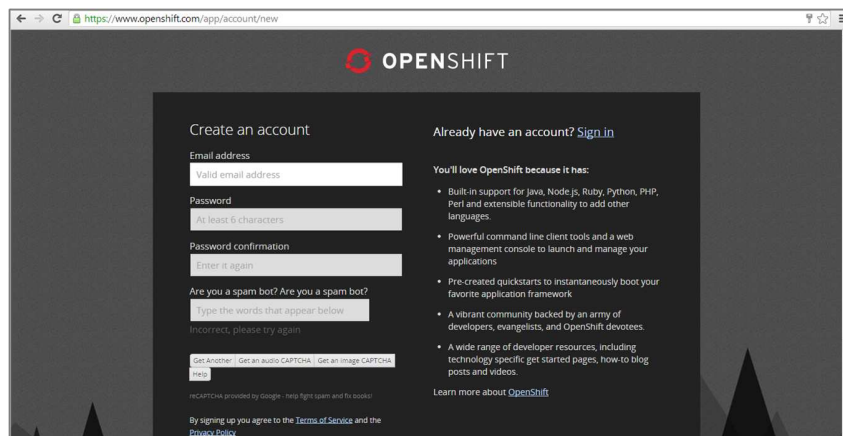


Figura 6. Pantalla de registro.

En el entorno de desarrollo de esta plataforma se necesita la instalación de herramientas para la interacción con los repositorios. OpenShift trabaja con repositorios para alojar las aplicaciones de sus usuarios. Las herramientas que se necesitan son Git y Ruby. Estas herramientas ayudan al usuario que por medio de la línea de comandos suba sus aplicaciones al repositorio donde se almacenara sus aplicaciones.

Para la creación de aplicaciones OpenShift brinda una opción grafica para la selección de las herramientas que se utilizaran en dicha aplicación. En la figura 7 se muestra como OpenShift ofrece sus herramientas.

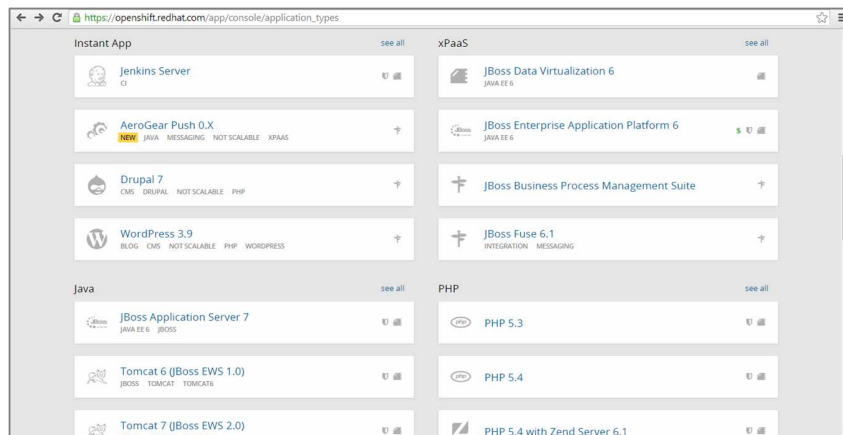


Figura 7. Herramientas ofrecidas por OpenShift.

OpenShift brinda al usuario un dominio para el acceso a sus aplicaciones. <http://NomApli-NomDominio.rhcloud.com>. Donde NomApli es el nombre de la aplicación, este nombre varía de acuerdo al nombre que el usuario le dé a cada aplicación. NomDominio es el nombre del dominio que OpenShift ofrece, este dominio es uno para cada usuario, el usuario no puede cambiarlo como con cada aplicación. Cuando se crea la aplicación y se entra a la dirección creada. OpenShift muestra una página de bienvenida significando esto que la aplicación esta lista para trabajar en ella. En la figura 8 se muestra la página de bienvenida.

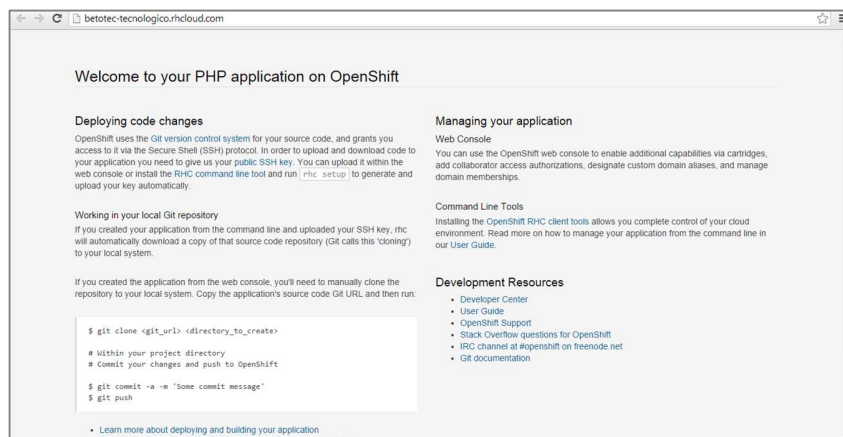


Figura 8. Página de bienvenida.

### 3.1.1.1 Desarrollo de un web service en OpenShift.

Para realizar un servicio web con lenguaje PHP se utilizó la herramienta NuSOAP, ya que NuSOAP es un kit de herramientas (ToolKit) para desarrollar servicios web [1]. Está compuesto por una serie de clases que hacen mucho más fácil el desarrollo de

estos. Provee soporte para el desarrollo de clientes y de servidores. Esta herramienta está basada en SOAP 1.1, WSDL 1.1 y HTTP 1.0/1.1.

Su instalación es muy sencilla ya que solo hay que descargar el archivo de la dirección: <http://sourceforge.net/projects/nusoap/> descomprimirlo y colocarlo dentro de la carpeta del proyecto.

Primero se necesita crear una aplicación dentro de OpenShift, solo basta con dar clic en el botón “add application” y proporcionar el nombre de la aplicación. A esta aplicación se le pone el nombre de login. Al crear la aplicación se crea el URL, para este caso es: <http://login-tecnologico.rhcloud.com>. Para poder trabajar en esta aplicación en la computadora del usuario primero se debe “clonar” el repositorio en donde está la aplicación.

```
Git clone <git url> <directorio a crear>
```

Donde “git url” es la dirección donde se encuentra el repositorio y “directorio a crear” es la dirección dentro de la computadora del usuario donde se almacenara la aplicación para poder trabajar en ella y después cargarla al repositorio.

Teniendo la aplicación en la computadora se procede a copiar la herramienta de NuSOAP al directorio creado y se procede a iniciar la construcción de la misma aplicación que se realizó anteriormente con lenguaje JAVA, script 14 y 15. El servicio web consta de 3 métodos “insertar”, “actualizar” y “eliminar”. Los cuales reciben como parámetros el nombre del usuario y la contraseña a excepción del método de eliminar, el cual solo recibe el nombre del usuario. En estos métodos se realiza una conexión a la base de datos y dependiendo del método se inserta, actualiza o elimina los dato, script 24.

```
function insertar($nom,$pass)
{
    define('OPENSIFT_MYSQL_DB_HOST', getenv('127.8.24.2'));
    define('OPENSIFT_MYSQL_DB_PORT', getenv('3306'));
    define('OPENSIFT_MYSQL_DB_USERNAME', getenv('adminYEeyGCW'));
    define('OPENSIFT_MYSQL_DB_PASSWORD', getenv('pp9WZxDaUnf1'));
    define('OPENSIFT_APP_NAME', getenv('login'));

    define('DB_HOST', getenv('OPENSIFT_MYSQL_DB_HOST'));
    define('DB_PORT',getenv('OPENSIFT_MYSQL_DB_PORT'));
    define('DB_USER',getenv('OPENSIFT_MYSQL_DB_USERNAME'));
    define('DB_PASS',getenv('OPENSIFT_MYSQL_DB_PASSWORD'));
    define('DB_NAME',getenv('OPENSIFT_APP_NAME'));
    $con=mysqli_connect(DB_HOST,DB_USER,DB_PASS,DB_NAME);
    $sql="INSERT INTO Usuarios (nombre, password) VALUES ('$nom','$pass')";
    mysqli_query($con,$sql);
    if (mysqli_error())
    {
        return false;
    }
    return true;
}
```

Script 24. Método inserta.

Para que NuSOAP pueda publicar el servicio y construir el documento WSDL se pide que dentro de la aplicación se describa también los parámetros que servirán para construir el documento.

```
$servidor->register('insertar',  
    array('nombre'=>'xsd:string','password'=>'xsd:string'),  
    array('resultado'=>'xsd:boolean'),  
    $ns, false,  
    'rpc',  
    'literal',  
    'Documentacion de insertar');
```

Script 25. Método para registrar el servicio web.

