ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

IMPLEMENTACIÓN DE UN PROTOTIPO DE SOFTWARE COMO SERVICIO (SAAS) PARA PEQUEÑAS Y MEDIANAS EMPRESAS

PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y REDES DE INFORMACIÓN

DANIEL OMAR NÚÑEZ ESPINOZA

danielnuneze@gmail.com

DIRECTOR: ING. DAVID MEJíA, MSc.

david.mejia@epn.edu.ec

Quito, Diciembre 2013

DECLARACIÓN

Yo, Daniel Omar Núñez Espinoza, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedo mis derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

_____

Daniel Omar Núñez Espinoza

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Daniel Omar Núñez Espinoza, bajo mi supervisión.

_____

Ing. David Mejía Navarrete, MSc.

DIRECTOR DE PROYECTO

AGRADECIMIENTOS

Agradezco por todo el apoyo brindado para la realización de este proyecto, en primer lugar, al Director del Laboratorio de Informática de la Facultad de Ingeniería Eléctrica y Electrónica, Ing. Xavier Calderón, por facilitarme los equipos y el lugar necesario para el desarrollo del prototipo.

A los ayudantes del Laboratorio de Informática por brindarme su ayuda en los momentos difíciles, así como la amistad brindada especialmente a Gabriela Morillo, Christian Jácome, Walter Salas y Lenin Vargas.

A toda la comunidad de usuarios y desarrolladores OpenStack internacional, quienes sin interés alguno compartieron conmigo su experiencia y consejo para el uso de la herramienta OpenStack. En especial, al coordinador de la Comunidad OpenStack Venezuela, Ing. Ender Mujica.

En general a la Escuela Politécnica Nacional y sus docentes, por enseñarme el valor del esfuerzo y dedicación constante para alcanzar grandes objetivos.

DEDICATORIA

Es este trabajo dedicado a toda mi familia, a mis queridos padres, quienes siempre me apoyaron en las largas y agotadoras jornadas de estudio. Mi madre que con su eterno amor y preocupación siempre dio un impulso en mi vida para avanzar y no darme por vencido. Mi padre por su comprensión y ayuda, fue quien acompaño muy de cerca mi vida universitaria y a quien podía acudir cuando mis fuerzas mermaban. Agradezco por su preocupación y cuidado a mi salud.

A mi amado hermano Cristian porque en los momentos en que más lo necesitaba, supo sacarme una sonrisa con sus ocurrencias.

Una mención especial a aquel hombre que estuvo junto a mí, quien compartía con-migo toda su sabiduría y experiencias de vida. Mi abuelito Mesías que en paz des-canse.

Mi familia, ñaño Jorge, tía Mirela y tío Pepe, mis queridos primos Juan, Carlos, Jhon y María José. Tanto cariño he recibido de ustedes que siempre fueron en mí un empuje para poner muy en alto a la familia Espinoza.

Quienes viajaron junto a mí en este largo camino universitario. Mis entrañables amigos: Pepe Valencia, Santiago Morejón, Fabricio Vélez, Sergio Narváez, David Zambonino, Sebastián Almagro, Liliana Guanga y Susan Feijó, quienes estuvieron en las buenas y las malas, en exámenes, proyectos, una simple reunión, en la Asociación de Estudiantes y tantas locuras que forman parte de los lindos momentos que siempre tendré presente.

CONTENIDO

ÍNDICE DE FIGURAS

Capítulo 1

Capítulo 2

Capítulo 3

Capítulo 4

ÍNDICE DE TABLAS

Capítulo 2

Capítulo 4

ÍNDICE DE CÓDIGOS

Capítulo 3

Capítulo 4

RESUMEN

El Presente Proyecto comprende el desarrollo de un sistema para la provisión de servicios de software para Pequeñas y Medianas Empresas (PYME), mediante la implementación de un Prototipo de Proveedor de SaaS (*Software as a Service*) usando los recursos de una infraestructura de nube privada desarrollada con el software OpenStack. Se plantea además, el costo de producción y precio de venta al público de cada uno de los servicios ofertados para que sean competitivos en el mercado actual y genere rentabilidad a mediano plazo.

En el primer capítulo se presentan conceptos sobre *Cloud Computing,* sus componentes, aplicaciones y usos; los diferentes modelos de servicio, sus características y principales escenarios de uso. Se describe además varias tecnologías para la creación y gestión de nubes privadas.

En el segundo capítulo se analiza con mayor detenimiento el software para la creación de nubes públicas y privadas denominado OpenStack; su arquitectura y la manera en que interactúan sus componentes para la provisión de recursos de una Infraestructura como Servicio (IaaS), y el modo en que los usuarios finales pueden hacer uso de estos recursos para su integración con soluciones externas. Este capítulo también abarca el proceso de instalación de la infraestructura de red y la configuración de los componentes de software de OpenStack en cada uno de los servidores.

El tercer capítulo describe todo el proceso de desarrollo del Prototipo de Proveedor de SaaS. Aquí se muestra la creación de los recursos de IaaS para su posterior manipulación con el API (Interfaz de Programación de Aplicaciones) de OpenStack, para de esta manera desarrollar los *scripts* que suministren la automatización al sistema. Por último, se muestra el desarrollo de la interfaz web con la cual el cliente puede interactuar para la administración de los servicios contratados.

El cuarto capítulo presenta los resultados de las pruebas realizadas con usuarios del sistema, así como la realimentación generada por parte del administrador ante las solicitudes de los usuarios. Además, se realiza un análisis financiero para la puesta en producción del Proveedor de SaaS con las características propuestas en los capítulos 2 y 3. Basado en este análisis se presenta el posible costo de producción y el precio de venta al público de cada uno de los servicios. Adicionalmente, se realiza una estimación de las ventas para un periodo de 3 años y la rentabilidad generada respecto a la inversión inicial.

Finalmente, en el capítulo final se presentan las principales conclusiones y recomendaciones obtenidas durante la elaboración de este Proyecto.

Se incluyen los anexos sobre los principales archivos de configuración de la infraestructura OpenStack, además del código desarrollado para la automatización y la página web de administración.

En el CD adjunto se puede encontrar el código de los scripts generados para la automatización del uso de los recursos de la infraestructura, así como el código de la página web de administración. Además se encuentra incluido el manual de usuario para el uso de la interfaz web de administración.

PRESENTACIÓN

El auge en la oferta de servicios de *Cloud Computing* en los últimos años ha provocado un cambio radical en la forma que las empresas usan los recursos de TI (Tecnologías de Información). Con el afán de reducir los costos de producción y mejorar sus ganancias, buscan soluciones efectivas en tecnologías de información de baja inversión inicial, poco mantenimiento y de rápido aprendizaje, que agilite las operaciones para ofrecer más y mejores servicios a sus clientes.

El *Cloud Computing* es un paradigma que permite ofertar servicios de cómputo a través de Internet, estos servicios no son más que aplicaciones que se ejecutan sobre una máquina física o posiblemente virtual cuya ubicación (física o virtual) le es totalmente desconocida al usuario del servicio, por otro lado, los datos pueden ser almacenados en ubicaciones desconocidas, la administración de estos sistemas pueden estar bajo la responsabilidad de un tercero y el acceso a estos servicios se lo puede realizar desde cualquier lugar. Son todas estas características las que ofrecen muchos beneficios a las PYME que comúnmente no tienen los recursos suficientes para establecer su propia infraestructura de TI.

La provisión de servicios de *Cloud Computing* se ofrece mediante tres modelos. El modelo IaaS acoge el uso de los recursos de la parte más baja de la infraestructura ya que deja bajo la administración del proveedor el manejo de la red, el almacenamiento, los servidores y la virtualización. En el modelo PaaS (*Platform as a Service*), el proveedor adicionalmente se encarga de la administración del Sistema Operativo, el *middleware y el runtime*. Por último, en el modelo SaaS el proveedor se encarga de la administración de todas las instancias para el correcto despliegue de las aplicaciones hacia el cliente final, por lo general por medio de una interfaz web.

Teniendo en cuenta la necesidad de las PYME, y tomando como base las ventajas que ofrece el modelo SaaS para la provisión de servicios, se plantea el Presente Proyecto de Titulación titulado “IMPLEMENTACIÓN DE UN

PROTOTIPO DE SOFTWARE COMO SERVICIO (SAAS) PARA PEQUEÑAS Y MEDIANAS EMPRESAS".

En este Proyecto se propone el diseño y la implementación de la infraestructura de *Cloud Computing* empleando el software OpenStack, el mismo que ha sido instalado y probado en servidores de tipo genérico. Sobre esta infraestructura se configuran los componentes básicos de una nube privada para la provisión de recursos computacionales y así se despliegan los servicios solicitados por los usuarios de la misma.

Los servicios con los que el Presente Proyecto demuestra la provisión de SaaS son: correo electrónico, CRM (*Customer Relationship Management*) y video conferencia web. Usando el API de OpenStack, se desarrolló un sistema automatizado basado en páginas web, para que un usuario pueda accesar a estas aplicaciones, configurarlas, usarlas y pague por ellas según los recursos de cómputo asignados.

Esta solución se presenta además como un proyecto empresarial viable que genere cierta rentabilidad por la inversión de capital inicial. Se exhiben estimaciones de ventas, considerando un reducido sector de las PYME en Ecuador como nicho de mercado, apalancado en el crecimiento de la demanda de servicios de *Cloud Computing*, especialmente servicios de tipo SaaS, a nivel latinoamericano.

CAPÍTULO 1
FUNDAMENTOS TEÓRICOS

1.1 *CLOUD COMPUTING*

Es un nuevo modelo de prestación de servicios de computación a través de Internet. Implica principalmente dos conceptos: Abstracción y Virtualización.

La Abstracción se refiere a que el usuario no se involucra en los detalles de la implementación. Desde un enfoque en donde las aplicaciones se ejecutan sobre un computador (físico o virtual) que no se especifica, los datos se almacenan en ubicaciones desconocidas, la administración de los sistemas está bajo responsabilidad de un tercero y se provee acceso a esta infraestructura desde cualquier lugar.

Por otro lado, la virtualización se refiere a la habilidad para crear sistemas que parezcan independientes ante los usuarios a través de mecanismos de compartición y asignación dinámica de los recursos de cómputo [1].

A diferencia de los servicios computacionales tradicionales, *Cloud Computing* permite aumentar el número de servicios basados en la red, donde los proveedores tienen la capacidad de ofrecer de manera rápida y eficiente una mayor cantidad de servicios. Los usuarios pueden acceder a un catálogo de servicios estandarizados y responder a las necesidades de su negocio, de forma flexible y adaptativa en caso de aumento de la demanda o de picos de trabajo, disfrutando de la transparencia e inmediatez bajo un modelo de pago bajo consumo. Según la IEEE[1] *Society*, *Cloud Computing* es un paradigma en el que la información se almacena de manera permanente en servidores de Internet y se envía a cachés temporales de cliente [2].

[1] IEEE: *Institute of Electrical and Electronics Engineers,* es una organización técnico-profesional mundial dedicada a la estandarización entre otras cosas.

El Laboratorio de Tecnologías de la Información, integrado por el Instituto Nacional de Estándares y Tecnologías (NIST por sus siglas en inglés) del Departamento de Comercio del Gobierno de Estados Unidos, han definido a *Cloud Computing* como:

¨*Cloud Computing* es un modelo que permite el acceso bajo demanda y a través de la red a un conjunto de recursos compartidos y configurables (redes, servidores, capacidad de almacenamiento, aplicaciones y servicios) que pueden ser rápidamente asignados y liberados con una mínima gestión por parte del proveedor del servicio¨ [3].

NIST también menciona cinco características esenciales que cualquier servicio de *Cloud Computing* debe ofrecer a sus usuarios:

1. Servicio bajo demanda.
2. Amplio acceso a la red.
3. Compartición de recursos.
4. Elasticidad.
5. Servicio a la medida.

1.1.1 SERVICIO BAJO DEMANDA

Un usuario determina explícitamente las capacidades de cómputo, tales como el tiempo de uso de los recursos de uno o varios servidores, según sea necesario, y de forma automática sin necesidad de interacción humana con el proveedor de servicios.

1.1.2 AMPLIO ACCESO A LA RED

Los recursos son accesibles y están disponibles a través de Internet, y son utilizados por una amplia variedad de dispositivos de usuario, como por ejemplo: teléfonos móviles, *tabletas*, portátiles y estaciones de trabajo.

1.1.3 COMPARTICIÓN DE RECURSOS

Los recursos de cómputo del proveedor (memoria, capacidad de procesamiento, almacenamiento, máquinas virtuales, etc.) son compartidos dinámicamente, y pueden ser asignados y reasignados de acuerdo a la demanda del consumidor. El cliente generalmente no tiene ningún control o conocimiento sobre la ubicación exacta de los recursos asignados, pero puede ser capaz de especificar la ubicación (por ejemplo, país, estado, o *Data Center*).

1.1.4 ELASTICIDAD

Los recursos de cómputo pueden ser asignados y liberados rápidamente, casi siempre de forma automática, para escalar rápidamente según la demanda de los usuarios. Esto genera en el consumidor la sensación de disponer de capacidades ilimitadas disponibles en cualquier momento.

1.1.5 SERVICIO A LA MEDIDA

El proveedor es capaz de medir el consumo real de los recursos de cómputo que el usuario está utilizando en tiempo real, lo que posibilita el pago por el uso efectivo de los mismos.

1.2 MODELOS DE INFRAESTRUCTURA DE *CLOUD COMPUTING*

Un modelo de despliegue de la infraestructura define el objetivo de la nube y la naturaleza en que la nube se encuentra físicamente.

1.2.1 NUBE PÚBLICA

Un usuario puede tener libre acceso a la infraestructura *cloud* de un proveedor con solo estar conectado a internet, como se muestra en la Figura 1.1. El término “público” no necesariamente significa libertad, a pesar de que puede ser libre o bastante económico de usar. Se debería entender de manera análoga a los servicios públicos convencionales, como: agua, electricidad, telefonía, transporte, a los que se puede acceder fácilmente mediante el pago por su consumo.

Los proveedores de nube pública suelen proporcionar a sus usuarios un mecanismo de control de acceso para que sus datos no sean públicamente visibles.

Figura 1. 1 Diagrama de Nube Pública [4]

1.2.2 NUBE PRIVADA

La organización dueña de la infraestructura física es la que mantiene, maneja y administra la infraestructura *cloud*, para ofrecer servicios a los usuarios. En la Figura 1.2 se representan los componentes de una red corporativa habitual, como la página web interna, el CRM[2], el servidor de aplicaciones, los sistemas de almacenamiento, y todo esto asegurado por medio de un *firewall*[3].

Ofrece muchas de las ventajas de un entorno de computación en la nube pública, tales como elasticidad y alta disponibilidad; la diferencia radica en que una nube privada maneja sus servicios, datos y procesos sin las restricciones de ancho de banda, exposiciones a la seguridad y requisitos legales que implica el uso de recursos de nube pública.

---

[2] CRM: *Customer Relationship Managment*, sistema informático de apoyo a la gestión de las relaciones con los clientes, a la venta y al marketing.

[3] *Firewall*: Es una parte de un sistema o una red que está diseñada para bloquear el acceso no autorizado, permitiendo al mismo tiempo comunicaciones autorizadas.

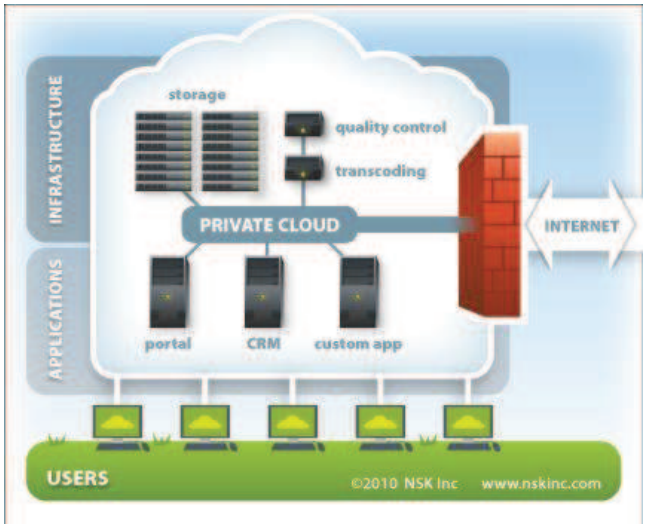


Figura 1. 2 Diagrama de Nube Privada [5]

1.2.3 NUBE HÍBRIDA

Es una combinación de uno o más tipos de *cloud*s (privada y pública como se observa en la Figura 1.3) que interactúan armónicamente para ofrecer servicios a usuarios privados.

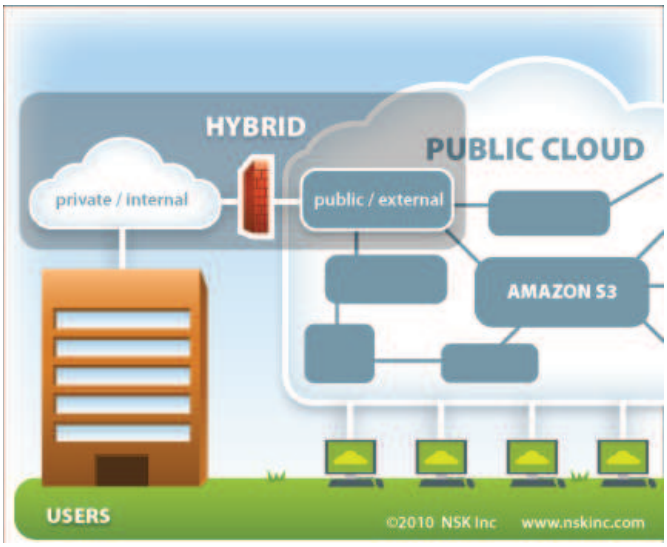


Figura 1. 3 Diagrama de Nube Híbrida [6]

En este modelo se suele subcontratar un proveedor de nube pública para el manejo de datos no críticos, manteniendo bajo su control los recursos y datos críticos dentro de su infraestructura *cloud* privada.

1.2.4 NUBE COMUNITARIA

Es controlada y utilizada por un grupo de organizaciones que tiene intereses en común, como misión, requerimientos de seguridad, políticas, etc. Los miembros de la o las comunidades comparten el acceso a los datos, recursos y servicios que provee su infraestructura. Por ejemplo, un grupo de organizaciones de atención sanitaria podría crear una nube comunitaria para almacenar historiales clínicos de pacientes.

1.3 ARQUITECTURA DE CAPAS Y MODELOS DE SERVICIO

Desde el punto de vista del proveedor, la arquitectura de *Cloud Computing* se compone de tres capas. Como se observa en la Figura 1.4 la capa más baja corresponde al Hardware, seguida por la virtualización de los recursos informáticos y por último la capa de servicios, que a su vez engloba a los diferentes tipos de servicios: IaaS, PaaS y SaaS.

Figura 1. 4 Arquitectura en capas de *Cloud Computing*

1.3.1 HARDWARE

En la Figura 1.5 se observa los componentes principales de hardware que soportan los servicios de *Cloud Computing*. Esta infraestructura envuelve un sin número de ordenadores apilados unos sobre otros, por lo general en uno o varios *Data Centers*.

El costo de estos *Data Centers* especializados varía enormemente dependiendo del tipo de cargas de trabajo que soporten. Por ejemplo, con la variación del almacenamiento de datos.

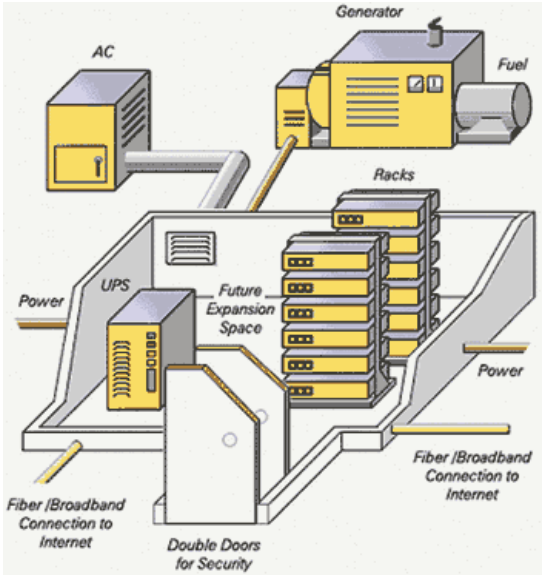


Figura 1. 5 Diagrama de Hardware de un *Data Center* [7]

1.3.1.1 Equipos de refrigeración

Los equipos activos presentes en un *Data Center* necesitan ser enfriados por sistemas especializados que reduzcan las temperaturas de las placas, chips, discos, etc. Cuando estos sistemas son enfriados mediante aire acondicionado, se requiere una gran cantidad de energía. Sin embargo, los *Data Centers* pueden diseñarse para emplear enfriamiento por agua, que es un sistema 40% más eficiente que un sistema de aire acondicionado [8].

1.3.1.2 CPU, memoria y discos

El uso de recursos de procesamiento, memoria y almacenamiento tiende a saturarse con una gran cantidad de datos y procesos excedentes (debido a las cargas de trabajo no previstas, o porque una aplicación o proceso no fue diseñado para ser eficiente), por lo que se tiene una ocupación valiosa de memoria, CPU y discos, innecesariamente. Un típico *Data Center* para *Cloud Computing* soporta el aprovisionamiento continuo de nuevos recursos cuando se presente un incremento en la carga de trabajo del sistema.

1.3.1.3 Almacenamiento y Redes

Es necesario que tanto el almacenamiento como las redes de datos sean administrados en conjunto para que sean eficientes. El rendimiento de todo el

sistema depende en gran medida en la forma en que los datos se transportan hasta los *clusters*[4] de procesamiento a través de las redes de comunicaciones. El empleo de mejores redes de almacenamiento evitará la compra innecesaria de hardware y software adicional, que es un problema recurrente en los *Data Centers* tradicionales.

1.3.1.4 Redundancia

Los *Data Centers* siempre deben mover sus datos a través de la red al sistema de respaldos y recuperación de desastres. Estos sistemas deben estar preparados para soportar las diversas cargas de trabajo del *Data Center*, sin afectar la eficiencia del servicio y no añadir mayores costos y complejidad.

1.3.1.5 Software Integrado

El tipo de software ligado a nivel de hardware representa un costo muy alto en el *Data Center* tradicional, simplemente porque hay cargas de trabajo para muchos sistemas operativos y sus elementos de software están para ayudar en su ejecución más eficiente sobre el hardware.

1.3.2 VIRTUALIZACIÓN

Virtualizar un computador significa aparentar que se trata de múltiples computadores o de un computador completamente diferente.

Involucra la simulación de un computador físico, el cual se encuentra alojado en un sistema anfitrión y se ejecuta a través de un software llamado “hipervisor”. Este software administra el hardware del anfitrión para crear un ambiente simulado donde el usuario puede crear máquinas virtuales con los recursos de hardware compatibles con el sistema operativo elegido.

1.3.2.1 Importancia de la Virtualización

Desde una perspectiva de negocio, hay muchas razones para utilizar virtualización. La mayoría están relacionadas con la consolidación de servidores.

---

[4] *Cluster*: Término que se aplica al conjunto o conglomerado de computadoras construidos mediante la utilización de hardware común y que se comportan como si fuesen una única computadora.

Simplemente, si se puede virtualizar un número de sistemas en un solo servidor físico, se ahorrará energía, espacio, capacidad de refrigeración y administración ya que se usarán los recursos de un solo servidor físico de manera más eficiente. Como puede ser difícil determinar el grado de utilización de un servidor, las tecnologías de virtualización soportan la migración en directo. La migración en directo permite que un sistema operativo y sus aplicaciones se muevan a un nuevo servidor para balancear la carga sobre el hardware disponible.

1.3.2.2 Tecnologías de Virtualización

1.3.2.2.1 KVM (Kernel-based Virtual Machine)

KVM es un software de código abierto que provee una solución completa de virtualización para Linux sobre hardware x86 con extensiones de virtualización (Intel VT[5] o AMD-V[6]). Se compone de un módulo de *kernel*, kvm.ko, que proporciona la infraestructura de virtualización de base y un módulo de procesador específico, intel.ko o amd.ko. QEMU proporciona toda la funcionalidad de hardware virtual para los sistemas virtualizados. La mayoría de distribuciones de GNU/Linux como Debian, Ubuntu, RHEL, Centos, Fedora, OpenSuSe, Mandriva tienen soporte por defecto de KVM, además de BSD, Solaris, Mac y Windows. También provee mayor seguridad entre las máquinas virtuales y una configuración más amigable para los administradores.

El componente de *kernel* de KVM está incluido en el código principal de Linux, a partir de febrero del 2006.

1.3.2.2.2 LXC – Linux Containers

Los contenedores proporcionan virtualización ligera que les permite aislar los procesos y los recursos sin la necesidad de establecer mecanismos de interpretación e instrucciones propias.

---

[5] Intel VT: *Intel Virtualization Technology*, es la tecnología de virtualización desarrollada por Intel para la arquitectura x86.

[6] AMD-V: Tecnología de virtualización de la compañía AMD (*Advanced Micro Devices, Inc.*) para arquitecturas x86 y nativa en todos los procesadores que saca al mercado.

Un único sistema operativo particiona los recursos administrados en grupos aislados llamados “contenedores”. Los contenedores pueden ejecutar las instrucciones nativas del CPU sin ningún mecanismo de interpretación especial.

Al proporcionar una manera de crear y entrar en los contenedores, un sistema operativo da a las aplicaciones la ilusión de que se ejecutan en una máquina separada, mientras que al mismo tiempo comparte muchos de los recursos subyacentes. Este intercambio a menudo se puede extender a otros archivos en los directorios que no necesiten ser escritos o editados.

Existe un ahorro significativo debido a la compartición de estos recursos, mientras que también proporciona el aislamiento de los procesos de cada uno de los sistemas operativos. Esto significa que los contenedores tienen sobrecarga significativamente más baja que otros modos de virtualización.

1.3.2.2.3 QEMU- Quick EMUlator

QEMU es un emulador de máquina que puede ejecutar sin modificar el sistema operativo virtual (como Windows o Linux) todas las aplicaciones en una máquina virtual. QEMU se ejecuta en varios sistemas operativos, tales como Linux, Windows y Mac OS X. Permite que las máquinas virtuales puedan ser fácilmente detenidas, y sus estados puedan ser inspeccionados, almacenados y restaurados. Además, se pueden simular dispositivos externos mediante la adición de nuevos dispositivos emulados.

Es un emulador de procesadores basado en la traducción dinámica de binarios (conversión del código binario de la arquitectura fuente, en código entendible por la arquitectura huésped).

Esta máquina virtual puede ejecutarse en cualquier tipo de microprocesador o arquitectura (x86, x86-64, PowerPC, MIPS[7], SPARC[8], etc.). Está licenciado en parte con la LGPL[9] y la GPL[10].

---

[7] MIPS: *Microprocessor without Interlocked Pipeline Stages*, nombre asignado a toda una familia de microprocesadores de arquitectura RISC desarrollados por MIPS *Technologies*.

1.3.2.2.4 UML – User Mode Linux

Crea un sistema Linux completamente funcional en un sistema Linux. UML tiene numerosas aplicaciones en el mundo de Linux. Muchos desarrolladores confían en UML para probar sus aplicaciones sin poner todo el sistema en peligro. Los usuarios de Linux pueden ejecutar UML para experimentar con versiones del *kernel* sin tener que preocuparse de parches nuevos o experimentales [9].

Los administradores de sistemas pueden usar UML para probar configuraciones. Es posible incluso ejecutar varias versiones de UML en el mismo equipo para simular una red.

UML es un *kernel* de Linux que se ejecuta como un proceso. La diferencia entre un *kernel* UML y un *kernel* ordinario es que el *kernel* UML no se comunica directamente con el hardware. En su lugar, los comandos pasan al *kernel* real del sistema que lo alberga, el cual controla la comunicación con el hardware. Debido a que el sistema virtual y el sistema anfitrión son ambos Linux con estructuras prácticamente idénticas, la comunicación pasa muy eficientemente del sistema virtual al anfitrión, requiriendo muy pocas cabeceras de abstracción o traducción.

1.3.2.2.5 VMWare ESX/ESXi

Es un hipervisor que abstrae los recursos de procesador, memoria, almacenamiento y redes en varias máquinas virtuales, cada una de las cuales puede ejecutar un sistema operativo y aplicaciones sin modificaciones.

Su reducido tamaño y su fiabilidad, similar a la del hardware, también permiten integrar VMware ESXi directamente en los servidores x86 estándar del sector de importantes fabricantes de servidores, como Dell, IBM, HP y Fujitsu-Siemens.

---

[8] SPARC: *Scalable Processor Architecture*, diseñada por Sun Microsystems, es una arquitectura RISC abierta con un conjunto de instrucciones reducidas.

[9] LGPL: *Lesser General Public License*, es una licencia de software creada por la *Free Software Foundation* que pretende garantizar la libertad de compartir y modificar el software cubierto por ella, asegurando que el software es libre para todos sus usuarios.

[10] GPL: *General Public License*, es una licencia creada por la *Free Software Foundation* en 1989, y está orientada principalmente a proteger la libre distribución, modificación y uso de software.

VMware ESX y ESXi se instalan directamente en el hardware del servidor, insertando una capa de virtualización sólida entre el hardware y el sistema operativo. VMware ESX y ESXi particionan el servidor físico en varias máquinas virtuales que se pueden ejecutar de forma paralela. Cada máquina virtual representa un sistema completo con procesadores, memoria, redes, almacenamiento y BIOS[11], de tal forma que el sistema operativo y las aplicaciones de software se pueden instalar y ejecutar en la máquina virtual sin modificación alguna. La arquitectura *Bare Metal*[12] otorga a VMware ESX y ESXi un control total sobre los recursos de servidor asignados a cada máquina virtual y proporciona un rendimiento casi nativo de la máquina virtual.

1.3.2.2.6 Xen

El hipervisor Xen fue creado por Keir Fraser y Ian Pratt, como parte del proyecto de investigación Xenoserver de la Universidad de Cambridge a finales de 1990. El hipervisor Xen es una capa de software que se ejecuta directamente en el hardware del equipo sustituyendo el sistema operativo permitiendo que el hardware de la computadora pueda ejecutar múltiples sistemas operativos al mismo tiempo. Soporta arquitecturas x86, x86-64, Itanium[13], Power PC y procesadores ARM[14]. El hipervisor Xen puede ejecutarse en una amplia variedad de dispositivos informáticos y actualmente soporta Linux, NetBSD, FreeBSD, Solaris, Windows entre otros. La comunidad Xen.org desarrolla y mantiene el hipervisor Xen como una solución libre licenciada bajo GNU[15] *General Public License* [10].

---

[11] BIOS: *Basic Input-Output System*. Es un programa informático escrito directamente en la memoria *flash* de la placa base. Controla las funciones básicas de la place base y sus componentes.

[12] *Bare Metal*: Es un hipervisor también denominado nativo, *unhosted.* Es un software de virtualización que se ejecuta directamente sobre el hardware.

[13] Itanium. Fue el primer microprocesador de la arquitectura Intel Itanium. La arquitectura se basa en un explícito paralelismo a nivel de instrucción, con el compilador tomando decisiones sobre qué instrucciones deben ejecutarse en paralelo.

[14] ARM: *Advanced RISC Machine,* es una arquitectura RISC (*Reduced Instruction Set Computer*) de 32 bits desarrollada por ARM Holdings. La relativa simplicidad de los procesadores ARM los hace ideales para aplicaciones de baja potencia.

[15] GNU: acrónimo recursivo que significa "GNU No es Unix" (*GNU is Not Unix*). Fue iniciado por Richard Stallman con el objetivo de crear un sistema operativo completamente libre.

1.3.2.2.7 Microsoft Hyper-V

Hyper-V es un programa de virtualización basado en un hipervisor para los sistemas de 64-bits y posteriormente soportado en arquitectura x86.

Permite crear un entorno de servidores virtualizados para mejorar la eficacia al aprovechar mejor los recursos de hardware. Esto es posible debido a que Hyper-V crea y administra máquinas virtuales y sus recursos de tal forma que cada máquina virtual se comporte como un sistema independiente que funciona en un entorno de ejecución aislado, lo que permite ejecutar varios sistemas operativos simultáneamente en un mismo equipo físico [11].

1.4 SERVICIOS DE *CLOUD COMPUTING*

Es prácticamente un acuerdo universal distinguir dentro del *Cloud Computing* tres niveles o modelos de servicio que puede prestar un proveedor, que se diferencian entre sí fundamentalmente por el grado de visión y control al que el usuario tiene acceso.

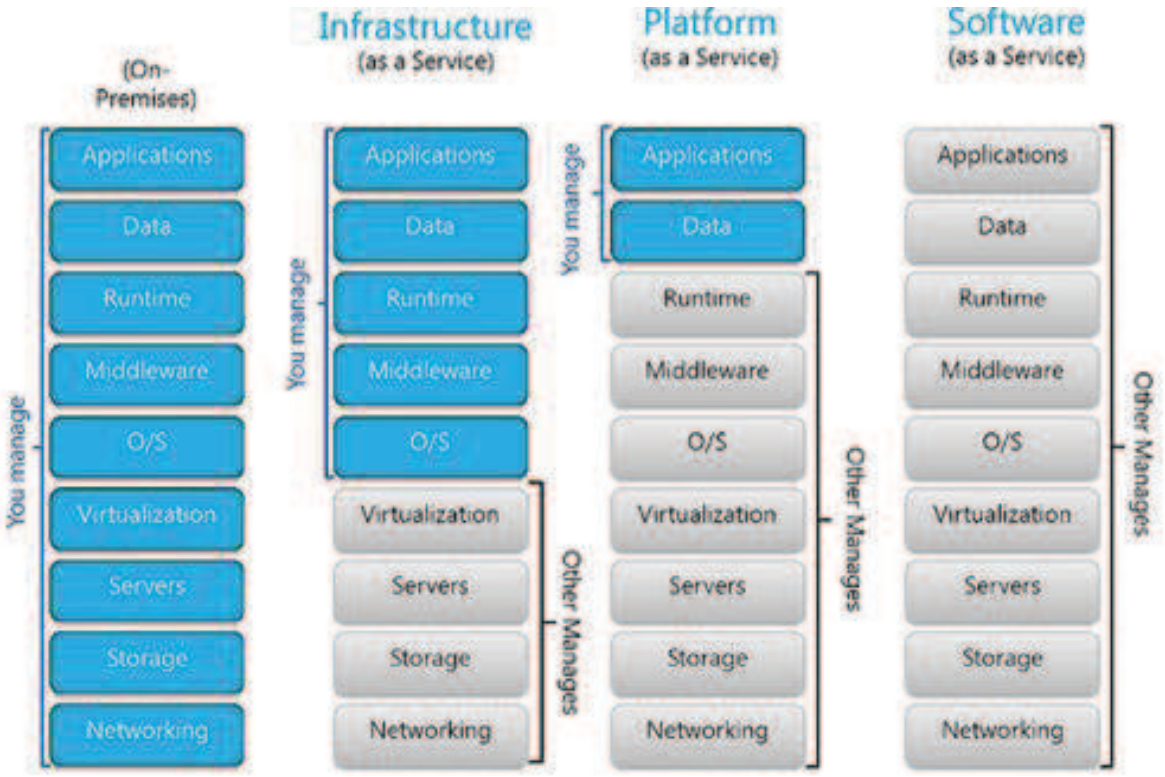


Figura 1. 6 Nivel de acceso del usuario a los recursos de cómputo según el servicio [12]

1.4.1 SOFTWARE COMO SERVICIO (SAAS)

Al usuario se le ofrece la capacidad de que las aplicaciones del proveedor se ejecuten sobre una infraestructura *cloud*. Siendo estas aplicaciones accesibles desde un cliente ligero como una interfaz web o una interfaz de programa. El consumidor no administra ni controla la infraestructura de nube subyacente incluyendo la red, servidores, sistemas operativos, almacenamiento y capacidades de la aplicación, como se observa en la Figura 1.6. Muchas de las aplicaciones del modelo SaaS están enfocadas en proveer funcionalidades a clientes empresariales a bajos precios, ofreciendo los mismos beneficios del software tradicional sin la complejidad asociada a la instalación, administración, soporte y costo inicial que esto representa. De esta manera, el cliente SaaS solo tiene la responsabilidad del uso del software contratado, y el proveedor es el responsable de mantener operativo todo el sistema *cloud* detrás de ese servicio.

1.4.1.1 Niveles de Madurez

Según Microsoft, las arquitecturas SaaS pueden ser clasificadas en una de cuatro niveles de madurez cuyos principales atributos son la facilidad de configuración, eficiencia, multiusuario y escalabilidad. Cada nivel se distingue del anterior mediante la adición de uno de estos tres atributos [13].

Los niveles de madurez descritos por Microsoft son:

1.4.1.1.1 Nivel 1, Ad-Hoc/Personalización

Cada cliente tiene una versión única y personalizada de la aplicación hospedada. La aplicación se ejecuta en su propia instancia en los servidores del huésped. La migración de una aplicación tradicional a este nivel de madurez de SaaS normalmente requiere el menor esfuerzo de desarrollo y reduce los costos operativos mediante la consolidación de hardware del servidor y la administración.

1.4.1.1.2 Nivel 2, Configurabilidad

El segundo nivel de madurez de SaaS ofrece una mayor flexibilidad a las aplicaciones a través de metadatos de configuración. En este nivel, muchos

clientes pueden utilizar una misma aplicación de distintas máquinas virtuales. Esto permite a un proveedor satisfacer las diversas necesidades de cada cliente mediante el uso de opciones de configuración detalladas. También permite que el proveedor pueda aliviar la carga de mantenimiento ya que es capaz de actualizar una base de código común.

1.4.1.1.3 Nivel 3, Eficiencia Multiusuario

Añade la capacidad multiusuario al segundo nivel. Esto resulta en un único programa que tiene la capacidad de servir a todos los clientes del proveedor. Este enfoque permite un uso más eficiente de los recursos del servidor, sin ninguna diferencia aparente para el usuario final, pero en última instancia, este nivel está limitado en su capacidad de escalar de forma masiva.

1.4.1.1.4 Nivel 4, Arquitectura escalable

En el cuarto nivel de madurez de SaaS, la escalabilidad se agrega mediante una arquitectura de varios niveles. Esta arquitectura es capaz de soportar una granja con equilibrio de carga de las máquinas virtuales de solicitud idéntica que se ejecutan en un número variable de servidores. La capacidad del sistema puede ser aumentada o disminuida de forma dinámica para satisfacer la demanda de carga mediante la adición o eliminación de servidores.

1.4.1.2 Características clave de SaaS

El despliegue de aplicaciones en una arquitectura orientada a servicios es un problema más complejo de lo que normalmente se encuentra en los modelos tradicionales de implementación de software. Como resultado, el precio de las aplicaciones SaaS se basa generalmente en el número de usuarios que pueden tener acceso al servicio. A menudo hay cargos adicionales por el uso de servicios de asistencia, ancho de banda y almacenamiento. Los flujos de ingresos de SaaS al proveedor son generalmente más bajos que los tradicionales generados por los costos de licencias de software. Sin embargo, la compensación por derechos de licencia más bajos es un flujo mensual de ingresos recurrentes, lo cual es visto por los directores financieros corporativos como un indicador más fiable de cómo el negocio está yendo de un trimestre a otro.

Las características clave son las siguientes:

- La gestión de la red está basada en el acceso al software en lugares centrales en lugar de un consumidor en un solo sitio, permitiendo a los clientes acceder a aplicaciones de forma remota a través de Internet.
- La entrega de aplicaciones se hace a partir de un modelo de uno-a-muchos, a diferencia del modelo tradicional de uno a uno.
- La actualización y mejoramiento centralizado de parches obvia la necesidad de descarga e instalación por parte del usuario.

1.4.2 PLATAFORMA COMO SERVICIO (PAAS)

Es la capa intermedia, nace a partir del modelo de distribución de las aplicaciones SaaS. Hace posible que todas las utilidades necesarias estén disponibles en la web para que el usuario pueda desarrollar y distribuir aplicaciones propias o adquiridas en la infraestructura de su proveedor, quién es el que ofrece la plataforma de desarrollo y las herramientas de programación. En este caso es el usuario el que mantiene el control de la aplicación pero no de toda la infraestructura subyacente.

El modelo PaaS hace que todas las instalaciones necesarias para apoyar el ciclo de vida completo de la construcción y la entrega de aplicaciones y servicios web estén completamente disponibles en Internet, todo sin descargas o instalación de software para desarrolladores, administradores de TI o usuarios finales.

Las empresas que prestan estos servicios ofrecen facilidades para el diseño de las aplicaciones y herramientas para el ciclo de desarrollo, pruebas, despliegue y *hosting*, así como oficinas virtuales, equipos de colaboración, integración con bases de datos, seguridad, escalabilidad, manejos de estados, paneles de control y muchas otras facilidades que una infraestructura propia difícilmente puede llegar a tener. Las organizaciones pueden redirigir una porción significativa de sus presupuestos a la creación de aplicaciones que proporcionen un valor empresarial real, en lugar de preocuparse por todos los problemas de infraestructura. El

modelo PaaS está impulsando así una nueva era de la innovación en masa. Ahora, los desarrolladores de todo el mundo pueden tener acceso a potencia de cálculo ilimitada. Cualquier persona con una conexión a Internet puede crear aplicaciones de gran alcance.

1.4.2.1 Características clave de PaaS

- Incluyen los servicios para desarrollar, probar, implementar, hospedar y administrar aplicaciones para apoyar el desarrollo de aplicaciones de ciclo de vida basadas en herramientas web.
- La interfaz de usuario suele proporcionar un cierto nivel de apoyo para simplificar la creación de interfaces de usuario.
- El apoyo a una arquitectura multiusuario ayuda a eliminar las preocupaciones de los desarrolladores sobre el uso de la aplicación por usuarios simultáneos.
- Los proveedores de PaaS suelen incluir servicios de gestión de concurrencia, escalabilidad y seguridad.
- Proveen integración con servicios web y bases de datos y capacidad de crear combinaciones de servicios web (conocidos como *mashups*).
- Capacidad para formar y compartir código, de forma predefinida o en equipos distribuidos que mejoran enormemente la productividad.
- Herramientas como un panel de instrumentos para ver el funcionamiento interno basados en mediciones tales como el rendimiento, el número de accesos simultáneos, etc.

1.4.3 INFRAESTRUCTURA COMO SERVICIO (IAAS)

Es la capa más baja, es el núcleo del servicio. El proveedor ofrece al usuario recursos computacionales de *Data Centers* consolidados, sean estos de procesamiento, almacenamiento o comunicaciones, que se pueden utilizar para el despliegue de cualquier sistema operativo y aplicaciones. El proveedor está encargado de todas las operaciones del *hosting* de los ambientes virtuales de los usuarios y de las operaciones propias del mantenimiento de la infraestructura real,

mientras que los usuarios mantienen el control absoluto y son responsables de todas las operaciones de despliegue de sus sistemas operativos y aplicaciones así como de las configuraciones.

Se considera como infraestructura a plataformas de virtualización de máquinas virtuales que incluyen el hardware de un computador, típicamente configurado en un *grid*[16] para escalabilidad horizontal masiva, acceso a redes de alta velocidad, incluyendo *routers*, *firewalls*, balanceadores de carga, conexión a Internet, y ambientes de almacenamiento virtual. Todos estos servicios son facturados bajo demanda y están sometidos a los SLA[17] que garantizan un mínimo de disponibilidad y confiabilidad para todo el servicio contratado.

La gestión que pertenece a los proveedores suelen incluir las implementaciones de las siguientes capas:

- El hardware (normalmente configurado como una red para escalabilidad masiva horizontal).
- Red de ordenadores (incluye *routers*, *firewalls*, balanceadores de carga, etc.).
- Conexión a Internet (a menudo a *backbones* de alta velocidad).
- Entorno de virtualización para ejecutar determinadas máquinas virtuales.
- Acuerdos de servicios.
- Herramienta de facturación de los recursos de cómputo.

1.5 SOFTWARE DE CREACIÓN Y GESTIÓN PARA *CLOUD*

1.5.1 CLOUDSTACK

CloudStack es un software *Open Source* que permite efectuar el despliegue, la configuración y la gestión de entornos de computación elástica. Permite construir cualquier tipo de *cloud* (privado, público e híbrido) y soporta los hipervisores Xen

[16] *Grid*. Computación distribuida o en malla, es un método para resolver problemas de computación masiva utilizando un gran número de computadores.
[17] SLA: *Service Level Agreement*, es un contrato escrito entre un proveedor de servicio y su cliente con objeto de fijar el nivel acordado para la calidad de dicho servicio.

Server y KVM. CloudStack es fácil de instalar. Existe poca documentación oficial disponible, lo que dificulta de forma considerable la realización de pruebas y la puesta a punto del entorno, porque hay distintas opciones parametrizables cuyos valores pueden ser incompatibles entre sí [14].

CloudStack destaca especialmente por su interfaz web que ofrece una gestión completa de la nube tanto para el administrador del sistema como para un usuario no privilegiado. Además, aporta gran cantidad de información, como la monitorización, las estadísticas de utilización de los recursos, la información del registro y las alertas.

CloudStack permite definir máquinas virtuales que el sistema mantendrá en funcionamiento sin intervención del administrador del sistema, efectuar la instalación de una nueva máquina virtual mediante una interfaz web empleando una imagen ISO[18], efectuar balanceo de carga entre máquinas virtuales y la posibilidad de acceder a la máquina virtual en modo gráfico por medio de la interfaz web.

Uno de los puntos más discutibles se refiere a la gestión de la infraestructura de hardware. CloudStack, en función del modo de red seleccionado, puede llegar a ser muy rígido a la hora de efectuar cambios en la infraestructura, impidiendo, por ejemplo, modificar de forma sencilla el rango de direcciones IP[19] que utilizan las máquinas virtuales.

CloudStack es un gestor muy recomendable en aquellos entornos en los que sea imprescindible efectuar la gestión de las máquinas por medio de interfaz web.

1.5.2 VCLOUD *DIRECTOR*

VMware vCLOUD *Director* permite a los clientes construir una infraestructura segura de *Cloud Computing* para nubes privadas, públicas e híbridas, disponer de

[18] ISO: Una imagen ISO es un archivo donde se almacena una copia exacta de un sistema de ficheros. Entre los usos más comunes están la distribución de sistemas operativos.
[19] IP: *Internet Protocol*, una dirección IP es una representación numérica que identifica a un dispositivo dentro de una red que utilice el protocolo IP.

los recursos de infraestructura en los *Data Centers* virtuales, y permitir que los recursos sean consumidos por los usuarios bajo demanda. vCLOUD *Director* maneja el *Data Center*, incluyendo los recursos de cómputo, almacenamiento y red, junto con las políticas pertinentes [15].

Varios niveles de servicios de máquinas virtuales se entregan totalmente encapsulados como vApps[20], utilizando OVF[21]. Mediante la creación de nubes híbridas seguras con vSphere[22] y VMware *v*Cloud *Director*, las organizaciones de TI pueden convertirse en verdaderos proveedores de servicios.

VMware vCloud *Director* aprovecha los estándares abiertos, tales como las API[23] de vCloud y de OVF para que los administradores puedan empaquetar y migrar cargas de trabajo en las nubes. Al encapsular múltiples servicios de máquinas virtuales y las políticas asociadas a redes en vApps, los usuarios finales de una nube pueden fácilmente compartir servicios con otros, y fácilmente pueden migrar servicios entre las nubes con facilidad [16].

Los usuarios finales pueden manipular los servicios como unidades flexibles y portátiles en lugar de que los servicios se encierren en un entorno de implementación específico.

vCloud dispone de una API abierta, basada en una API REST[24] que permite el acceso a los recursos de la nube, como la carga y descarga de vApps, gestión de catálogo, y otras operaciones.

---

[20] vApp. Es un contenedor para máquinas virtuales que ofrece control de recursos y la gestión de las máquinas virtuales [16].
[21] OVF: *Open Virtualization Format*, es un estándar para empaquetar y distribuir servicios virtualizados.
[22] vSphere. Es una plataforma de virtualización de la empresa VMware para la creación de infraestructuras en la nube.
[23] API: Interfaz de Programación de Aplicaciones, es el conjunto de procedimientos y funciones para la comunicación entre componentes de software.
[24] REST: Transferencia de Estado Representacional, es una técnica de arquitectura de software para sistemas distribuidos.

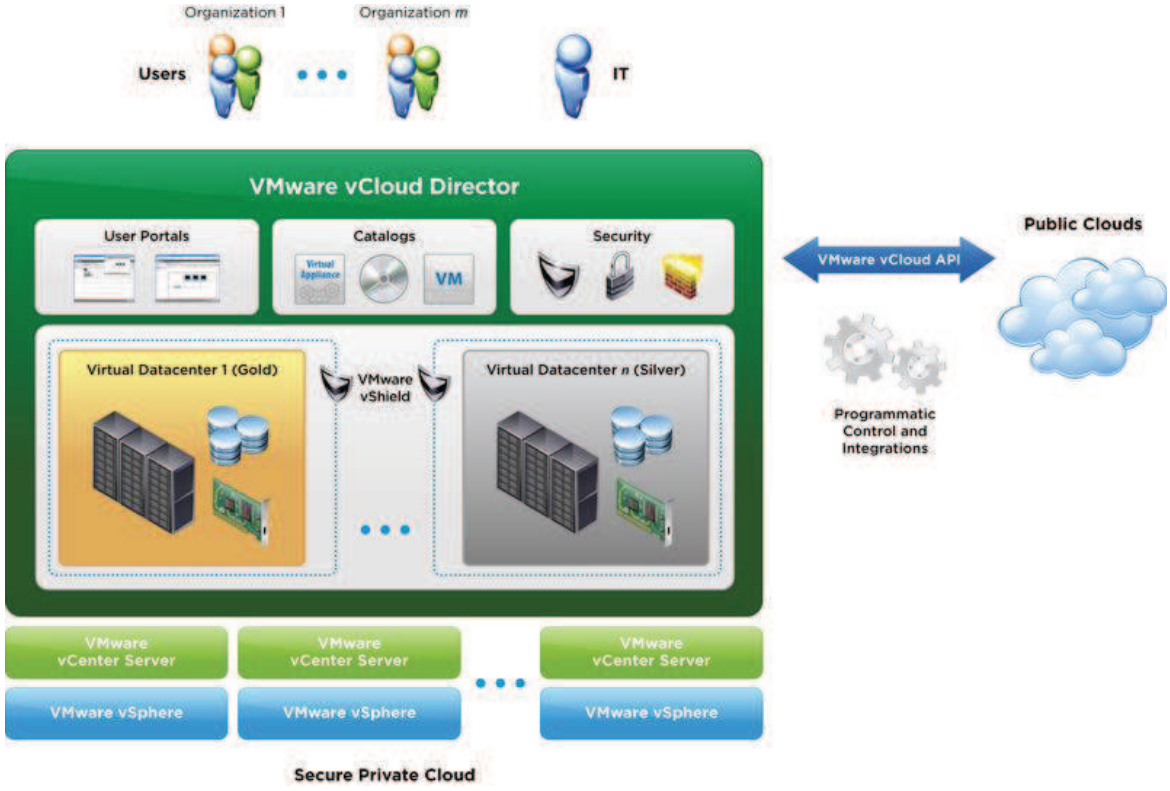


Figura 1. 7 Estructura para la provisión de servicios en la Nube según VMware [15]

Como se observa en la Figura 1.7, vCloud *Director* administra los recursos computacionales de uno o más centros de datos virtualizados por vSphere, donde el software de administración tiene la capacidad de segmentar los recursos para dar servicios específicos de procesamiento, almacenamiento y red totalmente separados entre sí por medio de la protección del módulo vShield[25]. Además es posible crear catálogos de máquinas virtuales, distintos portales de usuario y diferentes perfiles de seguridad.

1.5.3 EUCALYPTUS

Elastic Utility Computing Architecture for Linking Your Programs to Useful Systems, es software de código abierto auspiciado por Eucalyptus Systems Inc.

[25]vShield. Es un componente que añade seguridad a los *Data Center* virtuales mediante un conjunto de aplicaciones virtuales de seguridad creadas para vSphere.

para la implementación de nubes privadas o híbridas sobre *clusters*, compatible con la interfaz de Amazon EC2[26] para ofrecer Infraestructura como Servicio [17].

Permite crear un entorno propio de *Cloud Computing* con el objetivo de maximizar los recursos informáticos (*clusters*, estaciones de trabajo, máquinas de escritorio, etc.) y proporcionar un entorno de *Cloud Computing* para sus usuarios.

Funciona con la gran mayoría de distribuciones Linux, incluyendo Ubuntu, Red Hat *Enterprise* Linux (RHEL), CentOS, Suse Linux *Enterprise Server* (SLES), OpenSUSE, Debian y Fedora. De igual manera, puede usar una gran cantidad de tecnologías de virtualización, incluyendo vSphere, XEN y KVM para implementar las abstracciones que *Cloud Computing* ofrece.

Esta implementado con herramientas básicas de Linux y tecnologías de servicios web que lo hacen fácil de instalar y administrar.

1.5.4 OPENSTACK

Básicamente OpenStack es un software *Open Source* usado para la construcción de *cloud*s públicas y privadas.

Desde el punto de vista de software, OpenStack es una colección de proyectos de software libre mantenidos por la comunidad que incluyen varios componentes, siendo los más importantes:

- OpenStack *Compute*, con nombre clave **Nova**.
- OpenStack *Object Storage*, con nombre clave **Swift**.
- OpenStack *Image Service*, con nombre clave **Glance**.

A través de estos servicios, OpenStack proporciona una completa plataforma operativa para la administración y gestión de *cloud*s [18].

[26] EC2. Es un servicio web ofrecido por Amazon. Proporciona herramientas de computación en la nube de una manera flexible. Permite escalar los distintos servicios y necesidades de procesamiento de una manera fácil y ágil.

1.6 METODOLOGÍAS DE DESARROLLO DE SOFTWARE

Las metodologías de desarrollo de software son un conjunto integrado de procedimientos, técnicas, documentación y herramientas que permiten abordar de forma homogénea las actividades de un proyecto de creación de software.

Es un marco de trabajo que se usa para estructurar, planificar y controlar el proceso de desarrollo de sistemas de información. Una gran variedad de estos marcos de trabajo han evolucionado, cada uno con sus propias fortalezas y debilidades [21]. A continuación se explican los enfoques más generales que se desarrollan en varias metodologías específicas.

1.6.1 METODOLOGÍA EN CASCADA

Ordena rigurosamente las etapas del ciclo de vida del software, de manera que el inicio de cada etapa debe comenzar después de haber finalizado la etapa anterior (Figura 1.8).