La función “register” recibe como parámetros el nombre del métodos que se desea publicar, los datos de entrada, que datos recibe el método para su funcionamiento, los datos de salida, que es lo que responderá el servicio, script 25.

Después de registrar los métodos y parámetros del servicio solo falta publicarlo para que NUSOAP genere el archivo WSDL, script 26.

```
if (isset($HTTP_RAW_POST_DATA)) {  
    $input = $HTTP_RAW_POST_DATA;  
}else{  
    $input = implode("\r\n", file('php://input'));  
}  
$servidor->service($input);
```

Script 26. Publicación del servicio.

Finalizada la construcción de la aplicación se necesita cargarla al repositorio para que se haga accesible atreves de internet. En la línea de comandos debemos entrar al directorio donde se encuentra la aplicación. Una vez dentro del directorio se procede con los comandos, script 27:

```
git add .  
git commit -m "mensaje"  
git push
```

Script 27. Funciones de git.

Con estos comandos se indica que se desea agregar documentos al repositorio, se marca los cambios efectuados dentro de la aplicación con un mensaje y por último se manda los archivos hacia el repositorio.

En la figura 9 se muestra el servicio web publicado así como la opción para descubrir el archivo WSDL.

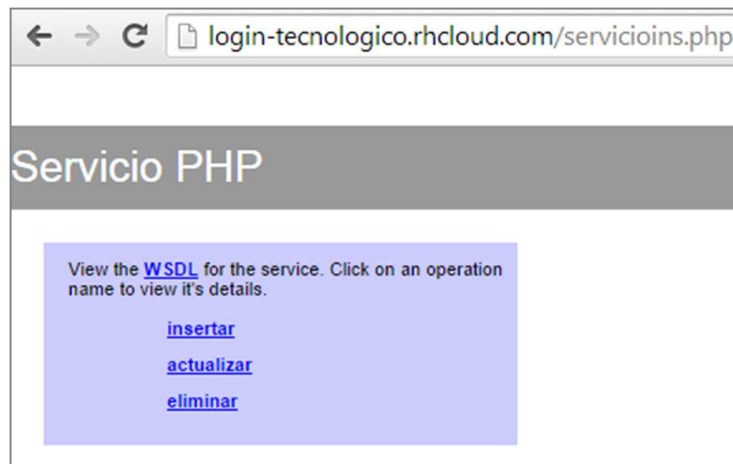


Figura 9. Publicación del servicio web.

Para consumir el servicio se construyó un cliente en el mismo lenguaje (PHP), el cual hace una petición al servicio y este le responde “true” si no hubo algún error o “false” si hubo algún error y no se pudo completar el proceso.

```
$par = array("nombre" => "beto", "password" => "angy");
```

Script 28. Asignación de parámetros.

La variable \$nom es un arreglo que tiene como identificador “nombre” y “password” dentro de ella para almacenar los valores que serán enviados como parámetro, script 28.

```
try{
    $client = new SoapClient("http://login-tecnologico.rhcloud.com/servicioins.php?wsdl");
    $result = $client->__SoapCall("insertar",$par);
    if ($result==true) {
        echo "<h1>Se inserto </h1>";
    }
    else {
        echo "<h1>NO se inserto</h1>";
    }
}catch (SoapFault $e){

    echo "Ups!! hubo un problema y no pudimos recuperar los datos. <br/>
    $e<hr/>";
}
```

Script 29. Consumo del servicio insertar.

Para consumir el servicio se crea una instancia del servicio dando como referencia la dirección del archivo WSDL y se invoca a la función “insertar” pasándole como parámetros los datos almacenados en \$par. Al almacenar el servicio responde en \$result para después hacer una comparación y verificar si se almaceno correctamente la información, script 29.



### 3.1.2 Marco conceptual de Azure.

Azure es la plataforma como servicio de Microsoft que permite a los desarrolladores crear rápidamente, hospedar, y escalar aplicaciones en un entorno de nube. Azure ofrece una amplia gama de recursos que soportan diversos lenguajes de programación como Java, .NET, Python, Ruby, PHP, Node.js, y un conjunto completo de herramientas de desarrollo para aumentar la productividad del desarrollador y acelerar la entrega de aplicaciones [28]. Para tener acceso a Azure se tiene que crear una cuenta con un correo electrónico válido y como es un proveedor comercial y a pesar de que ofrece un mes gratis es necesario introducir una tarjeta de crédito. En la figura 10 se muestra la pantalla que Azure ofrece para registrarse.

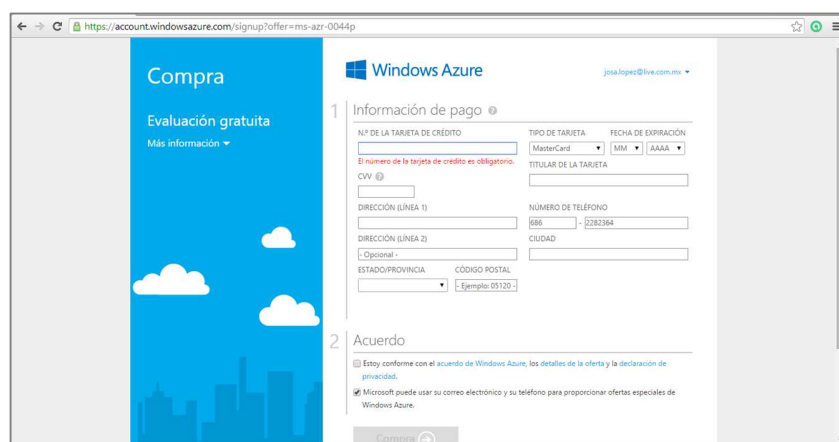
The image is a screenshot of a web browser displaying the Windows Azure registration page. The browser's address bar shows the URL: https://account.windowsazure.com/signup?offer=ms-azr-0044p. The page has a blue header with the 'Windows Azure' logo and a user email 'josa.lopez@live.com.mx'. On the left, there is a blue sidebar with the text 'Compra' and 'Evaluación gratuita'. The main content area is divided into two sections: '1 Información de pago' and '2 Acuerdo'. The 'Información de pago' section contains several input fields: 'Nº DE LA TARJETA DE CRÉDITO', 'TIPO DE TARJETA' (with a dropdown menu showing 'MasterCard'), 'FECHA DE EXPIRACIÓN' (with dropdowns for 'MM' and 'AAAA'), 'TITULAR DE LA TARJETA', 'CVV', 'DIRECCIÓN (LÍNEA 1)', 'DIRECCIÓN (LÍNEA 2)', 'NÚMERO DE TELÉFONO' (with a dropdown for '555' and a text field for '0282384'), 'CIUDAD', 'ESTADO/PROVINCIA' (with a dropdown), and 'CÓDIGO POSTAL' (with a dropdown showing 'Ejemplo: 05120'). A red error message is visible below the credit card number field: 'El número de la tarjeta de crédito es obligatorio.' The '2 Acuerdo' section has a checkbox for 'Estoy conforme con el acuerdo de Windows Azure, los detalles de la oferta y la declaración de privacidad.' and a checked checkbox for 'Microsoft puede usar su correo electrónico y su teléfono para proporcionar ofertas especiales de Windows Azure.' At the bottom, there is a 'Comprar' button.

Figura 10. Pantalla de registro.

En el entorno de desarrollo de esta plataforma se necesita la instalación de herramientas. Azure trabaja con repositorios para alojar las aplicaciones de sus usuarios. La herramienta que se necesita para trabajar con sitios web es WebMatrix. Esta herramienta ayuda al usuario a modificar e interactuar con sus aplicaciones y al terminar de editarla subir la aplicación a los servidores de Azure.

Para la creación de aplicaciones Azure brinda una opción de editor de texto online "visual studio online" pero como abrimos una cuenta de versión gratuita este editor tiene una caducidad de 1 hora así que se procedió a trabajar con WebMatrix localmente y subir la aplicación cuando se termine de editarla en nuestra computadora. En la figura 11 se muestra el editor de texto online.



Figura 11. Editor de texto online.

Azure brinda al usuario un dominio para el acceso a sus aplicaciones web. <http://servesc.azurewebsites.net> Donde servesc es el nombre del sitio del usuario, este nombre varía de acuerdo al nombre que el usuario le dé a su sitio. Cuando se crea la aplicación y se entra a la dirección dada. Azure muestra una página de bienvenida significando esto que la aplicación esta lista para trabajar en ella. En la figura 12 se muestra la página de bienvenida.



Figura 12. Pagina de bienvenida.

### 3.1.2.1 Desarrollo de un web service en Azure

Para la creación del servicio web en Azure se utiliza la misma información que en OpenShift. Para tal fin se utilizaron los scripts 25, 26, 28 y parte del script 24 ya que la conexión a la base de datos es diferente en este proveedor. En esta plataforma se debe realizar localmente la aplicación para después desplegarlo en los servidores de Azure. Se realiza la aplicación en el editor de texto WebMatrix. En la figura 13 se muestra la edición del código.

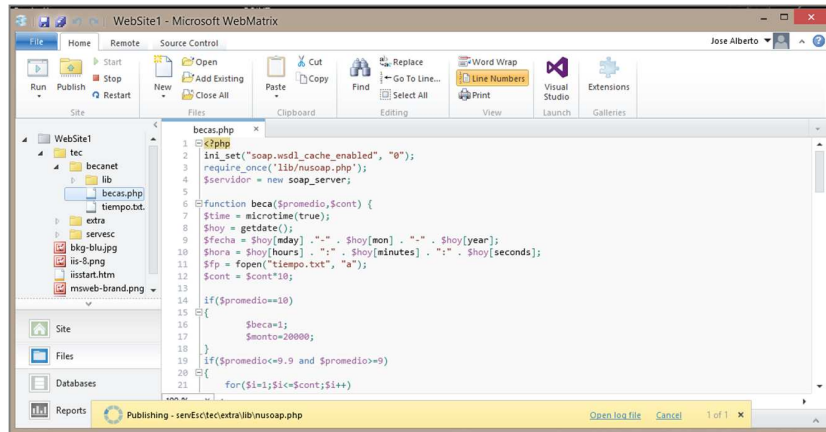


Figura 13. Creación del servicio web.

En la figura 14 se muestra el servicio web publicado así como la opción para descubrir el archivo WSDL.

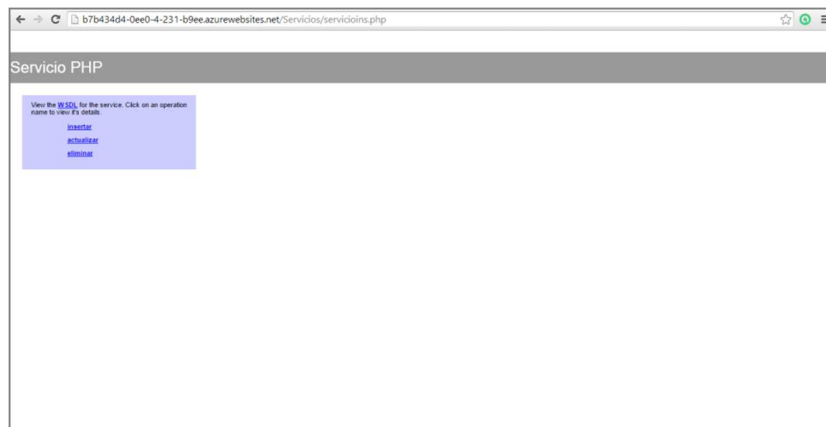


Figura 14. Publicación del servicio web.

Para consumir el servicio se construyó un cliente al igual que el que se realizó en OpenShift, para tal fin se utilizó el script 29.

### 3.1.3 Marco conceptual de Koding

Rigsby en su publicación en internet informa que Koding ofrece una nueva tendencia de desarrollo basado en la nube [20]. En el 2009 se creó la primera versión de esta plataforma. Koding es un entorno de desarrollo en línea con el objetivo de simplificar el desarrollo a sus usuarios, proporciona computación libre y desarrollo para todos. Lo hace ofreciendo máquinas virtuales (VMs) gratis para el desarrollo en la nube. La máquina virtual de Koding proporciona un sistema operativo Ubuntu, con una Terminal. Esta plataforma permite trabajar en el código; Python, PHP, C ++, Java y C. Esto lo hace en línea, por lo tanto es accesible desde cualquier parte del mundo. Incluso se puede compartir el trabajo en equipo. Koding ayuda a realizar un trabajo bien elaborado con rapidez, ya que brinda muchas herramientas para el desarrollo de este [26]. Para crear una cuenta solo se pide una cuenta de correo valida y un nombre de usuario.

En la figura 15 se muestra la página principal de Koding la cual muestra todas las opciones de su plataforma, así también un blog de los usuarios del sitio en el cual publican trabajos, dudas entre otras cosas permitiendo a los otros usuarios contribuir o ayudar en el trabajo de los demás.

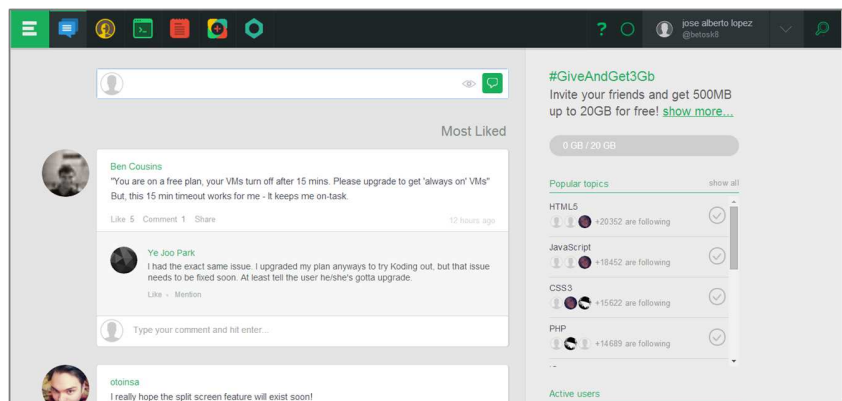


Figura 15. Página de inicio de Koding.

Koding también ofrece a sus usuarios la accesibilidad de trabajar en equipo en un mismo proyecto. Los involucrados comparten un repositorio el cual un administrador se encarga de brindarles derechos sobre los documentos del proyecto.

Dentro de las opciones que brinda Koding está la de tener una terminal, ya que está hecho con el sistema operativo Ubuntu, Koding nos permite interactuar directamente con la plataforma de igual manera como si se estuviera trabajando con este sistema operativo a través de la terminal. La figura 16 muestra la terminal con la que interactúa el usuario.

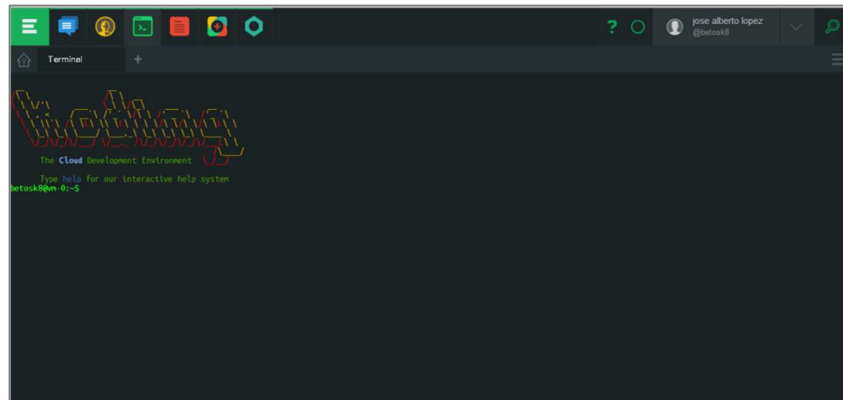


Figura 16. Terminal de Koding.

Para la edición de texto, Koding ofrece el editor Ace para facilitar a los usuarios la creación y modificación de sus archivos en el momento de interactuar con sus aplicaciones. En la parte izquierda se proyecta un árbol con los archivos del usuario almacenados en la plataforma y así se tenga un acceso visual y más rápido a los archivos. Esto se muestra en la figura 17.

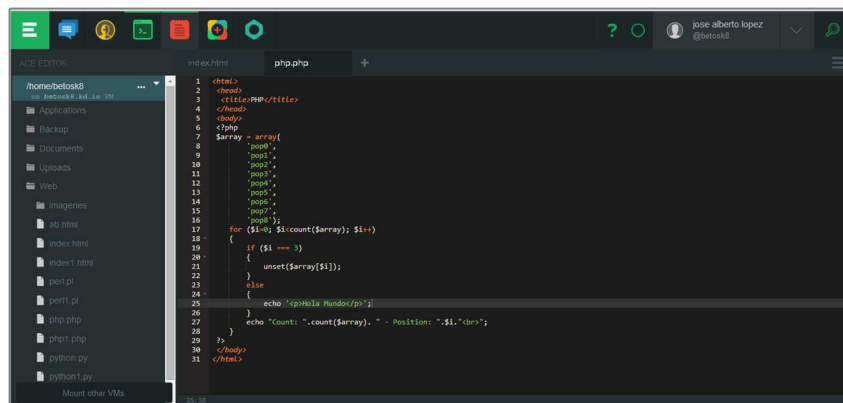


Figura 17. Editor de texto Ace en Koding.