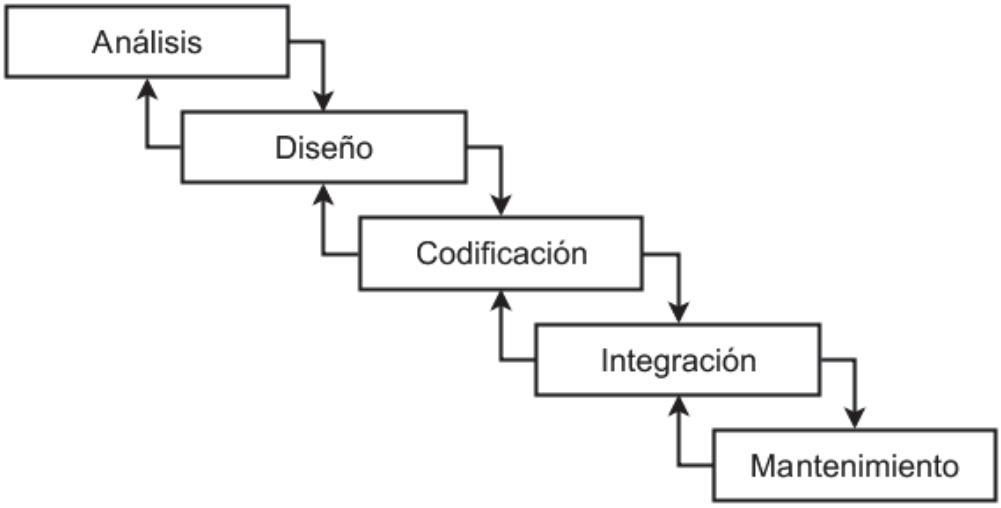


Figura 1. 8 Modelo de metodología en cascada [23]

Se cree que el modelo en cascada fue el primer modelo de proceso introducido y seguido ampliamente en la ingeniería de software. La innovación estuvo en la primera vez que la ingeniería del software fue dividida en fases claramente separadas [21].

1.6.2 METODOLOGÍA DE PROTOTIPADO

El prototipo define un conjunto objetivos generales, pero no identifica los parámetros de entrada, proceso y salida. Sin embargo, el hecho de que la versión de software sea incompleta, no implica que la aplicación desarrollada se contraponga con la función principal para la que fue diseñada [22].

El proceso comienza con la recolección de requisitos. El desarrollador y el cliente definen los objetivos globales del software. Entonces aparece un diseño rápido que lleva a la construcción del prototipo. El cliente evalúa el prototipo y se utiliza para refinar detalles con el fin de conseguir el objetivo final.

Figura 1. 9 Modelo de ciclo de vida de prototipos

Este proceso se repite hasta que satisfaga las necesidades del cliente, según el objetivo general del proyecto (Figura 1.9).

1.6.3 MÉTODOLOGÍA INCREMENTAL

Combina elementos del modelo en cascada con la filosofía interactiva de la construcción de prototipos. El proceso se divide en 4 partes: Análisis, Diseño, Código y Pruebas. Este proceso se repite con cada construcción de un nuevo prototipo (Figura 1.10). Con esto se mantiene al cliente en constante contacto con los resultados obtenidos en cada incremento.

Al igual que otros métodos de modelado, la metodología Incremental es de naturaleza interactiva pero se diferencia en que al final de cada incremento se entrega un producto completamente funcional.

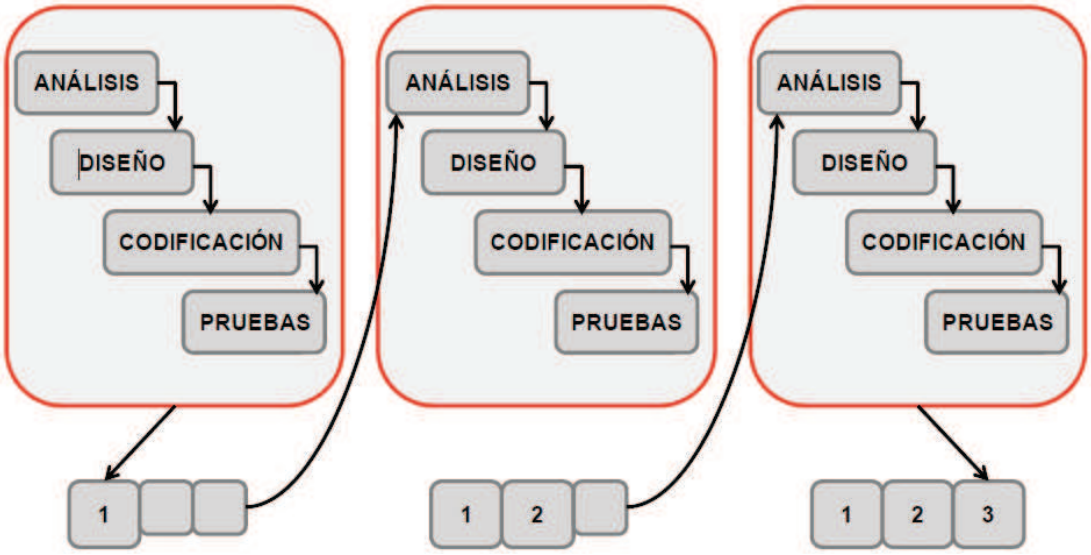


Figura 1. 10 Metodología incremental [22]

CAPÍTULO 2
IMPLEMENTACIÓN DEL PROTOTIPO DE PROVEEDOR DE SAAS

El Presente Proyecto propone la implementación de una infraestructura de *Cloud Computing* para la provisión de SaaS enfocado a solucionar las necesidades informáticas básicas de una empresa pequeña o empresas nacientes que carecen de recursos económicos y técnicos para montar su propia infraestructura de TI.

La solución propone que mediante el uso de los recursos físicos y computacionales de hardware común se pueda asignar el procesamiento, memoria y almacenamiento a las necesidades específicas de software que el cliente requiera, y mediante una interfaz web el usuario pueda acceder a sus aplicaciones de forma transparente.

El prototipo debe proveer aplicaciones listas para su uso, donde los clientes no tengan que preocuparse por el mantenimiento de toda la infraestructura necesaria para el despliegue de tal aplicación. Para ello, se usará la metodología de planificación que se muestra en la Tabla 2.1.

Fases	Descripción
Análisis	Se analizará y describirá el alcance de la solución a fin de identificar las características del servicio que debe proveer el prototipo a usuarios finales.
Diseño	Se diseñará la solución técnica para determinar hardware y la configuración de software necesaria para cumplir con los objetivos descritos en el análisis.
Construcción	Se detallará el proceso de construcción del prototipo, indicando la instalación, configuración y desarrollo de cada una de sus partes.
Pruebas	Se describirá el funcionamiento, y se realizará pruebas de comportamiento del sistema.

Tabla 2. 1 Metodología para la implementación del prototipo

El prototipo debe cumplir con las siguientes características:

- Proveer tres servicios que se consideran comunes entre las PYMEs.

- Utilizar hardware genérico y software libre para la implementación de la infraestructura.
- Proveer al usuario una única interfaz web para la administración de los servicios contratados.
- El mantenimiento y administración de la infraestructura necesaria para el despliegue de los servicios, estará a cargo del administrador del Proveedor de SaaS.

2.1 ANÁLISIS DE SERVICIOS

De acuerdo a estudios realizados sobre el nivel de uso de las tecnologías de comunicación en las PYMEs, las empresas que incorporan herramientas tecnológicas para su comunicación y manejo de relación con el cliente, incrementan notablemente su competitividad [23]. Los servicios tecnológicos que más destacan son:

- Correo Electrónico.
- Herramientas para administrar la relación con el cliente.
- Herramientas colaborativas y comunicación visual.

2.1.1 CORREO ELECTRÓNICO

El correo electrónico se ha convertido en la principal herramienta de comunicación en las empresas, tanto para el manejo de asuntos internos, como para la comunicación con los clientes [24], en comparación con herramientas de mensajería instantánea y programas de comunicación por voz [25].

Las empresas pueden optar por instalaciones en su propia infraestructura y bajo su administración, lo que conlleva el consumo de valiosos recursos tanto humanos como de dinero. Es por ello que las PYMEs confían en un proveedor externo especializado para el alojamiento y administración del servicio de correo electrónico.

En el mercado existen proveedores que ofrecen una gran variedad de características, agrupadas en varios tipos de planes con pago mensual o anual. Pero se observó que para proveer el servicio se debe ofrecer al menos las características que se describen en la Tabla 2.2.

Características	Descripción
Administración de cuentas	Interfaz de administración para la creación, cambio de contraseña y eliminación de las cuentas de correo.
Almacenamiento	Interfaz para visualizar el almacenamiento contratado
Administración de dominio	Interfaz para crear o eliminar un dominio de correo
Webmail	Interfaz web para el manejo del correo electrónico

Tabla 2. 2 Características básicas del servicio de correo electrónico

2.1.2 HERRAMIENTA PARA LA ADMINISTRACIÓN DE LA RELACIÓN CON EL CLIENTE

Para cualquier empresa, independientemente de su tamaño, necesita de un adecuado manejo de su fuerza de ventas, mejorar el nivel de servicio y soporte a sus clientes. Esto se consigue adecuando una herramienta informática a los procesos internos para administrar la información de los clientes y llevar un control sobre el ciclo de ventas.

La adopción del CRM (*Customer Relationship Management*) entre las empresas de Latinoamérica ha crecido paulatinamente, pero pocas lo han implementado con éxito en sus procesos. Esto, más la oferta de herramientas gratuitas de código abierto, han incentivado a las PYMEs en adoptar una herramienta CRM [26]. Las características descritas en la Tabla 2.3 describen a una herramienta CRM que se acopla fácilmente a las necesidades básicas de cualquier empresa.

Características	Descripción
Gestión de agenda	Creación de citas y generación de recordatorios
Gestión de prospectos	Seguimiento de clientes potenciales. Manejo de preventa
Gestión de cartera de clientes	Seguimiento de clientes y gestión de oportunidades
Facilidad de acceso	Interfaz web y/o aplicación móvil
Software Libre	Herramienta de código abierto con soporte comunitario y empresarial

Tabla 2. 3 Características principales de herramienta CRM

2.1.3 HERRAMIENTA COLABORATIVA

Se denomina herramientas de colaboración al conjunto de utilidades y servicios informáticos que permiten a distintos usuarios interactuar entre ellos, desempeñando un trabajo y compartiendo información con fluidez [27].

Este tipo de herramientas dinamiza la colaboración dentro y fuera de la empresa, ya que permite la compartición de un entorno de trabajo para editar información de forma simultánea, desde cualquier lugar, con la inmediatez que da una conexión a internet. Las características básicas de un software de colaboración, útil para las necesidades de una empresa, se encuentran descritas en la Tabla 2.4.

Características	Descripción
Video Conferencia	Creación de video conferencias con uno o más participantes
Compartición de escritorio	Visualización de escritorio de los participantes
Gestión de usuarios	Creación de usuarios, grupos, eventos, salas
Integración de audio y video	Integración de micrófono y camara web
Importación de documentos	Capacidad de compartir archivos de distintos formatos

Tabla 2. 4 Características básicas de una herramienta colaborativa

2.2 ANÁLISIS DE SOFTWARE IAAS

La infraestructura que soporta el servicio de software en la nube, se basa en la implementación de un sistema de creación y administración de IaaS. En la sección 1.5 se describen varias herramientas para la creación y gestión de Cloud; de ellas, se tomará CloudStack, Eucalyptus y OpenStack (plataformas *Open Source*) para realizar un análisis comparativo y determinar la más idónea para la implementación del Presente Proyecto.

En la Tabla 2.5 se detalla los puntos de comparación entre las diferentes plataformas.

En la Tabla 2.6 se muestra la comparación entre las plataformas OpenStack, Eucalyptus y CloudStack, de acuerdo a los aspectos mencionados en la Tabla 2.5 [28], [29], [17], [30], [14].

Característica	Detalle
Características de la Plataforma Open Source	
Lenguaje	Lenguajes de programación en que esta escrita la herramienta
Lanzamiento	Fecha en que el proyecto es formalmente presentado
Licencia	Licencia Open Source para su distribución o comercialización
Lineas de código	Número de líneas de código de producción y prueba
Desarrolladores	Número de contribuidores activos
Discusiones	Número de hilos mensuales en los foros oficiales
Mensajes	Número de mensajes mensuales en los foros
Participantes	Número de participantes activos mensuales en los foros
Funcionalidad	
Hypervisor	Soporte de tecnologías de virtualización
Monitoreo	Capacidad de monitorización de recursos
Facturación	Facturación por el uso de los recursos
Auntenticación	Autenticación de servicios y usuarios
Cuota	Definición de límites en el consumo de recursos
Seguridad	Control de acceso por políticas de seguridad
Migración	Capacidad de mover máquinas virtuales de un nodo a otro sin afectar su disponibilidad

Tabla 2. 5 Aspectos comparativos para plataformas de IaaS *Open Source*

	OpenStack	CloudStack	Eucalyptus
Lenguaje	Python, Shell scripts	Java, Python, Shell scripts	Java, C++, Python, Perl, Shell scripts
Lanzamiento	Julio, 2010	Agosto, 2008	Mayo, 2007
Licencia	Apache v2.0	Apache v2.0	GPL
Lineas de código	1300000	1200000	217000
Desarrolladores	1278	96	
Discusiones	700	580	115
Mensajes	2250	2300	450
Participantes	420	225	75
Hypervisor	KVM, XEN, Hyper-V,Vmware,	KVM, XEN, OVM, Hyper-V, Vmware	KVM, Vmware
Monitoreo	no	no	no
Facturación	no	si	no
Auntenticación	si	si	si
Cuota	Por proyecto	Por recurso	Por recurso
Seguridad	si	si	no
Migración	si	si	no

Tabla 2. 6 Comparación de plataformas de IaaS *Open Source*

Las tres plataformas ofrecen funcionalidades similares, cada una de ellas con mayor o menor experiencia, pero al ser soluciones *Open Source*, el factor más importante a tomar en cuenta es la madures del proyecto; es decir, el tiempo en que se está desarrollando, el tamaño de la comunidad de desarrolladores y la actividad que tienen sus foros para el intercambio de información. En este sentido, se observa que Eucalyptus es la plataforma más antigua, pero entre las tres, es la plataforma que menos actividad presenta en su desarrollo (según las discusiones, mensajes y participantes). Por otro lado, OpenStack presenta un crecimiento acelerado en la cantidad de desarrolladores que colaboran activamente en el proyecto, siendo actualmente la plataforma con la comunidad más activa.

Este factor ha sido determinante para elegir a OpenStack como la plataforma de desarrollo de Infraestructura como Servicio para el prototipo de Proveedor de SaaS que propone implementar el Presente Proyecto.

A continuación se explica en detalle las características y funcionalidades del software para la implementación de IaaS OpenStack.

2.2.1 OPENSTACK

La misión principal del proyecto es proporcionar un software que cubra el ciclo completo de despliegue de una plataforma de *Cloud Computing* que proporcione el poder desplegar de forma sencilla, escalable, elástica y que satisfaga las necesidades de proveedores de nubes privadas y públicas [19].

El proyecto OpenStack en su conjunto está diseñado como un sistema operativo de nube masivamente escalable. Para ello, cada uno de los servicios ha sido diseñado para trabajar en conjunto con los otros servicios. Esta integración se facilita a través de las API públicas que cada servicio ofrece. Si bien estas API son usadas por los demás servicios, también están disponibles para los usuarios finales de la nube permitiendo un mayor nivel de integración con soluciones externas [20].

2.2.1.1 Componentes de OpenStack

OpenStack en la versión Folsom tiene siete componentes fundamentales para el proyecto, los cuales son: *Compute, Object Storage, Identity, Dashboard, Block Storage, Image Service* y *Network*. A continuación se presenta un breve resumen de cada uno de estos elementos [31]:

- *Object Storage* (nombre código **Swift**) permite almacenar o recuperar archivos (pero no montar directorios como un servidor de ficheros). Varias compañías ofrecen servicios comerciales de almacenamiento basados en Swift. Estas incluyen KT, Rackspace (en la que se originó Swift) e Internap. Swift también se utiliza internamente en muchas grandes empresas para almacenar sus datos.

- *Image Service* (nombre código **Glance**) ofrece un catálogo y repositorio de imágenes de disco virtuales. Estas imágenes de disco son en su mayoría de uso común en OpenStack *Compute*. Aunque este servicio es opcional, técnicamente cualquier nube lo requerirá.

- *Compute* (nombre código **Nova**) ofrece servidores virtuales bajo demanda. Rackspace y HP proporcionan servicios informáticos comerciales construidos con Nova y también se utiliza internamente en empresas como Mercado Libre y la NASA (donde se originó).

- *Dashboard* (nombre código **Horizon**) es una interfaz web útil tanto para el administrador de la nube como para el usuario final, y consiste en un sistema modular para el acceso a todos los servicios de OpenStack. Con esta interfaz gráfica se puede realizar la mayoría de operaciones en la nube, como el lanzamiento de una instancia, la asignación de direcciones IP y el establecimiento de controles de acceso.

- *Identity* (nombre código **Keystone**) suministra autenticación y autorización de todos los servicios. También proporciona un catálogo de servicios dentro de la nube.

- *Network* (nombre código **Quantum**) ofrece conectividad de red como un servicio. Permite a los usuarios crear sus propias redes y luego conectar las interfaces entre sus máquinas virtuales. Quantum provee una API que permite la conexión de soluciones de distintas tecnologías así como de varios fabricantes de infraestructura de red.

- *Block Storage* (nombre código **Cinder**) proporciona almacenamiento persistente de bloques a las máquinas virtuales. Este proyecto nació a partir del código original de Nova (nova-volumen). Cabe recalcar que este es almacenamiento de bloques (o volúmenes) a diferencia de sistemas de almacenamiento como NFS[27] o SMB[28].

2.2.1.2 Arquitectura Conceptual

La Figura 2.1 muestra la perspectiva del operador de la nube como una vista simplificada de la arquitectura de todos los servicios (suponiendo que se estén usando juntos, que es la configuración más común).

- Horizon provee una interfaz web de los otros servicios OpenStack.
- Nova usa y coordina los recursos de cómputo, almacenamiento y provisión de máquinas virtuales y asocia metadatos a las imágenes creadas por Glance.
- Quantum provee conectividad de red virtual a Nova.
- Cinder provee almacenamiento de volúmenes para Nova.
- Glance permite almacenar discos virtuales como objetos en Swift.
- Todos los servicios y usuarios se autentican mediante Keystone para hacer uso de los recursos de la infraestructura.

[27] NFS: *Network File System*, es un protocolo utilizado como un sistema de archivos distribuido en un entorno de red de área local.
[28] SMB: *Server Message Block*, es un protocolo que permite la compartición de archivos, impresoras y más recursos entre nodos de una red.

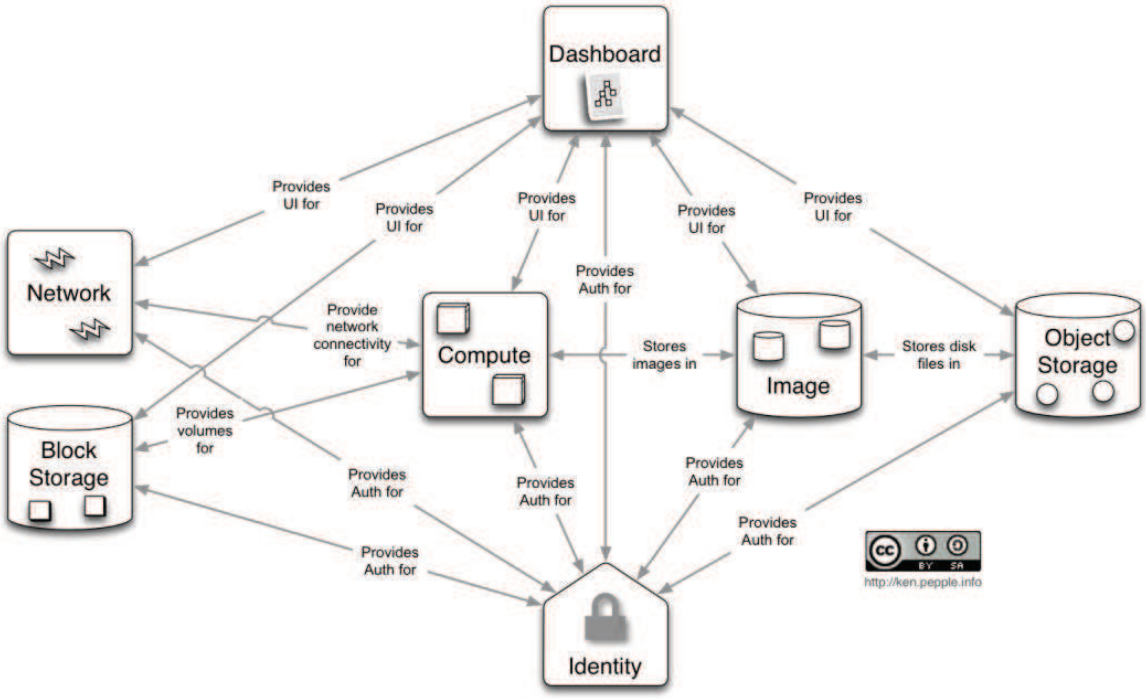


Figura 2. 1 Arquitectura simplificada de los componentes de OpenStack [31]

2.2.1.3 Arquitectura Lógica

La arquitectura lógica es mucho más compleja que la arquitectura conceptual. Tratando de ilustrar todas las combinaciones posibles de las comunicaciones entre los servicios, la Figura 2.2 describe la arquitectura más común de una nube OpenStack. Sin embargo, como OpenStack soporta una amplia variedad de tecnologías, ésta no representa la única arquitectura posible.

A continuación se explica con mayor detalle la función de cada componente de la arquitectura lógica que se presenta dentro de OpenStack.

2.2.1.3.1 Servicio de cómputo Nova

Es la parte principal de IaaS de OpenStack que provee servidores virtuales bajo demanda. No incluye ningún software de virtualización pero define los *drivers*[29] que interactúan con los mecanismos de virtualización que corren sobre el *host* de cómputo y exponen su funcionalidad a través de una API.

[29] *Driver* es un programa informático que permite al sistema operativo interactuar con un periférico, haciendo una abstracción del hardware y proporcionando una interfaz de administración.

Server (nova-api): Provee una interfaz al exterior para la interacción con la infraestructura *cloud*. Se comunica con los diferentes componentes a través de un servicio de encolamiento de mensajes (RabbitMQ *server*).

Message Queue (RabbitMQ Server): El *cloud controller* se comunica con los otros componentes como *Scheduler*, *Volume Controller*, *Network Controller* por AMQP (*Advance Message Queue Protocol*). Nova usa comunicación asincrónica para la respuesta a las solicitudes. Gracias al encolamiento de las peticiones, ningún requerimiento de los usuarios se queda sin responder por largo tiempo.

Compute Worker (nova-compute): Un *compute worker* hace frente al ciclo de vida de una instancia de un sistema operativo (máquina virtual). Hay varios en un sistema de *Cloud Computing* típico, donde una instancia se despliega en uno de los *compute workers* disponibles basándose en un algoritmo de asignación de recursos.

Network Controller (nova-*network*): Controla la configuración de red de los *hosts*. Estas operaciones incluyen asociar direcciones IP a nuevas máquinas virtuales, configurar las VLAN para los proyectos, implementar grupos de seguridad y configurar redes para los nodos de cómputo. Todas estas funciones se migraron a Quantum como un servicio totalmente separado de Nova, pero se incluye en la versión Folsom como una segunda alternativa (más limitada y menos flexible) para la provisión de conectividad de red en un ambiente OpenStack.

Volume Workers (nova-*volume*): Es usado para manejar los volúmenes LVM[30] para las máquinas virtuales. Este módulo optimiza las funciones de creación, asignación y eliminación de volúmenes para las máquinas virtuales. Provee una manera de entregar almacenamiento persistente para uso de las máquinas virtuales, como la asignación del disco principal a una instancia. Cuando el volumen se separa o cuando se termina la instancia, la información se mantiene y se retienen los datos almacenados en ella incluso cuando esta ha sido asignada a otra instancia.

[30] LVM: *Logical Volume Manager*, es una implementación de un administrador de volúmenes lógicos para el *kernel* de Linux.

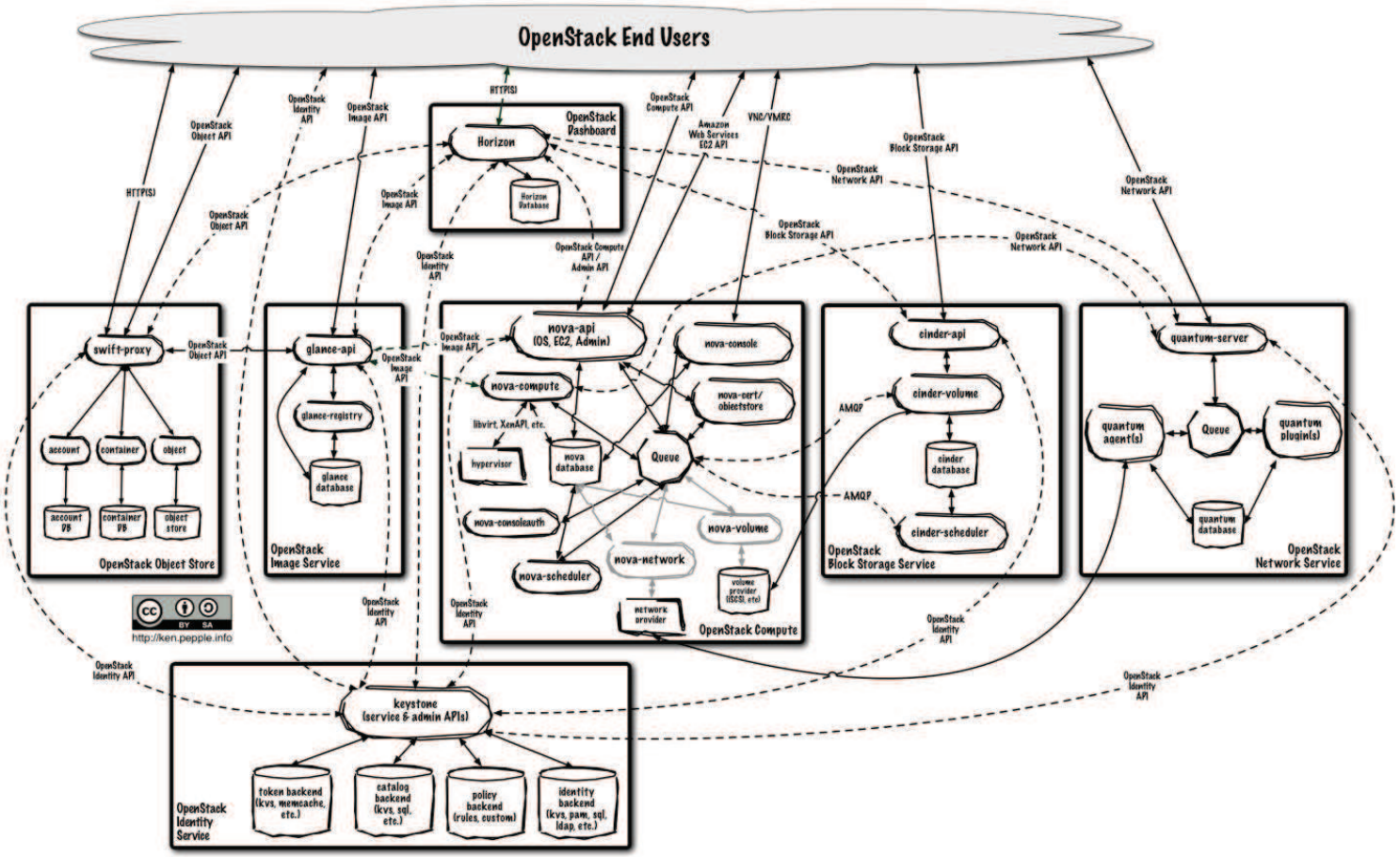


Figura 2. 2 Arquitectura Lógica de OpenStack [34]

Scheduler (nova-*scheduler*): El planificador mapea la llamadas de nova-API a los componentes correspondientes. Se ejecuta como un demonio[31] llamado "*nova-scheduler*" y recoge las solicitudes de *compute*, *network*, *volume* y los asigna del *pool* de recursos disponibles en función de un algoritmo de planificación. El planificador puede basar sus decisiones basándose en varios parámetros como carga, memoria, distancia física, arquitectura del CPU, etc.

Consola (nova-*console*): Nova provee la ejecución de los demonios *nova-console*, *nova-vncproxy*, *nova-consoleauth* para proveer el acceso a la consola de las máquinas virtuales por medio de la interfaz web Horizon.

2.2.1.3.2 Almacenamiento de Objetos Swift

Provee el almacenamiento de objetos, es decir, permite almacenar y recuperar archivos. Es un módulo con capacidad de ser altamente distribuido, alta disponibilidad que provee consistencia en el almacenamiento de objetos. Incluye los siguientes componentes:

Swift *Proxy Server*: Los usuarios interactúan con la configuración de Swift a través de un *Proxy Server* usando la API de Swift. Recibe y selecciona peticiones, encuentra el lugar donde se encuentra alojada las respectivas entidades y enruta las peticiones hacia ellas. También maneja fallos de las entidades, re-enrutando las peticiones hacia entidades de *backup*.

Swift *Object Server*: Es el responsable de manejar el almacenamiento, recuperar y eliminar los objetos alojados en el *storage* local. Los objetos son típicamente archivos binarios almacenados en el sistema de almacenamiento que contienen metadatos como un archivo de atributos extendidos.

Swift *Container Server*: Lista los objetos en un contenedor junto con sus metadatos asociados, en archivos SQL. También genera estadísticas como el número de objetos contenidos y el tamaño ocupado en el contenedor.

---

[31] Demonio. Se denomina a un proceso que se ejecuta en segundo plano, es decir que no interactúa con el usuario.

Swift *Account Server*: Lista los contenedores de la misma manera como el contenedor lista los objetos.

The Ring: Contiene información acerca de la locación física de los objetos almacenados dentro de Swift. Es una representación virtual del mapeo de nombres de las entidades con su ubicación física real. Esto es análogo a un servicio de indexación que varios procesos usan para buscar y localizar la posición real de las entidades en un clúster. Entidades como *Accounts*, *Containers*, *Objects* tienen sus anillos (*rings*) separados.

2.2.1.3.3 Servicio de provisión de imágenes Glance

Provee los servicios de descubrimiento, registro y recuperación de imágenes de máquinas virtuales. Es un repositorio y catálogo de imágenes de disco virtual que son comúnmente usados por *Compute*.

Puede ser configurado para usar uno de los tres tipos de almacenamiento disponibles:

- OpenStack *Object Store.*
- *Storage* S3[32] directamente.
- *Storage* S3 con OpenStack *Object Storage* como intermediario de acceso a S3.

Hay varios procesos que ejecutan Glance para asegurar la consistencia y la disponibilidad a través de él o de los *clusters* de almacenamiento.

Glance-Api: Es un servicio que permite almacenar y recuperar imágenes de máquinas virtuales de múltiples repositorios. Hasta la versión Diablo los repositorios pueden ser, un almacén propio de Amazon S3, un almacén de Swift, un sistema simple de ficheros, o un almacén HTTP[33].

---

[32] Amazon S3 es un servicio web de almacenamiento ofrecido por Amazon.

[33] HTTP: *HyperText Transfer Protocol*. Es el protocolo usado por cada transacción en la *World Wide Web*. Sigue el esquema de petición-respuesta entre los elementos de software de la arquitectura web (clientes, servidores).

Glance–*Registry*: Es un servicio que publica los metadatos para consultas internas por parte de *glance-server*. Usa una base de datos SQL para almacenar los metadatos.

2.2.1.3.4 Servicio de administración web Horizon

En una aplicación web modular hecha con Django[34] que provee un entorno amigable de administración de los recursos de la nube.

Al igual que con la mayoría de aplicaciones web, su arquitectura es bastante simple.

Horizon se instala normalmente a través del servidor web Apache. El código en sí se separa en un módulo reutilizable con la mayor parte de la lógica (interacciones con diversas API de OpenStack) y otro módulo de presentación (para que sea fácilmente adaptable para diferentes sitios).

Ya que se basa principalmente en la presentación de los recursos de otros servicios, almacena muy pocos datos en su propia base de datos.

Desde el punto de vista de la arquitectura de la red, este servicio tiene que ser accesible al cliente final, así como ser capaz de hablar con las API de cada servicio, para esto se necesita conectividad a las API con un rol de administrador.

2.2.1.3.5 Servicio de Autenticación Keystone

Provee identificación, autenticación, autorización y un catálogo de políticas para todos los servicios de OpenStack. Además valida las credenciales de autenticación de datos de los proyectos, los usuarios y sus roles dentro de la infraestructura. En un caso básico todos los datos son manejados por el mismo servicio, permitiendo el uso de este cuando las credenciales sean asociadas a los metadatos.

[34] Django es un *framework* de desarrollo web de código abierto escrito en Python. pone énfasis en el re-uso, la conectividad y extensibilidad de componentes, el desarrollo rápido y facilitar la creación de sitios web complejos.

2.2.1.3.6 Servicio de Red Quantum

Quantum ofrece conectividad de red como un servicio entre las interfaces de los dispositivos que gestionan los servicios de OpenStack. Quantum también permite a los usuarios crear sus propias redes y conectar las interfaces de red de las máquinas virtuales según su necesidad, posibilitando la creación de redes complejas y altamente escalables.

Al igual que muchos de los servicios de OpenStack, Quantum es altamente configurable debido a su arquitectura tipo *plugin*. Estos *plugins* dan cabida a equipos de redes y software de diferentes tecnologías y diferentes fabricantes.

Quantum hace uso de una cola de mensajería de información de las rutas entre el servidor y los diversos agentes, así como una base de datos para almacenar el estado de redes para los *plugins*.

Quantum interactúa principalmente con Nova, para ofrecer redes y conectividad de las máquinas virtuales entre ellas y hacia el exterior. El servidor Quantum acepta peticiones en su API y luego los envía al *plugin* adecuado para la acción. Los *plugins* compatibles con Quantum se encargan de realizar las acciones de conexión y asociación de puertos, creación de redes o subredes y proveer el direccionamiento IP.

El *plugin* Open vSwitch trabaja como parte del servicio de red Quantum, pero también tiene la capacidad de implementar diferentes *plugins* y agentes como: *switches* Cisco virtuales y físicos, Nicira NVP [32], los productos de NEC OpenFlow [33], [34], puentes Linux y el sistema operativo de Red Ryu [35].

El *plugin* Quantum OpenvSwitch consta de dos componentes [36]:

- Un *plugin* que procesa todas las llamadas a la API y almacena la lógica del modelo de red en la base de datos de Quantum.
- Un agente que se ejecuta en cada uno de los nodos que implementa el servicio *nova-compute*. Este agente recoge la configuración y las asignaciones directamente de la base de datos y se comunica con la

instancia de Open vSwitch en el nodo para configurar los flujos e implementar el modelo de datos lógico.

2.2.1.3.7 Almacenamiento de Bloques Cinder

Cinder separa la funcionalidad de bloques de almacenamiento persistente que antes era parte de *Compute* en su propio servicio. La API de almacenamiento de bloques permite la manipulación de volúmenes, tipos de volúmenes y los *snapshots*[35] de un volumen.

Cinder puede interactuar con una variedad de proveedores de almacenamiento a través de una arquitectura de controlador. En la actualidad, hay controladores para IBM, SolidFire, NetApp, Nexenta, Zadara, iSCSI Linux y otros proveedores de almacenamiento.

Al igual que *nova-scheduler*, el demonio *cinder-scheduler* escoge el almacenamiento de bloques óptimo en el nodo proveedor para la creación del volumen.

2.3 ANÁLISIS DE INTERFAZ WEB

Con el objetivo de ofrecer servicios en la modalidad SaaS, es necesario proveer una interfaz web amigable y de fácil entendimiento para el cliente, donde pueda solicitar nuevos servicios y manipular los aspectos básicos de éstos, como el aumento o disminución del almacenamiento, la suspensión o activación de los servicios. Esta página web deberá ser el único medio para que un cliente pueda administrar los servicios contratados.

Una herramienta que facilite la gestión del sistema será imprescindible para el Presente Proyecto, ya que al tratarse de Software como Servicio, en todos los puntos de la aplicación deberá existir la facilidad de administración y mantenimiento para el proveedor del sistema. Este nivel de modularidad se consigue con el patrón de arquitectura de Software “MVC” (Modelo, Vista, Controlador), el cual separa la lógica de negocio, los datos y la interfaz de usuario

[35] *Snapshot.* Es el estado de un sistema en un punto particular de tiempo. Se refiere a la copia del estado actual de un Sistema.

en tres componentes distintos, lo que facilitará la modificación de cada una de sus partes.

La herramienta deberá cumplir con las características descritas en la Tabla 2.7.

Características	Descripción
Arquitectura MVC	Patrón de desarrollo Modelo, Vista, Controlador
Interfaz de Administración	Interfaz que facilite la administración del modelo de
Framework de desarrollo web	Software que facilite el desarrollo web
Lenguaje Python	Desarrollo en lenguaje de programación Python

Tabla 2. 7 Características de la herramienta de desarrollo web

Entre los *frameworks* que cumplen con estas características están:

- Pylons: *Framework* web que enfatiza la flexibilidad y el desarrollo rápido.
- Django: *Framework* python que promueve el desarrollo rápido y el diseño limpio.
- TurboGears: Construido sobre Pylons. Es un conjunto de herramientas para construir aplicaciones web dinámicas.

Django es el *framework* por defecto en el desarrollo de la interfaz web de OpenStack (Horizon), además, tiene la comunidad de usuarios más grande en Español; razones por las que se ha decidido escoger a Django para el desarrollo del entorno web del Prototipo.

2.3.1 DJANGO

Django es un *framework* que sigue en parte las directrices de una arquitectura MVC[36], pero con algunas variaciones con la finalidad de hacerlo lo más funcional posible. Django denomina “MPV” (Modelo, Plantilla, Vista) a su arquitectura de desarrollo de páginas web. El Modelo se encarga del manejo de los datos, la Plantilla sirve para la presentación de la información y la Vista se encarga del control y funcionalidad de la aplicación web [37].

[36] MVC (Modelo, Vista, Controlador): Es una arquitectura de software que separa el manejo de los datos, la lógica de negocio y la interfaz de interacción con el usuario, como módulos independientes para facilitar el desarrollo de aplicaciones.

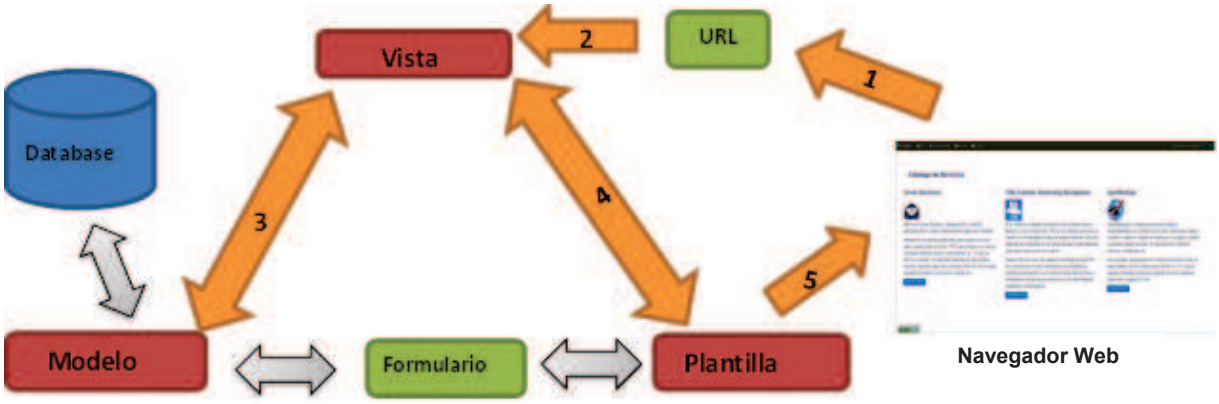


Figura 2. 3 Funcionamiento de Django

En la Figura 2.3 se muestra un diagrama básico del proceso de interacción de los diferentes módulos de Django para el funcionamiento de una página web. En el diagrama se presenta las siguientes interacciones:

1. Un usuario envía una solicitud a través del navegador web.
2. El módulo llamado *"URL"* interpreta la solicitud del usuario y la reenvía a la Vista correspondiente para que la procese.
3. La Vista interactúa con el Modelo, el cual hace directamente las consultas necesarias a la base de datos.
4. La Vista entrega la información procesada a la Plantilla, la cual generará el código HTML que presentará en el navegador web.
5. La Plantilla puede obtener datos directamente de los Modelos por medio de los Formularios, sin necesidad de la intervención de la Vista.

A continuación se definen los conceptos relevantes que intervienen en el desarrollo de una página web con el *framework* Django.

2.3.1.1 Modelos

Django define a un modelo como la fuente única para acceder a una tabla de una base de datos. Un modelo tiene nombre, campos y sus respectivos atributos, estos se verán reflejados en la base de datos.

Para que los modelos se repliquen en la base de datos es necesario definir un servidor de base de datos y crear ahí la base.

2.3.1.2 Formularios

Django define a un formulario como la colección de campos que son validados según un determinado tipo de dato. Los formularios se muestran como elementos HTML en una página web [38].

Un campo es una clase que es responsable de validar la información ingresada por el usuario. Por ejemplo, un campo de tipo "*EmailField*" valida que el texto ingresado por el usuario tenga el formato correcto de una dirección de correo electrónico.

2.3.1.3 URL

Django tiene una particular forma de mostrar las URL, ya que uno de sus principios es que las URL sean agradables y entendibles para el usuario. Es por esto que existe un archivo llamado *"urls.py"* que se encarga de redirigir las solicitudes del usuario hasta la vista apropiada y pasarle las variables que necesite para completar su trabajo.

2.3.1.4 Vistas

Una Vista es una función de Python que toma una petición web y devuelve una respuesta. Esta respuesta puede ser código HTML, o una redirección a otra página web, un error, una imagen, etc. Cabe señalar que las vistas solo se encargan del tratamiento de los datos, más no de la presentación de estos, esa tarea es de la Plantilla [39].

2.3.1.5 Plantillas

Django básicamente define como plantilla a un documento HTML con algunas funcionalidades extras. La Plantilla recibe los datos de la Vista y luego los organiza para su presentación en el navegador web. Las etiquetas que Django usa para las plantillas permiten que sea flexible para los diseñadores web al momento de añadir elementos como código JavaScript[37], maquetación con

[37] JavaScript: es un lenguaje de programación orientado a objetos para permitir mejoras en la creación de páginas web dinámicas.

CSS[38], código XML[39], etc. Las etiquetas permiten además, que la lógica del sistema siga permaneciendo en la Vista.

2.4 DISEÑO DEL PROTOTIPO

De acuerdo al análisis de los requerimientos que debe cumplir el prototipo de proveedor de SaaS, a continuación se presenta el diseño de la solución técnica para determinar el hardware y la configuración del software de los componentes que forman parte del prototipo.

2.4.1 NODOS

Es necesaria la instalación de tres tipos de nodos que realizarán los roles de controlador, cómputo y red. Se puede agregar tantos nodos de cómputo como se necesite, repitiendo los pasos de la sección de instalación del nodo *Compute*.

2.4.1.1 Nodo de Control

Se lo identifica dentro de la implementación con el nombre de "*Controller*". Será el encargado de dar la funcionalidad necesaria para administrar la infraestructura de OpenStack, como gestionar todos los recursos del *cloud*, interactuar con los clientes y ordenar a los nodos de virtualización que ejecuten las máquinas virtuales, pero en él no se ejecutarán máquinas virtuales.

La mayor parte de componentes del *cloud* y configuración se realizará en este equipo, pero comparado con los nodos de cómputo la carga de trabajo será pequeña, por lo que no es necesario un equipo de grandes prestaciones en este prototipo.

En el nodo *Controller* se instalarán los siguientes componentes: *nova-API*, *nova-novncproxy*, *nova-scheduler*, el servidor de Quantum, el *plugin* de Open vSwitch, *glance-API*, *glance-registry*, Keystone y Horizon.

---

[38] CSS: *Cascading Style Sheets*, hace referencia a un lenguaje de estilos usados para la presentación del aspecto y formato de un documento escrito con lenguaje de marcas.

[39] XML: *eXtensible Markup Language*s, es un lenguaje de marcas desarrollado por la *Word Wide Web Consorsium* que define la gramática para estructurar documentos grandes, siendo útil para la comunicación entre aplicaciones e intercambiar información.

2.4.1.2 Nodo de Cómputo

Se lo identifica dentro de la implementación con el nombre de *Compute*. Se instalará el hipervisor donde se ejecutan las máquinas virtuales y estará esperando las órdenes del nodo *Controller*. En cada nodo de cómputo se instalará los siguientes componentes: Nova *Compute*, y el *plugin* de Open vSwitch con su respectivo agente.

2.4.1.3 Nodo de Red

Se lo identifica dentro de la implementación con el nombre de *Network*. Provee los servicios de red como DHCP[40], acceso a la red, y enrutamiento de las máquinas virtuales generadas por Nova en el nodo de cómputo.

En el nodo *Network* se instalará los siguientes componentes: Quantum DHCP, el agente de capa 3 de Quantum, el *plugin* de Open vSwitch con su respectivo agente y el demonio Dnsmasq[41].

2.4.2 ESQUEMA DE RED

Cada uno de los nodos está conectado a tres redes (salvo *Compute*), lo que en terminología de OpenStack se conocen como una red pública, privada y de administración. Los términos *público y privado* no tienen el sentido habitual que tienen en redes IPv4, ya que en OpenStack la red pública se refiere a la que interacciona con los clientes y la privada que se utiliza para la intercomunicación de los componentes del *cloud*, especialmente para la transferencia de imágenes a los nodos en los que se ejecutan las máquinas virtuales.

---

[40] DHCP: *Dynamic Host Configuration Protocol,* es un protocolo que permite asignar una dirección IP automáticamente a un dispositivo que se conecta a una red de datos.

[41] Dnsmasq: Es un servidor ligero para proveer servicios de DNS (*Domain Name Server*), DHCP y TFTP (*Trivial File Transfer Protocol*) a una red pequeña.

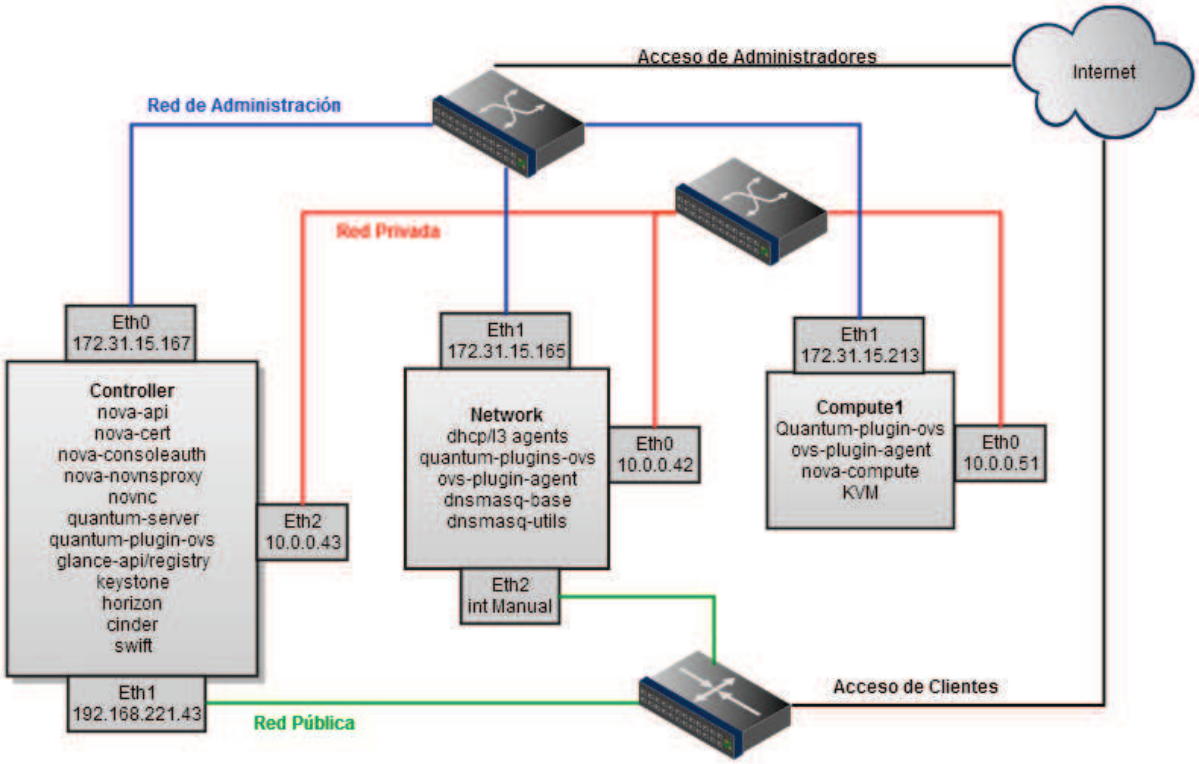


Figura 2. 4 Esquema de Red del Prototipo de Proveedor de SaaS

La red privada es una red aislada que no está conectada con otras redes, mientras que la red pública sí lo está, bien a una red local o directamente a Internet. La red de administración es aquella que conecta a todos los nodos para su configuración y administración, debe tener un adecuado control de acceso que permita solamente su entrada a los administradores de la infraestructura.

El esquema de red se presenta en la Figura 2.4, y el resumen del direccionamiento se muestra en la Tabla 2.8.

	Red Pública	Red Privada	Red de Administración
Controller	192.168.221.43	10.0.0.43	172.31.15.167
Compute		10.0.0.51	172.31.15.213
Network	int manual	10.0.0.42	172.31.15.165
Router	192.168.221.1		

Tabla 2. 8 Direccionamiento IP

2.4.2.1 Red de Administración

Esta red se usa para permitir funciones de administración de la infraestructura.

Es necesario configurar el acceso remoto al menos con el protocolo SSH[42] para acceder directamente a las consolas de los nodos, además de dar un acceso restringido a la interfaz de administración web (Horizon). Se requiere una dirección IP en cada nodo, en este caso se ha escogido la red 172.31.15.0/24. La dirección 172.31.15.167 para el nodo *Controller*, 172.31.15.213 para el nodo *Compute* y 172.31.15.165 para el nodo *Network*.

2.4.2.2 Red Pública

Se denomina red pública ya que se usa para asignar direcciones IP de un *pool* externo, para comunicar a las máquinas virtuales fuera de la nube OpenStack.

El servicio *nova-metadata* usa esta red para inyectar información dentro de las máquinas virtuales (por ejemplo las credenciales SSH) generadas por el nodo *Controller*.

Se ha elegido la red 192.168.221.0/24 y se ha asignado estáticamente la dirección 192.168.221.43 para el nodo *Controller*, mientras que se deja el resto del rango libre para asociar a las máquinas virtuales cuando requieran acceso al exterior de la red o a Internet.

2.4.2.3 Red Privada

Llamada también red de datos o red de máquinas virtuales. Esta red se usa para proveer conectividad a las máquinas virtuales OpenStack. Una red de datos separada de la administración permite un flujo de tráfico totalmente aislado y por lo tanto mayor seguridad para el funcionamiento de la infraestructura.

Las interfaces de cada nodo se conectan a esta red a través de túneles Open vSwitch (OVS-GRE[43] *tunnels*).

---

[42] SSH (*Secure Shell*): Es el nombre de un protocolo y del programa que lo implementa, y sirve para acceder a máquinas remotas a través de una red. Permite manejar por completo la computadora mediante un intérprete de comandos.

[43] GRE: *Generic Routing Encapsulation*, es un protocolo desarrollado originalmente por Cisco para el establecimiento de túneles a través de Internet. Puede transportar hasta 20 protocolos de red diferentes.

Se requiere una dirección IP por cada nodo, en este caso se ha elegido la red 10.0.0.0/24. Se emplea la dirección 10.0.0.43 para el nodo *Controller*, 10.0.0.51 para el Nodo *Compute* y 10.0.0.42 para el nodo *Network*.

2.4.2.4 Otros servicios de Red

- **DNS** (*Domain Name Server*): En esta configuración basta con configurar la asignación de nombres estáticamente para identificar cada nodo por su nombre, pero se puede usar un DNS público para la resolución de nombres externo (por ejemplo para la instalación de repositorios).
- **NTP** (*Network Time Protocol*): Se puede configurar todos los nodos para que sincronicen sus relojes con servidores NTP públicos (como ntp.ubuntu.com). Pero en esta implementación se sigue la sugerencia de implementar en el Nodo *Controller* el servidor NTP que sirva para que los otros nodos se sincronicen con él.
- ***Switches* y *Routers* Físicos:** El *switch* de la red de administración es marca Cisco modelo 2960 de 24 puertos 10/100/1000 no administrable. El *switch* de la red privada es marca TrendNet de 8 puertos 10/100 no administrable. Y por último se usa un *router* Linksys modelo WRT120N que divide los segmentos de la red pública y la red externa para el acceso de los clientes.

2.4.3 CARACTERÍSTICAS DE HARDWARE

La documentación oficial recomienda el uso de hardware que sirva para la implementación mínima de un ambiente de producción [40], por lo que las características de los servidores son bastante restrictivas para presupuestos limitados como al que se ajusta esta implementación.

Nodo	CPU	Disco	Memoria RAM	Interfaces de Red	Virtualización
Controller	1 CPU, *single* core	40GB	128MB	3 x 10/100Mbps	No
Network	1 CPU, *single* core	10GB	128MB	3 x 10/100Mbps	No
Compute	1 CPU, *single* core	180GB	4068MB	3 x 10/100Mbps	AMD-v o Intel VT

Tabla 2. 9 Requisitos de hardware mínimo para cada nodo

En la Tabla 2.9 se muestra los requisitos mínimos de hardware de los nodos de la infraestructura para el Prototipo de Proveedor de SaaS, de acuerdo a las funciones que cumple cada nodo según lo mencionado en la sección 2.4.1.

A continuación se justifica los requerimientos mínimos de procesamiento, memoria RAM, almacenamiento de disco e interfaces de red de cada uno de los nodos.

2.4.3.1 Procesamiento

El nodo "Compute" albergará las máquinas virtuales que ejecuten los servicios de los clientes. El hipervisor se encargará de proveer a cada máquina virtual un vCPU[44], y gracias a la propiedad de overcommiting[45] de OpenStack, cada núcleo del procesador físico puede emular 16 vCPUs [41]. Por lo tanto, de acuerdo al objetivo de proveer simultáneamente tres servicios a tres empresas diferentes, el nodo "Compute" deberá tener al menos un procesador de un solo núcleo con capacidad de virtualización. Los nodos "Controller" y "Network" solamente requieren procesar tareas de administración del sistema, por lo que no necesitan características específicas.