Esta plataforma ofrece muchas opciones extras para trabajar con las aplicaciones de los usuarios. Solo se necesita seleccionar la opción y Koding la instala automáticamente para que el usuario las empiece a utilizar inmediatamente después de haberla instalado. Algunas de las aplicaciones opcionales son: Wordpress, Drupal, Project Euler, Dropbox for Koding, Game of life, Laravel, Django, Ruby on rails, UML generator, Numbers, Brackets y Julia.

A modo que Koding ofrece un servicio en la nube, asigna una dirección a cada uno de sus usuarios para que puedan tener acceso cualquier persona incluso ajena a las que utilizan la plataforma. Por defecto el proveedor pone una página como muestra. En la figura 18 es un ejemplo de la página que Koding pone como inicial cuando se entra al sitio asignado a un usuario. La dirección es: <http://username.kd.io/>



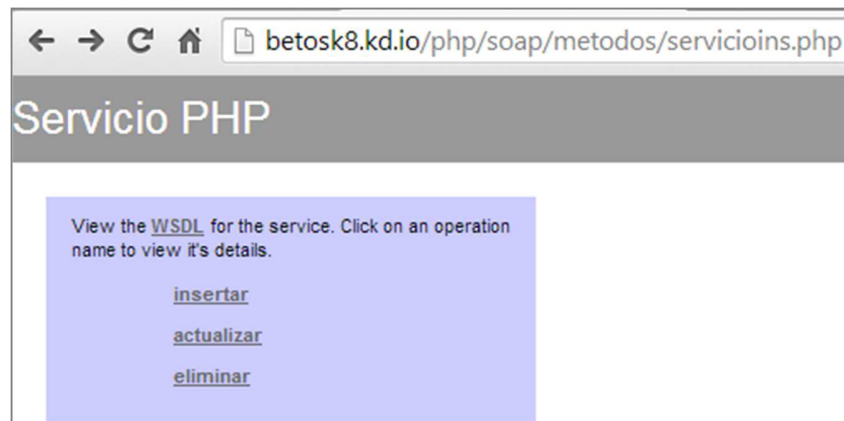


Figura 20. Publicación del servicio web.

Para consumir el servicio se construyó un cliente al igual que el que se realizó en OpenShift utilizando el script 29.

## 3.2 Diseño de aplicación.

Para realizar las evaluaciones se necesita crear una aplicación la cual deberá consumir servicios desplegados en las plataformas de los proveedores que serán evaluados. Se realizó un sistema el cual consume los tres servicios. Los servicios se crearán como aplicaciones independientes demostrando aquí el bajo acoplamiento.

Un servicio será el kardex, proporcionado por el “departamento” de servicios escolares, este servicio responderá a la petición la lista de las materias del usuario. Otro “departamento” es becanet, en este servicio el usuario proporciona su promedio y la aplicación responderá con la beca que puede ser otorgada dependiendo del promedio. Y por último tenemos el “departamento” de extraescolares, con este servicio el cliente recibirá una lista de las materias extraescolares disponibles. Los datos obtenidos de las evaluaciones serán almacenados en archivos de texto para después poder obtener la información.

En la figura 21 se muestra el diseño de la aplicación así como la interacción con los servicios.

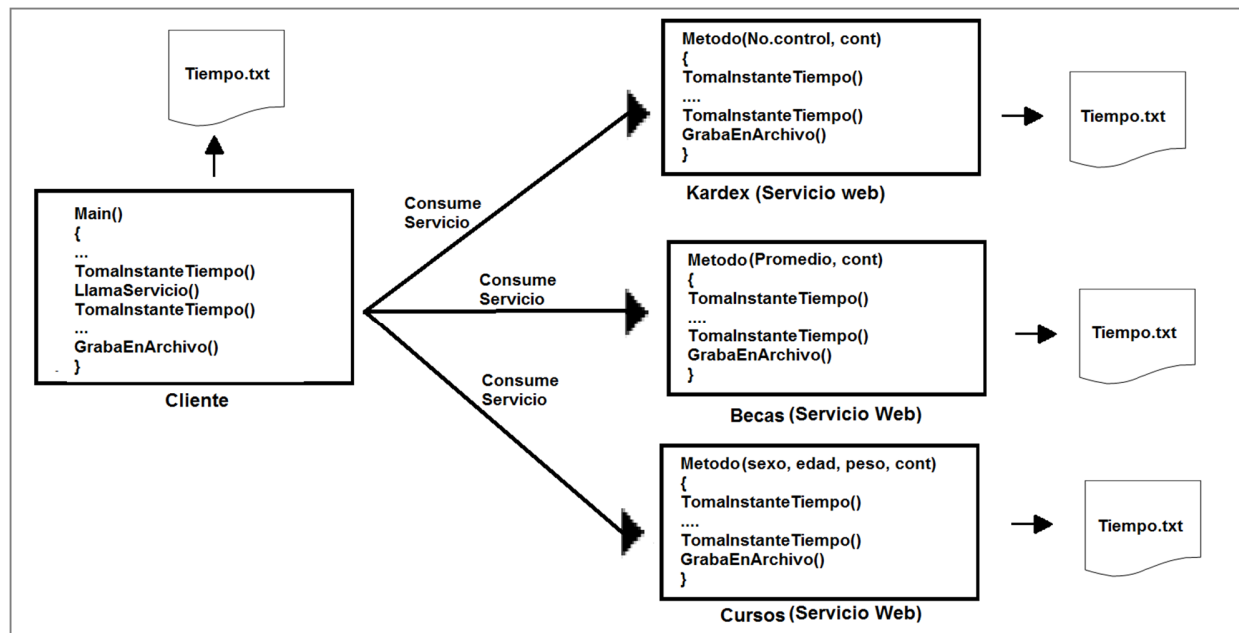


Figura 21. Diseño de la aplicación.

Al implementar esta aplicación se podrán orquestar los servicios al consumirlos y así poder realizar las evaluaciones.

### 3.3 Implementación.

Para efectos del trabajo de evaluación se creó una aplicación y por medio de ella realizar las evaluaciones y poder alcanzar el objetivo general de este trabajo. A pesar de que evaluaremos proveedores en la capa PaaS las evaluaciones se realizarán en la capa SaaS ya que usaremos un software como herramienta de evaluación.

Para la implementación de la aplicación se crearán tres servicios web becas, kardex y las materias extraescolares.

El servicio de becas recibe como parámetro el promedio del alumno para después tomar una decisión y saber qué tipo de beca será acreedor el alumno. Con fines de medir el tiempo de procesamiento por parte del proveedor, se toma el instante de tiempo en cuanto entra a la función beca después se realiza el procedimiento del método y antes de mandar la respuesta al cliente se vuelve a tomar el instante de tiempo y se hace una resta para conocer el lapso de tiempo que necesitó el proveedor para procesar la información como en el artículo de Mares [14]. Para que cambien la información que tiene que procesar se le agrega un parámetro extra y se realiza un ciclo con el número de iteraciones recibido en este parámetro. Ya por último se guarda el



lapso de tiempo en un archivo para después de varias pruebas poder graficar los resultados.

El servicio de kardex recibe como parámetros el número de control del alumno para fines de evaluación se envía un parámetro extra para indicar el número de materias que se van a regresar como respuesta. En esta función las materias son predefinidas y almacenadas en un arreglo, durante el proceso se seleccionaran aleatoriamente más materias para tener un total de materias indicadas por el parámetro extra las cuales serán con las que el servicio responderá a la petición del cliente. Al igual que en el servicio beca se tomara el instante de tiempo en cuanto entre a la función y antes de responder la petición del cliente, se realizara la resta para conocer el lapso de tiempo y será almacenado en un archivo.

El servicio de Extraescolares recibe como parámetros el sexo, la edad y el peso también para fines de evaluación se envía un parámetro extra para indicar el número de materias que se van a regresar como respuesta. Con estos parámetros el servicio toma una decisión para mandar una lista de materias dependiendo de los parámetros que se hayan mandado. Al igual que el servicio anterior al momento de entrar al método se toma el instante del tiempo y antes de contestar al cliente se vuelve a tomar el instante de tiempo después se realiza la resta y el resultado es almacenado en un archivo.