2.4.3.2 Memoria RAM

El hipervisor del nodo "Compute" asigna una porción de memoria a cada máquina virtual. Para que una máquina virtual ejecute adecuadamente los servicios de Correo Electrónico, CRM o Video Conferencia, se requiere de al menos 256MB de memoria de RAM, y el servicio con más capacidad tendrá 768MB de memoria RAM (ver sección 3.1.1). El factor de overcommiting para la memoria RAM es de 1,5 [41]; además, los requisitos indican que se deben ejecutar 9 servicios simultáneamente, y en el peor de los casos, cada uno de estos servicios tendrá 768MB de memoria RAM. Esto quiere decir que el nodo "Compute" deberá tener al menos una capacidad física de 4608MB de memoria RAM.

---

[44] vCPU: *virtual Central Processing Unit*, es la emulación de un procesador para un entorno de virtualización de servidores.

[45] *Overcommiting:* capacidad de incrementar el número de instancias que se pueden ejecutar en una nube OpenStack, a cambio de la reducción en el rendimiento de la instancias.

Los nodos “Controller” y “Network”, debido a su labor meramente administrativa, solamente deben tener el mínimo de memoria RAM para soportar un Sistema Operativo Linux, que en el caso de Ubuntu Server 12.04 es 128MB [41].

2.4.3.3 Almacenamiento

En una nube OpenStack, el controlador se encarga del almacenamiento de las imágenes de los sistemas operativos para generar máquinas virtuales. Una distribución de Linux para servidores, debidamente actualizada y con aplicativos adicionales instalados, llega a un máximo de 10GB; siendo éste el caso más extremo, el nodo “Controller” deberá tener al menos 40GB de capacidad de disco.

Por otro lado, cada máquina virtual que se ejecuta en el nodo de cómputo tiene asignada una capacidad de almacenamiento de disco determinada. Para la provisión de los servicios, se tendrá un máximo de 20GB por máquina virtual (ver sección 3.1.1). Es decir que, con el objetivo de soportar un máximo de 9 servicios simultáneamente, el nodo “Compute” deberá tener un disco con capacidad de al menos 180GB.

En el caso del nodo “Network”, la capacidad de almacenamiento requerida es mínima, ya que en ella se almacenará solamente los ficheros del sistema operativo y los scripts y archivos de configuración de OpenStack.

2.4.3.4 Interfaces de Red

De acuerdo al esquema de red descrito en la sección 2.4.2, los nodos “Controller” y “Network” tendrán tres interfaces de red cada uno, mientras que el nodo “Compute” tendrá conexión a través de dos interfaces de red.

Existirá momentos en que se deba transmitir gran cantidad de información en un periodo corto de tiempo, lo que sature a las interfaces de red, pero ésta situación no afectará el funcionamiento del sistema. Debido a que el proyecto es un prototipo, la velocidad de la interfaces de red no es relevante para la funcionalidad del Proveedor de SaaS.

Para el presente proyecto se tuvo la gentil colaboración del Laboratorio de Informática de la Facultad de Ingeniería Eléctrica y Electrónica, quienes facilitaron computadores con las características que se muestran en la Tabla 2.10, que cumplen con los requerimientos mínimos descritos en la Tabla 2.9.

Además, se facilitó el alojamiento de estos computadores en la sala de servidores ubicada en el séptimo piso del Edificio Eléctrica-Química.

	Procesador	Memoria	Disco	Red
Controller	64bits x86	2GB	160GB 7200rpm SATA	3x10/100Mbps
Network	64bits x86	1GB	160GB 7200rpm SATA	3x10/100Mbps
Compute	2x64bits x86	5GB	160GB 7200rpm SATA	2x10/100Mbps

Tabla 2. 10 Características de Hardware de los Nodos

2.4.4 INSTALACIÓN Y CONFIGURACIÓN

En el Anexo A se describe el proceso de instalación de toda la infraestructura OpenStack, tomando como referencia principal a [42], donde se muestra de manera sencilla el proceso de instalación sobre equipos netamente Cisco. También se ha tomado en cuenta documentación oficial de OpenStack [43], [44], así como manuales de contribuyentes independientes [45], [46], [47], [19], recomendaciones, información y ayuda de distintos foros.

Se pretende explicar el proceso de instalación de la manera más detallada posible, por lo que en cada cuadro de código se muestra:

- Comandos: son líneas específicas que se digitan en el *Shell*[46].
- Líneas de Configuración: son líneas que se editan, añaden o sustituyen en los ficheros de configuración.
- Resultado: líneas que se imprimen automáticamente en el *Shell* y que muestran información relacionada a la ejecución de algún comando.

[46] *Shell*, es el término usado en informática para referirse a un intérprete de comandos, interfaz de usuario tradicional de los sistemas operativos basados en Unix. Mediante las instrucciones el usuario puede comunicarse con el núcleo y ejecutar órdenes que permiten controlar el funcionamiento de la computadora.

Se especificará claramente el nombre del usuario, el nombre del nodo, el path absoluto y el permiso con que se ejecuten los comandos.

CAPÍTULO 3
ADMINISTRACIÓN, MONITOREO Y GESTIÓN DEL PROTOTIPO DE PROVEEDOR DE SAAS

Una vez instalada la infraestructura de red y configurados los componentes de OpenStack, se tiene listo el ambiente para crear máquinas virtuales, hacer imágenes de sistemas operativos, crear redes virtuales y principalmente hacer uso de todas la funcionalidades que ofrece OpenStack por medio de su API.

La construcción del Prototipo de Proveedor de SaaS se ha dividido en 3 secciones:

- Recursos IaaS.
- *Scripts*[47] de Automatización.
- Página Web.

Para la sección correspondiente al desarrollo de los scripts de automatización y la página web, se usará la metodología Incremental.

Por su naturaleza, la metodología incremental se acopla perfectamente ya que se tiene identificado claramente dos procesos sucesivos e independientes entre sí, a los cuales se puede aplicar los pasos de la metodología en cascada.

3.1 RECURSOS IAAS

Se refiere a la creación de recursos en la infraestructura OpenStack para ser usados en los *scripts* de automatización.

Gran parte de las acciones sobre la infraestructura OpenStack se puede realizar por medio de la interfaz web de Horizon, pero debido a que constantemente se

[47] *Script* es una porción de código para interactuar con el sistema operativo o con el usuario. Por lo general se almacena en un archivo simple de texto plano.

añaden nuevas funcionalidades, hay acciones que solamente se pueden efectuar por medio de la CLI[48] de OpenStack.

Las CLI de OpenStack son porciones de código abierto desarrollados en lenguaje Python que permiten hacer llamadas al API. Estos comandos se pueden ejecutar desde cualquier computador con sistema operativo Linux que tenga instalado el respectivo cliente. Por ejemplo, para ejecutar comandos del CLI de Nova se debe instalar el paquete *python-novaclient*.

En la Figura 3.1 se presenta un resumen del proceso requerido para la creación de los recursos de IaaS.

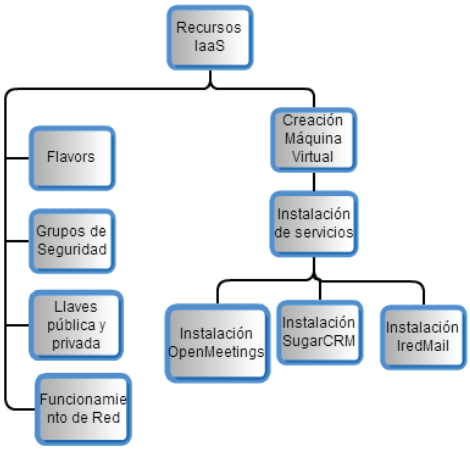


Figura 3. 1 Proceso de creación de los recursos de IaaS

3.1.1 *FLAVOR*

Flavor se denomina a la configuración de hardware de una máquina virtual. Cada *flavor* tiene una única combinación de: número de procesadores virtuales, espacio de disco y capacidad de memoria. Se pueden visualizar los *flavor* disponibles con el comando que se muestra en el Código 3.1.

```
1 $ nova flavor-list
```

Código 3. 1 Comando para listar los flavor disponibles

La manera más fácil de crear nuevos *flavors* es a través de la interfaz web de Horizon, la cual se muestra en la Figura 3.2.

[48] CLI: *Command Line Interface*, es un método de interacción con un programa informático por medio de una línea de texto simple.

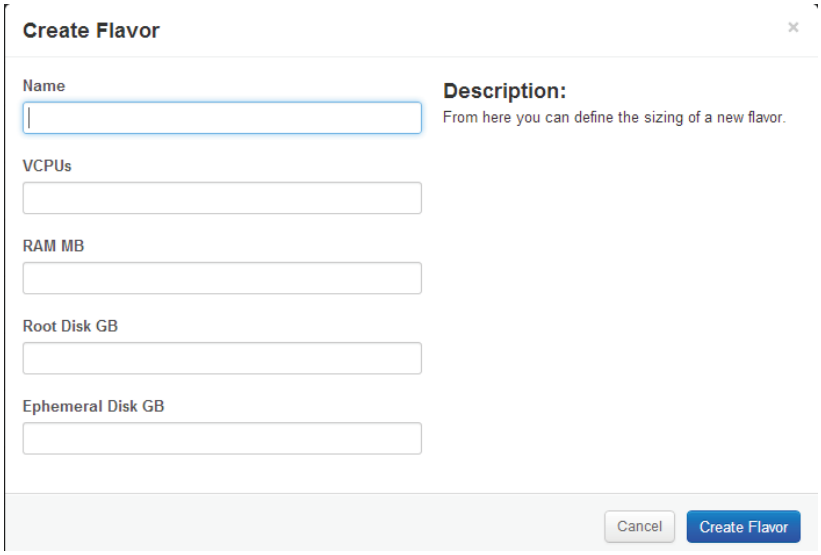


Figura 3. 2 Creación de un *Flavor*

Se definió que para el Presente Proyecto, los servicios de correo electrónico, CRM y video conferencia web pueden tener una capacidad de almacenamiento de 3, 10 o 20GB. Los servicios de correo electrónico y CRM se podrán ofrecer en una misma máquina virtual, por lo que la suma de la capacidad de cada uno de estos debe ser cubierta en la capacidad total de almacenamiento de la máquina virtual donde se alojen. Si los clientes desean ampliar la capacidad de sus servicios, debe existir un *flavor* que tenga las características de capacidad de disco que soporte ese crecimiento. Los *flavors* que se muestran en el Código 3.2 representan las combinaciones de capacidad de almacenamiento de los servicios.

```
$ nova Flavor-list
+---+-----------+-----------+------+-----------+------+-------+-------------+
|ID |    Name   | Memory_MB | Disk | Ephemeral | Swap | VCPUs | RXTX_Factor |
+---+-----------+-----------+------+-----------+------+-------+-------------+
| 2 |     6     |    512    |  6   |     0     |      |   1   |     1.0     |
| 6 |     23    |    1024   |  23  |     0     |      |   1   |     1.0     |
| 4 |     13    |    768    |  13  |     0     |      |   1   |     1.0     |
| 3 |     10    |    512    |  10  |     0     |      |   1   |     1.0     |
| 1 |     3     |    512    |  3   |     0     |      |   1   |     1.0     |
| 7 |     30    |    1024   |  30  |     0     |      |   1   |     1.0     |
| 5 |     20    |    768    |  20  |     0     |      |   1   |     1.0     |
| 8 |     40    |    1024   |  40  |     0     |      |   1   |     1.0     |
+---+-----------+-----------+------+-----------+------+-------+-------------+
```

Código 3. 2 *Flavors* creados

3.1.2 GRUPOS DE SEGURIDAD

Un grupo de seguridad y sus reglas dan al administrador la capacidad de especificar el tipo de tráfico que tiene permitido recibir una máquina virtual.

Cuando se genera una instancia, se la puede asignar a uno o más grupos de seguridad. Todo el tráfico entrante que no corresponde con una regla es denegado por defecto y las reglas se aplican automáticamente cuando se crean [48].

En la versión Grizzly de OpenStack está disponible una mejora de los grupos de seguridad, donde el administrador puede controlar el tipo de tráfico de ingreso y salida de las máquinas virtuales [49].

Los grupos de seguridad disponibles se visualizan con el comando de la línea 1 del Código 3.3, y las reglas de un grupo de seguridad con el comando de la línea 2, donde <nombre_secgroup> debe reemplazarse por el nombre del grupo de seguridad.

```
1 $ nova secgroup-list
2 $ nova secgroup-list-rules <nombre_secgroup>
```

Código 3. 3 Comando para listar los grupos de seguridad disponibles

En la Figura 3.3 se observa la ventana para la creación de un nuevo grupo de seguridad, para ello se accede a la interfaz de Horizon en la sección "*Access and Security".*

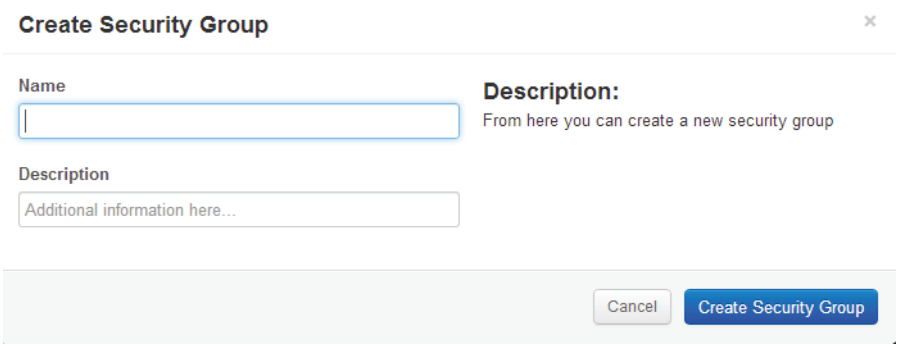


Figura 3. 3 Creación de grupos de seguridad

Para la inserción de reglas en un determinado grupo de seguridad, se emplea el botón "*Edit rules*" del respectivo grupo de seguridad, desplegándose una ventana como la que se observa en la Figura 3.4.

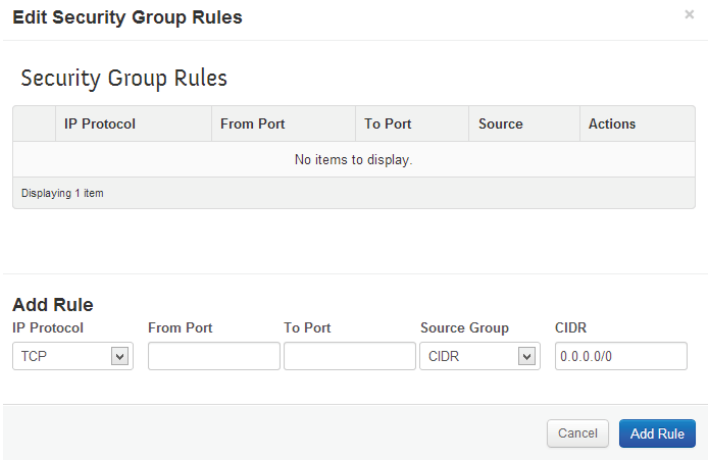


Figura 3. 4 Inserción de reglas en un grupo de seguridad

Ya que en el Presente Proyecto solo hay dos tipos de máquinas virtuales, cada una de éstas tiene su respectivo grupo de seguridad, las cuales representan en el Código 3.4.

Cada grupo de seguridad tiene las reglas que se presentan en el Código 3.5. El servicio de correo electrónico necesita la apertura de los puertos 25 (protocolo SMTP[49]), 110 (protocolo POP3[50]), 143 (protocolo IMAP[51]), además del puerto 80 (protocolo HTTP), 443 (protocolo HTTP/SSL[52]) para el acceso a webmail[53] y el protocolo ICMP[54] y SSH para la administración de la máquina virtual.

```
$ nova secgroup-list
+---------+-------------------------------------------------+
|   Name  |                   Description                   |
+---------+-------------------------------------------------+
| mailcrm | Puertos necesarios para el servicio de mail y crm |
|    op   |        Puertos necesarios para OpenMeetings      |
+---------+-------------------------------------------------+
```

Código 3. 4 Grupos de seguridad creados

[49] SMTP: *Simple Mail Transfer Protocol,* protocolo de capa aplicación para el intercambio de mensajes de correo electrónico.
[50] POP3: *Post Office Protocol,* protocolo de capa aplicación utilizado para obtener mensajes de correo electrónico almacenados en un servidor remoto.
[51] IMAP: *Internet Message Access Protocol,* es un protocolo de acceso a mensajes de correo electrónico almacenados en un servidor remoto. Se diferencia de POP3 ya que IMAP permite la visualización de los mensajes sin la necesidad de descargarlos.
[52] SSL: *Secure Sockets Layer*, es la evolución del protocolo TLS (*Transport Layer Security*), y ambos son protocolos que proporcionan autenticación y privacidad de la información transmitida en una red de datos, permitiendo comunicaciones seguras en Internet.
[53] Webmail: Es un cliente de correo electrónico que provee una interfaz web por la que se puede acceder al correo electrónico.
[54] ICMP: *Internet Control Message Protocol,* es un protocolo de notificación de errores del Protocolo de Internet (IP). Su uso más común es para saber si un dispositivo está disponible.

Mientras que el servicio de video conferencia web requiere la apertura de los puertos 1935 (protocolo RTMP[55]) para la transmisión de video en tiempo real, y el puerto 5080 para la descarga y subida de archivos por medio del protocolo HTTP.

```
$ nova secgroup-list-rules mailcrm
+-------------+-----------+---------+-----------+--------------+
| IP Protocol | From Port | To Port |  IP Range | Source Group |
+-------------+-----------+---------+-----------+--------------+
|     icmp    |     -1    |    -1   | 0.0.0.0/0 |              |
|     tcp     |     22    |    22   | 0.0.0.0/0 |              |
|     tcp     |     25    |    25   | 0.0.0.0/0 |              |
|     tcp     |     80    |    80   | 0.0.0.0/0 |              |
|     tcp     |    110    |   110   | 0.0.0.0/0 |              |
|     tcp     |    143    |   143   | 0.0.0.0/0 |              |
|     tcp     |    443    |   443   | 0.0.0.0/0 |              |
+-------------+-----------+---------+-----------+--------------+
$ nova secgroup-list-rules op
+-------------+-----------+---------+-----------+--------------+
| IP Protocol | From Port | To Port |  IP Range | Source Group |
+-------------+-----------+---------+-----------+--------------+
|     tcp     |    1935   |   1935  | 0.0.0.0/0 |              |
|     tcp     |    5080   |   5080  | 0.0.0.0/0 |              |
+-------------+-----------+---------+-----------+--------------+
```

Código 3. 5 Reglas definidas en los grupos de seguridad

3.1.3 LLAVES PÚBLICAS Y PRIVADAS

Un par de llaves (pública y privada) proporcionan una autenticación segura a las máquinas virtuales. Nova genera un par de llaves pública y privada, y envía la llave privada al usuario. Como parte del primer arranque de una máquina virtual, se añade la llave privada al archivo *authorized_keys*, y la llave pública se almacena de modo que pueda ser inyectada en las máquinas virtuales.

Se puede agregar un nuevo par de llaves con el comando del Código 3.6.

```
$ nova keypair-add --pub_key id_rsa.pub <key_name>
```

Código 3. 6 Comando para la creación de un par de llaves

Siendo `id_rsa.pub` el fichero que representa la llave pública ubicada en */.ssh/,* y `<key_name>` el nombre que se quiere dar al par de llaves.

Para el Presente Proyecto fue necesario crear un único par de llaves, el cual se observa en el Código 3.7. Este par de llaves no será distribuido a los usuarios ya

[55] RTMP: *Real Time Messaging Protocol*, protocolo diseñado por la empresa Macromedia para la trasmisión de video en tiempo real a través de Internet.

que solamente el administrador del sistema tendrá acceso a las máquinas virtuales.

```
$ nova keypair-list
+---------+-------------------------------------------------+
|   Name  |                   Fingerprint                   |
+---------+-------------------------------------------------+
| millave | dc:38:4a:d6:fa:9e:28:7a:c2:b7:dc:7e:0d:fc:f7:2e |
+---------+-------------------------------------------------+
```

Código 3. 7 Par de llaves creado

3.1.4 FUNCIONAMIENTO DE RED

Para entender el funcionamiento de red en OpenStack, es necesario explicar conceptos propios de Quantum.

3.1.4.1 *Tenant*

Se denomina *tenant* al nivel más alto de agrupación de recursos en una nube OpenStack. En un *tenant* se pueden definir cuotas de control, que son límites que el administrador establece para cada uno de los recursos, por ejemplo: creación de volúmenes, tamaño total de los volúmenes en GB, número de máquinas virtuales que pueden ser creadas, número de procesadores que pueden ser asignados a las máquinas virtuales, etc.

Para el Presente Proyecto se creó los *tenants* que se muestran en el Código 3.8.

```
$ keystone Tenant-list
+----------------------------------+--------------------+---------+
|                id                |        name        | enabled |
+----------------------------------+--------------------+---------+
| 621c3f810e494b3d918e52c9a0614070 |       admin        |   True  |
| 849f414c560e47b8b8cff1b645f0d20f |      services      |   True  |
+----------------------------------+--------------------+---------+
```

Código 3. 8 *Tenants* creados

El *tenant "admin"* permite controlar todos los recursos que se generen para la provisión de las máquinas virtuales, mientras que el *tenant "services"* sirve para agrupar los módulos Nova, Quantum, Keystone, y Glance, para así administrar de mejor manera el acceso a sus API.

3.1.4.2 Red

Una red es un dominio de *broadcast* virtual, la cual está reservada para un *tenant*. Cada *tenant* puede tener una o más redes, a su vez, una red puede tener varios

puertos y estos puertos pueden conectarse a interfaces virtuales. Cabe señalar que las subredes y los puertos siempre estarán asociados a una red.

Para el Presente Proyecto se creó las redes que se presentan en el Código 3.9. La red *"public"* sirve para asignar direcciones IP de un rango público a las máquinas virtuales, para que puedan comunicarse con el exterior de la nube OpenStack. La red *"net1"* da conectividad a las máquinas virtuales, además permite que el tráfico de datos este separado del tráfico de administración.

```
$ quantum net-list
+--------------------------------------+--------+--------------------------------------+
| id                                   | name   | subnets                              |
+--------------------------------------+--------+--------------------------------------+
| e3edb8c5-d61c-448d-88df-382d59053317 | public | 6f8287c0-3905-407f-bf30-d3dfac61c618 |
| f4a19a17-de1f-4f0b-854d-3594e05c64bd | net1   | 07b4fb1f-1b3e-45dc-99a7-41e7015bfd3e |
+--------------------------------------+--------+--------------------------------------+
```

Código 3. 9 Redes definidas

3.1.4.3 Subred

Una subred representa un bloque de direcciones IP que se pueden utilizar para su asignación a las máquinas virtuales.

Cada subred debe tener un CIDR[56] y debe estar asociada a una red. Las direcciones IP se asignan a partir de toda la subred CIDR o de una porción de ésta, según la configuración que desee el administrador. Una subred puede tener opcionalmente una puerta de enlace[57], una lista de servidores de nombres DNS y rutas predefinidas. Esta información se inserta en las interfaces asociadas a la subred. Para el Presente Proyecto se crearon las subredes que se presentan en el Código 3.10.

```
$ quantum subnet-list
+------------------+------------------+-------------------------------------------------------------+
| id       | name       | cidr             | allocation_pools                                            |
+----------+------------+------------------+-------------------------------------------------------------+
| 07b4fb1f | 10-subnet  | 10.10.10.0/24    | {"start": "10.10.10.2", "end": "10.10.10.254"}          |
| 6f8287c0 | 221-subnet | 192.168.221.0/24 | {"start": "192.168.221.100", "end": "192.168.221.250"} |
+----------+------------+------------------+-------------------------------------------------------------+
```

Código 3. 10 Subredes creadas

[56] CIDR: *Classless Inter-Domain Routing*, es un estándar de red para la interpretación de direcciones IP.
[57] Puerta de enlace, pasarela o también llamada *gateway*, es un dispositivo que permite interconectar redes.

La subred llamada *"10-subnet"* está asociada a la red *"net1"* para dar conectividad a las máquinas virtuales. La subred *"221-subnet"* está asociada a la red *"public"* y tiene asignado el segmento de red comprendido entre *"192.168.221.100"* y *"192.168.221.250"*, para que las máquinas virtuales tengan conexión al exterior de la nube OpenStack. En el Código 3.11 se muestra un ejemplo, donde a la máquina virtual recién creada (llamada *"prueba"*) se ha asignado la IP *"10.10.10.14"* de la red *"net1"*. Posteriormente con el comando de la línea 10 se asigna una IP de la red "*public*", en este caso se ha asociado la IP *"192.168.221.148"* al puerto *"5a4d2202-0051-412f-aac7-195b47b4f1fa"*.

```
1  $ nova list
2  +--------------------------------------+-----------+-----------+------------------+
3  |                  ID                  |    Name   |   Status  |     Networks     |
4  +--------------------------------------+-----------+-----------+------------------+
5  | 19173eac-e455-4b27-a30f-badd13df9985 |   prueba  |   ACTIVE  | net1=10.10.10.14 |
6  +--------------------------------------+-----------+-----------+------------------+
7  $ quantum port-list | grep 10.10.10.14
8  | 5a4d2202-0051-412f-aac7-195b47b4f1fa |      | fa:16:3e:f9:7a:42 | {"subnet_id":
9  "07b4fb1f-1b3e-45dc-99a7-41e7015bfd3e", "ip_address": "10.10.10.14"}     |
10 $ quantum floatingip-create --port_id 5a4d2202-0051-412f-aac7-195b47b4f1fa public
11 Created a new floatingip:
12 +---------------------+--------------------------------------+
13 | Field               | Value                                |
14 +---------------------+--------------------------------------+
15 | fixed_ip_address    | 10.10.10.14                          |
16 | floating_ip_address | 192.168.221.148                      |
17 | floating_network_id | e3edb8c5-d61c-448d-88df-382d59053317 |
18 | id                  | c221ea08-1833-476f-9ef1-cd7dcdf4619c |
19 | port_id             | 5a4d2202-0051-412f-aac7-195b47b4f1fa |
20 | router_id           | 7910055e-94e6-4fb7-9372-d9e768c1563b |
21 | Tenant_id           | 621c3f810e494b3d918e52c9a0614070     |
22 +---------------------+--------------------------------------+
```

Código 3. 11 Ejemplo para asociar una IP de red pública con una IP de la red privada

3.1.4.4 Interfaz virtual

Interfaz virtual o VIF (*Virtual Interface*), es la analogía con una interfaz de red física (NIC[58]). Las interfaces virtuales se conectan a una red Quantum por medio de puertos.

Un *bridge* Open vSwitch se comporta como un *switch* virtual, las VIF de las máquinas virtuales se conectan a sus puertos y éstos se pueden configurar como si fuesen interfaces de un *switch* físico normal, incluso soportan tráfico VLAN.

[58] NIC: *Network Interface Card*, tarjeta de red. Es un periférico que permite la comunicación con aparatos conectados entre sí.

3.1.4.5 Puerto

Se denomina a la interfaz de un *switch* virtual donde las VIF se conectan a una determinada red. El puerto tiene un estado administrativo que puede ser "*DOWN*" o "*ACTIVE*". Los puertos que están administrativamente "*DOWN*", no pueden recibir ni enviar tráfico.

Para que una máquina virtual tenga acceso a un domino de *broadcast*, ésta debe estar asociada a un puerto activo, y puede intercambiar tráfico con otras máquinas virtuales asociadas a puertos activos en la red. Cuando una dirección IP se asocia a un puerto, este puerto tiene una dirección MAC[59], lo cual implica que se asignó una dirección IP del grupo de direcciones de una subred.

En el Código 3.12 se muestra un ejemplo de las características de un puerto y como Quantum las inserta en la configuración de red de una máquina virtual.

```
$ nova list
+--------------------------+-----------+-----------+------------------+
|                  ID      |    Name   |   Status  |     Networks     |
+--------------------------+-----------+-----------+------------------+
| 19173eac-e455-4b27-a30f |   prueba  |   ACTIVE  | net1=10.10.10.14 |
+--------------------------+-----------+-----------+------------------+
$ quantum port-list | grep 10.10.10.14
| 5a4d2202-0051-412f-aac7 |      | fa:16:3e:f9:7a:42 | {"subnet_id": "07b4fb1f-1b3e-45dc-99a7-
41e7015bfd3e", "ip_address": "10.10.10.14"}     |
$ quantum port-show 5a4d2202-0051-412f-aac7
+----------------+-------------------------------------------------------------------------------+
| Field          | Value                                                                         |
+----------------+-------------------------------------------------------------------------------+
| admin_state_up | True                                                                          |
| device_id      | 19173eac-e455-4b27-a30f                                                       |
| device_owner   | compute:nova                                                                  |
| fixed_ips      | {"subnet_id": "07b4fb1f-1b3e-45dc-99a7", "ip_address": "10.10.10.14"} |
| id             | 5a4d2202-0051-412f-aac7                                                       |
| mac_address    | fa:16:3e:f9:7a:42                                                             |
| name           |                                                                               |
| network_id     | f4a19a17-de1f-4f0b-854d                                                       |
| status         | ACTIVE                                                                        |
| Tenant_id      | 621c3f810e494b3d918e52c9a0614070                                              |
+----------------+-------------------------------------------------------------------------------+
$ SSH ubuntu@10.10.10.14
Welcome to Ubuntu 12.04.2 LTS (GNU/Linux 3.2.0-38-virtual x86_64)
.
.
.
ubuntu@prueba:~$ ifconfig eth0
eth0      Link encap:Ethernet  HWaddr fa:16:3e:f9:7a:42
          inet addr:10.10.10.14  Bcast:10.10.10.255  Mask:255.255.255.0
          inet6 addr: fe80::f816:3eff:fef9:7a42/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1151 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1043 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:190605 (190.6 KB)  TX bytes:170570 (170.5 KB)
```

Código 3. 12 Ejemplo de la asociación de un puerto a una máquina virtual

[59] MAC: *Media Access Control*, es un identificador de 48 bits dividido en seis bloques hexadecimales, que identifica de forma única a una tarjeta o dispositivo de red.

El comando de la línea 1 del Código 3.12 lista las máquinas virtuales, en este caso la máquina virtual *"prueba"* está activa y tiene la dirección IP *"10.10.10.14"* en la red interna *"net1"*. Mediante el comando ejecutado en la línea 7 se obtiene el puerto asignado a la dirección IP *"10.10.10.14"*, identificado por medio del id *"5a4d2202-0051-412f-aac7"* como se presenta en la línea 8. El comando de la línea 10 lista las características del puerto, en la salida del comando (líneas 11 a 24) se evidencia el id *"19173eac-e455-4b27-a30f* en la línea 15. De esta manera se verifica que se ha asignado a la máquina virtual *"prueba"* la dirección IP *"10.10.10.14"* y la dirección MAC *"fa:16:3e:f9:7a:42"*. Estos datos se pueden comprobar accediendo directamente a la máquina virtual *"prueba"* por medio de SSH. En la línea 31 se observa que efectivamente Quantum asignó la dirección MAC *"fa:16:3e:f9:7a:42"* a la interfaz *"eth0".*

3.1.4.6 Router

Es un dispositivo que conecta el tráfico de una subred a una red externa. Cada *router* puede tener un *gateway* asignado a un puerto de una red externa y múltiples interfaces en una red interna. Una vez que la subred se conecta a una red externa, las máquinas virtuales asociadas pueden enviar tráfico al exterior a través del *router*. En el Código 3.13 se observa las características del *router* que conecta la red "*public*" con la red "*net*1".

El *router* denominado *"router1"* funciona como *gateway* externo para la red con id "e3edb8c5-d61c-448d-88df", mediante el comando de la línea 7 se comprueba que el ID efectivamente pertenece a la red "*public*", como se aprecia en la línea 8.

El comando de la línea 9 despliega las características de *"router1"*, su salida presenta el ID *"7910055e-94e6-4fb7-9372"* lo cual se observa en la línea 15. A fin de mostrar las interfaces virtuales y el direccionamiento creado, se ejecuta el comando de la línea 20.

De la línea 21 a la 24 se observa la interfaz llamada *"qr-6ebc1df6-95"*, la cual tiene la dirección IP "*10.10.10.1/24*", que es el *gateway* de la red interna *"net1"*.

```
1  $ quantum router-list
2  +--------------------------+---------+-----------------------------------------+
3  | id                       | name    | external_gateway_info                   |
4  +--------------------------+---------+-----------------------------------------+
5  | 7910055e-94e6-4fb7-9372 | router1 | {"network_id": "e3edb8c5-d61c-448d-88df"} |
6  +--------------------------+---------+-----------------------------------------+
7  $ quantum net-list | grep e3edb8c5-d61c-448d-88df
8  | e3edb8c5-d61c-448d-88df | public | 6f8287c0-3905-407f-bf30 |
9  $ quantum router-show router1
10 +-----------------------+-----------------------------------------+
11 | Field                 | Value                                   |
12 +-----------------------+-----------------------------------------+
13 | admin_state_up        | True                                    |
14 | external_gateway_info | {"network_id": "e3edb8c5-d61c-448d-88df"} |
15 | id                    | 7910055e-94e6-4fb7-9372-d9e768c1563b    |
16 | name                  | router1                                 |
17 | status                | ACTIVE                                  |
18 | Tenant_id             | 621c3f810e494b3d918e52c9a0614070        |
19 +-----------------------+-----------------------------------------+
20 $ ip netns exec qrouter-7910055e-94e6-4fb7-9372 ip addr list
21 8: qr-6ebc1df6-95: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN
22     link/ether fa:16:3e:81:33:30 brd ff:ff:ff:ff:ff:ff
23     inet 10.10.10.1/24 brd 10.10.10.255 scope global qr-6ebc1df6-95
24        valid lft forever preferred lft forever
25 11: qg-1e45d281-65: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN
26     link/ether fa:16:3e:15:44:50 brd ff:ff:ff:ff:ff:ff
27     inet 192.168.221.101/24 brd 192.168.221.255 scope global qg-1e45d281-65
28     inet 192.168.221.148/32 brd 192.168.221.148 scope global qg-1e45d281-65
29 $ ovs-vsctl list-br
30 br-ex
31 br-int
32 br-tun
33 $ ovs-vsctl list-ifaces br-int
34 qr-6ebc1df6-95
35 $ ovs-vsctl list-ifaces br-ex
36 eth2
37 qg-1e45d281-65
```

Código 3. 13 Características del router para la conexión entre la red externa e interna

De la línea 25 a la 28 se observa que la interfaz llamada *"qg-1e45d281-65"* tiene asociada la dirección IP *"192.168.221.101*" que es la dirección propia del *router,* y la dirección IP *"192.168.221.148"*, que es la dirección de la máquina virtual "*prueba*" en la red externa "*public*". El comando de la línea 29 permite presentar los *bridges* creados por Quantum.

El comando de la línea 33 muestra las interfaces asociadas al puente *"br-int", en este caso,* está asociado a *"qr-6ebc1df6-95"* que es la interfaz interna del *router*. El comando de la línea 35 muestra las interfaces asociadas al puente *"br-ex",* en este caso son las interfaces *"qg-1e45d281-65"*, que es la interfaz externa del *router*, y la interfaz física *"eth2"* del nodo "*network*", que es la interfaz que está conectada directamente al *router* externo *"linksys"* y que provee la conexión con Internet (observar la Figura 2.4).

3.1.5 CREACIÓN DE MÁQUINAS VIRTUALES

Para generar una nueva máquina virtual desde la CLI, se usa el comando del Código 3.14. Donde `<nombre_instancia>` es el nombre de la nueva máquina virtual, `<uuid_imagen>` es el identificador de la imagen alojada en Glance y `<nombre_flavor>` es el nombre que se da a la configuración que usará la nueva máquina virtual.

```
1 $ nova boot <nombre_instancia> --image "<uuid_imagen>" --flavor <nombre_flavor>
```

Código 3. 14 Creación de una máquina virtual desde CLI

Para visualizar las imágenes disponibles se puede utilizar cualquiera de los comandos de las líneas 1 y 2 del Código 3.15. Y se pueden visualizar los *flavors* disponibles con el comando de la línea 3.

```
1 $ glance image-list
2 $ nova image-list
3 $ nova flavor-list
```

Código 3. 15 Comandos para listar las imágenes disponibles y los *flavors*

En la Figura 3.5 se muestra la ventana de la interfaz de Horizon al haber elegido el botón *"Generar Instancia"*. Para el Presente Proyecto se generó una imagen de Ubuntu Server llamada "*precise*", como se observa en las líneas 14 a 17 del Código 2.8. A partir de esta imagen se generarán las máquinas virtuales en los que se configurarán los servicios de correo electrónico, CRM y video conferencia web.

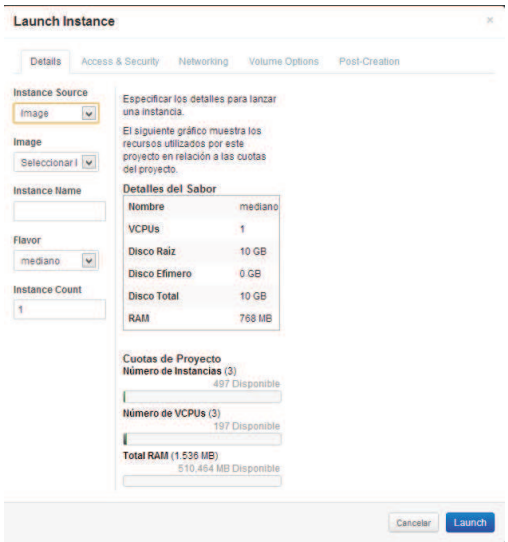


Figura 3. 5 Creación de una máquina virtual usando Horizon

3.1.5.1 Instalación de servicios

Los servicios que el prototipo de Proveedor SaaS ofrece son:

- Correo electrónico, por medio del software IredMail.
- CRM, por medio del software SugarCRM.
- Video conferencia web, por medio del software OpenMeetings.

3.1.5.1.1 Instalación de iRedMail

iRedMail es un proyecto de código abierto que tiene las principales funciones para la implementación de un servidor de correo. Entre sus componentes constan:

- Postfix: software para el manejo del protocolo SMTP [50].
- Dovecot: software para el manejo de los protocolos POP3 e IMAP [51].
- RoundCube: software que provee la interfaz *webmail* [52].
- ClamAV: software para el escaneo de virus [53].
- SpamAssassin: software para la detección y control de *spam* [54].

A partir de una nueva instancia de la imagen "*precise*", se pueden seguir los pasos sugeridos en el manual de instalación para Ubuntu 12.04 disponibles en la documentación oficial de iRedMail [55].

Debido a un error en el script de instalación se editó el archivo *virtual_domain_config.sh* para inhabilitar el cuadro de diálogo de la selección del dominio, el usuario administrador y su contraseña. La URL de acceso a la interfaz web de administración de iRedMail es *https://<ip_publica>/iredadmin*. La interfaz se presenta en la Figura 3.6.

Figura 3. 6 Interfaz de acceso a iRedMail

El acceso a la interfaz *webmail* es *https://<ip_ublica>/webmail*, (la interfaz web se presenta en la Figura 3.7). En ambos casos se puede acceder con el usuario administrador *"postmaster@domino.com"* y la contraseña "*password*".

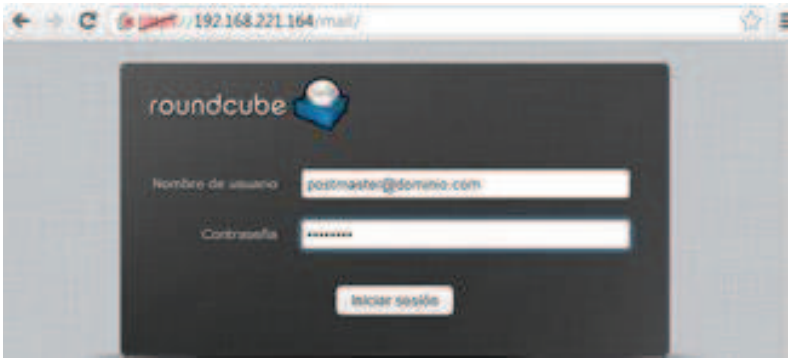


Figura 3. 7 Interfaz de acceso *webmail*

3.1.5.1.2 Instalación de SugarCRM

En la misma instancia donde se instaló el software iRedMail, se puede seguir las instrucciones para la instalación de SugarCRM versión 6.5 *Community Edition* para Ubuntu 12.04 [56].

El acceso a la interfaz web de SugarCRM es http://<ip_publica>/sugarcrm y se accede con las credenciales del usuario *"admin"* y su contraseña "*password*". El interfaz web se presenta en la Figura 3.8.

Figura 3. 8 Interfaz web de acceso a SugarCRM

Para demostrar la flexibilidad que se puede tener en la administración de los recursos IaaS para proveer servicios SaaS, se decidió instalar el software iRedMail y SugarCRM en una misma máquina virtual, con el objetivo de optimizar el uso de los recursos IaaS. Para esto, se procede a crear una imagen de la

máquina virtual por medio de la interfaz de Horizon. Se accede a las opciones de la máquina virtual y se elige el botón "*Create Snapshot*". En la Figura 3.9 se presenta el interfaz para la creación de una imagen.

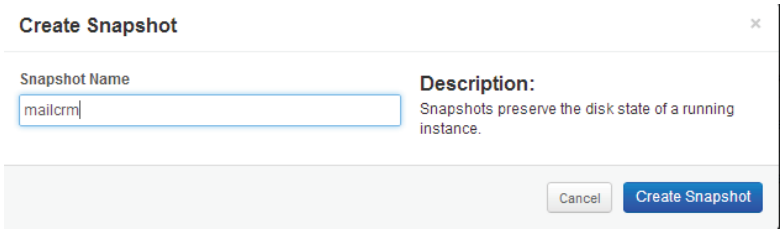


Figura 3. 9 Creación de la imagen "mailcrm"

3.1.5.1.3 Instalación de OpenMeetings

A partir de una nueva instancia de la imagen "precise", se puede seguir los pasos descritos en [57].

El acceso a la interfaz web de OpenMeetings es http://<ip_publica>:5080/openmeetings y se accede con el usuario "admin" y la clave "daniel". La interfaz web se presenta en la Figura 3.10.

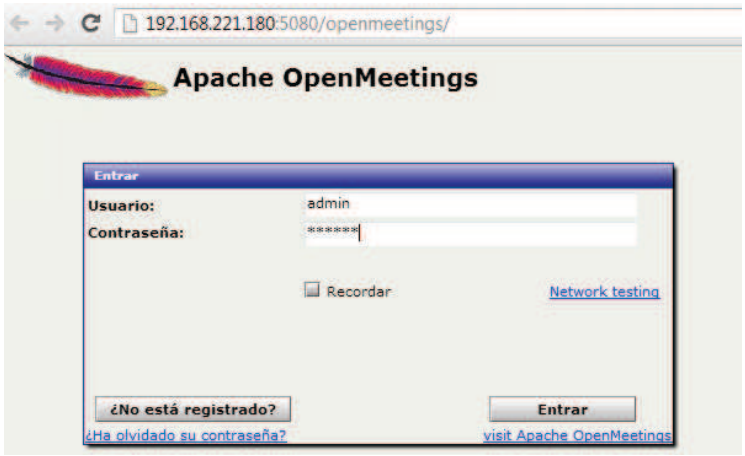


Figura 3. 10 Interfaz web de acceso a OpenMeetings

Cabe mencionar que debe estar creada la regla para el puerto 5080 en el grupo de seguridad correspondiente, para que la interfaz web se despliegue correctamente.

Ya que el servicio *"red5"* que habilita todos los componentes de OpenMeetings no se ejecuta automáticamente al iniciar la máquina virtual, se añadió el comando de

inicio de servicio en el archivo *"crontab"* para que el demonio cron[60] ejecute cada cinco minutos, como se observa en el Código 3.16.

```
$ sudo crontab -e
# m h  dom mon dow   command
*/5 * * * * sudo /etc/init.d/red5 start
```

Código 3. 16 Configuración de cron para inicio de servicio red5

Se decidió instalar el software OpenMeetings en una sola máquina virtual, con el objetivo de demostrar el nivel de automatización que puede tener un servicio SaaS totalmente libre de la configuración del administrador del sistema.

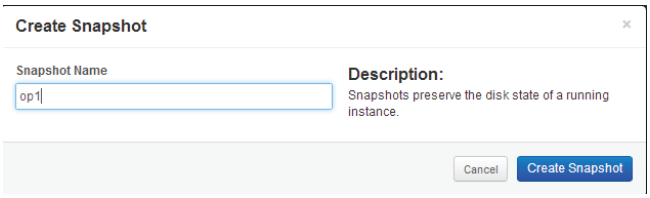


Figura 3. 11 Creación de la imagen de OpenMeetings

Para ello, se procede a crear una imagen de la máquina virtual donde está instalado OpenMeetings. En la interfaz de Horizon se accede a las opciones de la máquina virtual y se elige el botón "*Create Snapshot*" tal como se muestra en la Figura 3.11.

3.2 *SCRIPTS* DE AUTOMATIZACIÓN

Al tener lista la infraestructura, configurados y disponibles los recursos IaaS, ya se puede hacer uso del API de OpenStack para desarrollar pequeños programas para manipular los recursos IaaS y generar la automatización que este Proyecto requiere.

3.2.1 ANÁLISIS

Para determinar los requisitos que deben cumplir los *scripts*, es necesario analizar la interacción del usuario con el proveedor de SaaS. Las actividades que el prototipo debe ofrecer son:

[60] Cron. Es el administrador de procesos en segundo plano (demonio) que ejecuta procesos a intervalos de tiempo determinados.

- Generar los servicios de Correo Electrónico, CRM y Video Conferencia.
- Manipular la capacidad de almacenamiento del servicio contratado.
- Permitir la cancelación del servicio contrato en el instante que el usuario lo desee.

Para resolver las actividades anteriores, es necesario crear *scripts* que cumplan la siguiente funcionalidad:

- Crear máquinas virtuales con el servicio contratado por el cliente.
- Manipular el estado de la máquina virtual que albergue el servicio contratado por el cliente.
- Manipular la capacidad de disco asignado a la máquina virtual que alberga el servicio contratado por el cliente.

3.2.2 DISEÑO

En la Figura 3.12 se muestra un breve esquema de los *scripts* requeridos para el Presente Proyecto.

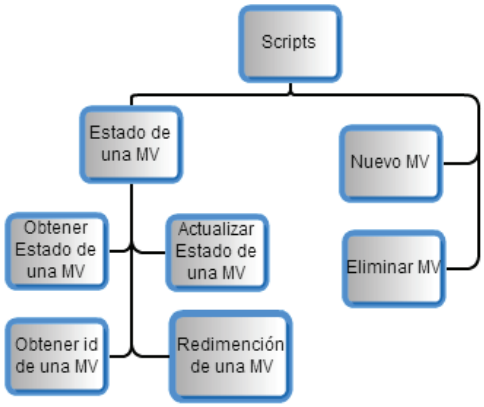


Figura 3. 12 Diagrama de los *scripts* de automatización

Para el uso del API de OpenStack se requiere establecer una conexión HTTP a la URL[61] específica del recurso que se desea consultar (usando el método *"GET"*) o cambiar (usando el método *"POST"*). Esta conexión debe incluir las credenciales de autenticación (*username*, *password* e ID del *Tenant*) generadas con el módulo

[61] URL: *Uniform Resource Locator,* es la cadena de caracteres que se asigna a todos los recursos de información para su localización en Internet.

de autenticación Keystone y especificar el formato de respuesta que puede ser JSON[62] o XML.

Se empleó el lenguaje de programación Python [58] ya que existe mejor compatibilidad con el API de OpenStack, en comparación con otros lenguajes. Además se usó funcionalidades propias del *shell* de Linux.

3.2.3 CODIFICACIÓN

3.2.3.1 Nueva máquina virtual

Se requiere crear máquinas virtuales con las características que el usuario elija mediante la aplicación web de acuerdo al tipo de servicio que se quiere contratar. Para ello es necesario que esas características sean recibidas por un *script* en forma de variables y que luego de procesar se entregue información para poder acceder a la nueva máquina virtual. El *script* que se adjunta en el Anexo B se llama *"newserver.py"*, éste recibe como variables, el nombre de la imagen y el nombre del *Flavor* que se desea usar, y devuelve el ID de la dirección IP pública asociada a la máquina virtual creada.

El *script* realiza el siguiente proceso:

1. Obtiene las variables de ingreso, que son el nombre de la imagen y el nombre del *Flavor*.
2. Hace una llamada HTTP al API de Keystone con las credenciales de acceso para obtener un *token*[63].
3. Obtiene el ID del *Flavor*.
4. Obtiene el ID de la imagen.
5. Crea una instancia de la imagen con las características del *Flavor*, además se añade el nombre del par de llaves *"millave"*, el ID de la red *"net1"* y el nombre del grupo de seguridad.

---

[62] JSON: *JavaScript Object Notation*, es un sencillo formato para el intercambio de datos. Forma parte del lenguaje JavaScript pero su simplicidad ha provocado que sea usado por la gran mayoría de lenguajes al momento de intercambiar datos.

[63] *Token*: es un conjunto de caracteres secreto asociado a un usuario y su contraseña. Tiene una fecha y hora de expiración.

6. Obtiene el puerto asociado a la máquina virtual recién creada.
7. Asigna una dirección IP de la red "*public*".
8. El *script* devuelve el ID de la IP pública asociada a la máquina virtual.

3.2.3.2 Eliminar una máquina virtual

El Proyecto ofrece la funcionalidad de cancelar el servicio contratado en el momento que el cliente requiera. Esto implica que se debe eliminar la máquina virtual que tiene ejecutando el servicio que el cliente estaba utilizando.

El *script* que se adjunta en el Anexo C se llama *"delete_server.py"*, el cual recibe como variable el ID de la máquina virtual, la elimina, y no devuelve ninguna información.

3.2.3.3 Estado de la máquina virtual

El prototipo permite que el cliente pueda manipular las características del servicio contratado, principalmente aumentar o disminuir las prestaciones y suspender o activar el servicio sin perder la información almacenada. Para ello, fue necesario crear los *scripts* que manipulen, por medio del API de OpenStack, el estado de las máquinas virtuales donde se alojan los servicios contratados.

3.2.3.3.1 Obtener estado de una máquina virtual

Es necesario saber el estado de la máquina virtual donde se está ejecutando el servicio contratado por el cliente. El *script* llamado *"get_dbservidor.py"* que se adjunta en el Anexo D, recibe como parámetros de entrada el ID de una máquina virtual y la palabra *"ip"* o la palabra *"estado"*.

Si el segundo parámetro de entrada es *"ip"*, el *script* devuelve la dirección IP pública asignada a la máquina virtual; caso contrario, (es decir si el segundo parámetro de entrada es *"estado"*) el *script* devuelve la palabra "*active*" si la máquina virtual está activa, o "*suspended*" si la máquina virtual está en pausa.

Cabe señalar que Nova denomina "*active*" al estado en que la máquina virtual se está ejecutando y usa memoria RAM, procesamiento en el CPU y almacenamiento de disco. Por otro lado, Nova denomina "*suspended*" al estado

en que la máquina virtual está suspendida o en pausa, esto significa que la máquina virtual no se está ejecutando y que ha liberado los recursos físicos tanto de memoria RAM como de procesamiento de CPU.

3.2.3.3.2 Actualizar estado de una máquina virtual

Para que el cliente pueda manejar el estado del servicio contratado, es necesario manipular directamente el estado de la máquina virtual donde está alojado. El *script* llamado *"set_dbservidor.py"* que se adjunta en el Anexo E, recibe como parámetros de entrada el ID de una máquina virtual y la palabra "*active*" o la palabra "*suspend*".

Si el segundo parámetro de entrada es "*active*", significa que se quiere activar o reanudar una máquina virtual. Esto implica que se use nuevamente recursos de memoria RAM y procesamiento de CPU.

Si el segundo parámetro es la palabra "*suspend*", significa que se quiere suspender o pausar una máquina virtual. Esto implica que se liberarán los recursos de memoria RAM y procesamiento de CPU.

3.2.3.3.3 Obtener el ID de una máquina virtual

Para facilitar la administración de las máquinas virtuales, fue necesario crear un *script* que indique el identificador de la máquina virtual asociado a una dirección IP pública. Cabe recordar que el *script "newserver.py"* solamente devuelve la dirección IP de una máquina virtual recién creada, es por ello la necesidad de crear el *script* llamado *"get_uuid.py"* presentado en el Anexo F, que recibe como parámetro de entrada una dirección IP, y devuelve el ID de la máquina virtual a la que pertenece.

3.2.3.3.4 Redimensionamiento de una máquina virtual

El prototipo proporciona al usuario la posibilidad de ampliar o disminuir las prestaciones del servicio contratado. Es decir aumentar o disminuir los recursos de memoria RAM, y almacenamiento en disco que el servicio contratado ocupa de la máquina virtual donde se ejecuta. Para ello se creó el *script* llamado *"resizevm2.py"* el cual se incluye en el Anexo G. El *script* recibe como parámetros

de entrada el ID de una máquina virtual y la cantidad de GB (en números) al que se desea redimensionar. Los posibles valores para el segundo parámetro en realidad corresponden al nombre de un *Flavor* previamente configurado, como se puede observar en el Código 3.2.

El *script* transforma el nombre del *Flavor* a su correspondiente ID para usarlo con el comando *"nova resize"* y ejecutar el redimensionamiento de la máquina virtual.

Esta operación suele demorar varios minutos ya que es un proceso que realiza directamente el hipervisor KVM en el respectivo nodo de cómputo.

Al finalizar el proceso la máquina virtual se encuentra en un estado llamado "*VERIFY_RESIZE*", significa que se debe ejecutar manualmente el comando *"nova resize-confirm"* para aceptar el redimensionamiento de la máquina virtual, o el comando *"nova resize-revert"* para retroceder a las características del *Flavor* anterior.