Para la creación del cliente se utilizara un lenguaje diferente al de los servicios. El cliente consumirá los tres servicios y para medir la latencia se necesita tomar el instante de tiempo antes de mandar la petición al servicio y después de que el servicio haya contestado, esto se realizara para los tres servicios. Como se necesita observar los resultados, estos serán almacenados en un archivo.

### 3.3.1 Implementación en la plataforma OpenShift.

Los servicios que serán desplegados en este proveedor se desarrollarán en lenguaje PHP. Se crearán 3 aplicaciones, una para cada servicio como ya se ha señalado se debe desarrollar localmente y después subir la aplicación al servidor del proveedor. En el servicio beca se toma el promedio recibido y lo busca en un rango de valores para indicar que beca le corresponde al alumno para esta calificación. El rango de tiempo se toma desde que inicia el método hasta antes de mandar la respuesta al cliente, esto para medir el tiempo que el proveedor necesitara para procesar la información. Para que exista una variante el método recibe un parámetro extra, este parámetro se utiliza para crear un ciclo y así poder tener una variante entre petición y petición del servicio. Y al final se almacena el valor en un archivo, script 30.

```
$time = microtime(true);  
$hoy = getdate();  
$fecha = $hoy[mday] . "-" . $hoy[mon] . "-" . $hoy[year];  
$hora = $hoy[hours] . ":" . $hoy[minutes] . ":" . $hoy[seconds];  
$fp = fopen("tiempobeca.txt", "a");  
$cont = $cont*10;
```

```

if($promedio==10)
{
    for($i=1;$i<=$cont;$i++)
    {
        $beca=1;
        $monto=20000;
    }
}
if($promedio<=9.9 and $promedio>=9)
{
    for($i=1;$i<=$cont;$i++)
    {
        $beca=2;
        $monto=15000;
    }
}
if($promedio<=8.9 and $promedio>=8)
{
    for($i=1;$i<=$cont;$i++)
    {
        $beca=3;
        $monto=10000;
    }
}
if($promedio<=7.9 and $promedio>=7)
{
    for($i=1;$i<=$cont;$i++)
    {
        $beca=4;
        $monto=10000;
    }
}
$arreglo[] = array( 'TipoBeca'=>$beca, 'Monto'=>$monto);
$time_end = microtime(true);
fputs($fp, $time . "|" . $time_end . "|" . ($time_end-$time) . "|" . $fecha . "|" .
$hora . PHP_EOL);
fclose($fp);
return $arreglo;

```

Script 30. Servicio beca.

El servicio kardex recibe el número de control. En este método hay materias que ya están predefinidas. La lista de materias con las que el servicio responde las genera aleatoriamente tomando las materias que ya están definidas y el número de materias será indicado con el parámetro recibido. Para poder medir el procesamiento de los datos se toma el lapso de tiempo que transcurre desde que inicia el método hasta antes de regresar la lista de materias. Y al final almacenamos este valor en un archivo, script 31.

```

$time = microtime(true);
$hoy = getdate();
$fecha = $hoy[mday] . "-" . $hoy[mon] . "-" . $hoy[year];
$hora = $hoy[hours] . ":" . $hoy[minutes] . ":" . $hoy[seconds];
$fp = fopen("tiempokardex.txt", "a");
$cont=$cont*10;
$materia[0]="mate 1";
$materia[1]="mate 2";
$materia[2]="mate 3";
$materia[3]="mate 4";

```

```

        $materia[4]="mate 5";
        $materia[5]="mate 6";
        $materia[6]="progra 1";
        $materia[7]="progra 2";
        $materia[8]="progra 3";
        $materia[9]="progra 4";
        for ($i = 0; $i <= $cont; $i++)
        {
            if ($i <10)
            {
                $nombre=$materia[$i];
                $calif=9+($i*.1);
            }
            else
            {
                $nombre=$materia[rand(0,9)];
                $calif=rand(6,10);
            }
        }
        $arreglo[] = array( 'Nombre'=>$nombre, 'Calif'=>$calif);
    }
    $time_end = microtime(true);
    fputs($fp, $time . "|" . $time_end . "|" . ($time_end-$time) . "|" . $fecha . "|" . $hora .
PHP_EOL);
    fclose($fp);
    return $arreglo;

```

Script 31. Servicio kardex.

El servicio de extraescolares recibe como parámetros el sexo, la edad y el peso. Dependiendo de los parámetros que el cliente mande, el proceso decide que materias le mandara como respuesta y la cantidad de materias serán indicadas por el parámetro extra. Al igual que los servicios anteriores se toma el lapso de tiempo y lo almacena en un archivo, script 32.

```

$time = microtime(true);
$hoy = getdate();
$fecha = $hoy[mday] . "-" . $hoy[mon] . "-" . $hoy[year];
$hora = $hoy[hours] . ":" . $hoy[minutes] . ":" . $hoy[seconds];
$fp = fopen("tiempoextra.txt", "a");
$cont=$cont*10;
if ($sexo=="h")
{
    if ($edad>=17&&$edad<=20)
    {
        if ($peso>=70&&$peso<=80)
        {
            $materias[0]= "h1770baseball";
            $materias[1]= "softball";
            $materias[2]= "basketball";
            $materias[3]= "atletismo";
            $materias[4]= "natacion";
            $materias[5]= "rapel";
            $materias[6]= "mecanica";
            $materias[7]= "albanil";
            $materias[8]= "tenis";
            $materias[9]= "clavados";
            $materias[10]= "carpinteria";
            $materias[11]= "soldadura";
        }
    }
}

```

```

        if ($peso>=81&&$peso<=95)
        {
            $materias[0]= "h1781fisico const";
            $materias[1]= "softball";
            $materias[2]= "basketball";
            $materias[3]= "atletismo";
            $materias[4]= "natacion";
            $materias[5]= "rapel";
            $materias[6]= "mecanica";
            $materias[7]= "alterofilia";
            $materias[8]= "gimnacio";
            $materias[9]= "clavados";
            $materias[10]= "carpinteria";
            $materias[11]= "soldadura";
        }
    }

    ...

    for ($i = 0; $i <= $cont; $i++)
    {
        if ($i <=11)
        {
            $nombre=$materia[$i];
        }
        else
        {
            $nombre=$materia[rand(0,11)];
        }
        $arreglo[] = array( 'Nombre'=>$nombre);
        }$time_end = microtime(true);
    fputs($fp, $time . "|" . $time_end . "|" . ($time_end-$time) . "|" . $fecha . "|" .
$hora . PHP_EOL);
    fclose($fp);
    return $arreglo;

```

Script 32. Servicio extraescolares.

Una vez terminado los servicios se procederá a desplegarlos en la plataforma OpenShift mediante los comandos de git. Para el servicio de beca la dirección será: <http://beca-servescolares.rhcloud.com/becas.php?wsdl>. La dirección para el servicio de kardex será: <http://serv-servescolares.rhcloud.com/kardex.php?wsdl>. Y por último el servicio de extraescolares será:

<http://extra-servescolares.rhcloud.com/Extrasserv.php?wsdl>.

### 3.3.2 Implementación en la plataforma Azure.

Los servicios que serán desplegados en este proveedor son iguales que en OpenShift pero para este proveedor se utilizara la herramienta WebMatrix. Se utilizaron los scripts 30, 31 y 32.

La dirección para kardex será:

<http://servesc.azurewebsites.net/tec/servesc/kardex.php?wsdl>.

Para el servicio de beca será:

<http://servesc.azurewebsites.net/tec/becanet/becas.php?wsdl> y

Por último la dirección para extraescolares será:

<http://servesc.azurewebsites.net/tec/extra/Extrasserv.php?wsdl>.

### 3.3.3 Implementación en la plataforma Koding.

Los servicios que serán desplegados en este proveedor son iguales que en OpenShift solo que aquí se puede desarrollar directamente en la plataforma de Koding. Se utilizaron los scripts 30, 31 y 32.

Para el servicio de kardex la dirección fue:

<http://ubkk17a3e4bb.betosk8.koding.io/Escolares/kardex.php?wsdl>.

Para el servicio de extraescolares:

<http://ubkk17a3e4bb.betosk8.koding.io/Extra/Extrasserv.php?wsdl>

Y para el servicio de beca es:

<http://ubkk17a3e4bb.betosk8.koding.io/becanet/becas.php?wsdl>.