Para brindar mayor automatización a este proceso, se creó el *script* llamado *"verify_resize.py"* que se muestra en el Anexo H. Este *script* se ejecuta a través del demonio *"cron"* con periodicidad de 1 minuto y lo que hace es listar todas las máquinas virtuales para verificar su estado una por una, si encuentra alguna máquina virtual con el estado "*VERIFY_RESIZE*" ejecuta el comando "*nova resize-confirm*" para aceptar el redimensionamiento, y se registra el éxito del redimensionamiento en un archivo de log llamado *"out.txt"*, como se muestra en el Código 3.17.

```
2013-04-23 14:45:03.538477 se confirmó resize de 82511101-f3e3-485a-80f0
2013-04-23 15:15:03.214064 se confirmó resize de d7336316-f296-43a2-9ace
2013-04-23 15:40:02.777483 se confirmó resize de 0cf0b52e-c4c0-40ef-8418
2013-04-23 16:00:02.519135 se confirmó resize de cd41e674-660a-4af3-99db
2013-04-23 16:19:02.909212 se confirmó resize de fcd83807-44e5-4f66-a574
2013-04-23 16:40:02.543058 se confirmó resize de d405b168-b630-4ea3-a8b8
```

Código 3. 17 Contenido del archivo de log out.txt

3.2.4 PRUEBAS

El Código 3.18 muestra la generación de una nueva máquina virtual, la manipulación de su estado y capacidad a través de los scripts de automatización.

```
$ python newserver.py test mailcrm
192.168.221.116
$ nova list
+--------------------------+-----------+-----------+------------------+
|       ID                 |    Name   |   Status  |     Networks     |
+--------------------------+-----------+-----------+------------------+
| 19173eac-e455-4b27-ab7f |    test   |   ACTIVE  | net1=10.10.10.11 |
+--------------------------+-----------+-----------+------------------+
$ quantum floatingip-list | grep 10.10.10.11
| 5a4d2202-0051-412f-aac7 |      | 192.168.221.116 | "ip_address": "10.10.10.11" |
$ python set_dbservidor.py 19173eac-e455-4b27-ab7f suspended
$ python get_dbservidor.py 19173eac-e455-4b27-ab7f estado
suspended
$ python set_dbservidor.py 19173eac-e455-4b27-ab7f active
$ nova list
+--------------------------+-----------+-----------+------------------+
|          ID              |    Name   |   Status  |     Networks     |
+--------------------------+-----------+-----------+------------------+
| 19173eac-e455-4b27-ab7f |    test   |   ACTIVE  | net1=10.10.10.11 |
+--------------------------+-----------+-----------+------------------+
$ python get_dbservidor.py 19173eac-e455-4b27-ab7f ip
192.168.221.116
$ python resizevm2.py 19173eac-e455-4b27-ab7f 20
$ nova list
+--------------------------+---------+-------------+------------------+
|          ID              |   Name  |    Status   |     Networks     |
+--------------------------+---------+-------------+------------------+
| 19173eac-e455-4b27-ab7f |   test  |VERIFY_RESIZE| net1=10.10.10.11 |
+--------------------------+---------+-------------+------------------+
$ python verify_resize.py
$ nova list
+--------------------------+-----------+-----------+------------------+
|                 ID       |    Name   |   Status  |     Networks     |
+--------------------------+-----------+-----------+------------------+
| 19173eac-e455-4b27-ab7f   |    test   |   ACTIVE  | net1=10.10.10.11 |
+--------------------------+-----------+-----------+------------------+
```

Código 3. 18 Pruebas de scripts de automatización

- Línea 1 a 10: El *script* "*new_server.py*" genera una nueva máquina virtual llamada "*test*". Esta se encuentra en estado "*Active*" y tiene la IP interna "10.10.10.11", la cual está asociada a la IP pública "192.168.221.116".
- Línea 11: El *script* "*set_dbservidor.py*" cambia el estado de la máquina virtual "test" a "*SUSPENDED*".
- Línea 12 y 13: El *script* "*get_dbservidor.py*" muestra que efectivamente máquina virtual está en estado "*SUSPENDED*".
- Línea 14 a 20: El *script* "*set_dbservidor.py*" cambia el estado de la máquina virtual a "*ACTIVE*".
- Línea 21 y 22: El *script* "*get_dbservidor.py*" muestra la IP pública asociada a la máquina virtual "*test*".

- Línea 23 a 28: El *script* "*resize_vm2.py*" realiza el redimensionamiento de la máquina virtual "*test*" al *flavor* llamado "20", y deja a la máquina virtual en el estado "*VERIFY_RESIZE*".
- Línea 29 a 35: El *script* "*verify_resize.py*" verifica el estado de la máquina virtual "*test*" y la pasa al estado "*ACTIVE*".

Al haber comprobado el correcto funcionamiento de cada *script*, se da por terminado el presente proceso, y se da paso al siguiente, donde se dará la funcionalidad al Prototipo de Proveedor de SaaS por medio de una Interfaz Web.

3.3 PÁGINA WEB

Una vez que los recursos IaaS están configurados, y se han creado los *scripts* para la automatización, ya es posible crear una interfaz web donde el cliente pueda interactuar con la administración de los servicios.

3.3.1 ANÁLISIS

Para satisfacer las necesidades de un servicio SaaS, el prototipo debe ofrecer las siguientes características:

- Tener una interfaz web amigable y de fácil entendimiento para el cliente.
- Cliente pueda solicitar nuevos servicios.
- Manipular los aspectos básicos de los servicios contratados, como el aumento o disminución del almacenamiento, la suspensión o activación de los servicios.
- Visualizar la facturación generada por el consumo de los servicios contratados.
- La página web sea el único medio para que un cliente pueda administrar los servicios contratados.

3.3.2 DISEÑO

En la Figura 3.13 se detalla brevemente las etapas de desarrollo de la página web.

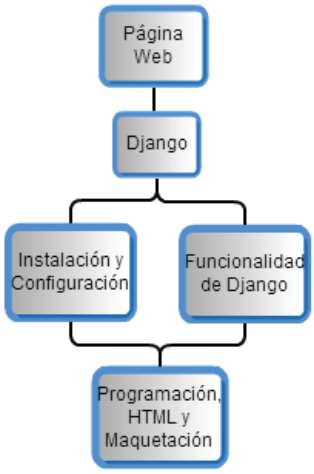


Figura 3. 13 Etapas de desarrollo de la página web

3.3.2.1 Instalación y configuración de Django

Se eligió el nodo *Network* para que funcione como servidor web. En este nodo está instalado Ubuntu Server 12.04, por lo que ya se encuentran instalados todos los componentes del lenguaje Python. La instalación de Django se describe en los comandos que se muestran en el Código 3.19.

```
1  root@network:~# wget https://www.djangoproject.com/download/1.4.5/tarball/
2  root@network:~# tar zxvf Django-1.4.5.tar.gz
3  root@network:~# cd Django-1.4.5/
4  root@network:~/Django-1.4.5# Python setup.py install
5  root@network:~# mkdir project
6  root@network:~# cd project
7  root@network:~/project# django-admin.py startproject saas
8  root@network:~# Python project/saas/manage.py runserver 0.0.0.0:8000
9  Validating models...
10
11 0 errors found
12 Django version 1.4.5, using settings 'saas.settings'
13 Development server is running at http://0.0.0.0:8000/
14 Quit the server with CONTROL-C.
```

Código 3. 19 Instalación de Django

Es necesario descargar directamente el código fuente de la última versión disponible en la página oficial del proyecto Django (línea 1), en este caso se usó la versión 1.4.5. En la línea 2 se extrae su contenido y se ejecuta el archivo *"setup.py"* para la instalación como se muestra en la línea 4. El comando de la línea 8 inicia el servidor indicando que se acepta conexiones desde cualquier dirección IP a través del puerto 8000.

La Figura 3.14 muestra que efectivamente se puede acceder al servidor de prueba de Django por medio de un navegador web.

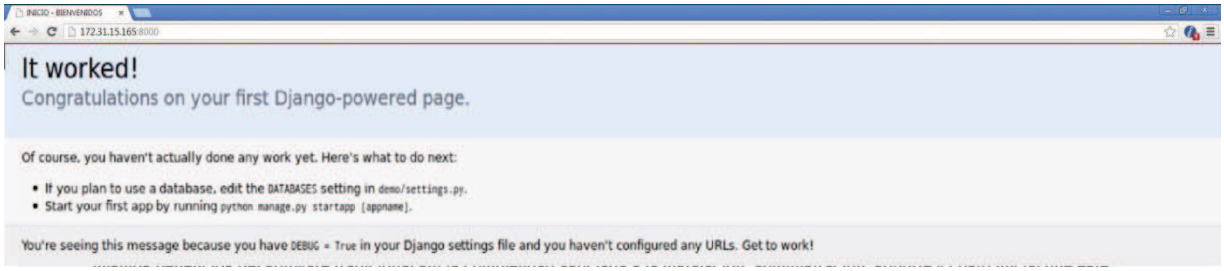


Figura 3. 14 Página web básica configurada con Django

A continuación se explica el modo en que Django organiza los recursos para la creación de páginas web.

3.3.2.1.1 Proyecto

Django define a un proyecto como la colección de aplicaciones y su configuración para una página web. Un proyecto puede contener varias aplicaciones.

Se definió el proyecto llamado *"saas"*. En la carpeta del proyecto *"saas"* están los archivos que se muestran en el Código 3.20.

```
1 root@network:~/project/saas/saas# ls
2 apps     settings.py   templates
3 __init__.py   media   urls.py    wsgi.py
```

Código 3. 20 Archivos del proyecto "saas"

3.3.2.1.2 App

Django llama "*app*" a una aplicación web que realiza alguna tarea específica. Una *app* puede ser parte de múltiples proyectos [59]. Para alojar las aplicaciones de la página web se creó la carpeta *"apps"* lo cual se realiza mediante la línea 1 del Código 3.21. Dentro de esta carpeta se creó el archivo *"__init__.py"* como se presenta en la línea 3. Este es un archivo vacío que indica a Python que la carpeta debe ser considerada como un paquete.

Para la página web se creó una sola *app* llamada *"pymes"* con el comando que se observa en la línea 4. Este comando crea la carpeta "*pymes*" y dentro de esta, los

archivos donde se configura la funcionalidad de la página web, como son: modelos, formularios, URL y las vistas.

```
root@network:~/project/saas/saas# mkdir apps
root@network:~/project/saas/saas#cd apps
root@network:~/project/saas/saas/apps# tounch __init__.py
root@network:~/project/saas/saas/apps# django-admin.py startapp pymes
root@network:~/project/saas/saas/apps# cd pymes && ls
admin.py      models.py      urls.py     forms.py      views.py
__init__.py   tests.py
```

Código 3. 21 Configuración de la aplicación "pymes"

3.3.3 CODIFICACIÓN

3.3.3.1 Modelos

```
root@controller:~# mysql -u root -pdaniel
mysql> use saas;
mysql> show tables;
+----------------------------+
| Tables_in_saas             |
+----------------------------+
| pymes_cliente              |
| pymes_cliente_servidores   |
| pymes_cuenta               |
| pymes_factura              |
| pymes_mail                 |
| pymes_reg_dominio          |
| pymes_reg_servicio         |
| pymes_servicio             |
| pymes_servicio_activo      |
| pymes servidor             |
| pymes servidor servicios   |
| pymes tipo                 |
+----------------------------+
12 rows in set (0.00 sec)
mysql> describe pymes_cliente;
+-----------+--------------+------+-----+---------+----------------+
| Field     | Type         | Null | Key | Default | Extra          |
+-----------+--------------+------+-----+---------+----------------+
| id        | int(11)      | NO   | PRI | NULL    | auto_increment |
| user_id   | int(11)      | NO   | UNI | NULL    |                |
| nombre    | varchar(100) | NO   |     | NULL    |                |
| apellidos | varchar(100) | YES  |     | NULL    |                |
| empresa   | varchar(200) | YES  |     | NULL    |                |
| status    | tinyint(1)   | NO   |     | NULL    |                |
+-----------+--------------+------+-----+---------+----------------+
```

Código 3. 22 Base de datos de la app "pymes"

Los modelos están en el archivo *"project/saas/saas/apps/pymes/models.py"* que se han incluido en el Anexo G.

Para que los modelos se repliquen en la base de datos es necesario definir un servidor de base de datos y crear ahí la base. En este caso, se usó el servidor MySQL presente en el nodo *Controller*. En el Código 3.22 se muestran las tablas con sus campos y atributos que Django creó a partir de los modelos definidos en el archivo *"models.py"*.

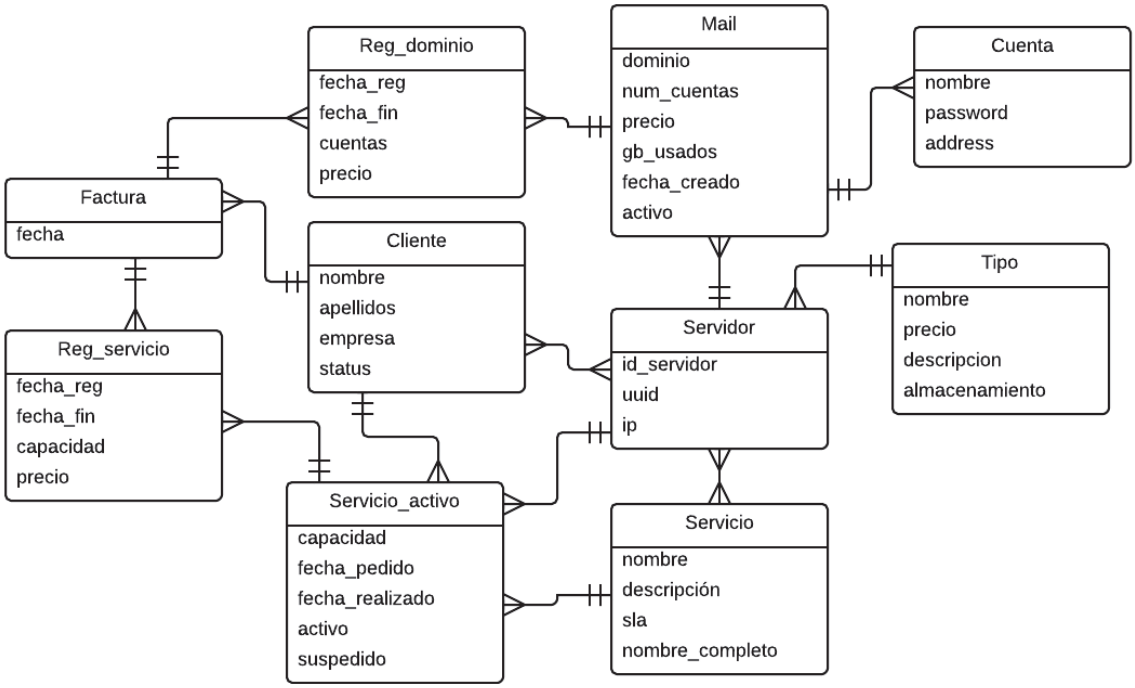


Figura 3. 15 Diagrama Entidad-Relación de la *app* "*pymes*"

En la Figura 3.15 se muestra un diagrama entidad-relación de la base de datos de la *app* "*pymes*".

3.3.3.2 Formularios

El usuario ingresa información en un determinado formulario; esta información es validada para posteriormente enviar esta información a las vistas o modelos.

Los formularios creados están en el archivo *"project/saas/saas/app/pymes/forms.py"* que se adjunta en el Anexo I.

3.3.3.3 URL

En el momento que una página se despliega en el navegar Web, se realiza una consulta al módulo URL para redirigir las consultas del usuario hasta la vista apropiada.

El archivo *"project/saas/saas/apps/pymes/urls.py"* se presenta en el Anexo J.

3.3.3.4 Vistas

La información recibida de una petición a un "URL", se procesa y trata para generar un resultado y devolverlo a la plantilla correspondiente.

El código desarrollado para la funcionalidad de la página web se encuentra en el archivo *"project/saas/saas/apps/pymes/views.py"* que se muestra en el Anexo K.

3.3.3.5 Plantillas

Las Plantillas creadas para cada página web del proyecto están alojadas en la ruta *"project/saas/saas/templates/pymes/"*, y se muestran en el Anexo L.

El sistema de plantillas de Django maneja el concepto de *"*Herencia de Plantillas*"* para reducir la duplicación y redundancia de áreas comunes en una página web con varios archivos HTML. Básicamente la herencia de plantillas permite construir una plantilla básica que contiene todas las partes comunes del sitio y define *"*bloques*"* que las plantillas descendientes pueden sustituir [39].

Se ha creado dos plantillas base que se encuentran en la ruta *"project/saas/saas/templates/"*. La plantilla llamada "*base.html*" se usa como esqueleto para que todas las plantillas hijas muestren la misma cabecera y pie de página. La plantilla *"servicios_base.html"* sirve para mostrar los precios y características de los servicios de correo electrónico, CRM y video conferencia web. El código de estas dos plantillas se muestra en el Anexo M.

3.3.3.6 Funcionalidad

Las vistas son la porción de código que da la funcionalidad a una aplicación web, y las plantillas se encargan de su presentación en el navegador. Es por esto que a continuación se explica las vistas contenidas en el archivo *"views.py"* y la interacción con su respectiva plantilla.

3.3.3.6.1 Contacto

Es básicamente un formulario para que el cliente pueda hacer comentarios, sugerencias o una solicitud al administrador. La plantilla *"contacto.html"* hereda de la plantilla *"base.html"*. Muestra el formulario *"ContactForm"* donde el usuario introduce la información correspondiente y se envía a la vista *"contacto_view"*, aquí se valida y envía como *e-mail* al administrador.

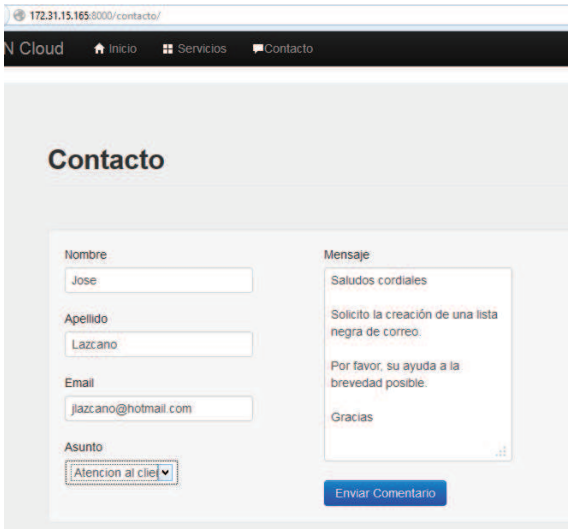


Figura 3. 16 Página con el formulario de contacto

En la Figura 3.16 se muestra la página web de contacto con un ejemplo del formulario lleno, y en la Figura 3.17 se muestra la respuesta al haber enviado el comentario.

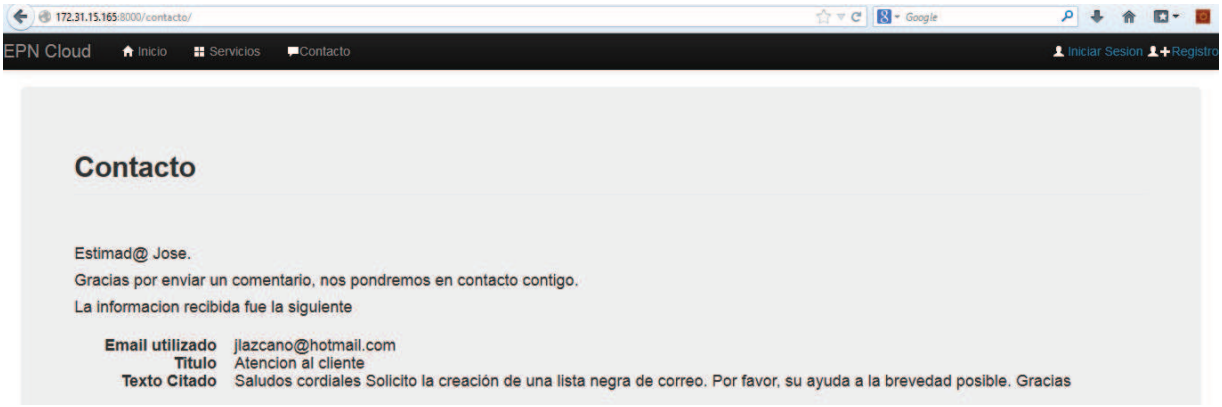


Figura 3. 17 Respuesta al envío de un comentario en la página de contacto

3.3.3.6.2 Login

La vista llamada "*login_view*" muestra el formulario "*LoginForm*", el cual valida los datos ingresados por el usuario y verifica si las credenciales de usuario y clave coinciden; de ser afirmativo, redirige a la página principal, caso contrario, muestra un mensaje de error indicando que el usuario y/o contraseña son inválidos. La página de *Login* se muestra en la Figura 3.18.

Figura 3. 18 Página de *Login*

3.3.3.6.3 Logout

La vista "*logout_view*" simplemente ejecuta la función de Django llamada "*logout*", lo que hace es terminar la sesión del usuario actual. Posteriormente la vista redirige a la página principal.

3.3.3.6.4 Registro

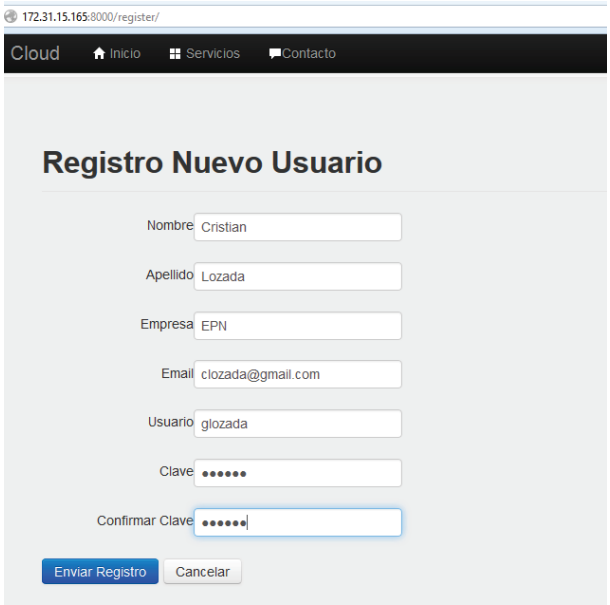


Figura 3. 19 Página de registro

La vista llamada "*register_view*" muestra el formulario llamado "*RegistrationForm*", valida los datos ingresados por el usuario y se crea un nuevo usuario en la base de datos. Finalmente muestra un mensaje exitoso si el formulario no tiene ningún error, caso contrario, se muestra un mensaje pidiendo que se ingrese nuevamente

los campos incorrectos. En la Figura 3.19 se muestra un ejemplo del registro de un nuevo usuario en el sistema.

3.3.3.6.5 Facturación

Es una página donde se muestra la facturación de los servicios contratados con el detalle de la fecha de inicio y fin del servicio. La vista *"facturacion_view"* obtiene el objeto *"factura"* del cliente y devuelve a la plantilla llamada *"facturacion.html"*, una colección de registros de servicios, registros de dominios (en el caso de haber contratado algún dominio adicional), registros de cuentas de correo electrónico (en el caso de haber contratado el servicio de Correo Limitado) y el valor total a pagar hasta la fecha de vencimiento de la factura; la plantilla lo único que hace es ordenar la información en un tabla legible, como se observa en la Figura 3.20.

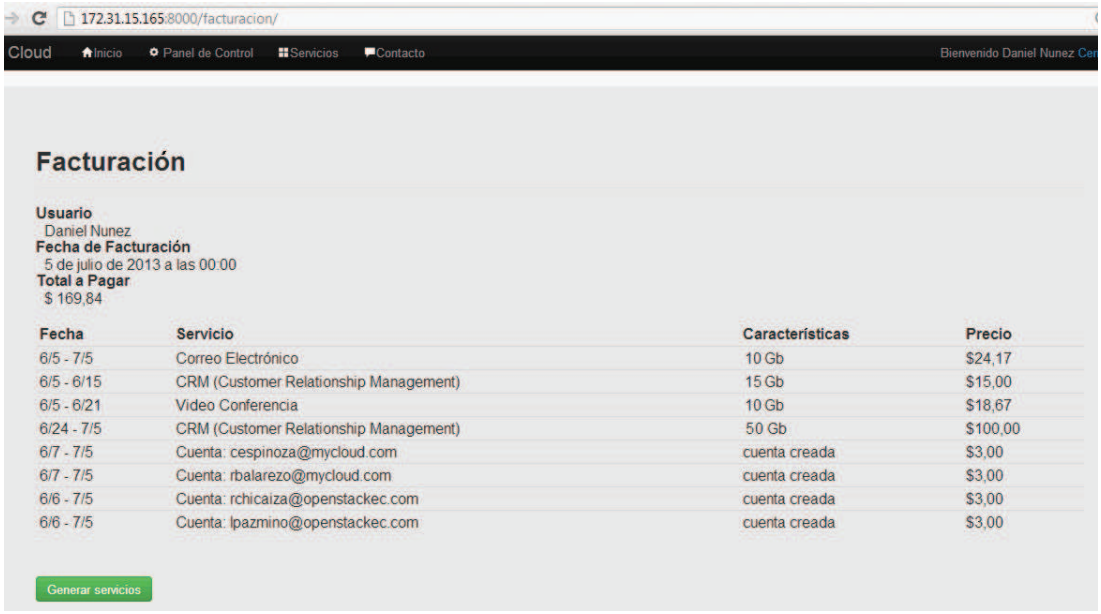


Figura 3. 20 Página de Facturación

3.3.3.6.6 Servicios

La vista *"servicios_view"* simplemente realiza un redireccionamiento a la vista del servicio que el usuario quiere contratar. La plantilla llamada *"servicios.html"* muestra los servicios que están contenidos en la base de datos y los ordena de forma legible para el usuario, como se observa en la Figura 3.21.

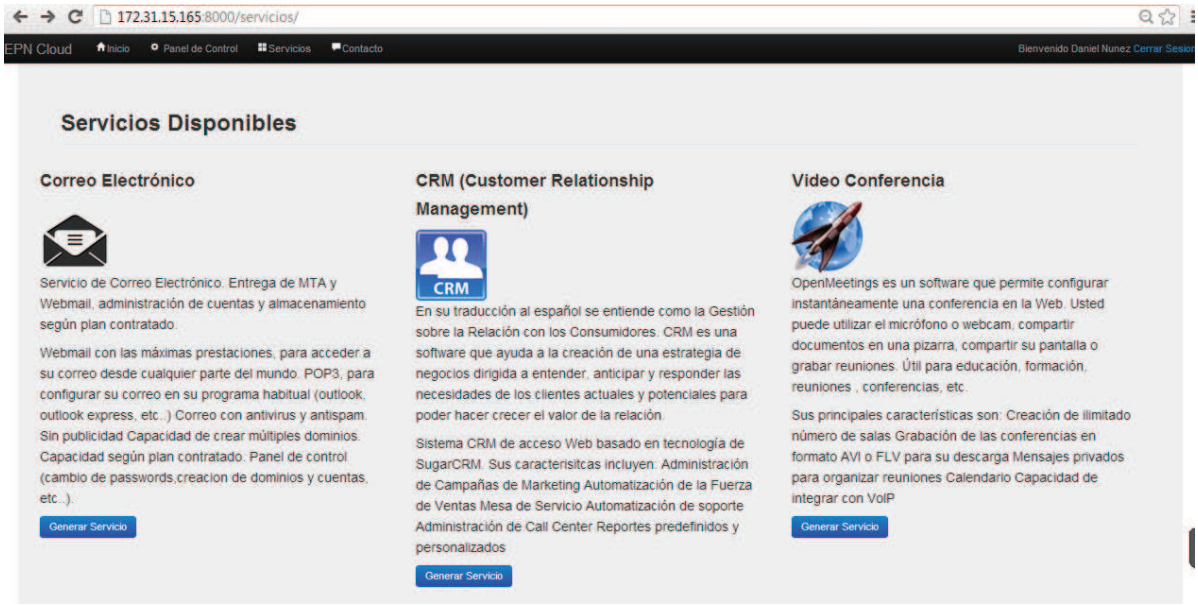


Figura 3. 21 Página de servicios

3.3.3.6.7 Correo Electrónico

El cliente puede contratar dos diferentes planes de servicio de correo electrónico. El Plan Ilimitado, permite la creación de cuentas ilimitadas y se factura un valor fijo mensual, mientras que el Plan Limitado factura por la creación de cada cuenta de correo creada.

La Figura 3.22 muestra la plantilla llamada *"correo_electronico.html"*, que permite al cliente escoger entre los dos tipos de planes del servicio de correo electrónico.

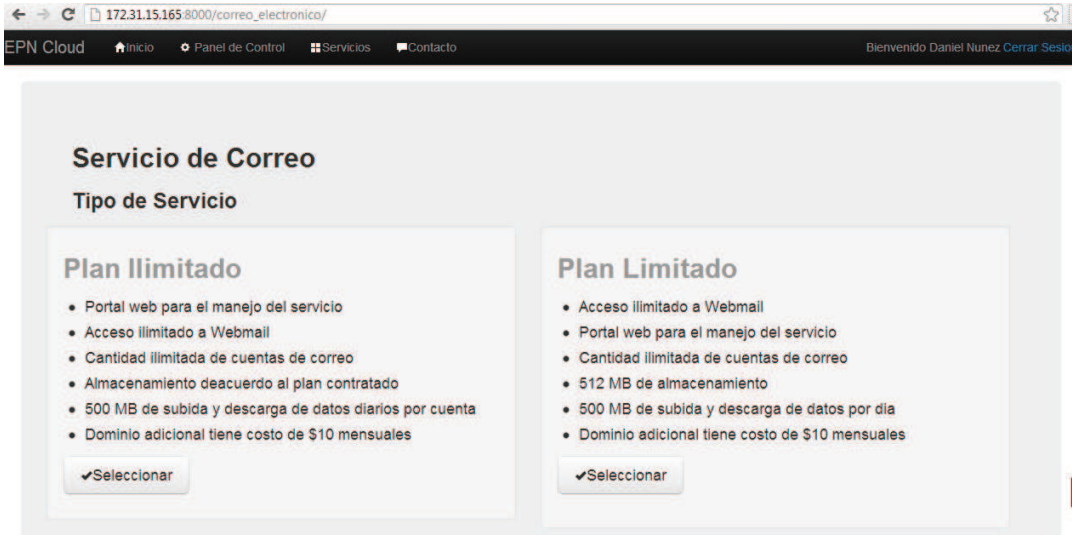


Figura 3. 22 Página con los tipos de servicios de Correo Electrónico

Si el cliente selecciona el servicio de correo en el Plan Ilimitado, se despliega la plantilla llamada *"mail.html"*, como se observa en la Figura 3.23. La vista llamada "*mail_view*" se ejecuta en el momento que el usuario haya seleccionado el plan que desea contratar.

La vista primeramente verifica si el usuario tiene el servicio de CRM activo; en caso de ser afirmativo, el usuario elije un plan y se añade la capacidad solicitada al servidor que tiene el servicio de CRM, ejecutando el *script "resizevm2.py"*. A continuación se envía por *e-mail* al administrador una solicitud para que acceda al servidor y active manualmente en el CLI de Linux los servicios del correo electrónico. Finalmente se crea el registro de facturación correspondiente.

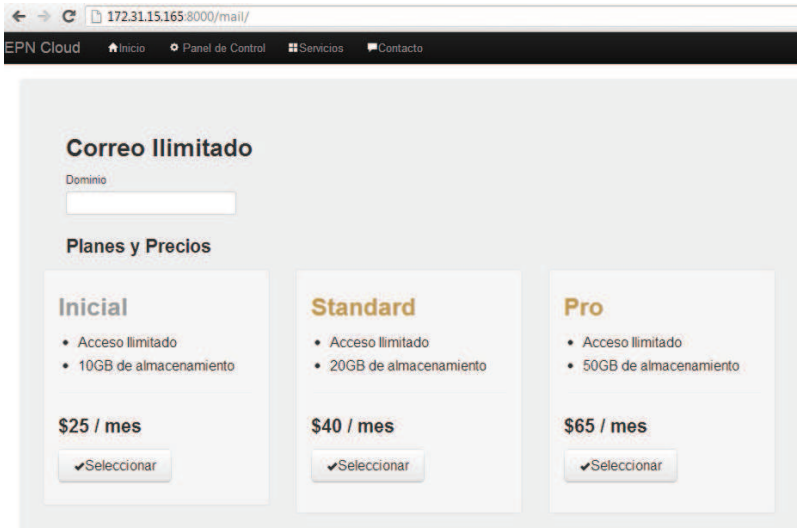


Figura 3. 23 Página del Servicio de Plan Ilimitado

En caso de que el usuario no tenga el servicio de CRM activo, se ejecuta el *script "newserver.py"* con las características del plan seleccionado. El ID del nuevo servidor y su dirección IP pública (extraída con el *script "get_uuid.py"*) se asocian al cliente en la base de datos y por último, se crea el respectivo registro de facturación.

En el caso de que el cliente escoja el servicio de correo tipo Limitado, se despliega la plantilla llamada *"nueva_cuenta.html"* que ejecuta la vista *"nueva_cuenta_view"*, como se observa en la Figura 3.24.

Figura 3. 24 Nueva cuenta en el servicio Plan Limitado

La vista asocia la cuenta con el dominio, ingresados en el formulario *"NewUserForm"*. Se verifica la existencia del dominio en la base de datos, después se asocia al usuario y finalmente se crea el registro de facturación correspondiente. Todas las cuentas que los clientes generen en el servicio de correo Plan Limitado, se crean en el servidor con dirección IP *"192.168.221.164"*, donde el administrador del sistema se encargará de gestionar los dominios y sus cuentas.

3.3.3.6.8 Configuración del Servicio de Correo Electrónico

Si un cliente tiene contratado el servicio de correo Plan Ilimitado, en la página del panel de control se mostrará un enlace que redirige a la plantilla llamada *"conf_mail.html"*. Aquí se muestra la información de los dominios contratados con sus respectivas cuentas, como se observa en la Figura 3.25.

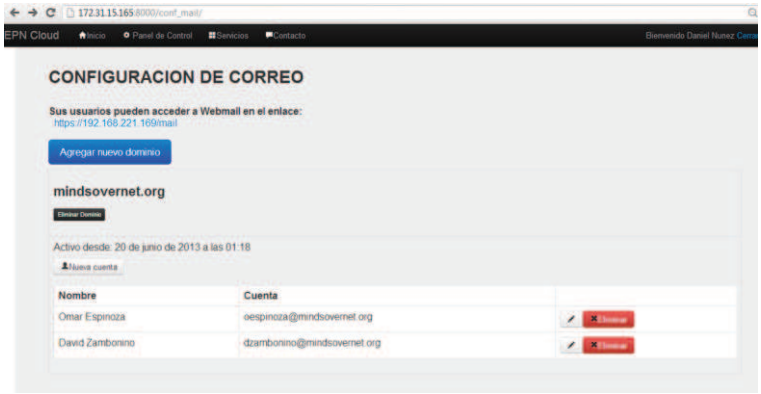


Figura 3. 25 Página de configuración del Servicio de Correo Plan Ilimitado

Cada botón y enlace en esta plantilla ejecuta la vista llamada *“conf_mail_view”*, donde el usuario puede: eliminar dominios, eliminar cuentas de correo, además de redireccionar a las respectivas vistas en el caso de elegir las opciones de: crear dominios, crear cuentas de correo y cambiar sus contraseñas.

En el caso de eliminar un dominio, este cambio se guarda en la base de datos y se modifica el registro de facturación correspondiente, poniendo como fecha final la fecha en que se realizó la eliminación.

3.3.3.6.9 Nuevo dominio

La plantilla llamada “*nuevo_domino.html*” se muestra cuando el usuario quiere agregar un dominio adicional en su servicio de correo electrónico. Esta plantilla tiene el formulario llamado *“DominioForm”* y un botón para contratar el nuevo dominio, como se observa en la Figura 3.26.

Cuando el usuario hace clic sobre el botón “Contratar”, se ejecuta la vista llamada “*nuevo_dominio_view*”, que recibe el nombre del nuevo dominio. Esta información se almacena en la base de datos y se crea un nuevo registro de facturación con la fecha en la que se realiza el registro, y el valor equivalente hasta la fecha de corte de la factura.

Figura 3. 26 Página para la creación de un nuevo dominio

3.3.3.6.10 Nueva cuenta de correo

La plantilla llamada *"new_user.html"* muestra el formulario "*NewUserForm*" y el dominio donde se va crear la nueva cuenta de correo, como se muestra en la Figura 3.27. Los datos ingresados por el usuario se envían la vista llamada "*new_user_view*". Aquí se valida la información ingresada, la cuenta de correo se asocia al dominio y se guarda en el base de datos. Esta información se envía por *e-mail* al administrador para que él cree la nueva cuenta de correo manualmente desde la consola web de iRedMail.

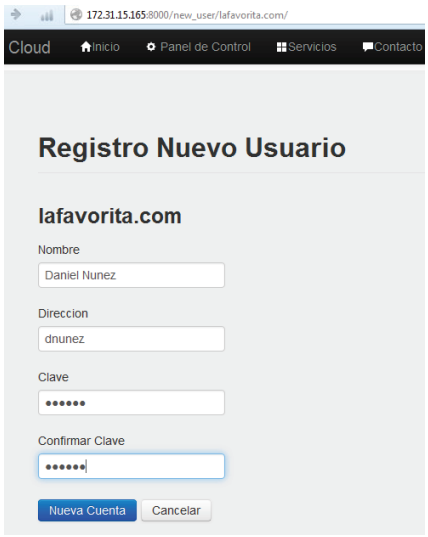


Figura 3. 27 Página para la creación de una cuenta de correo

Cabe mencionar que el software iRedMail no ofrece una API con la cual se pueda desarrollar la integración de sus funciones a la solución del Presente Proyecto, es por este motivo que las solicitudes que generen los clientes a través de la página web, requiere la intervención del administrador del sistema, quién realizará manualmente las configuraciones necesarias en la consola de administración.

3.3.3.6.11 Cambio de contraseña

La plantilla *"contrasena.html"* muestra el formulario llamado "*PassForm*" para cambiar la contraseña de una cuenta de correo electrónico, como se observa en la Figura 3.28. La información del formulario se envía a la vista llamada *"contrasena_view"*, donde se valida y se almacena en la base de datos. Este

cambio se envía por *e-mail* al administrador para que lo haga efectivo manualmente.

Figura 3. 28 Página para el cambio de contraseña de una cuenta de correo

3.3.3.6.12 CRM

La plantilla llamada *"crm.html"* ejecuta la vista llamada *"crm_view"* en el momento que el usuario haya seleccionado el plan que desea contratar, como se observa en la Figura 3.29.

Primeramente la vista verifica si el usuario tiene el servicio de correo electrónico activo; en caso de ser afirmativo, el usuario elije un plan y se añade la capacidad de disco solicitada al servidor que tiene el servicio de correo electrónico, ejecutando el *script "resizevm2.py"*. A continuación se envía por *e-mail* al administrador una solicitud para que acceda al servidor y active manualmente en el CLI de Linux los servicios del CRM. Finalmente se crea el registro de facturación correspondiente.

Figura 3. 29 Página del Servicio de CRM

En caso de que el usuario no tenga el servicio de correo electrónico activo, se ejecuta el *script "newserver.py"* con las características del plan seleccionado. El ID del nuevo servidor y su dirección IP pública (extraída con el *script "get_uuid.py"*) se asocian al cliente en la base de datos y por último, se crea el respectivo registro de facturación.

3.3.3.6.13 Video Conferencia Web

La plantilla llamada *"op.html"* ejecuta la vista llamada *"op_view"* en el momento que el usuario haya seleccionado el plan que desea contratar, como se observa en la Figura 3.30.

Figura 3. 30 Página de creación de servicio de video conferencia web

La vista crea una nueva máquina virtual ejecutando el *script "newserver.py"* con las características del plan seleccionado. El ID de la nueva máquina virtual y su dirección IP pública (extraída con el *script "get_uuid.py"*) se asocian al cliente en la base de datos. A continuación se crea una factura con la fecha de corte 30 días posterior a la fecha actual (en el caso de no tener creada una factura) y posteriormente se crea un registro de facturación con la fecha de creación del servicio y el valor equivalente hasta la fecha de corte de la factura.

3.3.3.6.14 Redimensionamiento

El usuario solicita la ampliación o reducción de la capacidad de almacenamiento del servicio contratado ingresando a la plantilla llamada *"resize.html"*, como se observa en la Figura 3.31, que a su vez ejecuta la vista llamada "*resize_view*".

El usuario elige un plan y se crea un nuevo registro de facturación, al mismo tiempo se ejecuta el *script "resizevm2.py"* con las características del plan seleccionado. Al registro antiguo se modifica la fecha final y se calcula el valor equivalente desde la fecha de contratación hasta la fecha de modificación.

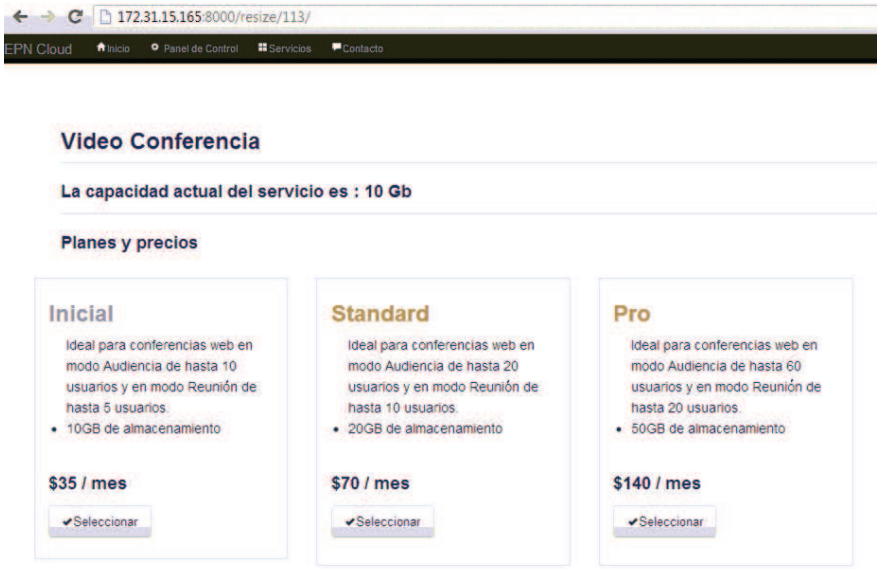


Figura 3. 31 Página para ampliar o reducir la capacidad de almacenamiento de un servicio

3.3.3.6.15 Panel de control

La plantilla llamada *"panel.html"* muestra los servicios contratados por el cliente, con el acceso a sus respectivas funciones de control, como se observa en la Figura 3.32.

Cada uno de los servicios listados tiene la opción de suspender, activar y cancelar. Cualquiera que sea la opción escogida por el usuario, ejecuta la vista llamada *"panel_view",* la cual primeramente verifica si el servicio es correo electrónico o CRM. Si los dos servicios están en la misma máquina virtual, se procede a enviar un *e-mail* al administrador para que por CLI de Linux, termine o

active los procesos correspondientes, y a continuación se genere un registro de facturación.

Si solamente un servicio está activo en el servidor; la opción “suspender” ejecuta el *script “set_dbservidor.py”* con la opción “*suspend*” para poner al servidor en un estado de suspensión donde se libera recursos de RAM y procesamiento. La opción “activar” ejecuta el *script “set_dbservidor.py”* con la opción “*active*” para poner al servidor en estado activo y vuelva a su funcionamiento normal. La opción cancelar ejecuta el *script “delete_server.py”* para eliminar definitivamente el servidor y liberar todos los recursos que estaba usando. Finalmente se genera un registro de facturación con las fechas correspondientes y el valor a pagar.

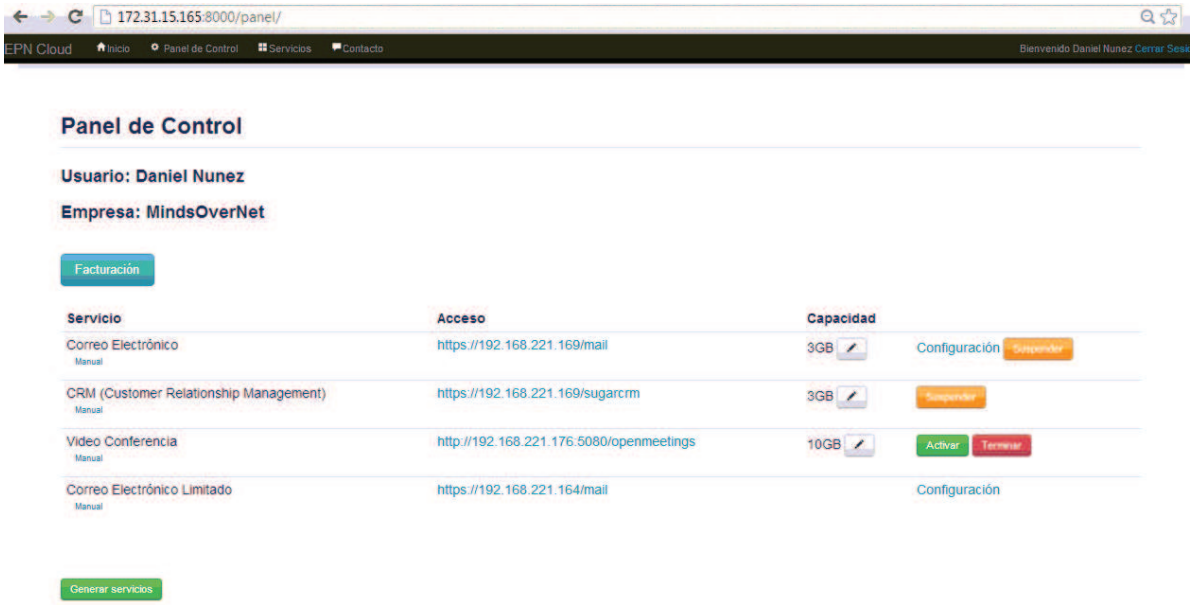


Figura 3. 32 Página de Panel de Control

3.3.4 PRUEBAS

La implementación de la interfaz web constituye el bloque final en la construcción del Prototipo de Proveedor de SaaS. En el siguiente capítulo se detalla la interacción del usuario con el sistema desarrollado, comprobando de esta manera el correcto funcionamiento de la interfaz web.

CAPÍTULO 4
INTERACCIÓN DEL USUARIO CON LAS APLICACIONES PROVISTAS POR EL PROTOTIPO DE PROVEEDOR DE SAAS

A continuación se muestra el uso del sistema desde el punto de vista de usuarios normales y también del administrador, constatando los recursos consumidos por las máquinas virtuales y la respuesta del administrador ante las solicitudes de los usuarios. Posteriormente se estudia el costo de producción de cada uno de los servicios y el precio de venta al público para que sean competitivos en el mercado actual, además de que la solución propuesta genere rentabilidad a mediano plazo.

4.1 DEFINICIÓN DE SERVICIOS

Un requerimiento del Presente Proyecto es proveer los servicios de correo electrónico, CRM y video conferencia web. Pero para proveer la flexibilidad al cliente a la hora de contratar un servicio, se plantea varios planes que ofrecen diferentes características.

4.1.1 CORREO ELECTRÓNICO

El servicio de correo electrónico consiste en la provisión de un medio para crear cuentas de correo electrónico y un portal web para el envío y recepción de *e-mails*. En el mercado existen varias empresas que ofrecen un servicio similar. Estas ofrecen la creación de cuentas ilimitadas con una capacidad de almacenamiento definida por cada cuenta de correo. La Tabla 4.1 muestra varias empresas que ofrecen el servicio de Correo Electrónico, se especifica el precio por cuenta de correo y su capacidad de almacenamiento.

Empresa	Precio	Capacidad de Almacenamiento por cuenta
Rackspace [60]	$2	25GB
Zoho [61]	$2.5	10GB
Gmail [62]	$5	25GB
GoDaddy [63]	$3	1GB

Tabla 4. 1 Empresas que ofrecen servicios de correo electrónico

Tomando estos precios como referencia, se plantea dos tipos de servicio de correo electrónico, donde los precios por cuenta de correo van desde $0.43 a $3.

Nota: La moneda indicada es el dólar americano (USD o $).

4.1.1.1 Servicio de Correo Electrónico Plan Ilimitado

Por un único pago mensual el cliente recibe:

- Portal de administración donde se puede suspender, activar o terminar el servicio.
- Acceso a webmail (Ej.: https://192.168.221.114/webmail), donde los usuarios podrán ingresar para enviar, recibir y administrar correos electrónicos.
- Creación ilimitada de cuentas de correo electrónico.
- Almacenamiento según plan contratado.
- 500MB de almacenamiento por cuenta (posibilidad de ampliar sin costo adicional).
- 500MB máximo de subida y descarga por día por cuenta.
- El costo de un dominio adicional es de $10 mensuales.

El cliente puede elegir entre las siguientes opciones:

4.1.1.1.1 Correo Electrónico Ilimitado Inicial

Este plan es ideal para la creación de un dominio de una empresa pequeña con un máximo de 20 cuentas de correo. El cliente recibe 10GB de almacenamiento y su precio es $25.

4.1.1.1.2 Correo Electrónico Ilimitado Standard

Plan ideal para la creación de un dominio de una PYME con un máximo de 50 cuentas de correo. Este plan también es conveniente para la creación de 2 o 3 dominios de empresas pequeñas con un máximo de 16 cuentas por dominio. El cliente recibe 20GB de almacenamiento y su precio es $40.

4.1.1.1.3 Correo Electrónico Ilimitado Pro

Plan ideal para la creación de un dominio de una empresa mediana con un máximo de 150 cuentas de correo. Este plan también es conveniente para la creación de 7 u 8 dominios de empresas pequeñas con un máximo de 15 cuentas por dominio. El cliente recibe 50GB de almacenamiento y su precio es $65.

4.1.1.2 Servicio de Correo Electrónico Plan Limitado

Por un único pago mensual de $3, el cliente recibe:

- Acceso a webmail (Ej.: https://192.168.221.114/webmail), donde el usuario podrán ingresar para enviar, recibir y administrar correos electrónicos.
- Portal de administración, donde se puede cambiar contraseñas, crear y eliminar cuentas.
- 500MB de almacenamiento.
- 500MB de subida y descarga por día.

4.1.2 SERVICIO DE CRM

El servicio de CRM consiste en la provisión del acceso a una instancia del software SugarCRM.

La empresa SugarCRM [64] ofrece la versión profesional de su software en modalidad SaaS por $35 por usuario al mes, con capacidad de almacenamiento de 15GB. Mientras que Vtiger CRM [65] también ofrece su servicio en modalidad SaaS por un precio de $12 por usuario al mes, con capacidad de almacenamiento de 5GB.

Tomando estos valores como referencia, se asignó el precio de $15 por usuario al mes con capacidad de almacenamiento de 5GB.

Por un único pago mensual, el cliente recibe:

- Portal de administración donde se puede suspender, activar o terminar el servicio.

- Acceso al portal web de una instancia de SugarCRM, donde los usuarios podrán hacer uso de todas las funcionalidades que el software SugarCRM ofrece (Ej.: https://192.168.221.114/sugarcrm).
- Almacenamiento según plan contratado.
- 5GB de almacenamiento estimado por usuario.
- Creación ilimitada de usuarios.

Se definen 3 planes para la contratación del servicio de CRM, éstas son:

4.1.2.1 CRM Inicial

Ideal para una pequeña empresa con 3 personas en el departamento de ventas. El precio es de $45. El cliente además recibe:

- 30GB por mes para subida y descarga de información.
- 15GB de almacenamiento.

4.1.2.2 CRM *Standard*

Ideal para una PYME con 10 personas en el departamento de ventas. El precio es de $150. El cliente recibe además:

- 100GB por mes para subida y descarga de información.
- 50GB de almacenamiento.

4.1.2.3 CRM Pro

Ideal para una empresa mediana o grande con 20 personas en el departamento de ventas. El precio es de $300. El cliente además recibe:

- 200GB por mes para para subida y descarga de información.
- 100GB de almacenamiento.

4.1.3 SERVICIO DE VIDEO CONFERENCIA WEB

El servicio de video conferencia web consiste en la provisión de una instancia del software OpenMeetings. Puede ser utilizado para presentaciones, formación en línea, conferencias web, pizarra de dibujo, edición colaborativa de documentos, etc.

Especialmente en este servicio, el precio del enlace a Internet encarece el costo de producción, por lo que el precio definido no se compara con el precio que ofrece un proveedor comercial. La empresa Gigapros [66] ofrece el servicio de *hosting* de OpenMeetings por un precio mensual de $59.95 con capacidad de 50GB de almacenamiento y tráfico ilimitado.

Por un único pago mensual, el cliente recibe:

- Portal de administración, donde se puede suspender, activar o terminar el servicio.
- Acceso a portal web de OpenMeetings, donde los usuarios podrán hacer uso de todas las funcionalidades que el software OpenMeetings dispone (Ej.: http://192.168.221.115/openmeetings).
- Almacenamiento según plan contratado.
- Capacidad ilimitada de creación de conferencias en modo Audiencia[64] y modo Reunión[65]

Se definen 3 planes para la contratación del servicio de Video Conferencia, éstas son:

4.1.3.1 Video Conferencia Inicial

Ideal para conferencias web en modo Audiencia de hasta 10 usuarios y en modo Reunión de hasta 5 usuarios. El plan tiene 10GB de almacenamiento para creación y subida de información. El precio es de $35.

4.1.3.2 Video Conferencia *Standard*

Ideal para conferencias web en modo Audiencia de hasta 20 usuarios y en modo Reunión de hasta 10 usuarios. El plan tiene 20GB de almacenamiento para creación y subida de información. El precio es de $70.

---

[64] Modo Audiencia se denomina a una conferencia web donde un solo usuario emite su señal de video y los otros usuarios reciben y atienden esa señal.

[65] Modo Reunión se denomina a una conferencia web donde todos los participantes pueden observar la trasmisión de video de cada participante.

4.1.3.3 Video Conferencia Pro

Ideal para conferencias web en modo Audiencia de hasta 60 usuarios y en modo Reunión de hasta 20 usuarios, con 50GB de almacenamiento para creación y subida de información. El precio es de $140.

Cabe mencionar que el control del tráfico de subida y descarga es un parámetro fundamental en la oferta de servicios de acceso remoto, como lo son los servicios antes mencionados, ya que generan costos de uso del enlace de Internet que deben ser cubiertos por el proveedor de SaaS. Este control queda fuera del alcance del Presente Proyecto, pero se recomienda implementar una solución para el monitoreo y control del consumo de recursos de red de las máquinas virtuales. OpenStack ofrece la función de monitoreo y medición de consumo de recursos con el proyecto Ceilometer [67], pero este no se encontraba disponible en la versión Folsom, con la cual se implementó toda la infraestructura.

4.2 PRUEBAS DE USUARIO

El Presente Proyecto fue diseñado para proveer servicios de correo electrónico, CRM y video conferencia web, por medio de una página web y en modo pago por consumo. Para probar la funcionalidad del prototipo, se ha diseñado un escenario donde empresas ficticias hacen uso de los servicios y manipulan la infraestructura de acuerdo a sus necesidades.

4.2.1 EMPRESA A

La empresa MindsOverNet es un emprendimiento de reciente constitución que se dedica a los servicios de Tecnologías de la Información, en ella trabajan apenas 4 personas que cubren las tareas de ingeniería, ventas y administración. Tiene un gran potencial de crecimiento tanto en ventas como en trabajadores por lo que requieren de herramientas descentralizadas y de bajo costo para soportar sus operaciones. Además tienen una constante comunicación con sus proveedores en el exterior y brindan capacitación a través de internet.

El usuario “Daniel Núñez” ingresa a la página principal del proveedor de SaaS y registra la empresa “MindsOverNet” como se muestra en la Figura 4.1.

Figura 4. 1 Registro del usuario "Daniel Núñez" y la empresa "MindsOverNet"

A continuación, el usuario decide contratar el Plan Ilimitado Inicial del servicio de correo electrónico. Para ello se dirige a la pestaña "*Servicios*", selecciona el servicio de correo electrónico y posteriormente el Plan Ilimitado Inicial como se observa en la Figura 4.2.

Figura 4. 2 Selección de Plan Ilimitado Inicial del servicio de correo para la empresa MindsOverNet

Seleccionando la pestaña "*Panel de Control*" se puede observar que efectivamente el servicio contratado ya está disponible para su uso. Como se observa en la Figura 4.3.

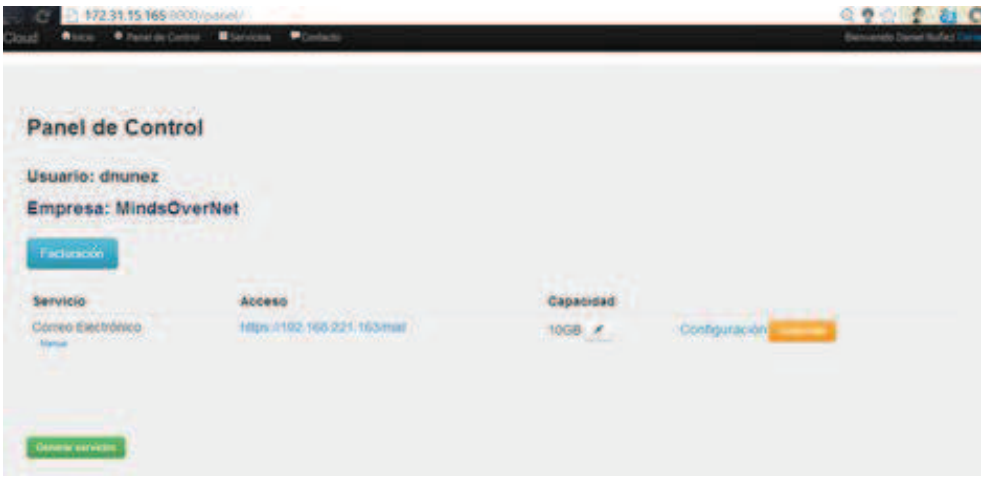


Figura 4. 3 Panel de Control de la empresa MindsOverNet para el servicio de correo electrónico

El usuario accede a la configuración del servicio de correo electrónico haciendo clic en el enlace "*Configuración*" para agregar 2 cuentas en el dominio "mindsovernet.com". Todos estos cambios se ven reflejados inmediatamente en la página de configuración de Correo Electrónico Ilimitado, como se observa en la Figura 4.4.

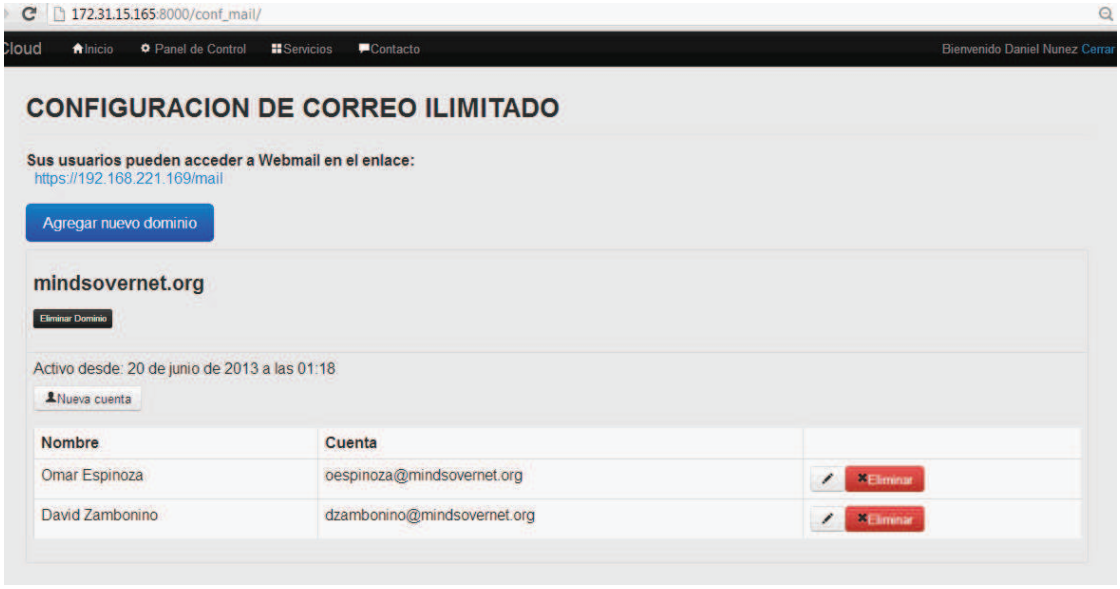


Figura 4. 4 Configuración de Correo Electrónico Ilimitado de la empresa "MindsOverNet"

Con cada una de estas solicitudes, se envía un *e-mail* al administrador del sistema indicando que se realice los cambios sobre el servidor correspondiente, como se observa en la Figura 4.5.

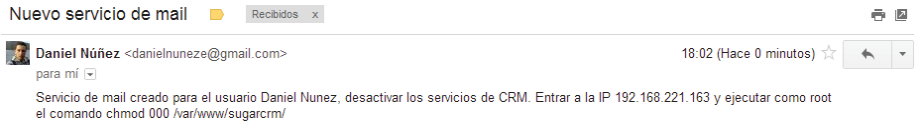


Figura 4. 5 *E-mail* enviado por el sistema al administrador

El administrador accede al servidor indicado en el *e-mail* y ejecuta los comandos necesarios. En la línea 4 del Código 4.1 se muestra el comando ejecutado en el servidor que aloja el servicio de Correo Electrónico de la empresa “MindsOverNet” para desactivar el acceso web de SugarCRM.

Cabe recordar que al crear un servicio de Correo Electrónico o CRM, siempre se genera una máquina virtual donde están instalados y activos ambos servicios; al haber contratado uno de ellos, el acceso web del otro servicio debe desactivarse manualmente por parte del administrador.

```
1 root@controller:~# ssh ubuntu@192.168.221.163
2 ubuntu@mailcrm:~$ su root
3 Password:
4 root@mailcrm:~# chmod 000 /var/www/sugarcrm/
5 root@mailcrm:~# df -h
6 Filesystem      Size  Used Avail Use% Mounted on
7 /dev/vda1       9.9G  1.8G  7.7G  19% /
8 udev            242M  8.0K  242M   1% /dev
9 tmpfs            99M  244K   99M   1% /run
```

Código 4. 1 Comandos para desactivar el acceso a SugarCRM

En la línea 7, se muestra que el servidor tiene una partición de 10GB de almacenamiento, lo que corresponde a la capacidad solicitada.

Posteriormente el usuario decide contratar el servicio de correo con el Plan Limitado para la creación de 2 dominios con 2 cuentas en cada dominio. Para ello accede a la pestaña “*Servicios*” y escoge el servicio de correo electrónico, se despliega una página con los tipos de servicios y escoge el Plan Limitado, y se presentará un formulario para crear una cuenta de correo. En este caso se crea la

cuenta “cespinoza” en el dominio “mycloud.com” como se observa en la Figura 4.6.

De igual manera, se envía un *e-mail* al administrador indicando la creación de la cuenta en el servidor correspondiente.

Figura 4. 6 Creación de nueva cuenta en el Plan Limitado del Servicio de Correo Electrónico

El administrador ingresa al portal web iRedAdmin para todas las solicitudes de creación de dominios y cuentas de correo. En la Figura 4.7 se muestra las cuentas creadas para el servicio de correo electrónico Plan Ilimitado de la empresa MindsOverNet.

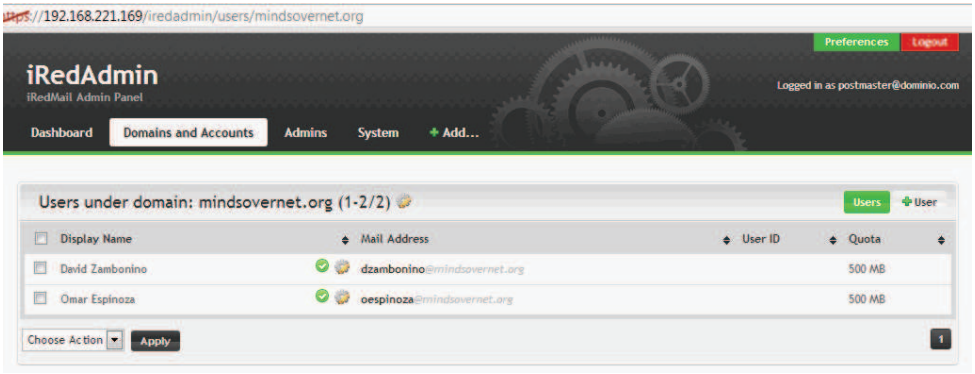


Figura 4. 7 Portal iRedAdmin del servicio de correo electrónico para la empresa "MindsOverNet"

Una vez hechos los cambios en el portal iRedAdmin, los usuarios de las cuentas de correo podrán hacer uso del servicio para enviar y recibir *e-mails*. En la Figura

4.8 se muestra un *e-mail* enviado desde la cuenta del usuario “dzambonino” para el usuario “oespinoza”.

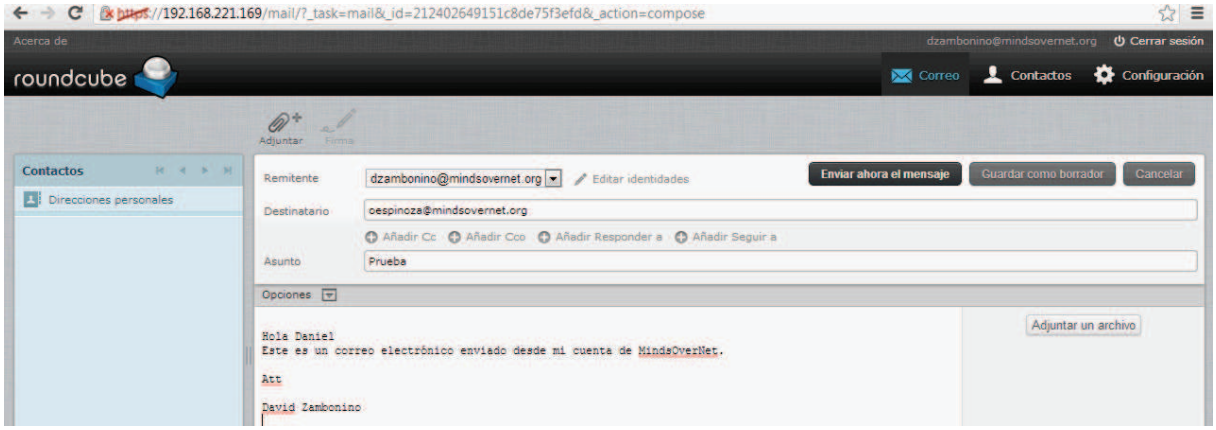


Figura 4. 8 *E-mail* del usuario "dzambonino" al usuario "oespinoza"

En la Figura 4.9 se muestra la bandeja de entrada del usuario “oespinoza” donde se observa que efectivamente se encuentra el *e-mail* enviado por “dzambonino”.

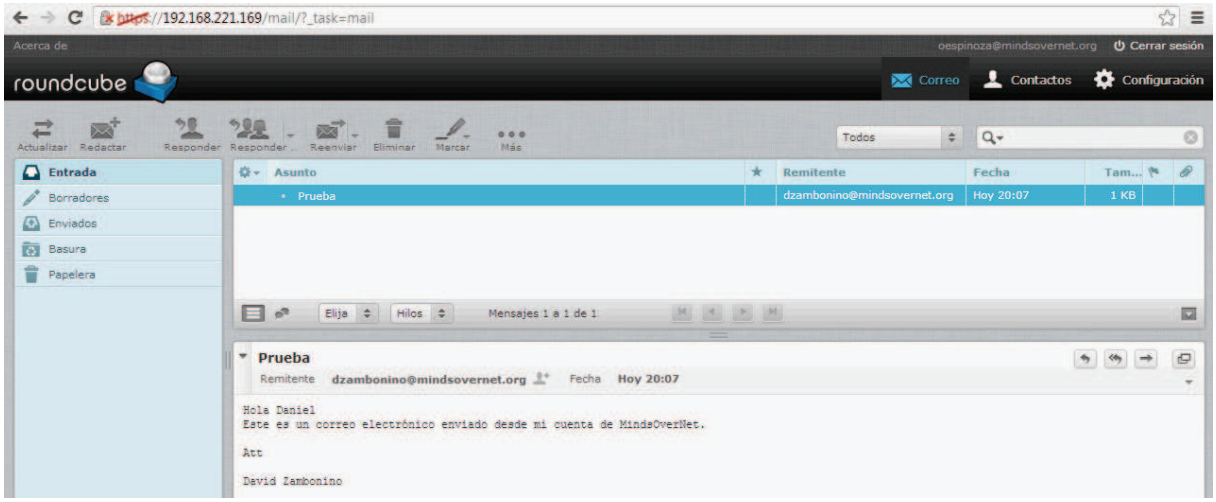


Figura 4. 9 *E-mail* recibido exitosamente del usuario "dzambonino" al usuario "oespinoza"

Cabe mencionar que solo se podrán enviar *e-mails* a cuentas del mismo dominio y no se podrá recibir *e-mails* desde Internet, ya que el dominio debe estar registrado en un servidor DNS y el registro MX[66] debe apuntar a la dirección IP pública de la máquina virtual que tiene el servicio de correo electrónico.

[66] Registro MX: *Mail eXchange record,* es un recurso de DNS que indica como debe ser enrutado un correo electrónico en Internet.

Ésta configuración no se realiza ya que queda fuera del alcance del Presente Proyecto.

A continuación el usuario decide contratar el Plan Inicial del servicio de CRM, como se observa en la Figura 4.10.

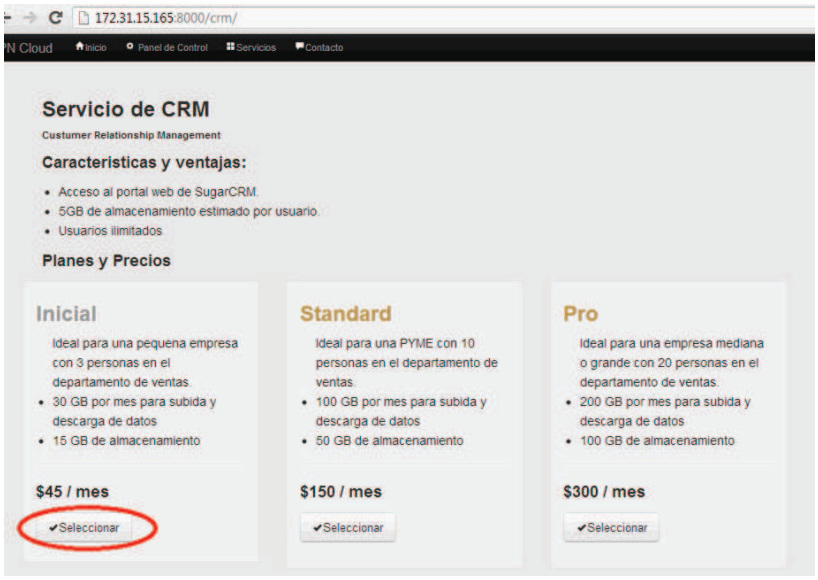


Figura 4. 10 Selección de plan "Inicial" del servicio de CRM

Inmediatamente el servicio de CRM se muestra en el panel de control y se envía un *e-mail* al administrador para que active el acceso al portal web de SugarCRM, como se muestra en la Figura 4.11.

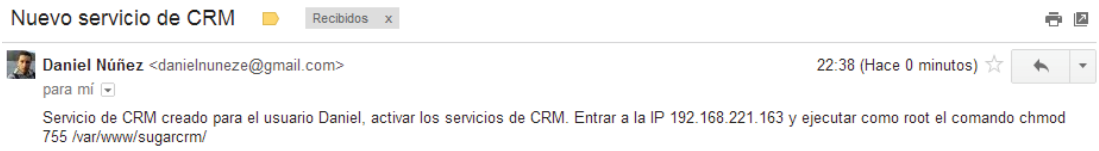


Figura 4. 11 *E-mail* recibido por el administrador para activar el acceso web a SugarCRM

El usuario decide también contratar el Plan Inicial del servicio de video conferencia web. Al elegirlo entre los servicios disponibles, inmediatamente se genera una máquina virtual con la instancia del software OpenMeetings, este quedará plenamente operativo después de aproximadamente 3 minutos, que es el tiempo que demora inicializar todos los servicios requeridos por OpenMeetings en la máquina virtual. El usuario podrá ingresar desde el enlace que se muestra en el

panel de control e inmediatamente podrá crear conferencia o reuniones, como se muestra en la Figura 4.12.

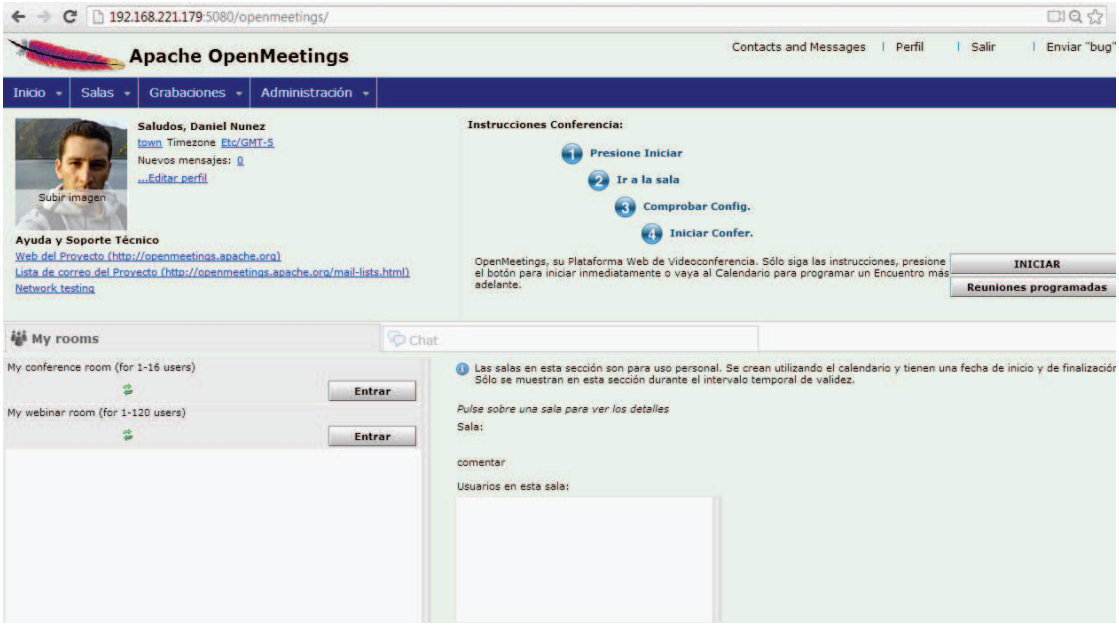


Figura 4. 12 Página de inicio de OpenMeetings del usuario Daniel Núñez

El panel de control de la empresa "MindsOverNet" con todos los servicios contratados y activos se muestra en la Figura 4.13.

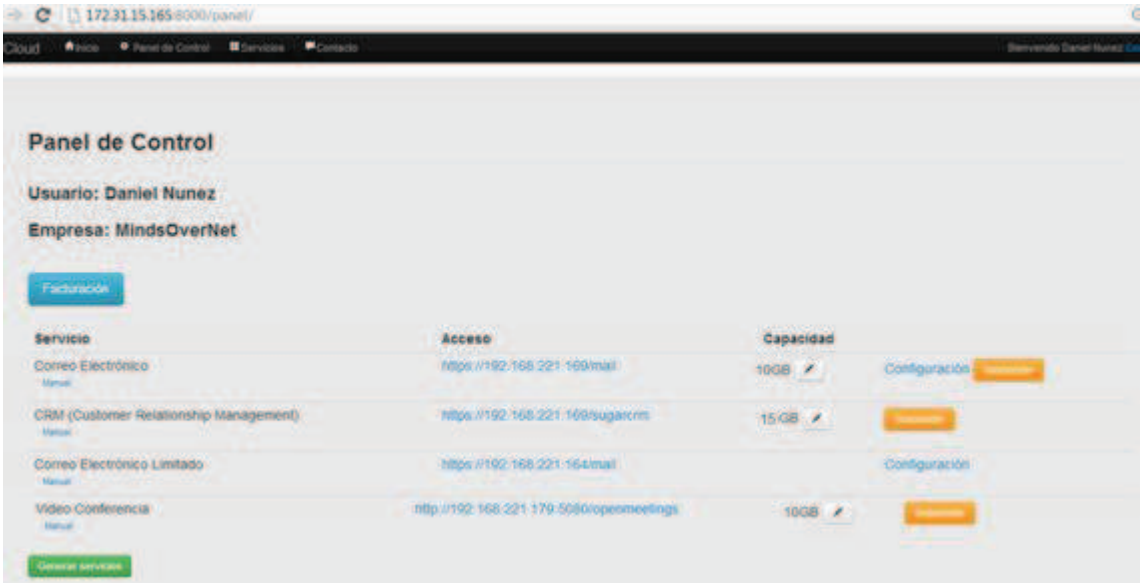


Figura 4. 13 Panel de Control de la empresa "MindsOverNet" con todos los servicios contratados y activos

Al cabo de 10 días de haber contratado el Plan Inicial del servicio de CRM, el usuario amplía su capacidad, haciendo clic sobre el botón con forma lápiz, y escoge el Plan *Standard,* como se muestra en la Figura 4.14.

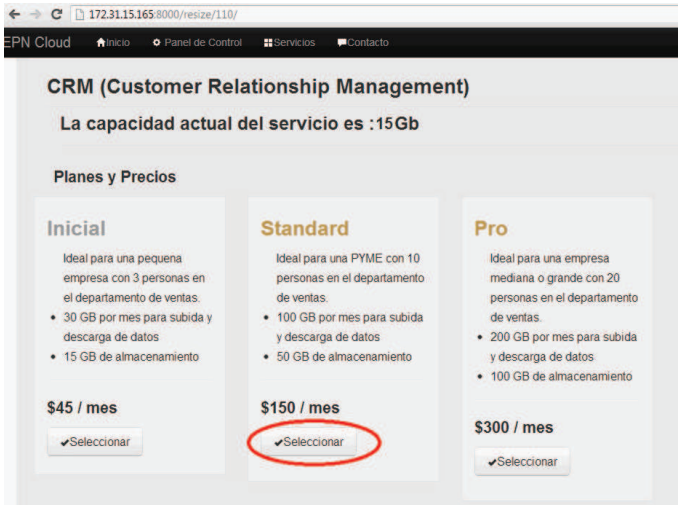


Figura 4. 14 Selección de Plan *Standard* para ampliación del servicio de CRM

Después de 15 días de haber contratado el Plan Inicial del servicio video conferencia web, el usuario decide cancelar el servicio, primeramente suspende la máquina virtual que tiene la dirección IP *"192.168.221.179"*.

Posteriormente en el panel de control hace clic sobre el botón *"Terminar"* para cancelar definitivamente el servicio, como se observa en la Figura 4.15.

Figura 4. 15 Selección del botón "*Terminar*" para cancelar el servicio de video conferencia web

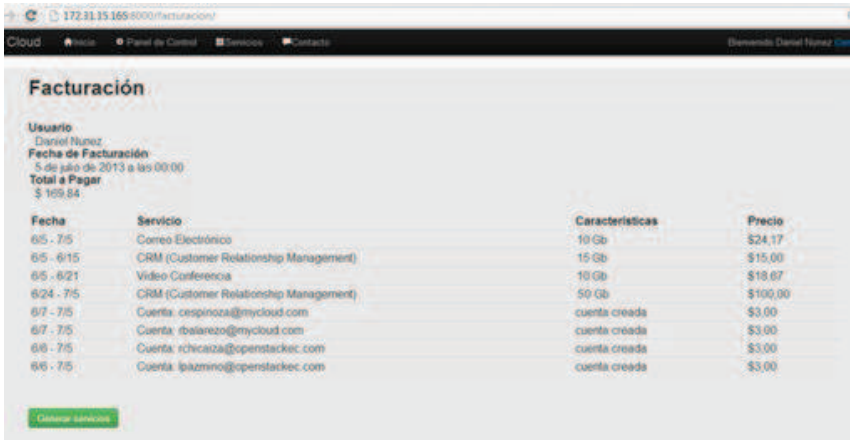


Figura 4. 16 Facturación del usuario "Daniel Núñez"

Con todos los cambios realizados por el usuario sobre sus servicios contratados, en la Figura 4.16 se muestra la facturación correspondiente al mes de uso de los servicios, con el detalle de todos los registros y sus respectivos valores.

4.2.2 EMPRESA B

La empresa “BigBang Comunicaciones” es una empresa de servicios de publicidad web y desarrollo de campañas comunicacionales. En esta empresa laboran 8 personas, pero debido a sus necesidades requieren optimizar el tiempo, por lo que necesitan de soluciones informáticas en la nube.

El usuario “José Luis Valencia” ingresa a la página principal del proveedor de SaaS y registra la empresa “BigBang Comunicaciones”. A continuación, decide contratar el Plan Ilimitado Inicial del servicio de correo electrónico con el dominio “bigbang.com”, y el Plan Inicial del servicio de CRM. Además decide agregar el dominio “radiofrecuenciaec.com”, como se muestra en la Figura 4.17.

Figura 4. 17 Creación de un dominio adicional en el servicio de correo Plan Ilimitado

De la misma manera, el usuario añade el dominio “tvdigitalec.com”. En estos 3 dominios el usuario crea dos cuentas en cada uno de ellos como se observa en la Figura 4.18.

Figura 4. 18 Dominios y cuentas de correo de la empresa "BigBang Comunicaciones"

Diez días después, el usuario decide suspender el servicio de CRM, por lo que se actualiza el registro de facturación con el equivalente al tiempo de uso del servicio y se crea un nuevo registro por el uso del espacio de disco de la máquina virtual suspendida. Al día 15, el usuario decide contratar el Plan Inicial del servicio de video conferencia web y al día 20 activa el servicio de CRM que estaba suspendido. Todos estos cambios se registran en la facturación correspondiente, como se observa en la Figura 4.19.

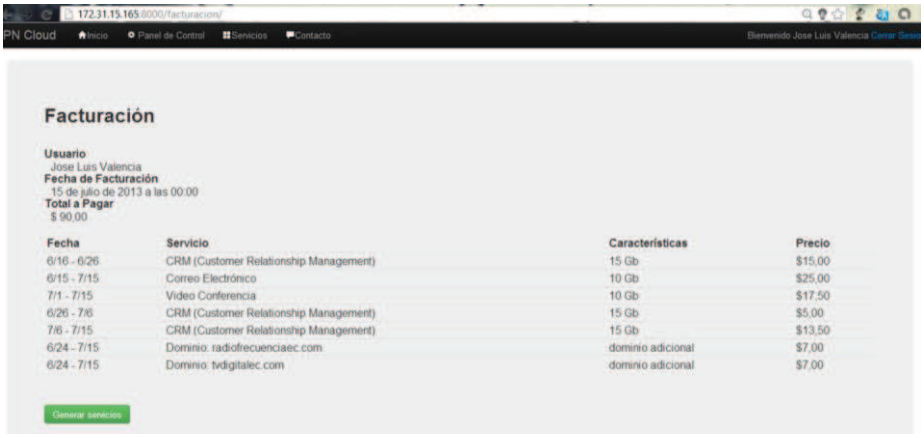


Figura 4. 19 Facturación del usuario "José Valencia"

4.2.3 EMPRESA C

La empresa “Grupo Microsistemas (GMS)” es una empresa con más de 30 años de trayectoria en la provisión de servicios de tecnología en Ecuador. En ella laboran aproximadamente 80 personas distribuidas en las ciudades de Quito, Guayaquil y Cuenca. Debido a que las herramientas que usan actualmente requieren mucho tiempo de mantenimiento, necesitan evaluar soluciones escalables que se acoplen al uso de sus empleados en las tres ciudades.

Para evaluar los servicios que provee el Proveedor de SaaS, el usuario “Edison Guanoluisa” ingresa a la página principal del proveedor de SaaS y registra la empresa “Grupo Microsistemas”; a continuación decide contratar el Plan *Standard* del servicio de video conferencia web.

Días después, el usuario contrata el Plan Inicial del servicio de CRM y el servicio de correo electrónico Plan Ilimitado con el dominio “gms.com.ec”. En este dominio, el usuario crea 3 cuentas de correo como se muestra en la Figura 4.20.

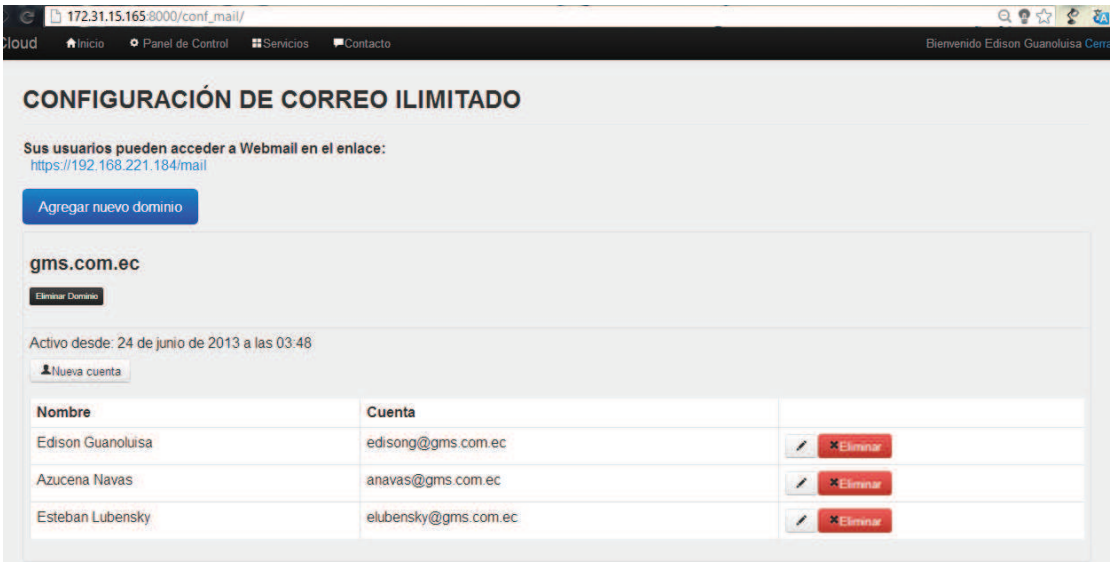


Figura 4. 20 Configuración del servicio de correo de la empresa "Grupo Microsistemas"

Todos los cambios realizados por el usuario “Edison Guanoluisa” se registran en la facturación correspondiente, como se observa en la Figura 4.21.

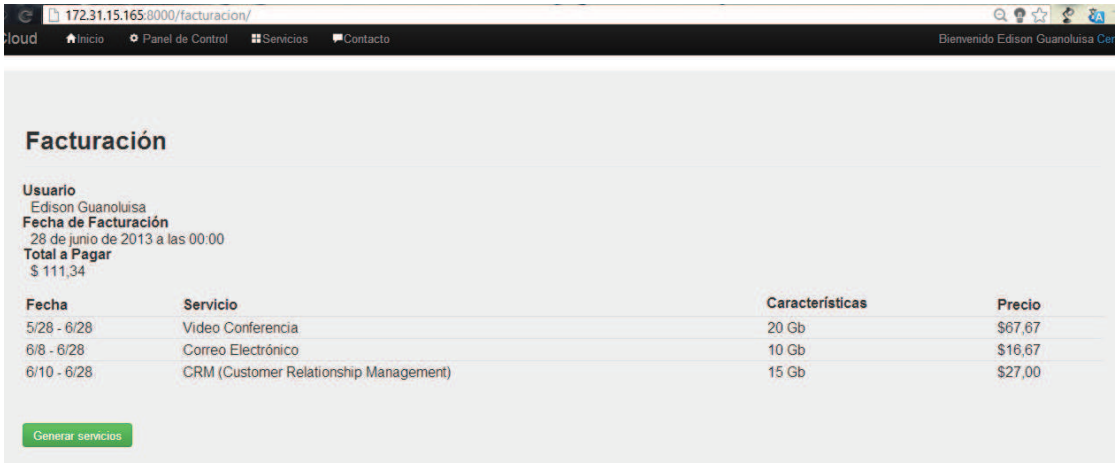


Figura 4. 21 Facturación de la empresa "Grupo Microsistemas"

4.2.4 RESULTADOS DE LAS PRUEBAS DE USUARIO

La interacción de las tres empresas sobre el sistema generó el uso de recursos de procesamiento, memoria RAM, almacenamiento en disco y direcciones IP públicas. Además de las intervenciones del administrador en satisfacer las solicitudes de cada usuario, especialmente en la creación de cuentas de correo y el despliegue adecuado de los servicios de correo electrónico y CRM.

El administrador tiene las siguientes herramientas para controlar, dar mantenimiento al sistema y resolver las solicitudes de los clientes:

4.2.4.1 Interfaz de administración web Horizon

La interfaz Horizon muestra información básica de las máquinas virtuales generadas por la interacción de los usuarios con el sistema. Como se mencionó anteriormente, Horizon es un proyecto bastante inmaduro que aún no proporciona el acceso a todas las funcionalidades de OpenStack, pero para el Presente Proyecto sirve como una interfaz gráfica básica, de fácil acceso, para saber el estado de la infraestructura OpenStack.

El administrador hace uso de esta herramienta para:

- Verificar la correcta creación de las máquinas virtuales.
- Verificar las direcciones IP privadas de las máquinas virtuales.
- Observar el redimensionamiento de las máquinas virtuales.

- Comprobar errores del sistema.
- Generar nuevas reglas en los grupos de seguridad.

En la Figura 4.22 se muestra una captura de la interfaz web Horizon del sistema, donde se muestran todas las máquinas virtuales creadas en las pruebas de usuario.

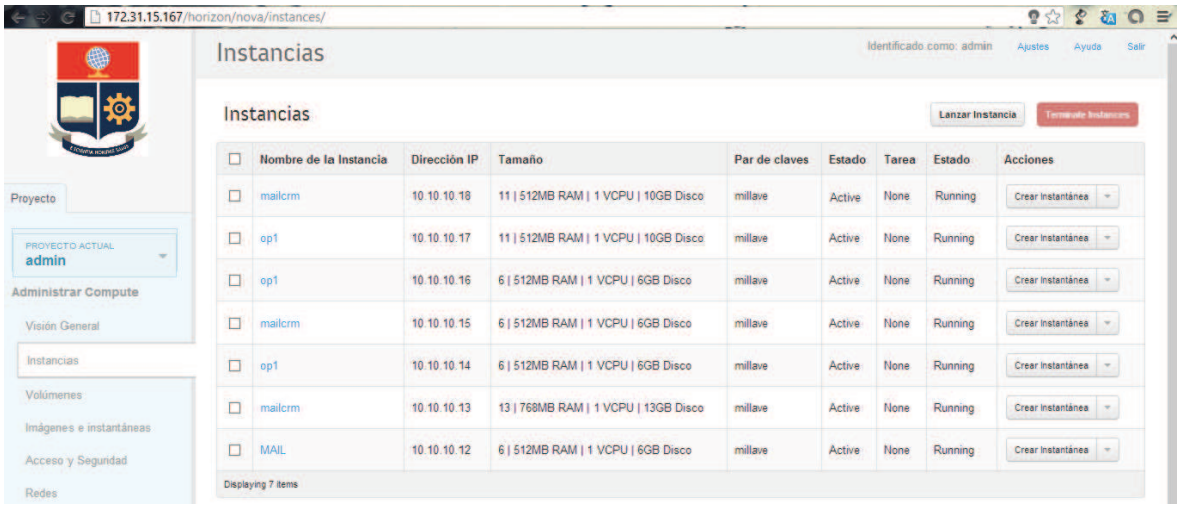


Figura 4. 22 Máquinas virtuales en la interfaz de administración web Horizon

4.2.4.2 Interfaz web de administración de Django

La interfaz web de administración de Django es una herramienta fundamental para la administración de los datos de los clientes, ya que muestra la información almacenada en la base de datos y permite su fácil edición de una manera totalmente transparente para el administrador.

El administrador hace uso de esta herramienta para:

- Insertar información relevante de los servicios.
- Verificar la correcta creación de los registros de facturación.
- Verificar los dominios y las cuentas que se deben crear.
- Comprobar que el estado de las máquinas virtuales corresponda con el estado mostrado en la interfaz web de Horizon.

En el Anexo O se muestran capturas de los registros de la base de datos de la información generada por las pruebas de usuario.

4.2.4.3 Consumo de recursos de infraestructura

A continuación se presentan las principales características del comportamiento de la infraestructura para la provisión de los servicios a los usuarios. Por tal objetivo, se ha utilizado el software de monitoreo llamado "PRTG" [68] para determinar el consumo de recursos de procesamiento, memoria y red en cada uno de los servidores de la infraestructura.

PRTG es una herramienta que tiene gran variedad de métodos para monitorear dispositivos de red, además ofrece una interfaz web amigable para la administración y configuración de los dispositivos a monitorear.

Para la monitorización del uso de los recursos de infraestructura, se instaló un agente SNMP[67] en cada uno de los servidores con las características de la Tabla 4.2.

Comunidad	epn
Locación	Lab Servidores
Contacto	danielnunenze@gmail.com

Tabla 4. 2 Características para la configuración SNMP

En la línea 1 del Código 4.2 se observa la instalación de los paquetes correspondientes al agente SNMP. En el fichero "*/etc/snmp/snmpd.conf*" se ingresa las características descritas en la Tabla 4.2, como se puede observar en las líneas 2 a 5. Además es necesario editar el fichero "*/etc/default/snmpd*" como se ver en la línea 6, para agregar la información de la línea 7. Por último se reinicia el servicio.

```
1 apt-get install snmpd
2 nano /etc/snmp/snmpd.conf
3 rwcommunity epn
4 syslocation "Lab Servidores"
5 syscontact danielnuneze@gmail.com
6 nano /etc/default/snmpd
7 SNMPDOPTS='-Lsd -Lf /dev/null -u snmp -I -smux -p /var/run/snmpd.pid -c
8 /etc/snmp/snmpd.conf'
9 /etc/init.d/snmpd restart
```

Código 4. 2 Configuración SNMP

[67] SNMP: *Simple Network Management Protocol,* es un protocolo de capa aplicación para intercambiar información de gestión entre dispositivos de red. Su objetivo principal es facilitar el monitoreo y gestión de redes de datos.

Se ha instalado el software PRTG en un computador con sistema operativo Windows según el proceso descrito en la documentación oficial [69]. Esta máquina se conectó en la red pública, mediante la cual se puede tener acceso a todos los servidores de la infraestructura.

El Anexo P muestra el proceso de inserción de dispositivos a la herramienta PRTG y la configuración para ingresar sensores al monitoreo.

	Evento	Comienzo	Final
1	Creación de máquina virtual con servicio de Correo Ilimitado Inicial de MindsOverNet	5:05:00 PM	
2	Creación de servicio de CRM de MindsOverNet	5:09:00 PM	
3	Creación de máquina virtual con servicio de Video Conferencia Inicial de MindsOverNet	5:10:00 PM	
4	Paso del plan Inicial al plan standard del servicio de CRM de MindsOverNet	5:12:00 PM	5:18:00 PM
5	Suspención del servicio de Video Conferencia de MindsOverNet	5:20:00 PM	5:21:00 PM
6	Cancelación del servicio de Video Conferencia de MindsOverNet	5:26:00 PM	
7	Creación de máquina virtual con servicio de Correo ilimitado de la BigBang Comunicaciones	5:28:00 PM	5:30:00 PM
8	Creación de servicio de CRM de BigBang Comunicaciones	5:32:00 PM	
9	Creación de máquina virtual con servicio de Video Conferencia Inicial de BigBang Comunicaciones	5:39:00 PM	5:41:00 PM
10	Creación de máquina virtual con servicio de Video Conferencia Inicial de GMS	5:45:00 PM	5:46:30 PM
11	Creación de máquina virtual con servicio de CRM de GMS	5:48:00 PM	5:49:30 PM
12	Creación de servicio de Corre Electrónico de MindsOverNet	5:50:00 PM	5:51:00 PM

Tabla 4. 3 Eventos destacados en las pruebas de usuario

Una vez que el servidor se encuentra configurado con los sensores para el monitoreo del procesamiento de CPU, consumo de memoria RAM y tráfico de red en la interfaces de las servidores, se procedió a realizar las pruebas de usuario que fueron descritas anteriormente.

La Tabla 4.3 muestra los eventos destacables en la generación de la pruebas de usuario junto con las tiempos de comienzo y final en aquellos casos que el evento dure varios minutos.

4.2.4.3.1 Monitoreo del procesamiento de CPU

En la Figura 4.23 se muestra las gráficas generadas por la herramienta PRTG correspondiente a la actividad del procesamiento de CPU de los servidores *Controller*, C2 y *Network*, respectivamente. Además se especifica los eventos relevantes de las pruebas de usuario descritos en la Tabla 4.3.

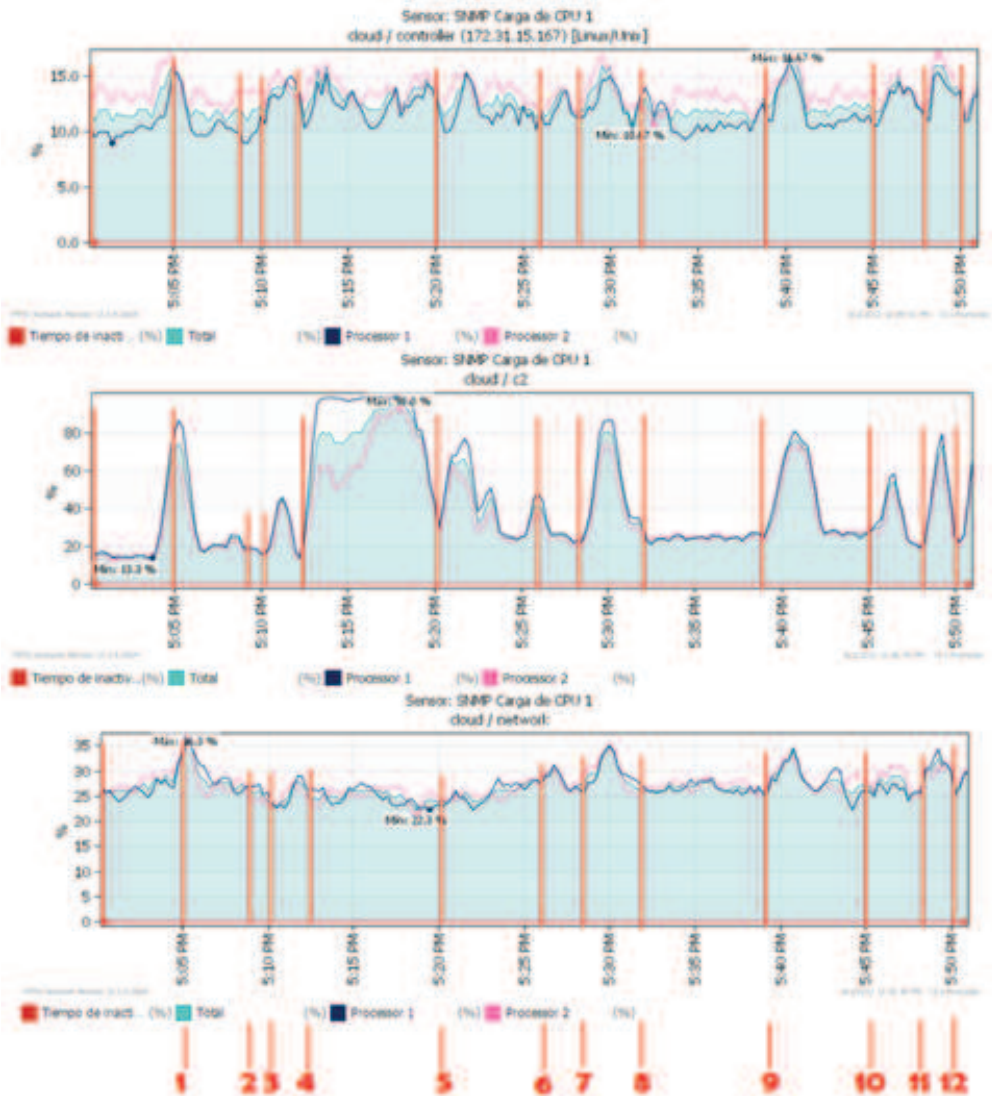


Figura 4. 23 Monitoreo de CPU de los servidores de la infraestructura

Analizando las gráficas se puede observar que existe un aumento del procesamiento en la CPU de cada servidor en el momento que se generan nuevas máquinas virtuales (evento 1, 3, 7, 9, 10 y 11), este aumento es más evidente en el servidor C2, donde se ejecutan los procesos de KVM para crear la máquina virtual con el servicio elegido por el usuario.

El evento 4 genera la mayor cantidad de procesamiento en el servidor C2, debido a que el paso del plan Inicial al plan Standard del servicio de CRM, implica la ampliación del almacenamiento de disco de la máquina virtual donde se ejecuta el

servicio. En este caso, esta operación dura aproximadamente 6 minutos, en los cuales la máquina virtual queda inaccesible.

4.2.4.3.2 Monitoreo de la memoria RAM

En la Figura 4.24 se muestra las gráficas generadas por la herramienta PRTG correspondiente a la memoria RAM disponible en los servidores *Controller*, C2 y *Network*, respectivamente. Además se especifica los eventos relevantes de las pruebas de usuario descritos en la Tabla 4.3.

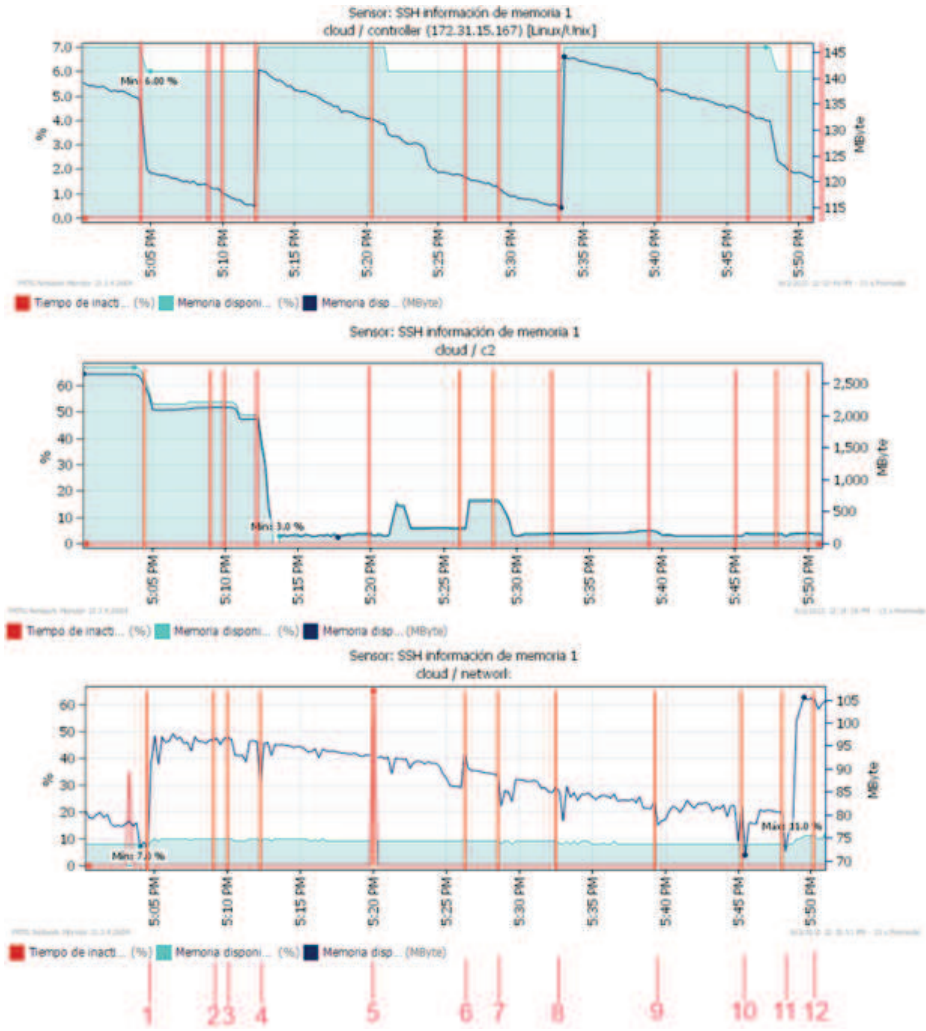


Figura 4. 24 Monitoreo de memoria de los servidores de la infraestructura

Al analizar las gráficas de la Figura 4.24 se observa que el servidor *Controller* se mantiene con poca memoria RAM disponible, pero no presenta variaciones abruptas al momento de generar nuevas máquinas virtuales. Esto se debe a que

este servidor solamente se encarga de generar las tareas que los servidores C2 y *Network* deben ejecutar.

El servidor C2 presenta un pronunciado descenso de la memoria RAM al producirse el evento 4. Esto se debe a que la información almacenada en la máquina virtual pasa a la memoria RAM del servidor C2 para poder hacer el redimensionamiento del disco virtual. El nodo *Network* presenta una reducción progresiva de memoria RAM disponible, es una variación aproximada de 15 MB en un periodo de tiempo aproximado de 50 minutos, lo que evidencia un correcto dimensionamiento de la memoria RAM para las tareas de red de la infraestructura.

4.2.4.3.3 Monitoreo de las interfaces de red

La red de Administración sirve para que el administrador de la infraestructura pueda acceder para realizar tareas de verificación del estado de los servidores, principalmente por medio del protocolo SSH. Por esta red también cruza tráfico de comunicación entre los servidores, generado por Glance, Nova y Quantum.

En la Figura 4.25 se muestra las gráficas generadas por la herramienta PRTG correspondiente a las interfaces de los servidores *Controller*, C2 y *Network* en la red de Administración. Además se especifica los eventos relevantes de las pruebas de usuario descritos en la Tabla 4.3.

Cabe señalar que lo que está sobre la línea “0” corresponde al monitoreo de la velocidad de entrada de paquetes en la interfaz (color celeste), mientras que lo que está bajo la línea 0 corresponde al monitoreo de la velocidad de salida de paquetes de la interfaz (color azul).

Analizando la Figura 4.25, se observa que existe constante envió y recepción de información en todos los servidores, esta información corresponde principalmente al intercambio de mensajes de estado que hace Nova (todos los nodos envían mensajes de estado a Nova-Scheduler) y mensajes para la ejecución de acciones al servicio de encolamiento RabbitMQ.

En la Figura 4.26 se muestra las gráficas generadas por la herramienta PRTG correspondiente a las interfaces de los servidores *Controller*, C2 y *Network* en la red Privada.

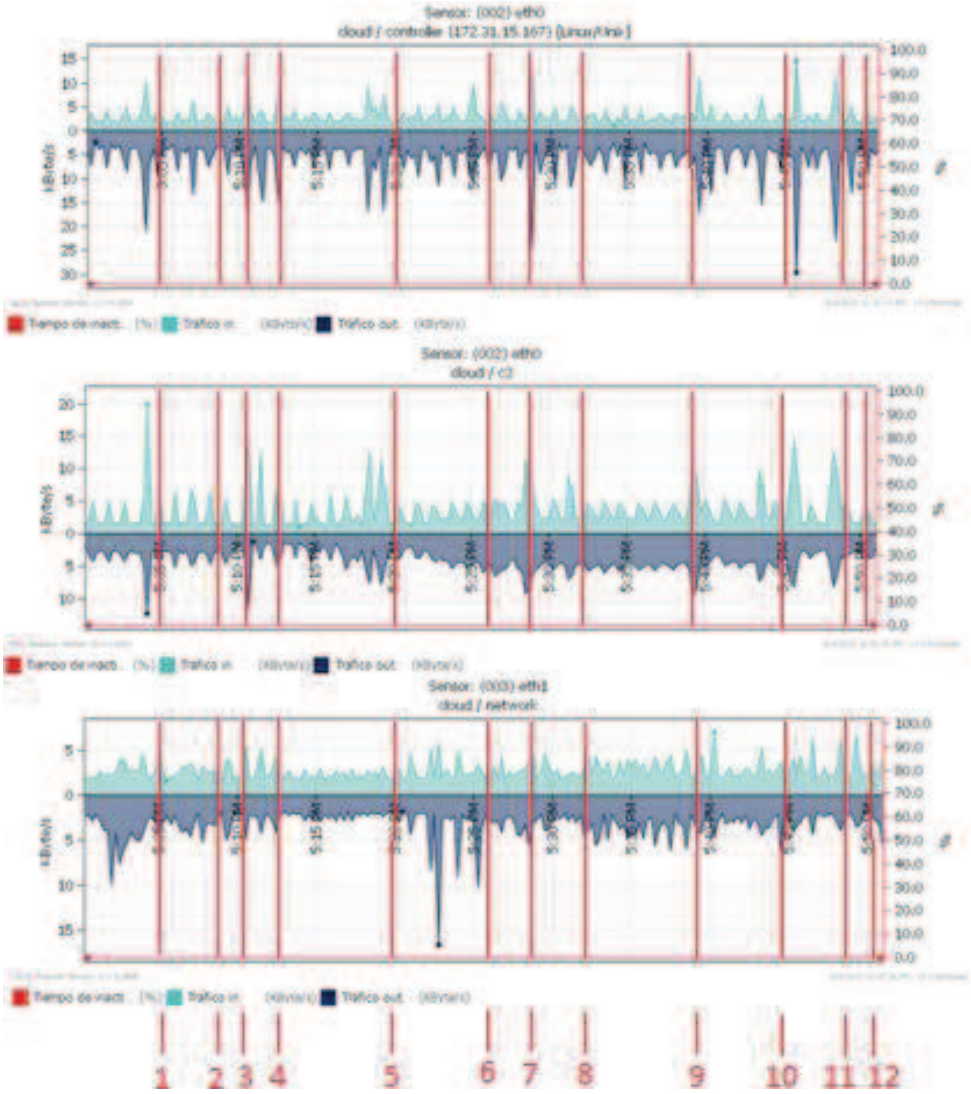


Figura 4. 25 Monitoreo del tráfico en la interfaces de la red de Administración

Esta red está principalmente establecida para el intercambio de objetos (en el caso de usar el proyecto Swift) o acceso a volúmenes de almacenamiento (en el caso de usar el proyecto Cinder).