## 3.4 Evaluación del desempeño de los “Enterprise Service Bus” de acuerdo a parámetros de orquestación.

Para la evaluación de los servicios el cliente debe correr en una maquina con las mismas características, para esto se utilizó una computadora con sistema operativo Ubuntu 12.10, memoria RAM de 1893 MB, procesador intel i3 3.10 GHz, en la cual serán desactivadas las funciones menos indispensables para que el cliente consuma los servicios con el mejor desempeño posible. Para de desarrollar el cliente se utilizó el programa Eclipse versión Juno. La computadora está conectada a una red con cable Ethernet y una tarjeta Gigabit Ethernet, la red se comparte con 14 nodos y una salida a Internet con una velocidad de 12 MB.

También otra característica que se debe tomar en cuenta es la transferencia de datos, la cual tiene que ser lo más parecida posible durante las pruebas. Para este fin se utilizó la página speedtest.net y se obtuvo:

52.85 Mbps Download speed y 3.85 Mbps Upload speed cuando se realizaron las pruebas en Azure.

64.61 Mbps Download speed y 5.80 Mbps Upload speed para Koding.

39.27 Mbps Download speed y 3.36 Mbps Upload speed para OpenShift.

Ya que se realizaron las pruebas se obtuvieron los siguientes resultados, para estas pruebas el cliente solicitó 156 veces el servicio con una respuesta de 100 materias por solicitud. Se realizaron 156 pruebas para después poder eliminar espurios. Se eliminaron los tres tiempos más altos y los 3 más bajos quedando así 150 muestras. Estos resultados son el promedio de latencia en segundos por el lado del cliente.

Servicio de Kardex	
Proveedor	Latencia
Azure	1.000793 segundos
OpenShift	0.288627 segundos
Koding	0.53692 segundos

Tabla 3. Latencia en el servicio de kardex.

En la tabla 3 se muestra la latencia que se obtuvo como promedio en el cliente para el servicio de kardex.

Servicio de ExtraEscolares	
Proveedor	Latencia
Azure	1.31736 segundos
OpenShift	0.26368 segundos
Koding	0.378927 segundos

Tabla 4. Latencia en el servicio de extraescolares.

En la tabla 4 se muestra la latencia que se obtuvo como promedio en el cliente para el servicio de extraescolares.

Servicio de Beca	
Proveedor	Latencia
Azure	0.529893 segundos
OpenShift	0.26778 segundos
Koding	0.266093 segundos

Tabla 5. Latencia en el servicio de beca.

En la tabla 5 se muestra la latencia que se obtuvo como promedio en el cliente para el servicio de beca.

Otros resultados obtenidos fueron el tiempo que necesitaba el proveedor para procesar la información que enviaría como respuesta al cliente. Estas respuestas varían entre 20 y 200 materias que el proveedor respondería al cliente. Para estas pruebas se tomaron 106 muestras y al igual que las pruebas anteriores se eliminaron los 3 tiempos más bajos y los 3 más altos, esto con el fin de eliminar espurios.

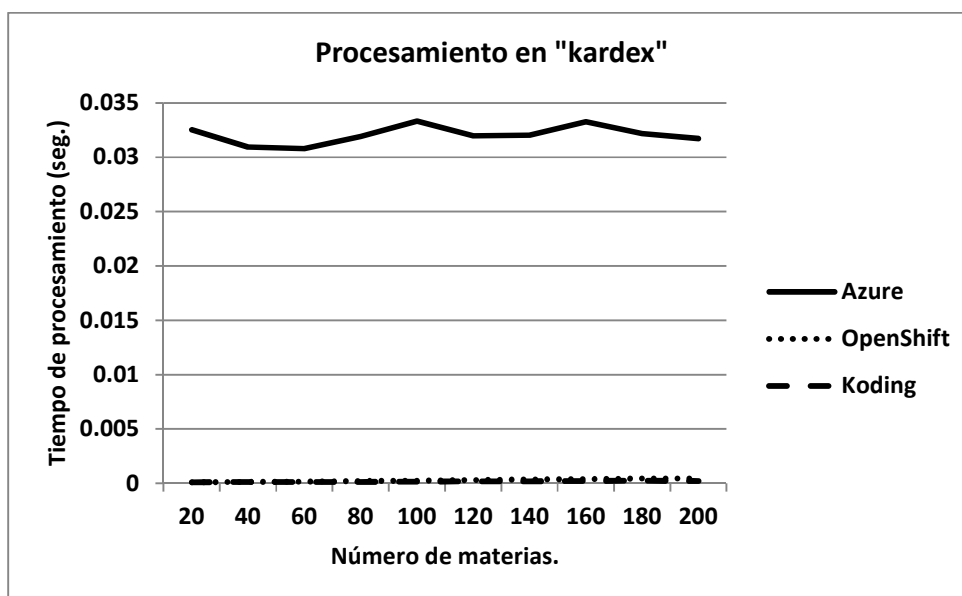


Figura 22-A. Tiempo de procesamiento para el servicio kardex.

En la figura 22-A se muestra la gráfica del tiempo que tardó el proveedor en atender la solicitud del servicio kardex. Debido a que Azure tomó más tiempo se creó otra gráfica para poder observar el comportamiento de los demás proveedores. Ya que al realizar un acercamiento a Azure desaparecía de la imagen. En la figura 22-B se realiza un acercamiento a los tiempos de los proveedores OpenShift y Koding para poder observar su comportamiento.

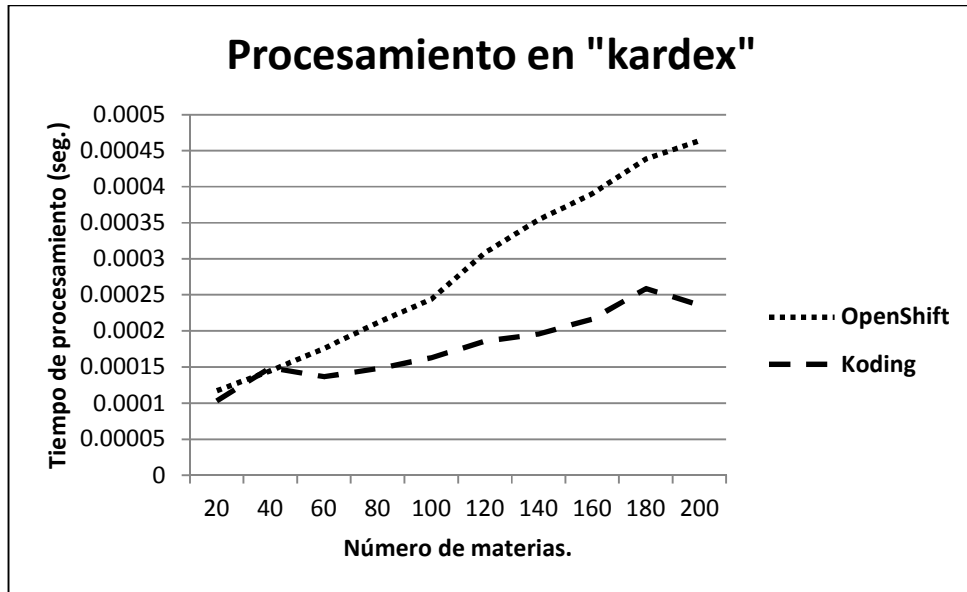


Figura 22-B. Tiempo de procesamiento para el servicio kardex.

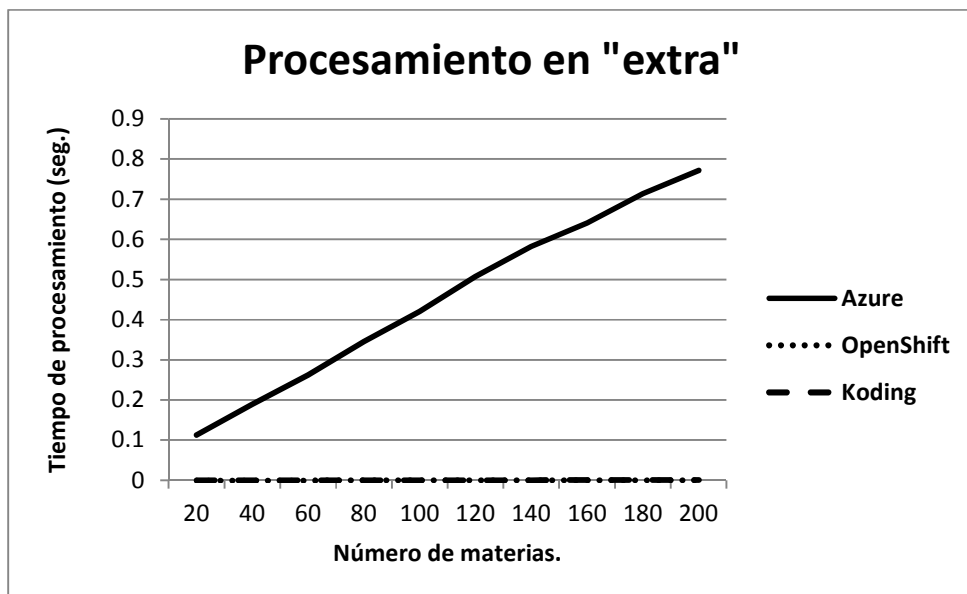


Figura 23-A. Tiempo de procesamiento para el servicio extra.