La actividad del tráfico en la red Privada es casi nula debido a que no existe generación de comunicación de las máquinas virtuales con los nodos *Controller* y *Network*, a excepción del tráfico generado en el minuto 5:05 y 5:30 entre el

servidor C2 y *Network,* donde los usuarios accedieron a las interfaces WEB de sus servicios, provocando que las respectivas máquinas virtuales envíen y reciban tráfico, principalmente HTTP.

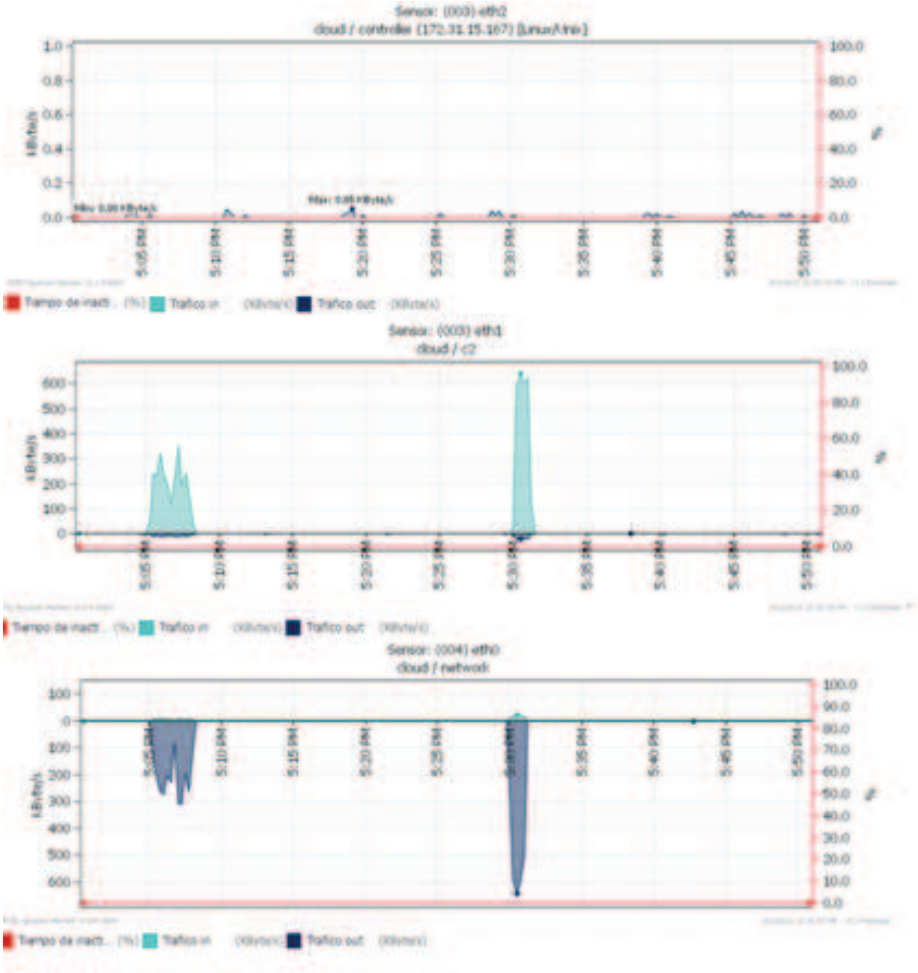


Figura 4. 26 Monitoreo del tráfico en las interfaces de la red Privada

En la Figura 4.27 se muestra las gráficas generadas por la herramienta PRTG correspondiente a las interfaces de los servidores *Controller* y *Network* en la red Pública. Además se especifica los eventos relevantes de las pruebas de usuario descritos en la Tabla 4.3.

Se observa que tráfico de entrada y salida en la interfaz del nodo Controller es mínima (menos de 1Kbps) y los picos que se muestran en la gráfica no corresponden a la generación de las máquinas virtuales de los clientes. Por otro lado, el nodo *Network* muestra una considerable generación de tráfico generado

en el minuto 5:05 y 5:30, debido a que usuarios accedieron a las interfaces WEB de sus servicios, provocando que las respectivas máquinas virtuales envíen y reciban tráfico HTTP.

Figura 4. 27 Monitoreo a las interfaces en la red Pública

Cabe recordar que la interfaz “eth2” del nodo *Network* se encuentra en modo manual, esto quiere decir que Quantum hace uso de esta interfaz para asignar las direcciones públicas por donde los usuarios pueden conectarse a sus servicios.

4.2.4.3.4 Prueba de Carga máxima de la infraestructura

En las pruebas de usuario se evidenció un normal funcionamiento de todos los servicios contratados por los usuarios, mientras que el uso de los recursos de infraestructura nunca alcanzó límites que comprometan el normal funcionamiento de las máquinas virtuales, como se pudo observar en las gráficas de consumo de recursos de la sección 4.2.4.3.3.

Para probar el límite de funcionamiento de la infraestructura instalada, se generó nuevas máquinas virtuales, dos con la imagen “mailcrm” y dos con la imagen “op”,

las cuatro máquinas virtuales con 512MB de memoria RAM, 1CPU y 6GB de disco, obteniendo un total de 11 máquinas virtuales, como se muestra en la Figura 4.28.

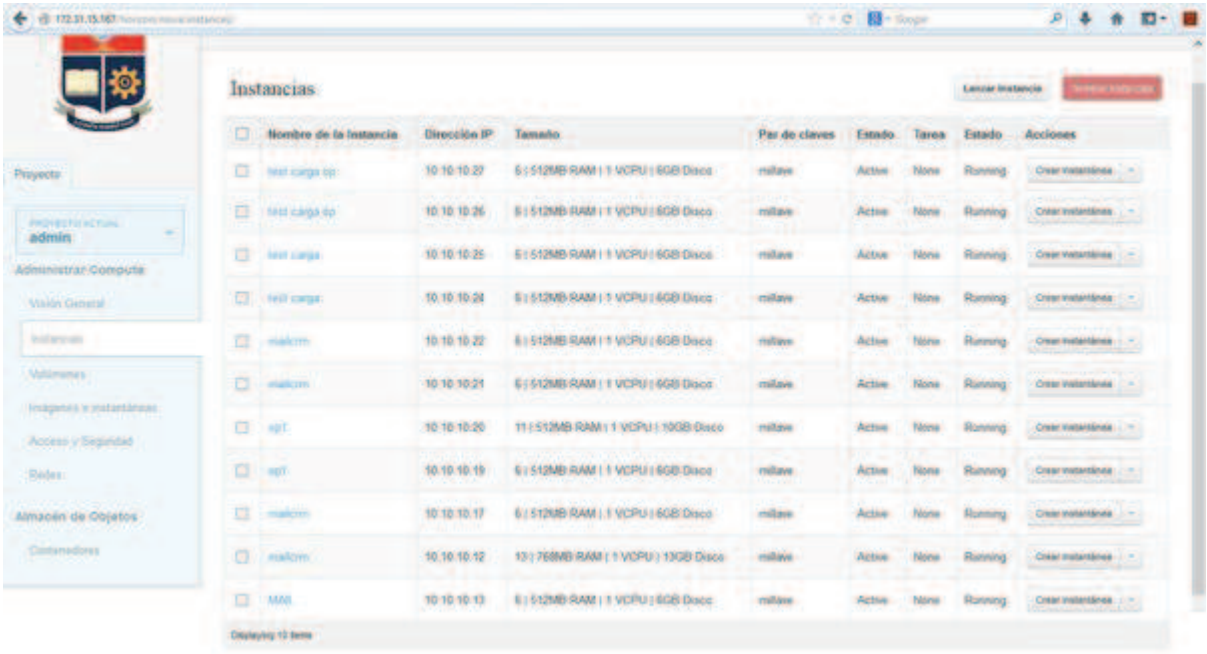


Figura 4. 28 Interfaz de Horizon con las máquinas virtuales creadas para prueba de carga de la infraestructura

Al intentar generar una quinta máquina virtual, Horizon muestra un error, que por medio de consola se pudo determinar que se ha alcanzado el límite en el consumo de los recursos disponibles en la infraestructura, en este caso la memoria RAM.

En la Tabla 4.4 se muestra el análisis de los recursos consumidos del nodo de virtualización C2 en la prueba de carga.

					Recursos Consumidos		
	RAM (MB)	CPU	Disco (GB)	# MVs	RAM (MB)	CPU	Disco (GB)
Flavor "6"	512	1	6	9	4608	9	54
Flavor "11"	512	1	10	1	512	1	10
Flavor "13"	768	1	13	1	768	1	13
				TOTAL	5888	11	77

Tabla 4. 4 Consumo de recursos de prueba de carga

Cabe recordar que el factor de overcommiting incrementa la disponibilidad de los recursos físicos del nodo de virtualización. En la Tabla 4.5 se muestra los recursos físicos, máximos disponibles, consumidos y libres.

Recursos del Nodo de Virtualización				Recursos Consumidos	Recursos Libres
Recursos Físicos		Factor de Overcommiting	Máximo Disponible		
Memoria RAM física en MB	3942	1,5	5913	5888	**-1946**
Núcleos de procesamiento	2	16	32	11	**-9**
Disco en GB (particion sda1)	143		143	77	**66**

Tabla 4. 5 Recursos consumidos y libres del nodo de virtualización en la prueba de carga

Los recursos libres que se muestran en la Tabla 4.5 corresponden a la resta de los recursos físicos, menos los recursos consumidos del nodo de virtualización. Estos valores se muestran en el log de Nova del nodo de virtualización C2, como se observa en Código 4.3.

```
2013-08-02 15:54:20 AUDIT nova.compute.resource_tracker [-] Free ram (MB): -1946
2013-08-02 15:54:20 AUDIT nova.compute.resource_tracker [-] Free disk (GB): 66
2013-08-02 15:54:20 AUDIT nova.compute.resource_tracker [-] Free VCPUS: -9
```

Código 4. 3 Recursos libres mostrados en el archivo nova-compute.log del nodo C2

Se observó que al tener ejecutándose las 11 máquinas virtuales simultáneamente, se presentó un aumento abrupto del procesamiento en el nodo C2, como se observa en la Figura 4.29. Esto ocasionó que las máquinas virtuales dejen de operar correctamente, hasta el punto de no responder a peticiones de red, provocando una total denegación de servicio.

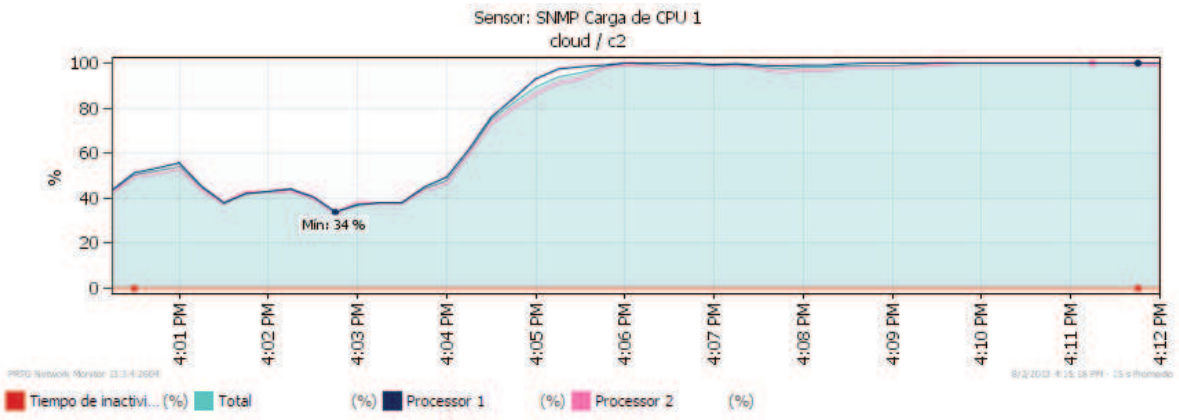


Figura 4. 29 Procesamiento en el nodo C2 con carga máxima

4.3 ANÁLISIS FINANCIERO

4.3.1 PROYECCIÓN DE VENTAS

El prototipo implementado está diseñado para dar soluciones de *Cloud Computing* bajo la modalidad de Software como Servicio para pequeñas y medias empresas. Se consideran como PYME a aquellas empresas que tienen menos de 100 empleados, que en Ecuador representan el 81.7% [70] de un total de 179.830 empresas registradas en el 2012 [71], es decir que la demanda potencial es de 146.921 empresas.

Se estima que la demanda real para la contratación de servicios de *Cloud Computing* son las PYMEs que tienen su propia página web. Se encontró que solo el 42.8% tienen un sitio web propio [72], que son aproximadamente 62.882 Pymes.

		Planes mensuales x año		
Servicio	**Precio**	**1**	**2**	**3**
CRM 15	$45	27	126	448
CRM 50	$150	23	87	339
CRM 100	$300	30	100	347
Mail1 10	$25	22	103	373
Mail1 20	$40	21	96	332
Mail1 50	$65	23	73	286
Mail 2	$3	1150	2796	5677
OM 10	$35	29	92	388
OM 20	$70	24	73	271
OM 50	$140	21	63	233
Venta Anual		$25,635	$85,418	$298,506
Creciento de Ventas anual promedio				**70.86%**
Clientes al tercer año				**492**

Tabla 4. 6 Proyección de Ventas

Debido a la competencia internacional en el segmento de *Cloud Computing* y la múltiple oferta de servicios, se considera como nicho de mercado solamente el 15% de la demanda real, es decir aproximadamente 9.432 empresas. De este valor se considera cubrir el 5% en los tres años posteriores al inicio de las

operaciones del proveedor de SaaS, es decir, el objetivo es alcanzar un aproximado de 472 clientes al tercer año de operaciones.

Según estas perspectivas, se ha hecho una proyección de ventas para un crecimiento promedio anual de 70% en tres años (ver Tabla 4.6).

4.3.2 COSTOS Y GASTOS

4.3.2.1 Costo de producción

Cuando un cliente hace uso de un servicio, implica que consume recursos de cómputo, memoria RAM y almacenamiento, según las características de la máquina virtual que acoge el servicio. El costo del consumo de estos recursos se calcula a partir del costo de adquisición, alojamiento y administración de la infraestructura, valores que constituyen un costo fijo mensual.

4.3.2.1.1 Infraestructura inicial

Para la operación del proveedor de SaaS se requiere el uso de la infraestructura descrita en la sección 2.4.

OpenStack tiene entre sus miembros *platinium* a la compañía HP[68] [74], empresa que produce servidores certificados por Canonical[69] para la instalación del sistema operativo Ubuntu Server [75]. De igual manera, la empresa Cisco figura como miembro *gold* en el proyecto OpenStack [30], por lo que se ha elegido sus productos para la instalación de la infraestructura del Presente Proyecto.

De acuerdo a la previsión de ventas descrita en la Tabla 4.6, el consumo de recursos de infraestructura muestra que el factor determinante para elegir un servidor de virtualización, radica en la cantidad de discos rígidos que pueda soportar. Es decir, de la gama de servidores que ofrece HP en el mercado, se debe escoger un servidor con la menor capacidad de memoria RAM y menor

[68] HP: Es una de las mayores empresas de tecnología, con sede en Palo Alto - California. Fabrica y comercializa hardware, software y brinda asistencia principalmente en proyectos de informática.
[69] Canonical: Empresa de desarrollo de software de código abierto con sede en Londres y oficinas en Boston, Shangai, São Paulo, Montreal y Taipei. Su principal producto es Ubuntu, que es un sistema operativo basado en Debian que se distribuye gratuitamente como software libre.

cantidad de procesadores, pero con la mayor cantidad de ranuras para discos rígidos.

En el Anexo Q se muestra el consumo mensual de recursos de infraestructura de acuerdo a la previsión de ventas. Esta tabla muestra que el consumo de memoria RAM llega hasta los 208GB aproximadamente, el almacenamiento en los discos alcanza los 15TB, y los procesadores virtuales llegan a un máximo aproximado de 430. Analizando estos datos se puede determinar que se necesitaría un servidor con al menos 2 procesadores de 16 núcleos cada uno, y al menos 15 bahías para discos rígidos. Existen servidores que ofrecen estas características, pero tienen una capacidad excesiva de memoria RAM (768GB de memoria RAM [76]), capacidad que no se utilizaría en el periodo de tiempo analizado, por lo que habría un desperdicio innecesario de recursos.

Por tal razón, se ha determinado que los equipos que se muestran en la Tabla 4.7 son los que mejor se adaptan al consumo de recursos del Proveedor de SaaS. Se especifica las principales características, la función que ocuparán los equipos dentro de la infraestructura y su precio aproximado.

Con el uso de los equipos mencionados en la Tabla 4.7, se alcanza a cubrir la capacidad máxima de almacenamiento de discos (8GB) en el mes número 31, justamente en el mismo mes donde se alcanza la capacidad máxima de procesadores virtuales (256 vCPU). Por tal motivo, en el mes 32 se prevé la adquisición de un segundo servidor de virtualización de las mismas características para cubrir la demanda de los usuarios.

Equipo Activo	Características	Función	Precio
HP Proliant DL165 G7 [77]	1 x AMD Opteron 6272 (2.1GHz/16 core), Memoria RAM 8GB (4x2GB) con capacidad hasta 384GB, 4 Interfaces RJ-45 x 1Gbps, chasis 1U en rack, 8 *slots* para discos SFF[70] de hasta 8TB.	Nodo de Cómputo	$2,149
HP Proliant DL120 G7 [78]	Procesador Intel Xeon E3-1200 (4 core, max 3.5GHz), Memoria RAM de 2GB con capacidad máxima de 32GB, 2 Interfaces RJ-45 x 1Gbps, chasis 1U en rack.	Nodo Controlador y Nodo de Red	2 x $1,023
HP 1TB 6G SAS 7.2K 2.5in [79]	Disco SAS[71], I/O 6GB/sec, 7.2krpm, 2.5" Hot Plug[72].	Disco para nodo de Cómputo	$294
Memoria RAM HP 16 GB [80]	HP 16GB Quad Rank[73], DDR3 SDRAM[74], velocidad de 1066Mhz.	Memoria para nodo de cómputo	$155
Router Cisco 1841 [81]	256MB de memoria DRAM[75], 2 Interfaces 10/100/1000 RJ-45, 1 USB, 1 Consola, 80 watts.	Interconexión de red pública e Internet	$422
Switch Cisco 2960S-24TS-S [82]	Switch no administrable de 24 puertos Ethernet 10/100/1000. Chasis 1U en rack. 30 Watts.	Switch de la red privada	$707
		TOTAL	$ 5,773

Tabla 4. 7 Equipos, características y precios

4.3.2.1.2 Recursos adicionales de infraestructura

Existen recursos que se añaden a la infraestructura según el aumento del volumen de clientes. La memoria RAM, el almacenamiento en disco, la velocidad del enlace de Internet y las direcciones IP públicas son recursos que deben ser agregados según la demanda de servicios y que los proveedores deberán

[70] SFF: *Small Form Factor*, es la denominación que se da a un disco de almacenamientos de 2.5 pulgadas de ancho.
[71] SAS: *Serial Attached* SCSI (*Small Computer System Interface*), es una tecnología de transferencia de información diseñada para sistemas intensivos de lectura y escritura y que requieran tiempos de acceso muy rápidos y lecturas o escrituras aleatorias. Su velocidad de rotación varía desde los 7.2KRPM a los 15KRPM.
[72] *Hot Plug*: Conexión en caliente, es la capacidad de un periférico de poder enchufarse o desenchufarse de un computador sin apagarlo.
[73] Un *rank* de memoria es un área de datos creado para que contenga una parte o todos los chips de la memoria RAM.
[74] DDR3 SDRAM: *Double Data Rate type three Synchronous Dynamic Random-Access Memory,* es un tipo de memoria RAM que permite usar integrados de 512MB a 8GB, siendo posible fabricar módulos de hasta 16GB.
[75] DRAM: *Dynamic Random Access Memory,* memoria RAM dinámica se usa principalmente como memoria primaria de un dispositivo. Se denomina dinámica ya que para almacenar un dato se requiere alojarlo y refrescarlo cada cierto periodo de tiempo.

suministrar de manera ágil e inmediata, optimizando así el uso de la infraestructura disminuyendo la subutilización de recursos.

	Marca y modelo o proveedor	Características	Precio
Disco	HP 1TB 6G SAS 7.2K 2.5in	Disco de 1TB de capacidad, velocidad rotacional de 7.2K rpm, interfaz SAS de 6Gb/s, hotplug.	$294
Memoria RAM	HP 16GB Quad Rank	Tarjeta de memoria RAM de 16GB, DDR3 SDRAM, velocidad de 1066 Mhz.	$155
IP pública	Telconet	Dirección IPv4 pública	$1.50

Tabla 4. 8 Elementos a añadir en la infraestructura inicial según crecimiento de clientes

En la Tabla 4.8 se muestra el equipamiento y recursos que se añadirá a la infraestructura inicial según el aumento del número de clientes.

Cada servicio además requiere de cierta cantidad de tráfico de datos del cliente por medio de Internet.

En el Anexo N se muestra la cotización de una empresa que ofrece el servicio de alojamiento de servidores en su centro de datos. En la Tabla 4.9 se muestra los valores relevantes de la cotización.

	Costo	Cantidad	Total
1 Mbps desde DC a Internet	$120	1	$120
Unidad de Rack	$150	4	$600
1 kVA	$280	1	$280
Soporte e intervención	$80	1	$80
		TOTAL	$1,080

Tabla 4. 9 Características del servicio de alojamiento y su costo

4.3.2.1.3 Capacidad de operación de la infraestructura

OpenStack tiene una característica importante para la optimización de recursos, llamada *overcommitting* [83]. Es la capacidad de incrementar el número de máquinas virtuales que puede alojar un nodo de cómputo simulando la ampliación del número de núcleos y memoria RAM, en detrimento del rendimiento de las máquinas virtuales. Nova usa por defecto los siguientes factores de asignación.

- Factor de asignación de CPU (CPU *ratio*): 1 a 16
- Factor de asignación de memoria RAM (RAM *ratio*): 1 a 1.5

Un factor de asignación de CPU de 1 a 16 significa que *Nova-Scheduler* asigna 16 núcleos virtuales por cada núcleo físico que tenga el nodo de cómputo. Por ejemplo, si un nodo tiene 2 procesadores y cada uno de éstos tiene 6 núcleos con tecnología *HyperThreading*[76], *Nova-Scheduler* mostrará que el nodo de cómputo tiene la disponibilidad de 384 vCPU para la creación de máquinas virtuales.

Un factor de asignación de memoria RAM de 1 a 1.5 significa que *Nova-Scheduler* puede asignar a las máquinas virtuales 1.5 veces más cantidad de memoria RAM de la que tenga el nodo físicamente. Por ejemplo, si un nodo de cómputo tiene 48GB, en él se podrán alojar máquinas virtuales hasta un total de 72GB de memoria RAM.

Capacidad máxima de la Infraestructura		
RAM	384GB x 1.5 (*RAM ratio*)	576GB
vCPU	16cores x 16 (*CPU ratio*)	256vCPU
Almacenamiento	8 slots x 1TB	8TB
Precio total de la Infraestructura		$11.396
Valor mensual a 36 meses		**$316,56**

Tabla 4. 10 Capacidad máxima de infraestructura y su precio

Tomando en cuenta esta característica especial de OpenStack, la Tabla 4.10 muestra la capacidad máxima de la infraestructura inicial, su precio aproximado y el valor mensual distribuido en 36 meses.

4.3.2.1.4 Recursos Consumidos

Se ha definido seis tipos de máquinas virtuales que podrán alojar los servicios solicitados por los clientes. La cantidad de máquinas virtuales que puede alojar la infraestructura depende mayormente del almacenamiento en disco y de la cantidad de las vCPU que puede alojar un nodo de cómputo.

[76] Tecnología propietaria de Intel presente en algunas familias de procesadores. Consiste en crear dos núcleos virtuales a partir de un núcleo real, pero el sistema operativo debe estar preparado para utilizar esta tecnología.

Según los valores que muestra la Tabla 4.10, en la Tabla 4.11 se indica la cantidad de máquinas virtuales que puede alojar la infraestructura a su máxima capacidad junto con su costo mensual.

Tipo de MV	RAM	vCPU	Disco	MV máximo	Costo x MV
MV1	256	1	10	256	$1.237
MV2	512	1	20	256	$1.237
MV3	768	1	50	160	$1.978
MV4	1024	1	100	80	$3.957
MV5	2048	1	200	40	$7.914
MV6	4096	1	400	20	$15.828

Tabla 4. 11 Costo de producción de cada tipo de máquina virtual

En la Tabla 4.12 se muestra el tipo de máquina virtual, el ancho de banda que ocupa cada uno de los servicios, y su costo total de producción.

Servicio	Tipo de MV	Internet [kbps]	Costo MV	Costo Internet	Costo Total
CRM 15	MV1	6.07	$1.24	$0.71	$1.95
CRM 50	MV2	20.23	$1.24	$2.37	$3.61
CRM 100	MV3	40.45	$1.98	$4.74	$6.72
Mail1 10	MV1	60.68	$1.24	$7.11	$8.35
Mail1 20	MV2	151.70	$1.24	$17.78	$19.01
Mail1 50	MV3	455.11	$1.98	$53.33	$55.31
OM 10	MV1	93.33	$1.24	$10.94	$12.17
OM 20	MV2	202.22	$1.24	$23.70	$24.93
OM 50	MV3	466.67	$1.98	$54.69	$56.67

Tabla 4. 12 Costo de producción de los servicios ofrecidos

El costo de producción del servicio de Correo Electrónico Plan Limitado varía según la cantidad de cuentas de correo alojadas en el servidor. En la Tabla 4.13 se muestra la máquina virtual, velocidad de conexión a Internet y su costo total de producción.

Servicio Correo Electrónico Limitado						
Cuentas	Tipo de MV	Internet [kbps]	Costo MV	Costo Internet	Costo Total	Costo x cuenta
20	MV1	60.68	1.2365	$7.11	$8.35	$0.41738
100	MV2	303.41	1.2365	$35.56	$36.79	$0.36792
300	MV3	910.22	1.9785	$106.67	$108.65	$0.36215
500	MV4	1,517.04	3.9569	$177.78	$181.73	$0.36347
1000	MV5	3,034.07	7.9139	$355.56	$363.47	$0.36347
2000	MV6	6,068.15	15.828	$711.11	$726.94	$0.36347

Tabla 4. 13 Costo de producción de servicio de correo electrónico Plan Limitado

En la Tabla 4.14 consta el costo de producción anual según la proyección de ventas descrita en la Tabla 4.6. Para el registro de los servicios, se menciona como “Mail 1” al servicio de correo electrónico Plan Ilimitado, y “Mail 2” al servicio de correo electrónico Plan Limitado.

		Planes mensuales x año		
Servicio	Costo	1	2	3
CRM 15	$1.948	27	126	448
CRM 50	$3.607	23	87	339
CRM 100	$6.719	30	100	347
Mail1 10	$8.348	22	103	373
Mail1 20	$19.014	21	96	332
Mail1 50	$55.312	23	73	286
Mail 2	variable	1150	2796	5677
OM 10	$12.174	29	92	388
OM 20	$24.934	24	73	271
OM 50	$56.666	21	63	233
Costo de Produccion Anual		$4,753.25	$15,480.52	$56,419.82

Tabla 4. 14 Costo de producción anual

4.3.2.2 Gasto Variable

Al costo de producción se debe añadir el gasto en los recursos que se han de adicionar a la infraestructura según el crecimiento de clientes.

Cabe mencionar que el precio de las tarjetas de memoria RAM, los discos y los nuevos nodos de cómputo, se cubren con un único pago al momento que se requieran instalar en la infraestructura. Mientras que el acceso a Internet y las direcciones IP públicas se deben pagar mensualmente al proveedor del

alojamiento. La suma de estos valores se lo denomina “gasto variable”, el cual se muestra en la Tabla 4.15.

Recursos Consumidos	1	2	3
Memoria RAM (Mb)	17152	59648	208128
Almacenamiento (Gb)	1192	4389	15316
Internet (Kbps)	37680	121839	443748
IP pública	232	825	3029
Nodo de cómputo adicional	0	0	1
Gasto variable anual	**$4,242**	**$15,990**	**$61,942**

Tabla 4. 15 Gasto variable anual

4.3.2.3 Gasto de Personal

Para mantener el buen funcionamiento de la infraestructura se requiere de personal constantemente pendiente de su monitoreo, cambios y mantenimiento; así como también personal adicional para atención al cliente, personal de ventas y contabilidad.

A inicio de las operaciones, la carga administrativa y de infraestructura la podrá cubrir una sola persona, pero a medida que aumente la cantidad de clientes, la carga administrativa irá aumento así como la necesidad de contratar nuevo personal. En la Tabla 4.16 se muestra la estimación de gastos de personal.

	1	2	3
# Promotores	1	1	1
# Empleados	1	2	4
Gasto Sueldo Promotor	$8,000	$11,000	$13,500
Gasto Sueldo Empleados	$4,800	$13,900	$30,150
IESS	$1,580.80	$3,075.15	$5,390.78
13er Sueldo	$1,177.43	$2,292.71	$3,740.63
14to Sueldo	$556.50	$821.50	$1,298.50
Fondo de Reserva	$88.89	$172.92	$303.13
Total	**$16,203.62**	**$31,262.28**	**$54,383.03**

Tabla 4. 16 Estimación del gasto de personal

4.3.2.4 Gasto Administrativo

En este rubro se detalla el gasto de arrendamiento del sitio que aloja la infraestructura, el arrendamiento de la oficina donde estará el personal

administrativo, así como también se especifica los valores por trámites legales para la constitución de la empresa, patente municipal y permisos de la Superintendencia de Compañías. Además el pago de servicios básicos, compra de útiles de oficina, etc. La Tabla 4.17 muestra un resumen anual de los rubros antes mencionados.

	1	2	3
Arriendo	$11,538	$11,538	$12,438
Suministros	$210	$355	$525
Servicios Basicos	$540	$840	$1,480
Generales	$106	$202	$338
Legales, permisos y patentes	$600	$200	$300
Total	**$12,994**	**$13,135**	**$15,081**

Tabla 4. 17 Estimación del gasto administrativo

4.3.3 RESULTADO OPERATIVO

El resultado operativo, o también conocido como EBIT[77] es el resultado de deducir el costo de producción, gasto variable, gasto personal y gasto administrativo, de las ventas en cada periodo. La Tabla 4.18 muestra el resultado operativo antes de deducir la depreciación de la infraestructura adquirida, la amortización del financiamiento, las utilidades y el impuesto a la renta.

	1	2	3
Ventas	$25,635	$85,418	$298,506
Costo de Producción	$4,753	$15,481	$56,420
Gasto Variable	$4,242	$15,990	$61,942
Gasto Personal	$16,458	$34,015	$54,580
Gasto Administrativo	$12,994	$13,135	$15,081
EBIT	**-$12,812**	**$6,798**	**$110,484**

Tabla 4. 18 Resultado Operativo (EBIT)

[77] EBIT: *Earnings Before Interest and Taxes*, es el valor que muestra el beneficio puro de la operación de una empresa, independientemente de cómo se ha financiado los activos necesarios para su operación y de los impuestos sobre los beneficios.

4.3.4 RESULTADO NETO

La utilidad líquida de un ejercicio económico es el resultado de deducir la depreciación del activo fijo y la amortización por financiamiento, del resultado operativo.

El Servicio de Rentas Internas impone en Ecuador la depreciación sobre activos fijos, en el caso de los equipos de computación corresponde al 33% anual [84]. Por lo tanto, el cálculo de la depreciación de la infraestructura inicial es de $1.732 anual.

Se estima que para el inicio de las operaciones del proveedor de SaaS se requiera el financiamiento del capital de trabajo por parte de una entidad bancaría o financiera. Un interés del 10.85% corresponde al interés para un crédito sobre capital de trabajo para el segmento PYMES a 3 años plazo, publicado por la Corporación Financiera Nacional en el mes de junio del 2013 [85].

Capital de trabajo son aquellos recursos que requiere una empresa para poder operar. En este caso es necesario cubrir los primeros seis meses de: costo de producción, gasto variable, gasto administrativo, gasto personal, así como también el valor de adquisición de la infraestructura inicial, como se muestra en la Tabla 4.19.

Costo de Producción	$1,656
Gasto Variable	$1,092
Gasto Personal	$6,474
Gasto Administrativo	$6,442
Infraestructura Inicial	$5,773
Capital de Trabajo	**$21,438**

Tabla 4. 19 Capital de Trabajo

Con los valores antes obtenidos, en la Tabla 4.20 se muestra la utilidad líquida y la participación laboral (en el caso de Ecuador es el 15% sobre la utilidad líquida),

se muestra también el rubro destinado al pago del Impuesto a la Renta[78], que corresponde al 22% del resultado de la resta de la participación laboral sobre la utilidad líquida.

	1	2	3
EBIT	-$12,558	$9,551	$110,681
Amortización	$8,750	$8,750	$8,750
Depreciación de activos	$1,732	$1,732	$1,732
Utilidad Líquida	-$23,039	-$931	$100,199
Participación Laboral	**$0**	**$0**	**$15,030**
Impuesto a la Renta	**$0**	**$0**	**$18,737**

Tabla 4. 20 Utilidad Líquida, Participación Laboral e Impuesto a la Renta

El resultado neto, es el resultado de deducir la depreciación del activo fijo, la amortización por financiamiento, la participación laboral y el impuesto a la renta, el resultado operativo. Estos valores se muestran en la Tabla 4.21.

	1	2	3
EBIT	-$12,558	$9,551	$110,681
Amortización	$8,750	$8,750	$8,750
Depreciación de activos	$1,732	$1,732	$1,732
Participación Laboral	$0	$0	$15,030
Impuesto a la Renta	$0	$0	$18,737
Resultado Neto	**-$23,039**	**-$931**	**$66,432**

Tabla 4. 21 Resultado Neto

4.3.5 INDICADORES

En un proyecto empresarial, es importante destacar la viabilidad y su rentabilidad en el tiempo, por la inversión de un capital inicial. La rentabilidad esperada para un proyecto de inversión, debe ser algo mayor que una inversión de bajo riesgo (por lo general se toma como referencia el interés para una póliza de acumulación, que a junio de 2013 es de 4.53% [86]) para justificar el esfuerzo y tiempo dedicado a la creación de la empresa. El VAN[79] y el TIR[80] se basan en el

[78] Impuesto a la Renta: es el impuesto que se grava sobre los ingresos o rentas, producto de actividades económicas percibidos durante un año, luego de descontar los costos y gastos incurridos para obtener o conservar dichas rentas.

[79] VAN: Valor Actual Neto, es un método de valoración de inversiones que permite calcular el valor presente de un determinado número de flujos de caja futuros, originados por una inversión.

resultado neto que tenga un proyecto en el transcurso del tiempo. Es decir, que si al invertir cierta cantidad de dinero se generan flujos de cajas positivos, en algún momento se recuperará lo invertido. Pero habrá que actualizar el valor de los ingresos futuros a la fecha actual de acuerdo a una tasa de interés referencial.

Resultado Neto o Flujo de Caja			
Inversión	1	2	3
-$21,438	-$23,039	-$931	$66,432
Tasa de Interés		10.85%	
VAN	TIR		
$5,226	16.63%		

Tabla 4. 22 Indicadores TIR y VAN del proyecto "Proveedor de SaaS para PYME"

En la Tabla 4.22 se muestra los indicadores VAN y TIR según la inversión inicial y el resultado neto obtenido en los 3 años de operaciones del proveedor de SaaS, respecto de la tasa de interés del crédito para capital de trabajo.

Otros indicadores que destacar son el crecimiento anual de las ventas, los costos y los gastos, el crecimiento anual de clientes y los empleos creados, los cuales se muestran en la Tabla 4.23.

	1	2	3	Promedio
Crecimiento de Ventas	71.19%	69.99%	71.38%	**70.86%**
Crecimiento de Costos	74.84%	69.30%	72.56%	**72.23%**
Crecimiento de Gastos	48.46%	44.62%	54.05%	**49.04%**
Crecimiento de Clientes	72.67%	68.92%	69.45%	**70.35%**
Empleos creados directamente en el proyecto				**5**

Tabla 4. 23 Indicadores relevantes del proyecto

[80] TIR: Tasa Interna de Retorno, es el interés que hace del valor neto igual a cero. Es un indicador para la rentabilidad de un proyecto, que de ser superior a una tasa referencial, el proyecto puede ser digno de inversión.

CAPÍTULO 5
CONCLUSIONES Y RECOMENDACIONES

5.1 CONCLUSIONES

- Según los niveles de madurez de una arquitectura de SaaS descritos en la sección 1.4.3.1.1, el Presente Proyecto demostró la posibilidad de proveer aplicaciones bajo el modelo SaaS con una arquitectura de nivel de madurez 1 (Ad-Hoc/Customización). Se concluye que se puede mejorar el nivel de madurez tratando directamente el código de las aplicaciones para que manejen dinámicamente los recursos de software según la variación en la demanda, optimizando así el uso de recursos de IaaS.

- Se concluye que el elevado precio de la salida internacional de Internet eleva significativamente el costo de producción de los servicios, generando un menor margen de ganancia.

- Una nube privada ofrece gran agilidad en el despliegue de servicios, pero su implementación está sujeta a una fuerte inversión inicial y un largo tiempo de aprendizaje y exitosa adopción. Por lo tanto, se concluye que este tipo de soluciones quedan fuera del alcance de las pequeñas empresas ya que no tienen el capital monetario y humano para afrontar este tipo de proyectos donde los beneficios se presenten en el mediano o largo plazo.

- En la configuración de la red de la Infraestructura OpenStack, se realizó pruebas con Nova-Network y Quantum. De esta experiencia se concluye que para el Presente Proyecto cualquiera de las dos tecnologías permite conseguir el funcionamiento del sistema de provisión de SaaS, ya que solamente se necesita generar una red para alojar las máquinas virtuales con las aplicaciones de los usuarios. Se decidió usar Quantum para la implementación final ya que es un proyecto en constante desarrollo y

amplio soporte por parte de la comunidad OpenStack, mientras que el desarrollo de Nova-Network terminará en la versión IceHouse [87].

- Se concluye que es totalmente factible alcanzar un alto crecimiento de ventas para un servicio de *Cloud Computing*, como el que se muestra en el Presente Proyecto (70%), ya que el mercado de *Cloud Computing* es aún un terreno poco explotado y se prevé que será altamente demando en los próximos años. Mucho más potencial tienen soluciones del tipo Software como Servicio que según estudios internacionales, es el segmento de mayor crecimiento comparado con soluciones tipo IaaS y tipo PaaS [88].

- Gracias a la gran escalabilidad que permite la arquitectura de OpenStack, se concluye que se puede desarrollar todo un entorno de *Cloud Computing* con equipamiento de hardware bastante modesto, como se demostró en la implementación del Presente Proyecto. Además, siempre es factible aumentar nodos de cómputo a medida que los requerimientos de memoria RAM y procesamiento crezcan.

- A partir de las pruebas de carga realizadas sobre el sistema, se determinó que para las características de consumo de recursos de infraestructura del Presente Proyecto, la mejor elección para el nodo de cómputo, es un servidor que tenga la mayor cantidad de núcleos por procesador, pero no más de 1 procesador. La propiedad de overcommiting de OpenStack permite extender la capacidad física de memoria RAM y número de vCPUs para la generación de máquinas virtuales, pero en la práctica, esta relación de crecimiento ocasiona que la generación de la máxima cantidad de máquinas virtuales esté limitada a la capacidad de memoria RAM.

- Para la creación de un entorno de *Cloud Computing* donde el principal requerimiento sea la creación de gran cantidad de máquinas virtuales, se concluye que los nodos de cómputo deben tener la mayor cantidad de

núcleos por procesador, y la mayor cantidad de procesadores. Ya que la cantidad de memoria RAM que puede alojar un servidor con estas características de procesamiento, no será limitante en la cantidad de máquinas virtuales que se puedan crear.

- Desde el punto de vista del proveedor, se puede administrar la infraestructura para optimizar al máximo el uso de recursos, siempre y cuando no afecte el servicio entregado al cliente. Esta premisa se demostró al facultar el despliegue de los servicios de Correo Electrónico y CRM en la misma máquina virtual de un cliente, como se muestra en la sección 3.1.5.1.

- Se concluye que el uso de una herramienta para la gestión de la configuración de código abierto, como Puppet [89], Chef [90], Vagrant [91] o Juju [92], facilitaría enormemente la automatización para generar servicios SaaS. Este tipo de herramientas realizaría la misma labor que los *scripts* de automatización desarrollados en el Presente Proyecto, con el beneficio adicional de optimizar el uso de los recursos de la infraestructura IaaS.

- Uno de los objetivos del Presente Proyecto fue dar al usuario la posibilidad de cambiar las prestaciones de su servicio contratado. Para ello se debía cambiar las características de almacenamiento de la máquina virtual donde se ejecuta la aplicación del usuario. En la implementación se realizó pruebas utilizando Swift para almacenar los archivos adicionales necesarios a la máquina virtual del usuario, de igual manera, se probó Cinder para asignar almacenamiento adicional en forma de volúmenes lógicos. En el primer caso se presentó el problema de poder contabilizar el espació real que ocupaban los archivos adicionales del usuario, en el segundo caso se presentó inestabilidad al interactuar con la máquina virtual del cliente. La solución definitiva para el aumento o reducción de la

capacidad de disco de una máquina virtual, se consigue con mayor facilidad a través de la propiedad de redimensionamiento de Nova (como se explica en la sección 3.2.3.4) lo que evita el uso de Swift y Cinder, que son los proyectos específicos de OpenStack para el manejo del almacenamiento.

- Se concluye que para la provisión de Software como Servicio es mejor externalizar la administración de los recursos IaaS, así los recursos humanos y económicos se usarían para investigación y desarrollo de nuevas funcionalidades de la aplicación. Además esta práctica abarataría costos de producción y mejoraría la calidad del servicio.

5.2 RECOMENDACIONES

- La implementación de los servicios se ha realizado usando software libre. Para la puesta en producción del prototipo se recomienda analizar más detenidamente el licenciamiento para la comercialización de cada software usado.

- La Fundación OpenStack hace un manejo minucioso del uso de la marca "OpenStack". Para la puesta en producción del prototipo se recomienda consultar la sección de "*Branding*" de la página web oficial, ya que se requiere la autorización directa del Departamento de Negocios de la Fundación OpenStack, para la utilización del logo "*Powered by OpenStack*" [93].

- En el caso del uso del software SugarCRM, para el proyecto se usó la versión *Community* que es libre y gratuita. Pero para uso comercial se recomienda consultar con el fabricante la posibilidad de ser *Partner* y poder distribuir SugarCRM con todas las funcionalidades, el soporte técnico y apoyo comercial [94]. El cliente tendría claros beneficios como: la

reducción en el tiempo de respuesta de soporte técnico, crédito o formas de pago flexible de acuerdo al tamaño de la implementación. Este costo de licenciamiento se debería añadir al costo de producción, lo que incrementaría el precio final de venta al público.

- iRedMail ofrece una solución comercial con más funcionalidades, soporte técnico y comercial [95]. Se recomienda adquirir la licencia para dar un mejor servicio en la provisión del servicio de correo electrónico.

- De llegar a implementar el proveedor de SaaS, se recomienda asegurar el crecimiento de las ventas con una fuerte campaña de publicidad web. Un estudio de mercado ayudaría a determinar los sitios web que más acceden las PYME en Ecuador, y con esa información contratar *AdWords*[81] y realizar publicidad más objetiva y por ende, con mayor efectividad.

- Si bien se ha diseñado una infraestructura para ofrecer servicios de *Cloud Computing* al mercado ecuatoriano. Se recomienda analizar la implementación de la infraestructura en un proveedor de alojamiento de servidores fuera del país. El análisis implicaría principalmente determinar la latencia generada por la distancia en la conexión hasta el cliente final, y su impacto en la calidad del servicio. De demostrarse una afectación leve a la calidad del servicio, se justificaría el alojamiento de la infraestructura en servidores del exterior, ya que ofrecen más servicios por un precio más reducido. Puede ser una buena alternativa el alquiler mensual de servidores, donde el proveedor se encargue del mantenimiento físico del hardware y que provea el acceso a Internet necesario. Servicios de este tipo ofrecen varias empresas como *BaremetalCloud* [96] o *HostLocation* [97].

[81] *AdWords* es el programa que utiliza Google para ofrecer publicidad en su buscador y por medio de otros sitios web que pertenecen a la red de publicidad de Google. Un usuario pagará a AdWords solamente por la cantidad de clics que reciba un enlace. De igual manera, el dueño de una página web cobrará por los clics sobre aquel anuncio.

- Se recomienda el uso del sistema operativo Ubuntu para la implementación de OpenStack, ya que existe documentación oficial que guía claramente en el proceso de instalación y configuración. Ubuntu además incluye por defecto el repositorio con los paquetes de todos los proyectos de OpenStack.
- Se recomienda el uso del lenguaje de programación Python, ya que se evidenció la adaptabilidad al entorno web (por medio del uso del *framework* Django) y su simplicidad para el desarrollo de *scripts* con el API de OpenStack.

- Una aplicación SaaS debe ser amigable. Procurar que el único requisito sea que el usuario sepa usar el dispositivo donde se muestre la aplicación. Por lo que se recomienda tener el asesoramiento de un profesional de desarrollo de interfaces e interactividad, para asegurar que la aplicación se adapte de la mejor manera al usuario final.

- El Prototipo de proveedor de SaaS no ha requerido explotar todas las características que ofrece OpenStack para el montaje de una nube privada. Por lo que se recomienda una mayor investigación en el campo de la Infraestructura como Servicio para aplicar las funcionalidades de los proyectos Cinder y Swift, que no han sido tratados en la implementación del Presente Proyecto.

- Al momento de crear una página con Django, donde se requiera el ingreso de información por parte del usuario, se recomienda crear un formulario con todos los campos necesarios, para que pueda ser importado directamente en una plantilla y se haga la validación de cada campo.

- Si se requiere hacer un cambio al modelo de base de datos de una aplicación de Django sin perder la información almacenada, se recomienda

realizar el cambio primeramente en el archivo de configuración del modelo (*model.py*) y después efectuar el cambio sobre la base de datos en el servidor.

- Se recomienda el uso del *framework* BootStrap [98] para la maquetación de una página web, ya que facilita la comprensión del lenguaje CSS. Esta herramienta incluye código HTML, CSS predefinido para la creación de botones, tablas, tipografía y otros componentes, así como la inclusión de extensiones JavaScript.

- El Presente Proyecto no cubre criterios de seguridad, por lo que se recomienda el análisis de la seguridad de una infraestructura de nube, mucho más para proyectos que pretendan brindar servicios de nube pública, ya que la alteración inadecuada de los archivos de configuración podría afectar el funcionamiento de la infraestructura y perturbar el servicio a los usuarios de la nube OpenStack.

- En comparación con la antigua tecnología de red (*nova-network*) que implementaba por defecto OpenStack, se recomienda el uso de Quantum ya que da un manejo más flexible para la interconexión de las máquinas virtuales evitando la configuración de VLAN y la adición de hardware adicional a la infraestructura de red.

- Se recomienda no utilizar más de 80% de los recursos disponibles de un nodo de virtualización con la generación de máquinas virtuales, ya que provocará lentitud en el funcionamiento de los servicios contratados, como se evidenció en las pruebas de carga máxima descritas en la sección 4.2.4.3.4.

REFERENCIA BIBLIOGRÁFICA

[1] B. Sosinsky, «Cloud Computing Bible,» de *Cloud Computing Bible*, Wiley Publishing Inc., 2011, p. 14.

[2] IEEE, «Towards a Formal Definition of a Computing Cloud,» de *Services (SERVICES-1), 2010 6th World Congress on*, Hawthorne, 2010.

[3] NIST, «The NIST Definition of Cloud Computing,» 2011.

[4] N. Inc., «http://www.nskinc.com/resources/blog/cloud-computing-101/,» NSK Inc., 2 Marzo 2010. [En línea]. http://www.nskinc.com/wp-content/uploads/2010/11/cloud-diagrams.png. [Último acceso: 15 Septiembre 2012].

[5] N. Inc., NSK Inc., 24 Noviembre 2010. [En línea]. http://www.nskinc.com/wp-content/uploads/2010/03/cloud-diagrams-private-cloud.png. [Último acceso: 15 Septiembre 2012].

[6] N. Inc, « http://www.nskinc.com/it-services/pavis-solutions/,» NSk Inc., Marzo 2010. [En línea]. http://www.nskinc.com/wp-content/uploads/2010/11/cloud-diagrams-hybrid.png. [Último acceso: Septiembre 2012].

[7] L. B. N. Laboratory, «http://eetd.lbl.gov/newsletter/nl12/eetd-nl12-6-datacenter.html,» 2003. [En línea]. http://eetd.lbl.gov/newsletter/nl12/images/data-center.gif. [Último acceso: 15 Septiembre 2012].

[8] I. H. Stauffer, «http://www.mundohvacr.com.mx,» 11 Octubre 2011. [En línea]. http://www.mundohvacr.com.mx/mundo/2011/10/ahorro-energetico-en-sistemas-de-data-center/. [Último acceso: 15 Septiembre 2012].

[9] F. Ciachi, «Linux dentro de Linux,» *Linux Magazine,* nº siete, pp. 21-24, 2006.

[10] Wikipedia, «http://es.wikipedia.org/wiki/Xen,» 6 Julio 2012. [En línea]. [Último acceso: 16 Septiembre 2012].

[11] M. A. A. Pomar, «http://www.josemariagonzalez.es,» 11 Abril 2011. [En línea]. http://www.josemariagonzalez.es/2011/04/19/microsoft-hyperv-capitulo-1-introduccion-hyperv.html. [Último acceso: 17 Septiembre 2012].

[12] Y. Chou, «http://blogs.technet.com,» 15 Noviembre 2010. [En línea]. http://blogs.technet.com/cfs-file.ashx/__key/CommunityServer-Blogs-Components-WeblogFiles/00-00-00-62-43-metablogapi/8551.image_5F00_12.png. [Último acceso: 17 Septiembre 2012].

[13] p. (seudonimo), «http://blogs.msdn.com,» 18 Agosto 2008. [En línea]. http://blogs.msdn.com/b/architectsrule/archive/2008/08/18/saas-maturity-model-according-to-forrester.aspx. [Último acceso: 17 Septiembr 2012].

[14] Apache Incubator, «Cloudstack,» Apache Incubator, 2012. [En línea]. http://cloudstack.org/. [Último acceso: 26 Septiembre 2012].

[15] VMware, «Vcloud Director overview,» VMware, 2012. [En línea]. http://www.vmware.com/products/vcloud-director/overview.html. [Último acceso: 26 Septiembre 2012].

[16] D. Davis, «Virtualization Admin,» 25 mayo 2012. [En línea]. http://www.virtualizationadmin.com/articles-tutorials/cloud-computing/vmware-vcloud/getting-started-vmware-vcloud-director-part1.html. [Último acceso: 26 Septiembre 2012].

[17] E. S. Inc., «Eucalyptus,» 2012. [En línea]. http://www.eucalyptus.com/.

[Último acceso: 26 Septiembre 2012].

[18] Canonical, «What is OpenStack,» 8 Septiembre 2012. [En línea]. http://www.youtube.com/watch?feature=player_embedded&v=SnsWf0hyDXc#!. [Último acceso: 20 Obtubre 2012].

[19] C. Albarez Barba, M. Á. Ibáñez Mompeán, A. Molina Coballes y J. Moreno León, «Administración de OpenStack Essex: Instalación, configuración y explotación,» Murcia, 2012.

[20] OpenStack, «Conceptual Architecture,» 2012. [En línea]. http://docs.openstack.org/folsom/openstack-compute/admin/content/conceptual-architecture.html. [Último acceso: 16 Diciembre 2012].

[21] Laboratorio Nacional de Calida del Software, «Ingeniería del Software: Metodologías y Ciclos de Vida,» Inteco, 2009.

[22] C. M. M. y. C. I. Mullo, Estudio de la Plataforma FFMPEG y desarrollo de una aplicación cliente/ servidor para un Wall View, Quito: Escuela Politécnica Nacional, 2013.

[23] SAP noticias Chile, «SAP, Sala de prensa,» SAP, 11 12 2012. [En línea]. http://latam.news-sap.com/2012/12/11/revelan-un-estudio-sobre-la-competitividad-en-nuevas-tecnologias-de-pymes-exportadoras-en-latinoamerica/. [Último acceso: 08 10 2013].

[24] A. L. M. Cerdán, «El correo electrónico en las Pymes para la comunicación y gestión del conocimiento,» *Universia Bussines Review,* vol. 5, pp. 70-79, 2005.

[25] Red Global de Exportación, «Internet y las nuevas tecnologías como herramientas para las PyMEs exportadoras,» RGX, 2012.

[26] Intel, «TI aplicada a las PYMEs,» DiálogoTI.

[27] BBVA, «BBVA con tu empresa,» BBVA, 23 Octubre 2012. [En línea]. http://www.bbvacontuempresa.es/fondo/comunicaci%C3%B3n-y-movilidad/herramientas-de-colaboraci%C3%B3n-para-mejorar-la-eficiencia-de-las-pymes. [Último acceso: 2013 Noviembre 11].

[28] Q. Jiang, «Open Source IaaS Community Analysis,» Enero 2013. [En línea]. http://www.qyjohn.net/?p=2733. [Último acceso: Marzo 2013].

[29] S. Baset, «Open Source Cloud Technologies,» IBM, San Jose, California, 2012.

[30] Openstack Fundation, «Companies Supporting The OpenStack Foundation,» [En línea]. http://www.openstack.org/foundation/companies/. [Último acceso: Mayo 2013].

[31] K. Pepple, «OpenStack Folsom Architecture,» 25 Septiembre 2012. [En línea]. http://ken.pepple.info/openstack/2012/09/25/openstack-folsom-architecture/. [Último acceso: 2013 Enero 5].