En la figura 23-A se muestra la gráfica del tiempo que tardó el proveedor en atender la solicitud del servicio extra. Debido a que Azure tomó más tiempo se creó otra gráfica para poder observar el comportamiento de los demás proveedores. Ya que al realizar un acercamiento a Azure desaparecía de la imagen. En la figura 23-B se realiza un acercamiento a los tiempos de los proveedores OpenShift y Koding para poder observar su comportamiento.



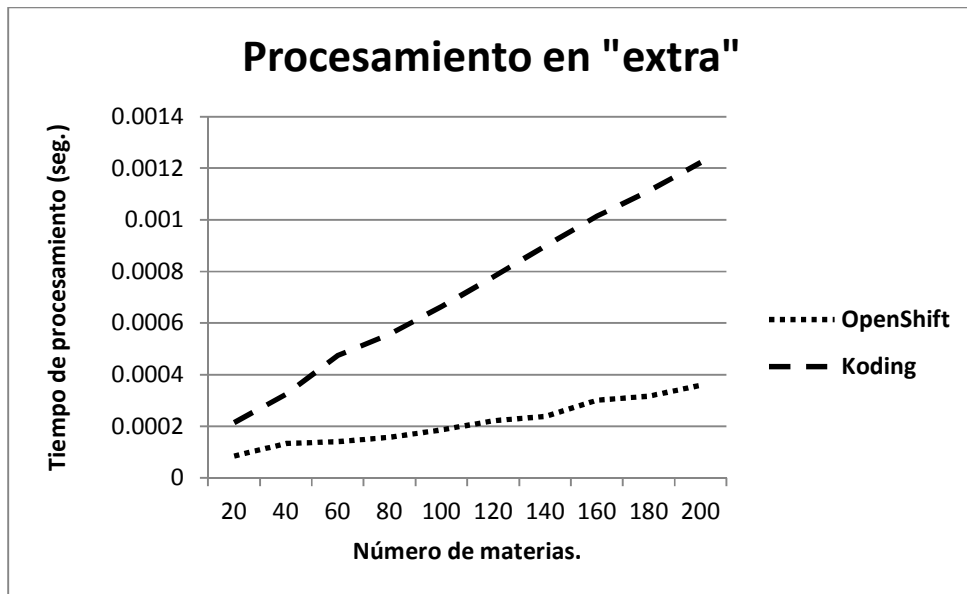


Figura 23-B. Tiempo de procesamiento para el servicio extra.

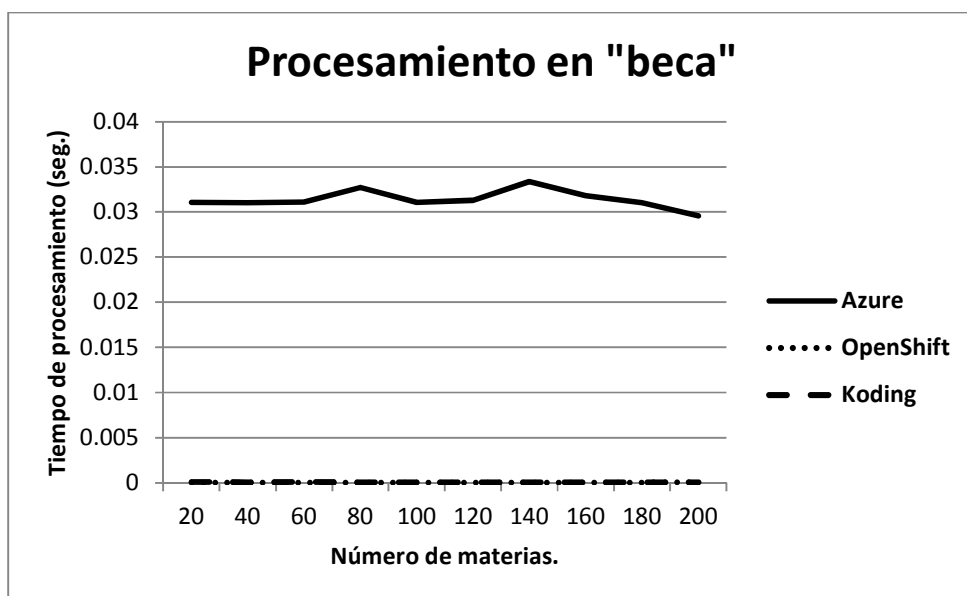


Figura 24-A. Tiempo de procesamiento para el servicio beca.

En la figura 24-A se muestra la gráfica del tiempo que tardó el proveedor en atender la solicitud del servicio beca. Debido a que Azure tomó más tiempo se creó otra gráfica para poder observar el comportamiento de los demás proveedores. Ya que al realizar un acercamiento Azure desaparecía de la imagen. En la figura 24-B se realiza un acercamiento a los tiempos de los proveedores OpenShift y Koding para poder observar su comportamiento.

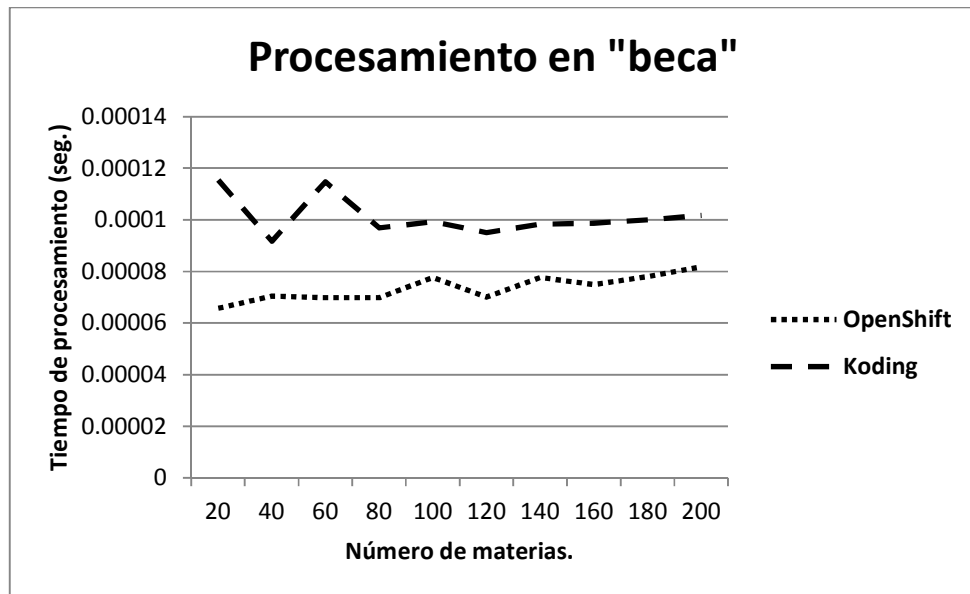


Figura 24-B. Tiempo de procesamiento para el servicio beca.

## Capítulo 4

### 4.1 Conclusiones

Tomando en cuenta los parámetros de QoS planteados en la propuesta se puede concluir que el mejor proveedor en la capa PaaS es OpenShift.

En comparación de OpenShift y Kodig, el primero mostro mejores resultados que el segundo, pero las características y herramientas de desarrollo son mejores en Kodig ya que brinda realmente una plataforma en la nube como tal, cosa que en OpenShift es limitado.

Por otro lado el proveedor Kodig ofrece una serie de herramientas para el desarrollo y despliegue de aplicaciones directamente en la nube, cosa que no se puede realizar con el proveedor OpenShift. Por lo tanto se puede mencionar que Kodig es una muy buena herramienta para el desarrollo y despliegue de aplicaciones debido a los resultados obtenidos, a la funcionalidad y herramientas que ofrece el proveedor a través de su plataforma. Kodig demostró un buen resultado en las pruebas que se realizaron a pesar de que es un proveedor gratuito, fue mejor que Azure.

Contrariamente de que Azure es un proveedor de paga fue el que más tiempo tomo para realizar los procesos, este proveedor ofrece un amplio repertorio de herramientas para el desarrollo y despliegue de aplicaciones pero todo bajo un costo.

Cabe mencionar que si el usuario o empresa cuentan con una buena solvencia económica y puede sustentar una suscripción a Azure, este podría ser una buena opción por toda la serie de herramientas que cuenta su plataforma y también al presentársele alguna serie de problemas, como Azure pertenece a Microsoft tiene un amplio respaldo en cuanto a soporte.

Por otro lado si el usuario o empresa no cuentan con tanto capital y desean implementar una aplicación con un buen funcionamiento OpenShift es el indicado, o también Kodig es una buena opción, tanto para este tipo de consumidores o usuarios en ambiente académico, ya que brinda una plataforma más amigable en comparación con OpenShift.

## 4.2 Bibliografía

[1] Ayala, D. and Nichol, S.

NuSOAP - SOAP Toolkit for PHP. **2013**

<http://sourceforge.net/projects/nussoap/>

[2] Bryant, C.

A Guide to Open Source Cloud Computing Software (PaaS).

**2014.**

<http://www.tomsitpro.com/articles/open-source-cloud-computing-software,2-754-6.html>

[3] Erl, T. and Mahmood, Z. and Puttini., R.

Cloud Computing, Concepts, Technology & Architecture.

Prentice Hall, **2013**.

[4] Geetha, E. and Suresh, T. and Rajani, K. and Ram, Ch. and Reddy, R. and Raghavendra, R.

Qos of web service: survey on performance and scalability.

B M S College of Engineering, Bangalore, India

M S Ramaiah Institute of Technology, Bangalore, India, **2013**

[5] Goyal, M. and Gupta, P. and Aggarwal, A. and Kumar, P.

QoS based trust management model for Cloud IaaS.

Parallel Distributed and Grid Computing (PDGC), 2012 2nd IEEE International Conference on, **2012**, 843-847

[6] Gusev, M. and Ristov, S.

Superlinear speedup in Windows Azure cloud.

Cloud Networking (CLOUDNET), 2012 IEEE 1st International Conference on, **2012**, 173-175.

[7] Harvey, C.

60 Open Source Apps You Can Use in the Cloud.

**2014**

<http://www.datamation.com/open-source/60-open-source-apps-you-can-use-in-the-cloud-1.html>

- [8] Joyanes, L.  
Computación en la Nube e innovaciones tecnológicas.  
Grupo de Investigación, Consultoría y Análisis en Tecnologías de la Información y la Comunicaciones (TIC) y Sociedad de la Información y el Conocimiento – (ICA -GISSIC).  
**2011**
- [9] Kanwal, A. and Masood, R. and Ghazia, U. and Shibli, M. and Abbasi, A.  
Assessment Criteria for Trust Models in Cloud Computing.  
Green Computing and Communications (GreenCom), 2013 IEEE and Internet of Things (iThings/CPSCoM), IEEE International Conference on and IEEE Cyber, Physical and Social Computing, **2013**, 254-261.
- [10] Ladan, M. I.  
Web Services Metrics: A Survey and A Classification.  
2011 International Conference on Network and Electronics Engineering IPCSIT vol.11,  
**2011**
- [11] Li, A. and Yang, X. and Kandula, S. and Zhang, M.  
Comparing Public-Cloud Providers.  
Internet Computing, IEEE, **2011**, 15, 50-53.
- [12] Lifka, D. and Foster, I. and Mehringer, S. and Parashar, M. and Redfern, R. and Stewart, C. and Tuecke, S.  
XSEDE Cloud Survey Report  
The Extreme Science and Engineering Discovery Environment, **2013**
- [13] Malawski, M. and Kuźniar, M. and Wójcik, P. and Bubak, M.  
How to Use Google App Engine for Free Computing.  
Internet Computing, IEEE, **2013**, 17, 50-59
- [14] Mares, O. and Villavicencio, A. and Delgadillo, O. and Ibañez, J. and Díaz, A.  
Análisis Comparativo entre .Net y Mono.  
Tercer Congreso Nacional de Tecnologías de la Información y Comunicación (Conatic 3.0). Tercer Congreso Nacional de Tecnologías de la Información y Comunicación (Conatic 3.0)., **2013**

- [15] Panettieri, J. and Katz, A.  
Top 100 Cloud Services Providers: 2013 Edition  
Talkin' Cloud, **2013**
- [16] Papazoglou, M. P.  
WEB SERVICES: PRINCIPLES AND TECHNOLOGY.  
Prentice Hall, **2008**
- [17] Peng, J. and Zhang, X. and Lei, Z. and Zhang, B. and Zhang, W. and Li, Q.  
Comparison of Several Cloud Computing Platforms.  
Information Science and Engineering (ISISE), **2009** Second International Symposium on, 2009, 23-27.
- [18] Qu, L. and Wang, Y. and Orgun, M.  
Cloud Service Selection Based on the Aggregation of User Feedback and Quantitative Performance Assessment.  
Services Computing (SCC), 2013 IEEE International Conference on, **2013**, 152-159.
- [19] Rimal, B. and Choi, E. and Lumb, I.  
A Taxonomy and Survey of Cloud Computing Systems.  
INC, IMS and IDC, 2009. NCM '09. Fifth International Joint Conference on, **2009**, 44-51.
- [20] Rigsby, J.  
Koding Makes Development Social With New Browser-Based Tool. **2012**.  
<http://www.cmswire.com/cms/web-experience/koding-makes-development-social-with-newbrowserbased-tool-016706.php>
- [21] Nadir, K. and Zang, T.  
Survey and comparison for Open and closed sources in cloud computing.  
International Journal of Computer Science Issues (IJCSI); Vol. 9 Issue 3, **2012**, p118
- [22] Shawky, D. and Ali, A.  
Defining a measure of cloud computing elasticity.  
Systems and Computer Science (ICSCS), 2012 1st International Conference on, **2012**, 1-5

- [23] Takako, P. and Estácio, G. and Kelner, J. and Sadok, D.  
A Survey on Open-source Cloud Computing Solutions.  
VIII Workshop em Clouds, Grids e Aplicações (WCGA), Federal University of  
Pernambuco, Brazil, **2010**.
- [24] Zhou, M. and Zhang, R. and Zeng, D. and Qian, W.  
Services in the Cloud Computing era: A survey.  
Universal Communication Symposium (IUCS), 2010 4th International, **2010**, 40-46
- [25] Williams, A.  
Survey: What are the Best Open Source Cloud Projects?  
**2014**  
<http://www.linux.com/news/enterprise/cloud-computing/781062-survey-what-are-the-best-open-source-cloud-projects>
- [26] Koding (2009). <http://www.koding.com>
- [27] OpenShift (2014). <https://www.openshift.com>
- [28] Azure (2014). <http://azure.microsoft.com>

## 4.3 Apéndice A

Script 1. Método insertar . . . . .	21
Script 2. Función para tomar el tiempo . . . . .	21
Script 3. Función para conectase a la basa de datos . . . . .	21
Script 4. Método para insertar a la base de datos . . . . .	21
Script 5. Operación matemática para conocer el lapso de tiempo . . . . .	22
Script 6. Función insertar solicitada desde el cliente . . . . .	22
Script 7. Función validaUsuario . . . . .	23
Script 8. Método para conectarse a la base de datos . . . . .	23
Script 9. Método para buscar al usuario y contraseña . . . . .	23
Script 10. Decisión del servicio . . . . .	23
Script 11. Método para capturar usuario y contraseña . . . . .	23
Script 12. Solicitud del servicio . . . . .	24
Script 13. Método inserta . . . . .	24
Script 14. Conexión a la base de datos . . . . .	24
Script 15. Función para insertar datos . . . . .	24
Script 16. Método actualiza . . . . .	25
Script 17. Método para actualizar el valor . . . . .	25
Script 18. Método para revisar el proceso . . . . .	25
Script 19. Método elimina . . . . .	25
Script 20. Función para eliminar el usuario . . . . .	25
Script 21. Método que realiza el cliente . . . . .	26
Script 22. Función actualiza del cliente . . . . .	27
Script 23. Método elimina del cliente . . . . .	27
Script 24. Método inserta . . . . .	30
Script 25. Método para registrar el servicio web . . . . .	31
Script 26. Publicación del servicio . . . . .	31
Script 27. Funciones de git . . . . .	31
Script 28. Asignación de parámetros . . . . .	32
Script 29. Consumo del servicio insertar. . . . .	32
Script 30. Servicio beca . . . . .	42
Script 31. Servicio kardex . . . . .	43
Script 32. Servicio extraescolares . . . . .	44