[32] Nicira Inc. , «Network Virtualization Platform,» [En línea]. http://nicira.com/en/network-virtualization-platform. [Último acceso: Enero 2013].

[33] NEC, «ProgrammableFlow Network,» 2013. [En línea]. http://www.necam.com/SDN/. [Último acceso: Febrero 2013].

[34] OpenFlow Org., «Openflow,» [En línea]. http://www.openflow.org/. [Último acceso: Enero 2013].

[35] Nippon Telegraph and Telephone Corporation, «Ryu,» [En línea]. http://osrg.github.com/ryu/. [Último acceso: Enero 2013].

[36] OpenvSwitch Org., «OVS Quantum Plugin Documentation,» [En línea]. http://openvswitch.org/openstack/documentation/. [Último acceso: Enero 2013].

[37] S. I. Montero, «Curso de Django,» Maestros del WEB, [En línea]. http://www.maestrosdelweb.com/editorial/curso-django-entendiendo-como-trabaja-django/. [Último acceso: Abril 2013].

[38] Django Software Fundation, «Django Documentation, Forms,» [En línea]. https://docs.djangoproject.com/en/dev/topics/forms/. [Último acceso: Marzo 2013].

[39] Django Software Fundation, «Design philosophies,» [En línea]. https://docs.djangoproject.com/en/dev/misc/design-philosophies/. [Último acceso: Abril 2013].

[40] OpenStack Foundation, «Compute and Image System Requirements,» OpenStack Foundation, [En línea]. http://docs.openstack.org/trunk/openstack-compute/install/apt/content/compute-system-requirements.html.

[41] Canonical Corp., «Ubuntu Server,» Canonical, [En línea]. https://wiki.ubuntu.com/PrecisePangolin/ReleaseNotes/UbuntuServer. [Último acceso: Septiembre 2013].

[42] C. Corp., «Cisco OpenStack Edition: Folsom Manual Install,» [En línea]. http://docwiki.cisco.com/wiki/Cisco_OpenStack_Edition:_Folsom_Manual_Install. [Último acceso: Enero 2013].

[43] OpenStack Fundation, «OpenStack Basic Install,» [En línea]. http://docs.openstack.org/folsom/basic-install/content/. [Último acceso: Diciembre 2012].

[44] OpenStack Fundation, «OpenStack Install and Deply - Ubuntu,» 2012-2013. [En línea]. http://docs.openstack.org/trunk/openstack-compute/install/apt/content/. [Último acceso: Enero 2013].

[45] B. Msekni, «OpenStack_Folsom_Install_Guide_WebVersion,» [En línea]. https://github.com/mseknibilel/OpenStack-Folsom-Install-guide/blob/master/OpenStack_Folsom_Install_Guide_WebVersion.rst. [Último acceso: Enero 2013].

[46] Z. VanDuyn, «Basic OpenStack Folsom Install Guide,» [En línea]. http://openstack-folsom-install-guide.readthedocs.org/en/latest/. [Último acceso: Enero 2013].

[47] E. Macchi, «OpenStack Folsom Guide. Guide for Ubuntu Precise,» 2012.

[48] OpenStack Fundation, «Manage Security Groups,» de *OPENSTACK COMPUTE ADMINISTRATION MANUAL - FOLSOM, 2012.2*, 2012, p. 155.

[49] OpenStack Fundation, «Security Groups,» de *OpenStack Networking Administration Guide Grizzly,2013.1*, 2013, p. 60.

[50] Postfix Organization, «The Postfix Home Page,» [En línea]. http://www.postfix.org/. [Último acceso: 15 Junio 2013].

[51] Dovecot Organization, «Dovecot Secure IMAP server,» [En línea]. http://www.dovecot.org/. [Último acceso: 15 Junio 2013].

[52] «Roundcube - open ource webmail software,» [En línea]. http://roundcube.net/. [Último acceso: 15 Junio 2013].

[53] SourceFire Inc., «ClamAV - Home,» SourceFire Inc, 2002. [En línea]. http://www.clamav.net/lang/en/. [Último acceso: 15 Junio 2013].

[54] Apache Fundation, «The Apache SpamAssassin Project,» [En línea].

http://spamassassin.apache.org/index.html. [Último acceso: 15 Junio 2013].

[55] Organización IredMail , «Install IredMail on Ubuntu,» [En línea]. http://www.iredmail.org/install_iredmail_on_ubuntu.html. [Último acceso: Febreo 2013].

[56] Anonimo, «http://www.e79.ca/,» [En línea]. http://www.e79.ca/blog/linux/installing-sugarcrm-6-5-ce-on-ubuntu-12-04-amd64/. [Último acceso: Febrero 2013].

[57] A. Bustos, «Instalación de Apache OpenMeetings 2.x en Ubuntu 12.04 y 12.10 - 32 o 64 bits,» 15 Febrero 2013. [En línea]. https://cwiki.apache.org/OPENMEETINGS/tutoriales-en-espaol-relacionados-con-openmeetings.data/Instalacion%20OpenMeetings%202.x%20en%20Ubuntu%2012.10%20y%2012.04.pdf. [Último acceso: Abril 2013].

[58] Python Software Fundation, «python,» [En línea]. http://www.python.org/. [Último acceso: Marzo 2013].

[59] Django Software Fundation, «Writing your first Django app, part 1,» [En línea]. https://docs.djangoproject.com/en/1.3/intro/tutorial01/. [Último acceso: Marzo 2013].

[60] RackSpace, «Email Hosting,» [En línea]. http://www.rackspace.com/email-hosting/webmail/. [Último acceso: Mayo 2013].

[61] ZOHO Corp., «Zoho Mail,» [En línea]. http://www.zoho.com/mail/zohomail-pricing.html. [Último acceso: Mayo 2013].

[62] Google, «Google Apps for Bussines,» [En línea]. http://www.google.com/intx/es-419/enterprise/apps/business/pricing.html. [Último acceso: Mayo 2013].

[63] Go Daddy Operating Company, «GoDaddy- Email Hosting,» [En línea]. http://es.godaddy.com/email/email-hosting.aspx. [Último acceso: Mayo 2013].

[64] SugarCRM Inc., «Sugar Professional CRM,» [En línea]. https://store.sugarcrm.com/product/professional. [Último acceso: Mayo 2013].

[65] Vtiger Inc., «Vtiger - Features and Pricing,» [En línea]. https://www.vtiger.com/crm/crm-features-pricing/pricing/. [Último acceso: Mayo 2013].

[66] GigaPros, «Openmeetings Hosting,» [En línea]. http://www.gigapros.com/portal/openmeetings-hosting. [Último acceso: Mayo 2013].

[67] OpenStack Fundation, «Ceilometer - Wiki,» [En línea]. https://wiki.openstack.org/wiki/Ceilometer. [Último acceso: 20 Junio 2013].

[68] Paessler, «PRTG,» [En línea]. http://www.es.paessler.com/prtg. [Último acceso: 12 Julio 2013].

[69] Paessler AG, «PRTG Network Monitor User Manual,» Nuremberg, 2013.

[70] Instituto Ecuatoria de Estadisticas y Censos, «Análisis coyuntural N3,» INEC, 2011.

[71] Diario El Comercio, «El INEC presentó un directorio de empresas y establecimientos,» 21 Agosto 2012. [En línea]. http://www.elcomercio.com.ec/negocios/INEC-presento-directorio-empresas-establecimientos_0_759524097.html. [Último acceso: Mayo 2013].

[72] L. M. C. A. y. L. G. Baralla, «La situación de las Pymes en América Latina,»

Fundación Mediterranea, 2012.

[73] A. Catalán, «El camino para que las pymes lleguen a las nubes,» Economía y Negocios, [En línea]. http://www.economiaynegocios.cl/noticias/noticias.asp?id=102762. [Último acceso: Mayo 2013].

[74] Openstack Fundation, «HP Information,» [En línea]. http://www.openstack.org/foundation/companies/profile/hp. [Último acceso: Mayo 2013].

[75] Canonical Ltd., «Ubuntu and HP Models,» [En línea]. http://www.ubuntu.com/certification/make/HP/. [Último acceso: Mayo 2013].

[76] Magitech, «Servidor HP ProLiant DL385P Gen8 AMD Opteron 6272 2P 642135-001,» [En línea]. http://www.magitech.pe/pc/servidor-hp-proliant-dl385p-gen8-amd-opteron-6272-2p-32gb-r-p420i-2gb-hot-plug-25-sff-2x750w-ps-svr-642135-001.html. [Último acceso: Julio 2013].

[77] Hewlett-Packard Development Company, «HP ProLiant DL165 G7,» [En línea]. http://shopping1.hp.com/is-bin/INTERSHOP.enfinity/WFS/WW-USSMBPublicStore-Site/en_US/-/USD/ViewProductDetail-Start?ProductUUID=X44Q7EN5FH8AAAExyTMHkCKi&CatalogCategoryID=.M4Q7EN5ThgAAAEufk8oLTIM. [Último acceso: Mayo 2013].

[78] Hewlett-Packard Development Company, «HP ProLiant DL120 G7,» [En línea]. http://shopping1.hp.com/is-bin/INTERSHOP.enfinity/WFS/WW-USSMBPublicStore-Site/en_US/-/USD/ViewStandardCatalog-Browse?CatalogCategoryID=PBkQ7EN5viYAAAEubq0oLTpy&jumpid=reg_r1002_usen_c-001_title_r0001. [Último acceso: Mayo 2013].

[79] Hewlett-Packard Development Company, «HP 1TB 6G SAS 7.2K rpm SFF,» [En línea]. http://shopping1.hp.com/is-

bin/INTERSHOP.enfinity/WFS/WW-USSMBPublicStore-Site/en_US/-/USD/ViewParametricSearch-SimpleOfferSearch?webform-id=WFSimpleSearch&DefaultButton=findSimple&WFSimpleSearch_NameOrID=605835-B21&findSimple=. [Último acceso: Mayo 2013].

[80] Hewlett-Packard Development Company, «HP 16GB Quad Rank,» [En línea]. http://h30094.www3.hp.com/product.aspx?cache=1809167456&culture=en-us&sku=10238517. [Último acceso: Mayo 2013].

[81] Cisco Systems, «Cisco 1841 Router,» [En línea]. http://www.cisco.com/en/US/prod/collateral/routers/ps5853/product_data_sheet0900aecd8016a59b.html. [Último acceso: Mayo 2013].

[82] Cisco Systems, «Cisco Catalyst 2960-S,» [En línea]. http://www.cisco.com/en/US/prod/collateral/switches/ps5718/ps6406/product_data_sheet0900aecd806b0bd8.html. [Último acceso: Mayo 2013].

[83] OpenStack Fundation, «Overcommitting,» [En línea]. http://docs.openstack.org/trunk/openstack-ops/content/overcommit.html. [Último acceso: Mayo 2013].

[84] Servicio de Rentas Internas, «Reglamento de Aplicación de la Ley de Régimen Tributario Interno. Art. 28.- Gastos generales deducibles,» [En línea]. http://www.sri.gob.ec/web/guest/depreciacion-acelarada-de-activos-fijos. [Último acceso: 05 Junio 2013].

[85] Corporación Financiera Nacional, «TASAS DE INTERÉS PRIMER PISO - Junio 2013,» Junio 2013. [En línea]. http://www.cfn.fin.ec/images/stories/pdfs/tasas_interes_primer_piso_junio2013.pdf. [Último acceso: 10 Junio 2013].

[86] Banco Central del Ecuador, «Tasas de Interés. Junio 2013,» [En línea].

http://www.bce.fin.ec/docs.php?path=documentos/Estadisticas/SectorMonFin/TasasInteres/Indice.htm. [Último acceso: 08 Junio 2013].

[87] M. Witt, Interviewee, *Cloud Services at Yahoo.* [Entrevista]. 22 Agosto 2013.

[88] Licencias OnLine, «Modelos de Servicio Cloud - SaaS, IaaS, PaaS,» 2013. [En línea]. http://www.licenciasonline.com/ec/es/cloud/modelos-de-servicio. [Último acceso: 16 Junio 2013].

[89] Puppet Labs, «Proyecto Puppet,» Puppet Labs, [En línea]. http://projects.puppetlabs.com/projects/puppet. [Último acceso: Noviembre 2013].

[90] Opscode Inc., «Chef,» Opscode, [En línea]. http://www.opscode.com/chef/. [Último acceso: Noviembre 2013].

[91] HashiCorp, «Vagrant,» 2013. [En línea]. http://www.vagrantup.com/. [Último acceso: Noviembre 2013].

[92] Canonical Ltd., «Juju Service Orquestation,» 2013. [En línea]. http://www.ubuntu.com/cloud/tools/juju. [Último acceso: Noviembre 2013].

[93] OpenStack Fundation, «How To License The Powered By OpenStack Logo,» [En línea]. http://www.openstack.org/brand/powered-by-openstack/. [Último acceso: Abril 2013].

[94] SugarCRM Inc., «SugarCRM Partner Program,» [En línea]. http://www.sugarcrm.com/partners. [Último acceso: Mayo 2013].

[95] IredMail Project, «iRedAdmin-Pro (web-based admin panel for iRedMail, full-featured edition),» 2012. [En línea]. http://www.iredmail.org/pricing.html. [Último acceso: Marzo 2013].

[96] BareMetalCloud, «Servidores Dedicados,» [En línea]. http://baremetalcloud.com/index.php/es/hardware-es/servidores-dedicados. [Último acceso: Marzo 2013].

[97] HostLocation, «Servidores Dedicados,» 2010. [En línea]. http://www.hostlocation.com/planos/dedicados.asp. [Último acceso: Junio 2013].

[98] M. O. y. Jacob, «Bootstrap,» [En línea]. http://twitter.github.io/bootstrap/. [Último acceso: 16 Junio 2013].

[99] C. C. U. C. D. Group, «White Paper Cloud Computing Use Cases Version 2.0,» Creative Commons, 2009.

[100] F. Páez, «http://www.cmigestion.es/2009/gestion-empresarial/virtualizacion-de-servidores-ahorro-de-costes-y-mayor-eficiencia/,» 22 Mayo 2009. [En línea]. http://www.cmigestion.es/blog2/images/Virtualizacion.jpg. [Último acceso: 15 Septiembre 2012].

[101] J. Smaldone, «http://blog.smaldone.com.ar/,» 20 Septiembre 2008. [En línea]. http://blog.smaldone.com.ar/2008/09/20/virtualizacion-de-hardware/. [Último acceso: 15 Septiembre 2012].

[102] O. KVM, «kvm.org,» 2011. [En línea]. http://www.linux-kvm.org/page/Main_Page. [Último acceso: 16 Septiembre 2012].

[103] M. Helsley, «LXC: Linux container tools,» IBM Corpotation, 2009.

[104] O. Qemu, «http://wiki.qemu.org/Main_Page,» [En línea]. [Último acceso: 16 Septiembre 2012].

[105] V. Inc., «http://vmware.connectedsocialmedia.com/,» [En línea]. http://media8.connectedsocialmedia.com/11/PID_013833/Podtech_VMware

_Disaster_Recovery_Datac.jpg. [Último acceso: 16 Septiembre 2012].

[106] O. XEN, «http://www.xen.org/,» [En línea]. [Último acceso: 16 Septiembre 2012].

[107] Soumyasch, «wikipedia.org,» 3 Enero 2008. [En línea]. http://commons.wikimedia.org/wiki/File:Viridian_Architecture.svg?uselang=es. [Último acceso: 16 Septiembre 2012].

[108] Microsoft, «http://blog.soporteti.net/hyper-v/hyper-v-configuracion-basica-de-una-maquina-virtual-video-tutorial/,» 20 Junio 2012. [En línea]. http://blog.soporteti.net/wp-content/uploads/2012/06/hyper-v.png. [Último acceso: 17 Septiembre 2012].

[109] N. especificado, «/www.tecnicobat.com,» 11 Abril 2011. [En línea]. http://www.tecnicobat.com/2012/04/11/entendiendo-la-arquitectura-de-hyper-v/. [Último acceso: 17 Septiembre 2012].

[110] Organización Wikipedia, «www.wikipedia.org,» 28 Julio 2012. [En línea]. http://es.wikipedia.org/wiki/Hyper-V. [Último acceso: 17 Septiembre 2012].

[111] Microsoft Corporation, «http://technet.microsoft.com,» Marzo 2010. [En línea]. http://technet.microsoft.com/es-es/library/cc816638(v=ws.10).aspx. [Último acceso: 17 Septiembre 2012].

[112] RackSpace, «Openstack Software,» RackSpace, [En línea]. http://www.rackspace.com/cloud/private/openstack_software/. [Último acceso: 2 Febrero 2013].

[113] R. C. Builders, «devstack,» [En línea]. http://devstack.org/. [Último acceso: 2 Febrero 2013].

[114] StackOps, «stackops-distro-community-edition,» StackOps, [En línea]. http://www.stackops.com/products/stackops-distro-community-edition/.

[Último acceso: 2 Febrero 2013].

[115] K. Sunitha, «answers.launchpad.net,» [En línea]. https://answers.launchpad.net/quantum/+question/210248. [Último acceso: Enero 2013].

[116] OpenStack Fundation, «Nova Concepts and Introduction,» [En línea]. http://docs.openstack.org/developer/nova/nova.concepts.html. [Último acceso: Enero 2013].

[117] F. Rodenas, «El diario de Ferdy,» [En línea]. http://www.rodenas.org/ferdyblog/2011/04/21/openstack-cactus/. [Último acceso: Enero 2013].

[118] D. Parrilla, «UN VISTAZO RÁPIDO A LA HISTORIA DE OPENSTACK,» [En línea]. http://www.nubeblog.com/2011/01/10/un-vistazo-rapido-a-la-historia-de-openstack/. [Último acceso: Enero 2013].

[119] Django Software Foundation, «Django Project,» [En línea]. https://www.djangoproject.com/. [Último acceso: Febrero 2013].

[120] Microsoft Corporation, «Windows Server 2008 R2 Hyper-V Component Architecture RTM COMPLETED SP1 RTM for Printing,» 2010.

[121] s. (alias), «OpenStack running on my Laptop !,» 8 Septiembre 2012. [En línea]. http://ssklogs.blogspot.com/2012/09/openstack-running-on-my-laptop.html. [Último acceso: 16 Marzo 2013].

[122] The Apache Software Foundation, «Commercial Support for OpenMeetings,» 2012. [En línea]. http://openmeetings.apache.org/commercial-support.html. [Último acceso: Junio 2013].

[123] K. Pepple, «Ken Pepple Bog,» GitHub Pages, 25 Septiembre 2012. [En

línea]. http://26a0ff8ca8ba32139f7d-db711c577a50b6bdc946ea71aaca027d.r97.cf1.rackcdn.com/openstack-logical-arch-folsom.jpg. [Último acceso: 13 Noviembre 2012].

[124] M. T. Jones, «An overview of virtualization methods, architectures, and implementations,» 29 DIciembre 2006. [En línea]. http://www.ibm.com/developerworks/library/l-linuxvirt/. [Último acceso: Abril 2012].

[125] Universidad Politécnica de Madrid, «Entornos de programación. Conceptos, funciones y tipos.,» [En línea]. http://lml.ls.fi.upm.es/ep/entornos.html. [Último acceso: Mayo 2013].

ANEXOS

ANEXO A

INSTALACIÓN Y CONFIGURACIÓN DE LA INFRAESTRUCTURA OPENSTACK

PREPARACIÓN DEL SISTEMA

Es necesario ejecutar el Código A-1 en cada uno de los nodos para preparar el sistema para su actualización y posteriormente, descargar los paquetes del repositorio oficial del proyecto OpenStack de Cisco.

```
1 root@Controller:~# nano /etc/apt/sources.list.d/cisco-OpenStack-mirror_folsom-proposed.list
2 deb http://128.107.252.163/OpenStack/cisco folsom-proposed main
3 deb-src http://128.107.252.163/OpenStack/cisco folsom-proposed main
4 root@Controller:~# gpg --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys E8CC67053ED3B199
5 root@Controller:~# gpg --armor --export E8CC67053ED3B199 | apt-key add -
6 root@Controller:~# apt-get update && apt-get dist-upgrade -y
```

Código A-1 Preparación del sistema

En la línea 1 se crea un nuevo fichero que contiene las fuentes de los repositorios de OpenStack CISCO que se especifica en las líneas 2 y 3. En la línea 4 y 5 se agrega la clave pública del repositorio Cisco OpenStack y se valida ante los servidores de Ubuntu. Por último en la línea 6 se actualiza el sistema y la distribución para que se descarguen los paquetes del nuevo repositorio.

```
1 root@compute:~# apt-get install -y cpu-checker
2 root@compute:~# kvm-ok
3 INFO: /dev/kvm exists
4 KVM acceleration can be used
```

Código A-2 Verificación de soporte de virtualización

Para que un computador pueda cumplir la función de nodo de cómputo, su procesador debe tener la capacidad de soportar virtualización de forma nativa. La mayoría de procesadores actuales tienen esta característica pero es posible activarla o desactivarla en la configuración de la tarjeta madre. En este caso, es necesario instalar una herramienta para verificar si KVM puede ejecutar virtualización, como se observa en la línea 1 del Código A-2. Posteriormente se ejecuta el comando de la línea 2, en caso afirmativo, se mostrarán las líneas 3 y 4

que indican que se puede proceder con la instalación de los componentes de KVM.

NODO CONTROLLER

Configuración de Red

Editar el archivo */etc/network/interfaces* con las líneas que se muestran en el Código A-3, las cuales permiten configurar las interfaces de red con la información presentada anteriormente.

```
auto lo
iface lo inet loopback
auto eth2
iface eth2 inet static
       address 172.31.15.167
       netmask 255.255.255.0
       network 172.31.15.0
       broadcast 172.31.15.255
       gateway 172.31.15.251
       dns-nameservers 8.8.8.8
auto eth0
iface eth0 inet static
address 10.0.0.43
netmask 255.255.255.0
auto eth1
iface eth1 inet static
address 192.168.221.43
netmask 255.255.255.0
```

Código A-3 Interfaces de Red del nodo *Controller*

Sincronización de Tiempo

En el Código A-4 se muestra los comandos para convertir al nodo *Controller* en un servidor de tiempo para que los demás nodos puedan solicitarle información de fecha y hora.

```
apt-get install -y ntp
sed -i 's/server ntp.ubuntu.com/server ntp.ubuntu.com\nserver 127.127.1.0\nfudge
127.127.1.0 stratum 10/g' /etc/ntp.conf
service ntp restart
```

Código A-4 Configuración NTP en el nodo *Controller*

MySQL

En el Código A-5 se muestra la instalación del servidor de base de datos MySQL. En la línea 1 se instala los paquetes necesarios para el funcionamiento del motor de base de datos, en la línea 2 se indica que el servidor pueda responder peticiones desde cualquier origen y en la línea 3 se reinicia el servicio.

```
1 apt-get install -y mysql-server python-mysqldb
2 sed -i 's/127.0.0.1/0.0.0.0/g' /etc/mysql/my.cnf
3 service mysql restart
```

Código A-5 Instalación y configuración de MySQL

En la línea 1 del Código A-6, se accede a la consola de MySQL, entre las línea 2 a 6 se crean las bases de datos de cada módulo de OpenStack. Entre las líneas 7 a 11 se crean usuarios y se les concede todos los permisos sobre las bases de datos creadas. Por facilidad en la instalación se decidió usar una única contraseña (“daniel”) para todas las bases de datos, aunque esto pondría en riesgo una implementación en producción, por lo que no se recomienda usar una única contraseña.

```
1  mysql -u root -pdaniel
2  mysql>CREATE DATABASE keystone;
3  mysql>CREATE DATABASE glance;
4  mysql>CREATE DATABASE quantum;
5  mysql>CREATE DATABASE nova;
6  mysql>CREATE DATABASE cinder;
7  mysql>GRANT ALL ON keystone.* TO 'keystone_admin'@'172.31.15.167' IDENTIFIED BY
8  'daniel';
9  mysql>GRANT ALL ON glance.* TO 'glance'@'172.31.15.167' IDENTIFIED BY 'daniel';
10 mysql>GRANT ALL ON quantum.* TO 'quantum'@'172.31.15.167' IDENTIFIED BY 'daniel';
11 mysql>GRANT ALL ON nova.* TO 'nova'@'172.31.15.167' IDENTIFIED BY 'daniel';
12 mysql>GRANT ALL ON cinder.* TO 'cinder'@'172.31.15.167' IDENTIFIED BY 'daniel';
   mysql>quit;
```

Código A-6 Creación de bases de datos y sus permisos

RabbitMQ

En el Código A-7 se detalla la instalación de RabbitMQ que servirá como servidor de encolamiento de mensajes. Los demás servicios se comunican con él por lo que se necesita cierto nivel de seguridad y segmentación de los mensajes para un mejor desempeño.

```
1 root@Controller:~# apt-get install -y rabbitmq-server
2 root@Controller:~# rabbitmqctl add_user OpenStack_rabbit_user daniel
3 root@Controller:~# rabbitmqctl set_permissions -p / OpenStack_rabbit_user ".*" ".*" ".*"
4 root@Controller:~# rabbitmqctl add_vhost /quantum
5 root@Controller:~# rabbitmqctl set_permissions -p /quantum OpenStack_rabbit_user ".*"
6 ".*" ".*"
```

CódigoA- 7 Configuración de RabbitMQ

En la línea 2 se crea el usuario “*OpenStack_rabbit_user*” con la contraseña “daniel” para que los servicios hagan uso autenticándose previamente del servicio RabbitMQ, como se observa en la línea 3. Debido a que hay una carga mayor del

módulo Quantum, en la línea 4 se crea un Vhost[82] para poder separar las peticiones del resto de servicios. De igual manera estas peticiones son autenticadas con el usuario "*OpenStack_rabbit_user*" como se observa en la línea 5.

Instalación de KEYSTONE

El Código A-8 describe el proceso de instalación y configuración del servicio Keystone.

```
1  root@Controller:~# apt-get install -y keystone
2  admin_token   = keystone_admin_token
3  connection = mysql://keystone_admin:daniel@172.31.15.167/keystone
4  root@Controller:~# service keystone restart
5  root@Controller:~# keystone-manage db_sync
6  ADMIN_PASSWORD=${ADMIN_PASSWORD:-daniel}
7  export SERVICE_TOKEN="keystone_admin_token"
8  export SERVICE_ENDPOINT="http://172.31.15.167:35357/v2.0"
9  SERVICE_TENANT_NAME=${SERVICE_TENANT_NAME:-services}
10 root@Controller:~# chmod +x keystone-data.sh
11 root@Controller:~# ./keystone-data.sh
12 MYSQL_USER=keystone_admin
13 MYSQL_DATABASE=keystone
14 MYSQL_HOST=172.31.15.167
15 MYSQL_PASSWORD=daniel
16 KEYSTONE_REGION=RegionOne
17 SERVICE_TOKEN=keystone_admin_token
18 SERVICE_ENDPOINT="http://172.31.15.167:35357/v2.0"
19 MASTER="172.31.15.167"
20 root@Controller:~# export SERVICE_TOKEN="keystone_admin_token"
21 root@Controller:~# export SERVICE_ENDPOINT="http://172.31.15.167:35357/v2.0/"
22 root@Controller:~# chmod +x keystone-data.sh
23 root@Controller:~# ./keystone-data.sh
24 root@Controller:~# nano openrc
25 export OS_TENANT_NAME=admin
26 export OS_USERNAME=admin
27 export OS_PASSWORD=daniel
28 export OS_AUTH_URL="http://172.31.15.167:5000/v2.0/"
29 export OS_AUTH_STRATEGY=keystone
30 export SERVICE_TOKEN=keystone_admin_token
31 export SERVICE_ENDPOINT=http://172.31.15.167:35357/v2.0/
32 root@Controller:~# source openrc
```

Código A-8 Configuración de Keystone

En la línea 1 se instalan los paquetes necesarios. Se edita el fichero */etc/keystone/keystone.conf* con la información de las líneas 2 y 3. En la línea 4 y 5 se reinicia el servicio y se sincroniza la base de datos, respectivamente.

[82] Vhost es una forma de agrupar colas, mensajes y permisos de usuarios que no podrán ser vistas o usadas por otro vhost. Es de gran utilidad si se desea compartir el mismo servidor de encolamiento.

Mediante el *script keystone-data.sh* que se muestra en el Código A-9, se editan las variables que se muestran en las líneas 6 a 9. Se da permisos de ejecución al fichero recién creado y se lo ejecuta en la línea 11.

```
1  #!/bin/sh
2  # Keystone Datas
3  # Description: Fill Keystone with datas.
4  # Mainly inspired by http://www.hastexo.com/resources/docs/installing-OpenStack-essex-20121-ubuntu-
5  1204-precise-pangolin
6  # Written by Martin Gerhard Loschwitz / Hastexo
7  # Modified by Emilien Macchi / StackOps
8  # Support: OpenStack@lists.launchpad.net
9  # License: Apache Software License (ASL) 2.0
10 #
11 ADMIN_PASSWORD=${ADMIN_PASSWORD:-password}
12 SERVICE_PASSWORD=${SERVICE_PASSWORD:-$ADMIN_PASSWORD}
13 export SERVICE_TOKEN="password"
14 export SERVICE ENDPOINT="http://localhost:35357/v2.0"
15 SERVICE TENANT NAME=${SERVICE TENANT NAME:-service}
16
17 get_id () {
18     echo `$@ | awk '/ id / { print $4 }'`
19 }
20 # Tenants
21 ADMIN_TENANT=$(get_id keystone tenant-create --name=admin)
22 SERVICE_TENANT=$(get_id keystone tenant-create --name=$SERVICE_TENANT_NAME)
23 DEMO_TENANT=$(get_id keystone tenant-create --name=demo)
24 INVIS_TENANT=$(get_id keystone tenant-create --name=invisible_to_admin)
25 # Users
26 ADMIN_USER=$(get_id keystone user-create --name=admin --pass="$ADMIN_PASSWORD" --
27 email=admin@domain.com)
28 DEMO_USER=$(get_id keystone user-create --name=demo --pass="$ADMIN_PASSWORD" --
29 email=demo@domain.com)
30 # Roles
31 ADMIN_ROLE=$(get_id keystone role-create --name=admin)
32 KEYSTONEADMIN_ROLE=$(get_id keystone role-create --name=KeystoneAdmin)
33 KEYSTONESERVICE_ROLE=$(get_id keystone role-create --name=KeystoneServiceAdmin)
34 # Add Roles to Users in Tenants
35 keystone user-role-add --user-id $ADMIN_USER --role-id $ADMIN_ROLE --tenant-id $ADMIN_TENANT
36 keystone user-role-add --user-id $ADMIN USER --role-id $ADMIN ROLE --tenant-id $DEMO TENANT
37 keystone user-role-add --user-id $ADMIN_USER --role-id $KEYSTONEAD-MIN_ROLE --tenant-id
38 $ADMIN_TENANT
39 keystone user-role-add --user-id $ADMIN_USER --role-id $KEYSTONESERV-ICE_ROLE --tenant-id
40 $ADMIN_TENANT
41 # The Member role is used by Horizon and Swift
42 MEMBER_ROLE=$(get_id keystone role-create --name=Member)
43 keystone user-role-add --user-id $DEMO USER --role-id $MEMBER ROLE --tenant-id $DEMO TENANT
44 keystone user-role-add --user-id $DEMO USER --role-id $MEMBER ROLE --tenant-id $INVIS TENANT
45 # Configure service users/roles
46 NOVA_USER=$(get_id keystone user-create --name=nova --pass="$SERVICE_PASSWORD" --tenant-id
47 $SERVICE_TENANT --email=nova@domain.com)
48 keystone user-role-add --tenant-id $SERVICE_TENANT --user-id $NOVA_USER --role-id $ADMIN_ROLE
49
50 GLANCE USER=$(get id keystone user-create --name=glance --pass="$SERVICE PASSWORD" --tenant-id
51 $SERVICE_TENANT --email=glance@domain.com)
52 keystone user-role-add --tenant-id $SERVICE_TENANT --user-id $GLANCE_USER --role-id $ADMIN_ROLE
53
54 SWIFT_USER=$(get_id keystone user-create --name=swift --pass="$SERVICE_PASSWORD" --tenant-id
55 $SERVICE_TENANT --email=swift@domain.com)
56 keystone user-role-add --tenant-id $SERVICE_TENANT --user-id $SWIFT_USER --role-id $ADMIN_ROLE
57
58 RESELLER ROLE=$(get id keystone role-create --name=ResellerAdmin)
59 keystone user-role-add --tenant-id $SERVICE_TENANT --user-id $NOVA_USER --role-id $RESELLER_ROLE
60
61 QUANTUM_USER=$(get_id keystone user-create --name=quantum --pass="$SERVICE_PASSWORD" --tenant-id
62 $SERVICE_TENANT --email=quantum@domain.com)
63 keystone user-role-add --tenant-id $SERVICE TENANT --user-id $QUAN-TUM USER --role-id $ADMIN ROLE
64
65 CINDER USER=$(get id keystone user-create --name=cinder --pass="$SERVICE PASSWORD" --tenant-id
66 $SERVICE_TENANT --email=cinder@domain.com)
67 keystone user-role-add --tenant-id $SERVICE_TENANT --user-id $CINDER_USER --role-id $ADMIN_ROLE
```

Código A-9 Script para generación de credenciales de Keystone

Mediante el script keystone-endpoints.sh que se muestra en el Código A-10, se editan las variables que se muestran en las líneas 12 a 19. Después es necesario exportar las credenciales de Keystone como se observa en las líneas 20 y 21, posteriormente se asignan los permisos de ejecución al fichero recién creado y se ejecuta en la línea 23. En la pantalla se deberá observar cómo se crean los usuarios, tenants[83] y endpoints[84] correspondientes a casa servicio de OpenStack. En la línea 24 se crea un fichero que contiene las variables de ambiente, para poder cargarlas con facilidad cuando sea necesario como se lo realiza en la línea 32, las variables de ambiente del archivo se detallan entre la línea 25 a la 31.

```
1  #!/bin/sh
2  # Keystone Endpoints
3  # Description: Create Services Endpoints
4  # Mainly inspired by http://www.hastexo.com/resources/docs/installing-OpenStack-essex-20121-ubuntu-
5  1204-precise-pangolin
6  # Written by Martin Gerhard Loschwitz / Hastexo
7  # Modified by Emilien Macchi / StackOps
8  # Support: OpenStack@lists.launchpad.net
9  # License: Apache Software License (ASL) 2.0
10 # MySQL definitions
11 MYSQL_USER=keystone
12 MYSQL_DATABASE=keystone
13 MYSQL_HOST=localhost
14 MYSQL_PASSWORD=password
15 # Keystone definitions
16 KEYSTONE_REGION=RegionOne
17 SERVICE_TOKEN=password
18 SERVICE_ENDPOINT="http://localhost:35357/v2.0"
19 # other definitions
20 MASTER="192.168.0.1"
21
22 while getopts "u:D:p:m:K:R:E:S:T:vh" opt; do
23   case $opt in
24     u)
25       MYSQL_USER=$OPTARG
26       ;;
27     D)
28       MYSQL_DATABASE=$OPTARG
29       ;;
30     p)
31       MYSQL_PASSWORD=$OPTARG
32       ;;
33     m)
34       MYSQL_HOST=$OPTARG
35       ;;
36     K)
37       MASTER=$OPTARG
38       ;;
39     R)
40       KEYSTONE_REGION=$OPTARG
41       ;;
42     E)
43       export SERVICE_ENDPOINT=$OPTARG
44       ;;
45     S)
46       SWIFT_MASTER=$OPTARG
```

[83] OpenStack denomina *tenant* a un contenedor de recursos que forma la estructura de la organización dentro del servicio *Compute*.

[84] OpenStack denomina *endpoint* al URL y puerto que Keystone define para un servicio de Openstack.

```
      ;;
    T)
      export SERVICE TOKEN=$OPTARG
      ;;
    v)
      set -x
      ;;
    h)
      cat <<EOF
Usage: $0 [-m mysql hostname] [-u mysql username] [-D mysql database] [-p mysql password]
       [-K keystone_master ] [ -R keystone_region ] [ -E keystone_endpoint_url ]
       [ -S swift_master ] [ -T keystone_token ]

Add -v for verbose mode, -h to display this message.
EOF
      exit 0
      ;;
    \?)
      echo "Unknown option -$OPTARG" >&2
      exit 1
      ;;
    :)
      echo "Option -$OPTARG requires an argument" >&2
      exit 1
      ;;
  esac
done
if [ -z "$KEYSTONE_REGION" ]; then
  echo "Keystone region not set. Please set with -R option or set KEY-STONE_REGION variable." >&2
  missing_args="true"
fi
if [ -z "$SERVICE TOKEN" ]; then
  echo "Keystone service token not set. Please set with -T option or set SER-VICE_TOKEN variable." >&2
  missing_args="true"
fi
if [ -z "$SERVICE_ENDPOINT" ]; then
  echo "Keystone service endpoint not set. Please set with -E option or set SER-VICE ENDPOINT
variable." >&2
  missing args="true"
fi
if [ -z "$MYSQL_PASSWORD" ]; then
  echo "MySQL password not set. Please set with -p option or set MYSQL_PASSWORD variable." >&2
  missing_args="true"
fi
if [ -n "$missing_args" ]; then
  exit 1
fi
keystone service-create --name nova --type compute --description 'OpenStack Com-pute Service'
keystone service-create --name cinder --type volume --description 'OpenStack Vol-ume Service'
keystone service-create --name glance --type image --description 'OpenStack Image Service'
keystone service-create --name swift --type object-store --description 'OpenStack Storage Service'
keystone service-create --name keystone --type identity --description 'OpenStack Identity'
keystone service-create --name ec2 --type ec2 --description 'OpenStack EC2 service'
keystone service-create --name quantum --type network --description 'OpenStack Networking service'
create_endpoint () {
  case $1 in
    compute)
    keystone endpoint-create --region $KEYSTONE_REGION --service-id $2 --publicurl
'http://'"$MASTER"':8774/v2/$(tenant id)s' --adminurl 'http://'"$MASTER"':8774/v2/$(tenant id)s' --
internalurl 'http://'"$MASTER"':8774/v2/$(tenant_id)s'
    ;;
    volume)
    keystone endpoint-create --region $KEYSTONE_REGION --service-id $2 --publicurl
'http://'"$MASTER"':8776/v1/$(tenant_id)s' --adminurl 'http://'"$MASTER"':8776/v1/$(tenant_id)s' --
internalurl 'http://'"$MASTER"':8776/v1/$(tenant_id)s'
    ;;
    image)
    keystone endpoint-create --region $KEYSTONE REGION --service-id $2 --publicurl
'http://'"$MASTER"':9292/v2' --adminurl 'http://'"$MASTER"':9292/v2' --internalurl
'http://'"$MASTER"':9292/v2'
    ;;
    object-store)
    if [ $SWIFT MASTER ]; then
      keystone endpoint-create --region $KEYSTONE REGION --service-id $2 --publicurl
'http://'"$SWIFT_MASTER"':8080/v1/AUTH_$(tenant_id)s' --adminurl 'http://'"$SWIFT_MASTER"':8080/v1' --
internalurl 'http://'"$SWIFT_MASTER"':8080/v1/AUTH_$(tenant_id)s'
    else
      keystone endpoint-create --region $KEYSTONE_REGION --service-id $2 --publicurl
'http://'"$MASTER"':8080/v1/AUTH_$(tenant_id)s' --adminurl 'http://'"$MASTER"':8080/v1' --internalurl
'http://'"$MASTER"':8080/v1/AUTH_$(tenant_id)s'
```

```
127     fi
128     ;;
129     identity)
130     keystone endpoint-create --region $KEYSTONE_REGION --service-id $2 --publicurl
131 'http://'"$MASTER"':5000/v2.0' --adminurl 'http://'"$MASTER"':35357/v2.0' --internalurl
132 'http://'"$MASTER"':5000/v2.0'
133     ;;
134     ec2)
135     keystone endpoint-create --region $KEYSTONE_REGION --service-id $2 --publicurl
136 'http://'"$MASTER"':8773/services/Cloud' --adminurl 'http://'"$MASTER"':8773/services/Admin' --
137 internalurl 'http://'"$MASTER"':8773/services/Cloud'
138     ;;
139     network)
140     keystone endpoint-create --region $KEYSTONE_REGION --service-id $2 --publicurl
141 'http://'"$MASTER"':9696/' --adminurl 'http://'"$MASTER"':9696/' --internalurl
142 'http://'"$MASTER"':9696/'
143     ;;
144   esac
145 }
146 for i in compute volume image object-store identity ec2 network; do
147   id=`mysql -h "$MYSQL_HOST" -u "$MYSQL_USER" -p"$MYSQL_PASSWORD" "$MYSQL_DATABASE" -ss -e "SELECT id
148 FROM service WHERE type='"$i"';"` || exit 1
149   create endpoint $i $id
150 done
```

Código A-10 Script para la creación de Endpoints de Keystone

Instalación de GLANCE

En el Código A-11 se describe el proceso de instalación y configuración del servicio Glance.

```
1  root@Controller:~# apt-get install -y glance
2  paste.filter_factory = keystone.middleware.auth_token:filter_factory
3  auth_host = 172.31.15.167
4  auth_port = 35357
5  auth_protocol = http
6  admin_tenant_name = services
7  admin_user = glance
8  admin_password = daniel
9  sql_connection = mysql://glance:daniel@172.31.15.167/glance
10 flavor = keystone
11 root@Controller:~# service glance-api restart; service glance-registry restart
12 glance-manage db_sync
13 service glance-api restart; service glance-registry restart
14 root@Controller:~# wget http://cloud-images.ubuntu.com/precise
15 /current/precise-server-cloudimg-amd64-disk1.img
16 root@Controller:~# glance add name="precise" is_public=true container_format=ovf
17 disk_format=qcow2 < precise-server-cloudimg-amd64-disk1.img
18 root@Controller:~# glance image-list
19 +--------------------+---------+-------------+------------------+-----------+--------+
20 | ID                 | Name    | Disk Format | Container Format | Size      | Status |
21 +--------------------+---------+-------------+------------------+-----------+--------+
22 | 6b1a40e4-6416-4f27 | precise | qcow2       | ovf              | 250150912 | active |
23 +--------------------+---------+-------------+------------------+-----------+--------+
```

Código A-11 Configuración de Glance

En la línea 1 se instala los paquetes necesarios de Glance. A continuación se añade la información de las líneas 2 a 8 en los ficheros */etc/glance/glance-api-paste.ini* y */etc/glance/glance-registry-paste.ini* para autenticar el servicio Glance ante Keystone. Después, se modifican los archivos */etc/glance/glance-api.conf* y */etc/glance/glance-registry.conf* añadiendo la información de las líneas 9 y 10 para indicar la ubicación y credenciales de conexión con la base de datos. Finalmente

se sincroniza la base de datos y se reinician los servicios con la finalidad de que surtan efecto los nuevos cambios, lo cual se realiza en las líneas 11 a 13.

Al ya tener instalado y configurado Glance, ya se puede agregar una imagen al repositorio. Para esto primeramente se descarga una imagen, en la línea 16 y 17 se presenta la descarga de la imagen del sistema operativo Ubuntu 12.04. En la línea 18 y 19 se presenta el comando para agregar una imagen al repositorio de imágenes de Glance. Como se puede observar, el comando de la línea 18 muestra que efectivamente la imagen de Ubuntu 12.04 se encuentra disponible para su uso.

Instalación de QUANTUM

En el Código A-12 se describe el proceso de instalación y configuración de los servicios de red, tanto de Quantum como de Open vSwitch.

```
1  root@Controller:~# apt-get install -y quantum-server quantum-plugin-openvswitch
2  allow_overlapping_ips = False
3  fake_rabbit = False
4  rabbit_virtual_host=/quantum
5  rabbit_userid=OpenStack rabbit user
6  rabbit_password=daniel
7  rabbit_host=172.31.15.167
8  rabbit_port=5672
9  sql_connection = mysql://quantum:daniel@172.31.15.167/quantum
10 enable_tunneling = True
11 tunnel_id_ranges = 1:1000
12 integration_bridge=br-int
13 tunnel_bridge = br-tun
14 network_vlan_ranges=
15 tenant_network_type=gre
16 [filter:authtoken]
17 paste.filter_factory = keystone.middleware.auth_token:filter_factory
18 auth_host=172.31.15.167
19 auth_port = 35357
20 auth_protocol = http
21 admin_tenant_name=services
22 admin_user=quantum
23 admin_password=daniel
24 root@Controller:~# service quantum-server restart
```

Código A-12 Configuración de Quantum en el nodo *Controller*

En la línea 1 está la instalación de los paquetes de Quantum y el *plugin* de Open vSwitch. Se edita el fichero */etc/quantum/quantum.conf* y se agregan las líneas 2 a 8 para habiliatar la autenticación ante el servidor de encolamiento de mensajes. El *plugin* de Open vSwitch se configura para que trabaje con túneles GRE y puentes entre las interfaces virtuales y físicas. Para ello se edita el fichero

/etc/quantum/plugins/openvswitch/ovs_quantum_plugin.ini con la información de las líneas 9 a 15.

Se agrega las credenciales de autenticación para identificarse ante Keystone, editando el fichero */etc/quantum/api-paste.ini* con las líneas 16 a 23. Y por último se reinicia el servicio Quantum, como se lo realiza en la línea 24.

Instalación de NOVA

En el Código A-13 se describe el proceso de instalación y configuración de los servicios de Nova en el nodo *Controller*.

```
1  root@Controller:~# apt-get install -y nova-api nova-cert novnc
2  nova-consoleauth nova-scheduler nova-novncproxy
3  paste.filter factory = keystone.middleware.auth token:filter factory
4  auth_host = 172.31.15.167
5  auth_port = 35357
6  auth_protocol = http
7  auth_uri = http://172.31.15.167:35357/v2.0
8  admin_tenant_name = services
9  admin user = nova
10 admin_password = daniel
11 root@Controller:~# nova-manage db sync
12 root@Controller:~# cd /etc/init.d/; for i in $( ls nova-* ); do sudo service $i
13 restart; done
14 root@Controller:~# nova-manage service list
15
16 Binary           Host           Zone       Status     State Updated_At
17 nova-consoleauth Controller     nova       enabled    :-)   2013-01-30 07:26:50
18 nova-cert        Controller     nova       enabled    :-)   2013-01-30 07:26:50
19 nova-scheduler   Controller     nova       enabled    :-)   2013-01-30 07:26:50
```

Código A-13 Configuración de Nova en el nodo *Controller*

En la línea 1 y 2 se instalan los paquetes necesarios de cada servicio de Nova. Se modifica el fichero */etc/nova/api-paste.ini* con la información de las líneas 3 a 10. A continuación se debe reemplazar la información del fichero */etc/nova/nova.conf* con las líneas de configuración del Código A-14. Por último se sincroniza la base de datos y se reinician los servicios, como se observa en las líneas 11 y 12 respectivamente.

El comando de la línea 13 muestra que efectivamente están activos los servicios instalados.

```
1 [DEFAULT]
2 logdir=/var/log/nova
3 state_path=/var/lib/nova
4 lock_path=/run/lock/nova
5 verbose=True
6 api_paste_config=/etc/nova/api-paste.ini
7 ec2_listen=172.31.15.167
```

```
rabbit_port=5672
rabbit virtual host=/
rabbit password=daniel
rabbit userid=OpenStack rabbit user
rabbit_host=172.31.15.167
metadata_listen=192.168.221.43
sql_connection=mysql://nova:daniel@172.31.15.167/nova
root_helper=sudo nova-rootwrap /etc/nova/rootwrap.conf

# Auth
use_deprecated_auth=false
auth_strategy=keystone

# Imaging service
glance_api_servers=172.31.15.167:9292
image_service=nova.image.glance.GlanceImageService

#VNC configuration
novncproxy_port=6080
novncproxy_host=0.0.0.0
novnc_enabled=true
novncproxy base url=http://172.31.15.167:6080/vnc auto.html

#Network settings
network_api_class=nova.network.quantumv2.api.API
quantum_url=http://172.31.15.167:9696
quantum_auth_strategy=keystone
quantum_admin_auth_url=http://172.31.15.167:35357/v2.0
quantum_admin_password=daniel
quantum_admin_username=quantum
quantum admin tenant name=services
libvirt use virtio for bridges=True
connection_type=libvirt
libvirt_vif_driver=nova.virt.libvirt.vif.LibvirtHybirdOVSBridgeDriver
firewall_driver=nova.virt.libvirt.firewall.IptablesFirewallDriver
linuxnet_interface_driver=nova.network.linux_net.LinuxOVSInterfaceDriver
```

Código A-14 Contenido de archivo nova.conf para el nodo Controller

Nota: en el campo "*State*" debe estar el símbolo de una cara feliz " :-) ", si el servicio no tiene ningún inconveniente, caso contrario se mostrará "xxx".

Instalación de HORIZON

En el Código A-15 se muestra la instalación del módulo Horizon. Tan solo es necesario instalar los paquetes necesarios y se reinicia los servicios, según se observa en la línea 1 y 2 respectivamente.

```
1 root@Controller:~# apt-get install OpenStack-dashboard memcached -y
2 root@Controller:~# service apache2 restart; service memcached restart
```

Código A-15 Instalación de Horizon

De esta manera ya está disponible para el acceso a través del URL[85] http://172.31.15.167/horizon. Al ingresar a este sitio web, se desplegará un formulario de *login* similar al presentado en la Figura 1. Las credenciales

[85] URL: *Uniform Resource Locator,* es la cadena de caracteres que se asigna a todos los recursos de información para su localización en Internet.

admin/daniel (configurado anteriormente en el módulo Keystone) dan acceso a la interfaz de administración.

Figura A-1 Formulario de *Login* Horizon

NODO NETWORK

Configuración de Red

Se edita el archivo */etc/network/interfaces* con las líneas que se muestran en el Código A-16. Es necesario aclarar que la interfaz eth2 se usa para la creación de puentes hacia las interfaces de las máquinas virtuales, es por ello que debe estar configurada en modo manual.

```
auto lo
iface lo inet loopback
auto eth1
iface eth1 inet static
        address 172.31.15.165
        netmask 255.255.255.0
        network 172.31.15.0
        broadcast 172.31.15.255
        gateway 172.31.15.251
        dns-nameservers 8.8.8.8
auto eth2
iface eth2 inet manual
        up ifconfig $IFACE 0.0.0.0 up
        up ip link set $IFACE promisc on
        down ifconfig $IFACE down
auto eth0
iface eth0 inet static
        address 10.0.0.42
        netmask 255.255.255.0
```

Código A-16 Configuración de red en el nodo *Network*

Sincronización de tiempo

En el Código A-17 se muestra la configuración del servicio NTP.

```
1 root@network:~# apt-get install -y ntp
2 root@network:~# nano /etc/ntp.conf
3 server 172.31.15.167
4 root@network:~# service ntp restart
```

Código A-17 Configuración de NTP en el nodo *Network*

En la línea 1 se instala el paquete necesario. El archivo */etc/ntp.conf* se edita de tal manera para que solo haga peticiones de actualización de tiempo al nodo *Controller*, agregando la línea 3. Por último se reinicia el servicio (línea 4).

Instalación de QUANTUM

En el Código A-18 se describe el proceso de instalación y configuración del servicio de red Quantum.

```
root@network:~# apt-get -y install quantum-plugin-openvswitch quantum-plugin-openvswitch-
agent quantum-dhcp-agent quantum-l3-agent
root@network:~# apt-get install -y dnsmasq-base dnsmasq-utils
root@network:~# kernel_version=`cat /proc/version | cut -d " " -f3`
root@network:~# apt-get install -y dkms openvswitch-switch openvswitch-datapath-dkms
linux-headers-$kernel_version
root@network:~# apt-get autoremove openvswitch-datapath-dkms
root@network:~# apt-get install -y dkms openvswitch-switch openvswitch-datapath-dkms
linux-headers-$kernel_version
root@network:~# /etc/init.d/openvswitch-switch restart
root@network:~# ovs-vsctl add-br br-int
root@network:~# ovs-vsctl add-br br-ex
root@network:~# ovs-vsctl add-port br-ex eth2
paste.filter_factory = keystone.middleware.auth_token:filter_factory
auth_host=172.31.15.167
auth_port = 35357
auth_protocol = http
admin_tenant_name=services
admin_user=quantum
admin_password=daniel
sql_connection=mysql://quantum:daniel@172.31.15.167/quantum
enable_tunneling = True
tunnel_id_ranges = 1:1000
integration_bridge=br-int
tunnel_bridge = br-tun
network_vlan_ranges=
tenant_network_type=gre
local_ip = 10.0.0.42
auth_url = http://172.31.15.167:35357/v2.0
auth_region = RegionOne
admin_tenant_name = services
admin_user = quantum
admin_password = daniel
metadata_ip = 192.168.221.43
metadata_port = 8775
use_namespaces = True
allow_overlapping_ips = False
fake_rabbit = False
rabbit_virtual_host=/quantum
rabbit_userid=OpenStack_rabbit_user
rabbit_password=daniel
rabbit_host=172.31.15.167
rabbit_port=5672
root@network:~# service quantum-plugin-openvswitch-agent restart
root@network:~# service quantum-dhcp-agent restart
root@network:~# service quantum-l3-agent restart
```

Código A-18 Configuración de Quantum en el nodo *Network*

En la línea 1 y 2 se instalan los paquetes necesarios para Quantum. El agente de capa 3 (L3_agent) usa por defecto “dnsmasq” como servidor DHCP. Para ello es necesario instalar los paquetes respectivos según lo presentado en la línea 3.

En este punto es necesario reiniciar el nodo *Network*. A continuación se deben instalar los paquetes adicionales para corregir un error de instalación del OVS, ésto se muestra en las líneas 4 a 10.

Una vez corregido este inconveniente, el cual se presenta por una incompatibilidad de OVS con la versión de Quantum, que impide la creación de puentes y túneles, se procede con las líneas 11 y 12 que permiten crear los *bridges*[86] “br-int” y “br-ex” respectivamente, en la línea 13 se asocia la interfaz eth2 al *bridge* “br-ex”.

El *bridge* de integración (br-int) sirve para conectar las máquinas virtuales con las interfaces físicas de sus respectivos nodos de cómputo a través del *bridge* externo (br-ex), este último es el que se conecta con la red externa.

Se edita el fichero */etc/quantum/api-paste.ini* y se agrega el contenido de las líneas 14 a 20 que servirán para autenticar el servicio Quantum ante Keystone. Se edita el fichero */etc/quantum/plugins/openvswitch/ovs_quantum_plugin.ini* con las líneas 21 a 28. El *plugin* de Open vSwitch se configura para que trabaje con túneles GRE y puentes entre las interfaces virtuales y físicas, para ello se debe actualizar el fichero */etc/quantum/l3_agent.ini* con la información de las líneas 29 a 36. De igual manera también se agrega la línea 36 al fichero */etc/quantum/dhcp_agent.ini* para habilitar la opción de crear varios routers virtuales y que puedan trabajar al mismo tiempo.

Editar los valores de conexión con el servidor de encolamiento RabbitMQ en el fichero */etc/quantum/quantum.conf* con las líneas 37 a 43. Por último, reiniciar los servicios según se lo presentado en las líneas 44 a 46.

---

[86] *Bridge* se denomina al dispositivo que interconecta segmentos de red, permitiendo la transferencia de datos entre redes en base a la dirección MAC de destino de cada paquete.

NODO DE CÓMPUTO

Configuración de Red

Se edita el archivo */etc/network/interfaces* y se agregan las líneas que se muestra en el Código A-19.

```
auto lo
iface lo inet loopback
auto eth0
iface eth0 inet static
        address 172.31.15.213
        netmask 255.255.255.0
        network 172.31.15.0
        broadcast 172.31.15.255
        gateway 172.31.15.251
        dns-nameservers 8.8.8.8
auto eth1
iface eth1 inet static
        address 10.0.0.51
        netmask 255.255.255.0
```

Código A19 Configuración de red en el nodo *Compute*

Sincronización de tiempo

En el Código A-20 se muestra la configuración del servicio NTP.

```
root@network:~# apt-get install -y ntp
root@network:~# nano /etc/ntp.conf
server 172.31.15.167
root@network:~# service ntp restart
```

Código A-20 Configuración de NTP en nodo *Compute*

En la línea 1 se instala el paquete necesario. El archivo */etc/ntp.conf* se edita para que solo se permita hacer peticiones de actualización de tiempo al nodo *Controller*, agregando la línea 3. Por último se reinicia el servicio (línea 4).

Instalación de KVM

En el Código A-21 se describe el proceso de instalación y configuración del hipervisor KVM, el cual se encargará de ejecutar las máquinas virtuales.

```
root@compute:~# apt-get install -y qemu-kvm libvirt-bin
cgroup_device_acl = [
 "/dev/null", "/dev/full", "/dev/zero",
 "/dev/random", "/dev/urandom",
 "/dev/ptmx", "/dev/kvm", "/dev/kqemu",
 "/dev/rtc", "/dev/hpet","/dev/net/tun"
]
root@compute:~# service libvirt-bin restart
```

Código A-21 Configuración de KVM

Proceder a la instalación de KVM mediante la línea 1. Editar el fichero */etc/libvirt/qemu.conf*, comentar lo presentado en las líneas 2 a 7 y agregar "/dev/net/tun". Por último reiniciar el servicio (línea 8).

Instalación de QUANTUM

En el Código A-22 se describe el proceso de instalación y configuración del módulo de red Quantum, Open vSwitch y los respectivos agentes.

```
1  root@compute:~# apt-get -y install quantum-plugin-openvswitch quantum-plugin-
2  openvswitch-agent
3  sql_connection=mysql://quantum:daniel@172.31.15.167/quantum
4  enable_tunneling = True
5  tunnel_id_ranges = 1:1000
6  integration_bridge=br-int
7  tunnel_bridge = br-tun
8  network_vlan_ranges=
9  tenant_network_type=gre
10 local_ip = 10.0.0.51
11 allow_overlapping_ips = False
12 fake_rabbit = False
13 rabbit_virtual_host=/quantum
14 rabbit_userid=OpenStack_rabbit_user
15 rabbit_password=daniel
16 rabbit_host=172.31.15.167
17 rabbit_port=5672
18 root@compute:~# ovs-vsctl add-br br-int
19 root@compute:~# service quantum-plugin-openvswitch-agent restart
```

Código A-22 Configuración de Quantum en el nodo *Compute*

En la línea 1 se instalan los paquetes necesarios para Quantum y Open vSwitch. Se edita el fichero */etc/quantum/plugins/openvswitch/ovs_quantum_plugin.ini* y se agregan las líneas 3 a 10. Editar los valores de conexión con el servidor de encolamiento RabbitMQ en el fichero */etc/quantum/quantum.conf* según muestra lo presentado en las líneas 11 a 17.

Se procede a crear *bridges*, como se muestra en la línea 18. Finalmente se reinician los servicios para que surtan efecto las configuraciones (línea 19).

Instalación de NOVA

En el Código A-23 se describe el proceso de instalación y configuración de los servicios de Nova en el nodo *Compute*.

```
1 root@compute:~# apt-get install -y nova-compute
2 [filter:authtoken]
3 paste.filter_factory = keystone.middleware.auth_token:filter_factory
4 auth_host = 172.31.15.167
5 auth_port = 35357
6 auth_protocol = http
```

```
auth_uri = http://172.31.15.167:35357/v2.0
admin_tenant_name = services
admin_user = nova
admin_password = daniel
libvirt_type=kvm
libvirt_ovs_bridge=br-int
libvirt_vif_type=ethernet
libvirt_vif_driver=nova.virt.libvirt.vif.LibvirtHybridOVSBridgeDriver
libvirt_use_virtio_for_bridges=True
root@compute:~# cd /etc/init.d/; for i in $( ls nova-* ); do sudo service $i restart; done
root@compute:~# nova-manage service list
Binary           Host                 Zone         Status     State Updated_At
nova-consoleauth Controller           nova         enabled    :-)   2013-01-31 16:15:00
nova-cert        Controller           nova         enabled    :-)   2013-01-31 16:15:00
nova-scheduler   Controller           nova         enabled    :-)   2013-01-31 16:15:00
nova-compute     compute              nova         enabled    :-)   2013-01-31 16:15:00
```

Código 23 Configuración de Nova en el nodo *Compute*

En la línea 1 se instala el paquete Nova. Se modifica el fichero */etc/nova/api-paste.ini* con la información de las líneas 2 a 10. Se modifica el fichero */etc/nova/nova-compute.conf* con las líneas 11 a 15 para editar los parámetros de virtualización y de conexión con Open vSwtch. A continuación se reemplaza la información que se encuentra en el Código A-24 en el fichero */etc/nova/nova.conf.*

```
[DEFAULT]
logdir=/var/log/nova
state_path=/var/lib/nova
lock_path=/run/lock/nova
verbose=True
api_paste_config=/etc/nova/api-paste.ini
s3_host=172.31.15.167
ec2_host=172.31.15.167
rabbit_port=5672
rabbit_virtual_host=/
rabbit_password=daniel
rabbit userid=OpenStack rabbit user
rabbit host=172.31.15.167
metadata host=192.168.221.43
sql_connection=mysql://nova:daniel@172.31.15.167/nova
root_helper=sudo nova-rootwrap /etc/nova/rootwrap.conf
connection_type=libvirt
use deprecated auth=false
auth strategy=keystone
keystone ec2 url=http://172.31.15.167:5000/v2.0/ec2tokens
# Imaging service
glance_api_servers=172.31.15.167:9292
image_service=nova.image.glance.GlanceImageService
# VNC configuration
vnc_enabled=true
vncserver_proxyclient_address=172.31.15.168
novncproxy base url=http://172.31.15.167:6080/vnc auto.html
vncserver_listen=172.31.15.168
# Network settings
network_api_class=nova.network.quantumv2.api.API
quantum_url=http://172.31.15.167:9696
quantum_auth_strategy=keystone
quantum admin auth url=http://172.31.15.167:35357/v2.0
quantum connection host=localhost
quantum_admin_password=daniel
quantum_admin_username=quantum
quantum_admin_tenant_name=services
libvirt_vif_driver=nova.virt.libvirt.vif.LibvirtHybridOVSBridgeDriver
linuxnet_interface_driver=nova.network.linux_net.LinuxOVSInterfaceDriver
firewall_driver=nova.virt.libvirt.firewall.IptablesFirewallDriver
# Compute
compute_driver=libvirt.LibvirtDriver
```

Código A-24 Contenido del archivo nova.conf

Ahora se pueden reiniciar todos los servicios de Nova con el código presentado en la línea 16. En el nodo *Controller* ya se debe visualizar que el nodo *Compute* esta sincronizado mediante el comando de la línea 17, el resultado se presenta en las líneas 18 a 23, en las que se puede observar que efectivamente el servicio *nova-compute* del nodo *Compute* esta sincronizado con la infraestructura.

Nota: Para la instalación de nodos de cómputo adicionales, se debe realizar exactamente el mismo proceso descrito en la sección correspondiente a la instalación del nodo de cómputo, solamente se debe cambiar el direccionamiento en la red privada y la red de administración.

ANEXO B

SCRIPT DE CREACIÓN DE UN NUEVO SERVIDOR

```
#!/usr/bin/Python
import base64
import httplib
import json
import urllib
import time
import commands
from urlparse import urlparse
import os
import sys
os.environ['OS_TENANT_NAME'] = 'admin'
os.environ['OS_USERNAME'] = 'admin'
os.environ['OS_PASSWORD'] = 'daniel'
os.environ['OS_AUTH_URL'] = 'http://172.31.15.167:5000/v2.0/'
os.environ['OS_AUTH_STRATEGY'] = 'keystone'
os.environ['SERVICE_TOKEN'] = 'keystone_admin_token'
os.environ['SERVICE_ENDPOINT'] = 'http://172.31.15.167:35357/v2.0/'
url = "172.31.15.167:5000"
ospassword = "daniel"
params = '{"auth":{"passwordCredentials":{"username": "admin", "password":"daniel"},
"tenantId":"621c3f810e494b3d918e52c9a0614070"}}'
headers = {"Content-Type": "application/json"}
conn = httplib.HTTPConnection(url)
conn.request("POST", "/v2.0/tokens", params, headers)
response = conn.getresponse()
data = response.read()
dd = json.loads(data)
conn.close()
apitoken = dd['access']['token']['id']
sc = dd['access']['serviceCatalog']
m = range(len(sc))
apiurlt = urlparse(apiurl)
params3 = urllib.urlencode({})
headers3 = { "X-Auth-Token":apitoken, "Content-type":"application/json" }
conn3 = httplib.HTTPConnection(apiurlt[1])
conn3.request("GET", "%s/flavors" % apiurlt[2], params3, headers3)
response3 = conn3.getresponse()
data3 = response3.read()
n = len(json.loads(data3) ["flavors"])
m = range(n)
sflavor = '%s' %sys.argv[3]
Flavor_uuid=""
for i in m:
        if dd3["flavors"][i]["name"] == sflavor:
                Flavor_uuid = dd3["flavors"][i]["id"]
image = sys.argv[1]
llave = 'millave'
red = "f4a19a17-de1f-4f0b-854d-3594e05c64bd"
nombrevm = sys.argv[2]
secgroup = nombrevm
serverid = commands.getoutput("nova boot --image %s --Flavor %s --key_name %s --nic net-id=%s  --
security_group %s %s | grep -w id | awk '{print $4}'" %
(image,Flavor uuid,llave,red,secgroup,nombrevm))
time.sleep(45)
apiurl_quantum="http://172.31.15.167:9696/v2.0"
apiurlt_quantum=urlparse(apiurl_quantum)
idpuerto = commands.getoutput("quantum port-list -- --device_id %s | grep -w subnet_id | awk '{print
$2}'" %serverid)
floatingipid = commands.getoutput("quantum floatingip-create public | awk '/ id / {print $4}'")
time.sleep(3)
ippublica = commands.getoutput("quantum floatingip-list | awk '/ %s / {print $5}'" %floatingipid)
print "%s" %ippublica
fk = '{"floatingip": {"port_id": "%s"}}' %idpuerto
headers6 = { "X-Auth-Token":apitoken, "Accept":"application/json" }
conn6 = httplib.HTTPConnection(apiurlt_quantum[1])
conn6.request("PUT", "%s/floatingips/%s.json" % (apiurlt quantum[2],floatingipid), fk, headers6)
response6 = conn6.getresponse()
data6 = response6.read()
dd6 = json.loads(data6)
conn6.close()
```

ANEXO C

SCRIPT PARA ELIMINAR UNA MÁQUINA VIRTUAL

```
#!/usr/bin/Python

import commands
import os
import sys

os.environ['OS_TENANT_NAME'] = 'admin'
os.environ['OS_USERNAME'] = 'admin'
os.environ['OS_PASSWORD'] = 'daniel'
os.environ['OS_AUTH_URL'] = 'http://172.31.15.167:5000/v2.0/'
os.environ['OS_AUTH_STRATEGY'] = 'keystone'
os.environ['SERVICE_TOKEN'] = 'keystone_admin_token'
os.environ['SERVICE_ENDPOINT'] = 'http://172.31.15.167:35357/v2.0/'

uuid = sys.argv[1]
os.system("nova delete %s" %uuid)
```

ANEXO D

SCRIPT PARA OBTENER EL ESTADO DE UNA MÁQUINA VIRTUAL

```
#!/usr/bin/Python

import os
import sys
import commands

os.environ['OS_TENANT_NAME'] = 'admin'
os.environ['OS_USERNAME'] = 'admin'
os.environ['OS_PASSWORD'] = 'daniel'
os.environ['OS_AUTH_URL'] = 'http://172.31.15.167:5000/v2.0/'
os.environ['OS_AUTH_STRATEGY'] = 'keystone'
os.environ['SERVICE_TOKEN'] = 'keystone_admin_token'
os.environ['SERVICE_ENDPOINT'] = 'http://172.31.15.167:35357/v2.0/'

serverid = sys.argv[1]
if sys.argv[2] == 'ip':
    idpuerto = commands.getoutput("quantum port-list -- --device_id %s | grep -w subnet_id | awk
'{print $2}'" %serverid)
        floatingipid = commands.getoutput("quantum floatingip-create public | awk '/ id / {print
$4}'")
        ippublica = commands.getoutput("quantum floatingip-list | awk '/ %s / {print $5}'"
%floatingipid)
        print "%s" %ippublica
if sys.argv[2] == 'estado':
        estado = commands.getoutput(" nova show %s | awk '/ OS-EXT-STS:vm_state / {print $4}'"
%serverid)
        print "%s" %estado
```

ANEXO E

SCRIPTS DE ACTUALIZACIÓN DE ESTADO DE UNA MÁQUINA VIRTUAL

```
#!/usr/bin/Python

import commands
import os
import sys

os.environ['OS_TENANT_NAME'] = 'admin'
os.environ['OS_USERNAME'] = 'admin'
os.environ['OS_PASSWORD'] = 'daniel'
os.environ['OS_AUTH_URL'] = 'http://172.31.15.167:5000/v2.0/'
os.environ['OS_AUTH_STRATEGY'] = 'keystone'
os.environ['SERVICE_TOKEN'] = 'keystone_admin_token'
os.environ['SERVICE_ENDPOINT'] = 'http://172.31.15.167:35357/v2.0/'

serverid = sys.argv[1]
estado = commands.getoutput(" nova show %s | awk '/ OS-EXT-STS:vm_state / {print $4}'" %serverid)
#print "estado es %s" %estado
if sys.argv[2] == 'suspend' and estado == 'suspended':
    print "suspend"
if sys.argv[2] == 'active' and estado == 'active':
        print "active"
if sys.argv[2] == 'suspend' and estado == 'active':
        os.system("nova suspend %s" %serverid)
if sys.argv[2] == 'active' and estado == 'suspended':
        os.system("nova resume %s" %serverid)
```

ANEXO F

SCRIPT QUE OBTIENE ID DE MÁQUINA VIRTUAL

```
#!/usr/bin/Python

import os
import sys
import commands

os.environ['OS_TENANT_NAME'] = 'admin'
os.environ['OS_USERNAME'] = 'admin'
os.environ['OS_PASSWORD'] = 'daniel'
os.environ['OS_AUTH_URL'] = 'http://172.31.15.167:5000/v2.0/'
os.environ['OS_AUTH_STRATEGY'] = 'keystone'
os.environ['SERVICE TOKEN'] = 'keystone admin token'
os.environ['SERVICE ENDPOINT'] = 'http://172.31.15.167:35357/v2.0/'

ip = sys.argv[1]
floatingip = commands.getoutput("quantum floatingip-list | grep %s | awk '{print $2}'" %ip)
port id = commands.getoutput("quantum floatingip-show %s | grep port id | awk '{print $4}'"
%floatingip)
uuid =  commands.getoutput("quantum port-show %s | grep device id | awk '{print $4}'" %port id)
print "%s" %uuid
```

ANEXO G

SCRIPT DE REDIMENSIONAMIENTO DE UNA MÁQUINA VIRTUAL

```
#!/usr/bin/Python

#Python ./resizevm <uuid> <3,6,11,13,20,23,30,40>
import os
import sys
import commands

os.environ['OS_TENANT_NAME'] = 'admin'
os.environ['OS_USERNAME'] = 'admin'
os.environ['OS_PASSWORD'] = 'daniel'
os.environ['OS_AUTH_URL'] = 'http://172.31.15.167:5000/v2.0/'
os.environ['OS_AUTH_STRATEGY'] = 'keystone'
os.environ['SERVICE_TOKEN'] = 'keystone_admin_token'
os.environ['SERVICE_ENDPOINT'] = 'http://172.31.15.167:35357/v2.0/'

if sys.argv[2] == "3":
    fl = "255396c8-11c9-4fdc-8b29-abdb971a54ec"
if sys.argv[2] == "6":
        fl = "04d41ad8-7e54-4c5b-9a96-bf697081438a"
if sys.argv[2] == "10":
        fl = "ff5ae6ec-ca6a-4778-8521-2d23dbab173c"
if sys.argv[2] == "13":
        fl = "19f3afea-f1da-4209-b239-9a5bb1ae0f69"
if sys.argv[2] == "20":
        fl = "9a580b2d-f0d5-4eb2-808f-aafb77dd4ec2"
if sys.argv[2] == "23":
        fl = "082bdbf9-6124-4ef7-9056-d677c24bb1d9"
if sys.argv[2] == "30":
        fl = "7c0ac399-993b-49e2-ab1e-6fc138114101"
if sys.argv[2] == "40":
        fl = "a0a2f50f-b9b7-45b6-b1fd-2f1c133256e2"
if sys.argv[2] == "11":
        fl = "ff5ae6ec-ca6a-4778-8521-2d23dbab173c"
os.system("nova resize %s %s" %(sys.argv[1],fl))
```

ANEXO H

SCRIPT DE VERIFICACIÓN DE REDIMENSIÓN DE UNA MÁQUINA VIRTUAL

```
#!/usr/bin/Python
import base64
import httplib
import json
import urllib
import commands
from urlparse import urlparse
import os
import sys
import time
import datetime

os.environ['OS_TENANT_NAME'] = 'admin'
os.environ['OS_USERNAME'] = 'admin'
os.environ['OS_PASSWORD'] = 'daniel'
os.environ['OS_AUTH_URL'] = 'http://172.31.15.167:5000/v2.0/'
os.environ['OS_AUTH_STRATEGY'] = 'keystone'
os.environ['SERVICE_TOKEN'] = 'keystone_admin_token'
os.environ['SERVICE_ENDPOINT'] = 'http://172.31.15.167:35357/v2.0/'

url = "172.31.15.167:5000"
params = '{"auth":{"passwordCredentials":{"username": "admin", "password":"daniel"},
"tenantId":"621c3f810e494b3d918e52c9a0614070"}}'
headers = {"Content-Type": "application/json"}
conn = httplib.HTTPConnection(url)
conn.request("POST", "/v2.0/tokens", params, headers)
response = conn.getresponse()
data = response.read()
dd = json.loads(data)
conn.close()

apitoken = dd['access']['token']['id']
sc = dd['access']['serviceCatalog']
m = range(len(sc))
foundNovaURL = False
for i in m:
    ss = sc[i]['name']
        if ss == 'nova':
                apiurl = sc[i]['endpoints'][0]['publicURL']
                foundNovaURL = True
                break
apiurlt = urlparse(apiurl)
params3 = urllib.urlencode({})
headers3 = { "X-Auth-Token":apitoken, "Content-type":"application/json" }
conn3 = httplib.HTTPConnection(apiurlt[1])
conn3.request("GET", "%s/servers" % apiurlt[2], params3, headers3)
response3 = conn3.getresponse()
data3 = response3.read()
dd3 = json.loads(data3)
conn3.close()

n = len(dd3["servers"])
m = range(n)
L = []
for i in m:
        serverid = dd3["servers"][i]["id"]
        L.append(serverid)
for id in L:
        params4 = urllib.urlencode({})
        headers4 = { "X-Auth-Token":apitoken, "Content-type":"application/json" }
        conn4 = httplib.HTTPConnection(apiurlt[1])
        conn3.request("GET", "%s/servers/%s" % (apiurlt[2], id), params3, headers3)
        response3 = conn3.getresponse()
        data3 = response3.read()
        dd3 = json.loads(data3)
        estado = dd3["server"]["status"]
        if estado == "VERIFY_RESIZE":
                os.system("nova resize-confirm %s" %id)
                now = datetime.datetime.now()
                print "%s se confirmó resize de %s" % (now, id)
```

ANEXO I

ARCHIVO MODELS.PY

```
from django.db import models
from django.contrib.auth.models import User
from django.db.models.signals import post save
from datetime import datetime

class tipo(models.Model):
        nombre          = models.CharField(max_length=50)
           precio          = models.DecimalField(max_digits=6,decimal_places=2)
           descripcion           = models.TextField(max_length=300)
           almacenamiento        = models.IntegerField()

        def   unicode  (self):
                return self.nombre

class servicio(models.Model):
        nombre          = models.CharField(max_length=20)
        descripcion     = models.TextField(max length=300)
        sla             = models.TextField(max length=500)
           nombre completo       = models.CharField(max length=100)
        def __unicode__(self):
                return self.nombre

class servidor(models.Model):
        estado          = models.CharField(max_length=15)
        uuid            = models.CharField(max_length=100)
        ip              = models.CharField(max length=20)
        servicios       = models.ManyToManyField(servicio,null=True,blank=True)
           tipo                 = models.ForeignKey('pymes.tipo',null=True,blank=True)
        def __unicode__(self):
                return self.ip

class mail(models.Model):
           dominio    = models.CharField(max length=100)
           num_cuentas= models.IntegerField()
           servidor   = models.ForeignKey('pymes.servidor',null=True,blank=True)
           precio     = models.DecimalField(max_digits=6,decimal_places=2)
           gbusados   = models.DecimalField(max_digits=6,decimal_places=1,null=True,blank=True)
           fecha_creado  = models.DateTimeField(auto_now=False,auto_now_add=True)
           activo= models.BooleanField(default=True)
           def __unicode__(self):
                    return self.dominio

class cuenta(models.Model):
           nombre     = models.CharField(max_length=100)
           password   = models.CharField(max_length=100)
           address    = models.CharField(max length=50)
           mail       = models.ForeignKey('pymes.mail',null=True,blank=True)

           def __unicode__ (self):
                    direccion = "%s@%s" %(self.address,self.mail)
                    return direccion
class perfilCliente(models.base.ModelBase):
        def  prepare(self):
                super(perfilCliente, self)._prepare()

class cliente(models.Model):
        __metaclass__    = perfilCliente
        user            = models.OneToOneField(User)
        nombre          = models.CharField(max_length=100)
        apellidos       = models.CharField(max length=100)
        empresa         = models.CharField(max length=200)
        status          = models.BooleanField(default=True)
        servidores     = models.ManyToManyField(servidor,null=True,blank=True)
        def __unicode__(self):
                nombreCompleto = "%s %s"%(self.nombre,self.apellidos)
                return nombreCompleto
        def create cliente user callback(sender, instance, created, **kwargs):
                if created:
                        self.objects.create(user=instance)
                post_save.connect(create_cliente_user_callback, sender=User, weak=False)

class servicio_activo(models.Model):
           cliente    = models.ForeignKey('pymes.cliente',null=True,blank=True)
           servidor   = models.ForeignKey('pymes.servidor',null=True,blank=True)
```

```
75            servicio   = models.ForeignKey('pymes.servicio',null=True,blank=True)
76            capacidad  = models.IntegerField()
77            fecha pedido          = models.DateTimeField(auto now=False,auto now add=True)
78            fecha realizado       = models.DateTimeField(blank=True,null=True)
79            fecha_ultima_edicion  = models.DateTimeField(auto_now=True)
80            activo                = models.BooleanField(default=True)
81            suspendido            = models.BooleanField(default=False)
82
83            def   unicode  (self):
84            nombreCompleto = "%s-%s-%s"%(self.cliente,self.servicio,self.capacidad)
85 #                   return self.capacidad
86                 return nombreCompleto
87
88 class factura(models.Model):
89            fecha_factura         = models.DateTimeField(blank=True,null=True)
90            cliente               = models.ForeignKey('pymes.cliente',null=True,blank=True)
91
92            def   unicode  (self):
93                      descr = "%s %s" %(self.cliente, self.fecha_factura)
94                      return descr
95
96 class reg dominio(models.Model):
97            fecha reg  = models.DateTimeField(blank=True,null=True)
98            fecha fin  = models.DateTimeField(blank=True,null=True)
99            precio     = models.DecimalField(max_digits=6,decimal_places=2)
100           mail       = models.ForeignKey('pymes.mail',null=True,blank=True)
101           cuentas    = models.IntegerField()
102           factura    = models.ForeignKey('pymes.factura',null=True,blank=True)
103
104           def __unicode__(self):
105                     descr = "%s-%s-%s" %(self.mail.dominio,self.fecha reg,self.fecha fin)
106                     return descr
107
108 class reg_servicio(models.Model):
109         fecha_reg     = models.DateTimeField(blank=True,null=True)
110         fecha_fin     = models.DateTimeField(blank=True,null=True)
111         precio        = models.DecimalField(max digits=6,decimal places=2)
112            capacidad  = models.IntegerField()
113         serv act      = models.ForeignKey('pymes.servicio activo',null=True,blank=True)
114         factura       = models.ForeignKey('pymes.factura',null=True,blank=True)
115
116         def __unicode__(self):
117 #               descr = "%s-%s-%s" %(self.serv_act,strptime(self.fecha_reg,"%d %b
118 %y"),strptime(self.fecha_fin,"%d %b %y"))
119                     descr = "%s-%s-%s" %(self.serv_act.servicio,self.fecha_reg,self.fecha_fin)
120                     return descr
```

ANEXO J

ARCHIVO FORMS.PY

```
from django import forms
from django.contrib.auth.models import User
from django.forms import ModelForm
from saas.apps.pymes.models import cliente, cuenta

class ContactForm(forms.Form):
    Email    = forms.EmailField(widget=forms.TextInput())
    Texto    = forms.CharField(widget=forms.Textarea())
    Nombre   = forms.CharField(widget=forms.TextInput())
    Apellido = forms.CharField(widget=forms.TextInput())

class LoginForm(forms.Form):
    username = forms.CharField(widget=forms.TextInput())
    password = forms.CharField(widget=forms.PasswordInput(render_value=False))

class RegistrationForm(ModelForm):
    username = forms.CharField(label=(u'UserName'))
    mail        = forms.EmailField(label=(u'EmailAddress'))
    password    = forms.CharField(label=(u'Password'),
widget=forms.PasswordInput(render_value=False))
    passworduno = forms.CharField(label=(u'VerifyPassword'),
widget=forms.PasswordInput(render_value=False))

        class Meta:
                model = cliente
                exclude = ('user',)
class DominioForm(forms.Form):
    dominio = forms.CharField(widget=forms.TextInput())

class PassForm(forms.Form):
    password = forms.CharField(label=(u'Password'), widget=forms.PasswordInput(render_value=False))
    passworduno = forms.CharField(label=(u'VerifyPassword'),
widget=forms.PasswordInput(render value=False))

class NewUserForm(ModelForm):
    passw     = forms.CharField(label=(u'Password'), widget=forms.PasswordInput(render_value=False))
    passwuno  = forms.CharField(label=(u'VerifyPassword'),
widget=forms.PasswordInput(render_value=False))
    class Meta:
        model = cuenta
        exclude = ('mail','password',)
```

ANEXO K

ARCHIVO URLS.PY

```
from django.conf.urls import patterns, url, include
from django.contrib import admin

urlpatterns = patterns('saas.apps.pymes.views',
    url(r'^$','index view',name='vista principal'),
    url(r'^login/$','login_view',name='vista_login'),
    url(r'^logout/$','logout_view',name='vista_logout'),
    url(r'^register/$','register_view',name='vista_register'),
    url(r'^mail/$','mail_view',name='vista_mail'),
    url(r'^nuevo_dominio/$','nuevo_dominio_view',name='vista_nuevo_dominio'),
    url(r'^resize/(?P<serv>.*)/$','resize_view',name='vista_resize'),
    url(r'^crm/$','crm view',name='vista crm'),
    url(r'^facturacion/$','facturacion view',name='vista facturacion'),
    url(r'^op/$','op_view',name='vista_op'),
    url(r'^servicios/$','servicios_view',name='vista_servicios'),
    url(r'^contacto/$','contacto_view',name='vista_contacto'),
    url(r'^panel/$','panel view',name='vista panel'),
    url(r'^conf mail/$','conf mail view',name='vista conf mail'),
    url(r'^cambio dominio/(?P<domini>.*)/$','cambio dominio view',name='vista cambio dominio'),
    url(r'^contrasena/(?P<addr>.*)/$','contrasena_view',name='vista_contrasena'),
    url(r'^new_user/(?P<domini>.*)/$','new_user_view',name='vista_new_user'),
)
```

ANEXO L

ARCHIVOS DE LAS PLANTILLAS

L.1 PLANTILLA CONTACTO.HTML

```
{% extends 'base.html'%}
{% block title %}Contactanos {% endblock %}
{% block content %}
<div id="legend">
        <legend><h2>Contacto</h2></legend>
</div>
<br>
{% if info_enviado %}
    Estimad@ {{nombre}}.<br>
        Gracias por enviar un comentario, nos pondremos en contacto contigo.
        <br>
        La informacion recibida fue la siguiente<br>
        <dl class="dl-horizontal">
        <dt>Email utilizado</dt><dd> {{email}}</dd>
        <dt>Titulo</dt> <dd>{{titulo}}</dd>
        <dt>Texto Citado </dt> <dd>{{texto}}</dd></dl>
{% else %}
        <form action="." method="POST" class="well">
                <div class="row">
                {% csrf token %}
                        <div class="span3">
                                {% if form.Nombre.errors %}<div class="control-group error">{% endif
%}
                                <div class="control-group">
                                <label for="nombre" class="control-label">Nombre </label>{{
form.Nombre }}
                                {% if form.Nombre.errors %}<span class="help-inline"> Nombre
requerido</span>{% endif %}</div>

                                <div class="control-group">
                                <label for="apellido" class="control-label">Apellido </label>{{
form.Apellido }}</div>

                                {% if form.Email.errors %}<div class="control-group error">{% endif
%}
                                <div class="control-group">
                                <label for="email" class="control-label">Email </label>{{ form.Email
}}
                                {% if form.Email.errors %}<span class="help-inline"> Email
requerido</span>{% endif %}</div>

                        <label>Asunto</label>
                        <select id="subject" name="subject" class="span6">
                                <option value="na" selected="">Escoja uno:</option>
                                <option value="Atencion al cliente">Atencion al cliente</option>
                                <option value="Sugerencias">Sugerencias</option>
                                <option value="Soporte de Servicio">Soporte de Servicio</option>
                        </select>
                        </div>
                        <div class="span3">
                        {% if form.Texto.errors %}<div class="control-group error">{% endif %}
                        <div class="control-group">
                        <label for="texto" class="control-label">Mensaje </label>{{ form.Texto
}}<br>
                        {% if form.Email.errors %}<span class="help-inline"> Email
requerido</span>{% endif %}</div>
                        <button class="btn btn-primary" type="submit">Enviar Comentario</a>
                        </div>
                <!--{{form.as_p}}-->
</div>
</form>
{% endif %}
{% endblock %}
```

PLANTILLA LOGIN.HTML

```
{% extends "base.html" %}
{% block title %} Login {% endblock %}
{% block content %}
<div id="legend">
    <legend><h2>Login</h2></legend>
</div>
<form action="" method="post" class="form-horizontal">
        {% csrf_token %}
        {% if form.errors %}<p>Porfavor corregir los siguientes campos:</p>{% endif %}
        {% if form.username.errors %}<div class="control-group error">{% endif %}
        <div class="control-group">
        <label for="username" class="control-label">Nombre de Usuario </label>{{ form.username }}
        {% if form.nombre.errors %}<span class="help-inline"> Usuario requerido</span>{% endif
%}</div>
        {% if form.password.errors %}<div class="control-group error">{% endif %}
        <div class="control-group">
        <label for="password" class="control-label">Clave </label> {{ form.password }}
        {% if form.password.errors %}<span class="help-inline"> Clave requerida</span>{% endif
%}</div>

        <button class="btn btn-success" type="submit"> Iniciar Sesion </button>

</form>
<!--<p>Olvido su password? <a href="/resetpassword/">Reset!</a></p>-->

{% endblock %}
```

PLANTILLA INDEX.HTML

```
{% extends 'base.html' %}
{% block title %} INICIO - BIENVENIDOS {% endblock %}
{% block content %}
<div class="hero-unit">
  <h1> Software como Servicio</h1><br>
  <h3><p> Aplicaciones del modelo SaaS están enfocadas en proveer funcionalidades a clientes
empresariales a bajos precios,
    ofreciendo los mismos beneficios del software tradicional sin la complejidad asociada a la
instalación,
         administración, soporte y costo inicial que este representa. De esta manera, el cliente
solo tiene la responsabilidad del uso del software contratado
         en modo pago por consumo. </p></h3>
    <p> <a class="btn btn-primary btn-large">Saber más</a>
  </p>
</div>

{% endblock %}
```

PLANTILLA REGISTER.HTML

```
{% extends "base.html" %}
{% block title %} Registro {% endblock %}
{% block content %}
<form action="" method="post" class="form-horizontal">
<fieldset>
          <legend><h2>Registro Nuevo Usuario</h2></legend>
{% csrf token %}
{% if form.nombre.errors %}<div class="control-group error">{% endif %}
    <div class="control-group">
        <label for="nombre" class="control-label">Nombre </label>{{ form.nombre }}
        {% if form.nombre.errors %}<span class="help-inline"> Nombre requerido</span>{% endif
%}</div>

        <div class="control-group">
        <label for="apellido" class="control-label">Apellido </label>{{ form.apellidos }}</div>

```

```
        <div class="control-group">
        <label for="empresa" class="control-label">Empresa </label> {{ form.empresa }}</div>
        {% if form.mail.errors %}<div class="control-group error">{% endif %}

        <div class="control-group">
        <label for="mail" class="control-label">Email </label> {{ form.mail }}
        {% if form.mail.errors %}<span class="help-inline"> Mail requerido</span>{% endif %}</div>

        {% if form.username.errors %}<div class="control-group error">{% endif %}
        <div class="control-group">
        <label for="username" class="control-label">Usuario </label> {{ form.username }}
        {% if form.username.errors %}<span class="help-inline"> Usuario requerido</span>{% endif
%}</div>

        {% if form.password.errors %}<div class="control-group error">{% endif %}
        <div class="control-group">
        <label for="password" class="control-label">Clave </label> {{ form.password }}
        {% if form.password.errors %}<span class="help-inline"> Clave requerida</span>{% endif
%}</div>

        {% if form.passwordduno.errors %}<div class="control-group error">{% endif %}
        <div class="control-group">
        <label for="passwordduno" class="control-label">Confirmar Clave </label> {{ form.passwordduno
}}
        {% if form.passwordduno.errors %}<span class="help-inline"> Clave requerida</span>{% endif
%}</div>
        <input type="submit" class="btn btn-primary" value="Enviar Registro"
name="registro"></input>
        <input type="submit" class="btn" value="Cancelar" name="cancelar"></input></div>
</fieldset>
</form>
{% endblock %}
```

PLANTILLA FACTURACION.HTML

```
{% extends "base.html" %}
{% block title %} Facturacion {% endblock %}
{% block content %}
<legend><h2>Facturacion</h2></legend>
<dl>
<dt>Usuario</dt><dd>{{cliente}}</dd>
<dt>Fecha de Facturacion</dt><dd>{{tiempo}}</dd>
<dt>Total a Pagar</dt><dd> $ {{total}}</dd></dl>
<table class="table table-stripted table-condensed">
<thead><th>Fecha</th><th>Servicio</th><th>Caracteristicas</th><th>Precio</th></thead>
<tbody>
{%for reg in reg_servicios%}
<tr><td>{{reg.fecha_reg.month}}/{{reg.fecha_reg.day}} -
{{reg.fecha fin.month}}/{{reg.fecha fin.day}}</td><td>{{reg.serv act.servicio.nombre completo}}</td
><td>{{reg.capacidad}} Gb</td><td>${{reg.precio}}</td></tr>
{%endfor%}
{%for reg in reg_dominios%}
<tr><td>{{reg.fecha_reg.month}}/{{reg.fecha_reg.day}} -
{{reg.fecha_fin.month}}/{{reg.fecha_fin.day}}</td><td>Dominio:
{{reg.mail.dominio}}</td><td>{{reg.cuentas}} cuentas</td><td>${{reg.precio}}</td></tr>
{%endfor%}
</tbody></table>

<br><a href="{% url vista_servicios %}" class="btn btn-success"> Generar servicios </a>
{%endblock%}
```

PLANTILLA SERVICIOS.HTML

```
{% extends "base.html" %}
{% block title %} Servicios Disponibles {% endblock %}
{% block content %}
{{mensaje}}
<div id="legend">
<legend><h2>Catalago de Servicios</h2></legend>
```

```
</div>
<div class="row">
    {% for ser in servicios %}
                <div class="span4"><h3>{{ser.nombre completo}}</h3>
                <img src="/media/images/{{ser.nombre}}_icon.png" class="img-rounded">
                <p>{{ser.descripcion}}</p>
                <p>{{ser.sla}}</p>
                <form action="." method="POST">
                {% csrf token %}
                <input type="hidden" name="servicio" value="{{ ser.nombre }}"></input>
                <input type="submit" class="btn btn-primary" value="Generar Servicio"></input>
                </form></div>
        {%endfor%}
{% endblock %}

```

PLANTILLA MAIL.HTML

```
{% extends "servicios_base.html" %}
{% block titulo %} MAIL {% endblock %}
{% csrf_token %}
```

PLANTILLA CONF_MAIL.HTML

```
{% extends "base.html" %}
{% block title %} CONFIGURACION MAIL {% endblock %}
{% block content %}
<div id="legend">
        <legend><h2>CONFIGURACION DE MAIL</h2></legend>
</div>
{{mensaje}}
{% if messages %}
<ul class="messages">
    {% for message in messages %}
    <li{% if message.tags %} class="{{ message.tags }}"{% endif %}>{{ message }}</li>
    {% endfor %}
</ul>
{% endif %}
<dl><dt>Sus usuarios pueden acceder a Webmail en la direccion:</dt>
<dd><a href="https://{{server.ip}}/mail">https://{{server.ip}}/mail</a></dd></dl>
<div class="btn-toolbar">
    <a class="btn btn-primary btn-large" href="{% url vista nuevo dominio %}">Agregar nuevo
dominio</a>
</div>
{% for dom in dominios %}
        <table class="table table-bordered">
        <tr><td><h3>{{dom.dominio}}</h3><form action="" method="POST">{% csrf_token %}
                <input class="btn btn-inverse btn-mini" type="submit" name="deldom" value="Eliminar
Dominio">
                <input type="hidden" name="domi" value="{{dom.dominio}}"></input></form>
        </td></tr>
        <tr><td>Activo desde: {{dom.fecha_creado}}
        <br>Cuentas: {{dom.num_cuentas}}
        <br>Precio Mensual: ${{dom.precio}}
        <div class="btn-toolbar">
                <form action="" method="post">{% csrf token %}
                <button class="btn" type="submit" name="domi" value="{{ dom.dominio }}"><i
class="icon-pencil"></i>Cambiar</button>
                <button class="btn"  name="new_cuenta" value="{{dom.dominio}}" type="submit"><i
class="icon-user"></i>Nueva cuenta</button>
                </form>
        </div>
        <table class="table table-hover">
        <tr><th>Nombre</th><th>Direccion</th><th></th></tr>
        {% for cue in cuentas %}
                <tr>
                {%if cue.mail == dom%}
               <td>{{cue.nombre}}</td><td>{{cue.address}}@{{dom.dominio}}</td>
                        <td><table><form action="" method="POST">
```

```
                        {% csrf_token %}
                        <a class="btn" href="/contrasena/{{cue.address}}"><i class="icon-
pencil"></i></a>
                        </form>
                        <form action="" method="POST">
                        {% csrf_token %}
                        <input type="hidden" name="del" value="{{cue.address}}"></input>
                        <button class="btn btn-danger" type="submit"><i class="icon-
remove"></i>Eliminar</button>
                        </form></table></td>
                {%endif%}
                </tr>
        {%endfor%}
        </table>
</table>
{%endfor%}
{%endblock%}
```

PLANTILLA NUEVA_DOMINIO.HTML

```
{% extends "base.html" %}
<HEAD>
        <TITLE>{% block titulo %}Nuevo Dominio {% endblock %}</TITLE>
</HEAD>
{% block content %}
{% csrf_token %}
<div id="legend">
<legeng><h2>Pricing Servicio de Correo</h2></legend></div>
<form action="" method="post">
                {% csrf token %}
                <input type="hidden" name="servicio" value="mail"></input>
{% if form.dominio.errors %}<div class="control-group error">{% endif %}
        <div class="control-group">
        <label for="dominio" class="control-label">Dominio </label>{{ form.dominio }}
        {% if form.dominio.errors %}<span class="help-inline"> Dominio requerido</span>{% endif
%}</div>
<div class="row">
<div class="span3">
    <div class="well">
                <h2 class="muted">Basico</h2>
                <ul>
                        <li>Acceso Ilimitado</li>
                        <li>10 Usuarios(Mailboxes)</li>
                        <li>Soporte via página web</li>
                </ul>
                <hr>
                <h3>$10 / mes</h3>
                </hr>
                <form action="" method="post">{% csrf token %}
                <p><button class="btn btn-large" name= "basic" value="basic" type="submit"><i
class="icon-ok"></i>Seleccionar</button></p></form>
        </div>
</div>
<div class="span3">
        <div class="well">
                <h2 class="text-warning">Standard</h2>
                <ul>
                        <li>Acceso Ilimitado</li>
                        <li>50 Usuarios(Mailboxes)</li>
                        <li>Soporte via e-mail</li>
                </ul>
                <hr>
                <h3>$50 / mes</h3>
                </hr>{% csrf token %}
                <p><button class="btn btn-large" name= "standard" value="standard" type="submit"><i
class="icon-ok"></i>Seleccionar</button></p>
        </div>
</div>
<div class="span3">
        <div class="well">
                <h2 class="text-info">Pro</h2>
                <ul>
                        <li>Acceso Ilimitado</li>
                        <li>100 Usuarios(Mailboxes)</li>
                        <li>Soporte telefónico</li>
```

```
                </ul>
                <hr>
                <h3>$100 / mes</h3>
                </hr>{% csrf token %}
                <p><button class="btn btn-large" name= "pro" value="pro" type="submit"><i
class="icon-ok"></i>Seleccionar</button></p>
        </div>
</div>
</form>
</div>
{%endblock%}
```

PLANTILLA CAMBIO_DOMINIO.HTML

```
{% extends "base.html" %}
{% block title %} Dominio {% endblock %}
{% block content %}

<legend><h2>Combio de Plan</h2></legend>
<h3>Dominio {{dom.dominio}} tiene {{cuentas_actuales}} cuentas de {{dom.num_cuentas}}
contratadas.</h3><br>
<div class="row">
<div class="span3">
        <div class="well">
                <h2 class="muted">Basico</h2>
                <ul>
                        <li>Acceso Ilimitado</li>
                        <li>10 Usuarios(Mailboxes)</li>
                        <li>Soporte via página web</li>
                </ul>
                <hr>
                <h3>$10 / mes</h3>
                </hr>
                <form action="" method="post">{% csrf_token %}
                <p><button class="btn btn-large" name= "basic" value="basic" type="submit"><i
class="icon-ok"></i>Seleccionar</button></p></form>
        </div>
</div>
<div class="span3">
        <div class="well">
                <h2 class="text-warning">Standard</h2>
                <ul>
                        <li>Acceso Ilimitado</li>
                        <li>50 Usuarios(Mailboxes)</li>
                        <li>Soporte via e-mail</li>
                </ul>
                <hr>
                <h3>$50 / mes</h3>
                </hr>
        <form action="" method="post">{% csrf token %}
                <p><button class="btn btn-large" name= "standard" value="standard" type="submit"><i
class="icon-ok"></i>Seleccionar</button></p></form>
        </div>
</div>
<div class="span3">
        <div class="well">
                <h2 class="text-info">Pro</h2>
                <ul>
                        <li>Acceso Ilimitado</li>
                        <li>100 Usuarios(Mailboxes)</li>
                        <li>Soporte telefónico</li>
                </ul>
                <hr>
                <h3>$100 / mes</h3>
                </hr>
                <form action="" method="post">{% csrf_token %}
                <p><button class="btn btn-large" name= "pro" value="pro" type="submit"><i
class="icon-ok"></i>Seleccionar</button></p></form>
        </div>
</div>
</form>
</div>
{% endblock %}
```

PLANTILLA NEW_USER.HTML

```
{% extends "base.html" %}
{% block title %} Nueva Cuenta {% endblock %}
{% block content %}
<fieldset>
          <legend><h2>Registro Nuevo Usuario</h2></legend>
<h3>{{ dominio }}</h3>
<h4>{{mensaje}}</h4>
<form action="" method="post">
{% csrf_token %}
{% if form.nombre.errors %}<div class="control-group error">{% endif %}
    <div class="control-group">
        <label for="nombre" class="control-label">Nombre </label>{{ form.nombre }}
        {% if form.nombre.errors %}<span class="help-inline"> Nombre requerido</span>{% endif
%}</div>

{% if form.address.errors %}<div class="control-group error">{% endif %}
        <div class="control-group">
        <label for="address" class="control-label">Direccion </label>{{ form.address }}
        {% if form.address.errors %}<span class="help-inline"> Direccion requerida</span>{% endif
%}</div>

{% if form.passw.errors %}<div class="control-group error">{% endif %}
        <div class="control-group">
        <label for="passw" class="control-label">Clave </label>{{ form.passw }}
        {% if form.passw.errors %}<span class="help-inline"> Clave requerida</span>{% endif %}</div>

{% if form.passwuno.errors %}<div class="control-group error">{% endif %}
        <div class="control-group">
        <label for="passw" class="control-label">Confirmar Clave </label>{{ form.passwuno }}
        {% if form.passwuno.errors %}<span class="help-inline"> Clave requerida</span>{% endif
%}</div>

        <input type="submit" class="btn btn-primary" value="Nueva Cuenta" name="new cuenta"></input>
        <input type="submit" class="btn" value="Cancelar" name="cancelar"></input></div>
</form>
</fieldset>
{% endblock %}
```

PLANTILLA CONTRASENA.HTML

```
{% extends "base.html" %}
{% block title %} Cambio de Clave {% endblock %}
{% block content %}
<legend><h2>{{cuenta}}</h2></legend>
<form action="" method="post">
{% csrf token %}
{% if form.errors %}<p> Corregir los campos: </p> {% endif %}
<div class="register_div">
        {% if form.password.errors %}<p class="error">{{ form.password.errors }}</p>{% endif %}
        <p><label for="password"{% if form.password.errors %} class="error"{% endif
%}>Password:</label> {{ form.password }}</p>
</div>
<div class="register_div">
        {% if form.passworduno.errors %}<p class="error">{{ form.passworduno.errors }}</p>{% endif
%}
        <p><label for="password1"{% if form.passworduno.errors %} class="error"{% endif %}>Verify
Password:</label>{{ form.passworduno }}</p>
</div>
<input type="hidden" name="cue" value="{{cuenta.address}}"></input>
<input value="Cambiar" name="cambiar" class="btn btn-success" type="submit"/></input>
<input type="submit" value="Cancelar" class="btn" name="cancelar"></input>
</form>
{% endblock %}
```

PLANTILLA CRM.HTML

```
{% extends "servicios_base.html" %}
{% block titulo %} CRM {% endblock %}
{% csrf token %}
```

PLANTILLA OP.HTML

```
{% extends "servicios_base.html" %}
{% block titulo %} OPENMETTINGS {% endblock %}
{% csrf_token %}
```

PLANTILLA RESIZE.HTML

```
{% extends "servicios_base.html" %}
{% block titulo %} {{servicio nombre}} {% endblock %}
{% block servicio %}
<legend><H2>{{serv_actual.servicio.nombre_completo}}</H2></legend>
<h3>La capacidad actual del servicio es : {{serv_actual.capacidad}} Gb<h3/>
{% endblock %}
{% csrf_token %}
```

PLANTILLA PANEL.HTML

```
{% extends "base.html" %}
{% block title %} PANEL DE CONTROL {% endblock %}
{% block content %}
<legend><h2>Panel de Control</h2></legend>
<h3>Usuario: {{user}}</h3>
<br><br>
<a class="btn btn-info btn-large" href="{% url vista_facturacion %}">Facturacion</a><br><br>
<table class="table table-hover">
<thead>
<th>Servicio</th><th>Acceso</th><th>Capacidad</th><th></th></thead>
<tbody>
{%if servicios%}
{% csrf token %}
{%for serv in servicios%}
    {%if serv.servicio.nombre == 'mail'%}
                <td>{{serv.servicio.nombre_completo}}<br>
                <a href="http://trac.roundcube.net/wiki/User_Guide_ES" class="btn btn-link btn-
small">Manual</a></td>
                <td><a
href="https://{{serv.servidor}}/mail">https://{{serv.servidor}}/mail</a></td>
                <td><form action="" method="post">
                {% csrf_token %}
                {{serv.capacidad}}GB
                <input type="hidden" name="servi" value="{{ serv.id }}"></input>
                <button class="btn" type="submit"><li class="icon-
pencil"></li></button><br></td></form>
                <td><form action="" method="post">
                {% csrf_token %}
                {% if serv.suspendido == None %}Empty{%else%}{%if serv.suspendido%}
                                    <input type="hidden" name="servicio" value="{{
serv.servicio.nombre }}"></input>
                                    <input type="hidden" name="server" value="{{
serv.servidor.uuid }}"></input>
                                    <br><input type="submit" value="Activar" class="btn btn-
success" name="_active">
                                    <input type="submit" class="btn btn-danger"
value="Terminar" name="_end"></td></tr>
                                    {%else%}
```

```
                                        <a href="{% url vista_conf_mail %}">Configuracion</a>
                                        <input type="hidden" name="servicio" value="{{
serv.servicio.nombre }}"></input>
                                        <input type="hidden" name="server" value="{{
serv.servidor.uuid }}"></input>
                                        <input type="submit" value="Suspender" class="btn btn-
warning" name="_suspend"><br></td></tr>{%endif%}{%endif%}
                </form>
                {%endif%}
                {%if serv.servicio.nombre == 'crm'%}
                        <td>{{serv.servicio.nombre_completo}}<br>
                        <a
href="http://support.sugarcrm.com/02_Documentation/01_Sugar_Editions/05_Sugar_Community_Edition/Sug
ar_Community_Edition_6.5/Sugar_Community_Edition_Application_Guide_6.5.0" class="btn btn-link btn-
small">Manual</a></td>
                        <td><a
href="https://{{serv.servidor}}/sugarcrm">https://{{serv.servidor}}/sugarcrm</a></td>
                        <td><form action="" method="post">
                        {% csrf_token %}
                        {{serv.capacidad}}GB
                        <input type="hidden" name="servi" value="{{ serv.id }}"></input>
                        <button class="btn" type="submit"><li class="icon-
pencil"></li></button><br></td></form>
                        <td><form action="" method="post">
                        {% csrf_token %}
                        {% if serv.suspendido = None %}Empty{%else%}{%if serv.suspendido%}
                                        <input type="hidden" name="servicio" value="{{
serv.servicio.nombre }}"></input>
                                        <input type="hidden" name="server" value="{{
serv.servidor.uuid }}"></input>
                                        <br><input type="submit" value="Activar" class="btn btn-
success" name=" active">
                                        <input type="submit" class="btn btn-danger"
value="Terminar" name="_end" ></td></tr>
                                {%else%}
                                        <input type="hidden" name="servicio" value="{{
serv.servicio.nombre }}"></input>
                                        <input type="hidden" name="server" value="{{
serv.servidor.uuid }}"></input>
                                        <input type="submit" value="Suspender" class="btn btn-
warning" name="_suspend"><br></td></tr>{%endif%}{%endif%}</form>
                {%endif%}
                {%if serv.servicio.nombre == 'op'%}
                        <td>{{serv.servicio.nombre_completo}}<br>
                        <a href="https://cwiki.apache.org/OPENMEETINGS/manuals-user-openmeetings-
various.data/Manual%20usuario%20OpenMeetings%201.9.1%20espanol.pdf" class="btn btn-link btn-
small">Manual</a></td>
                        <td><a
href="http://{{serv.servidor}}:5080/openmeetings">http://{{serv.servidor}}:5080/openmeetings</a><br
>
                        <td><form action="" method="post">
                        {% csrf token %}
                        {{serv.capacidad}}GB
                        <input type="hidden" name="servi" value="{{ serv.id }}"></input>
                        <button class="btn" type="submit"><li class="icon-
pencil"></li></button><br></td></form>
                        <td><form action="" method="post">
                        {% csrf_token %}
                        {% if serv.suspendido = None %}Empty{%else%}{%if serv.suspendido%}
                                        <input type="hidden" name="servicio" value="{{
serv.servicio.nombre }}"></input>
                                        <input type="hidden" name="server" value="{{
serv.servidor.uuid }}"></input>
                                        <bbr><input type="submit" value="Activar" class="btn btn-
success" name="_active">
                                        <input type="submit" class="btn btn-danger"
value="Terminar" name=" end"></td></tr>
                                {%else%}
                                        <input type="hidden" name="servicio" value="{{
serv.servicio.nombre }}"></input>
                                        <input type="hidden" name="server" value="{{
serv.servidor.uuid }}"></input>
                                        <input type="submit" value="Suspender" class="btn btn-
warning" name=" suspend"><br></td></tr>{%endif%}{%endif%}</form>
                {%endif%}
{%endfor%}
</tbody></table>
<br><br><br>
{% else %}
No tiene servicios contratados<br>
```

```
{%endif%}
<a href="{% url vista servicios %}" class="btn btn-success"> Generar servicios </a>
{% endblock %}
```

ANEXO M

PLANTILLAS BASE

PLANTILLA BASE.HTML

```
<!DOCTYPE HTML>
<HTML>
<HEAD>
        <TITLE>{% block title %} {% endblock %}</TITLE>
    <link rel = "stylesheet" href="/media/bootstrap/css/bootstrap.css">
        <link rel = "stylesheet" href="/media/bootstrap/css/bootstrap-responsive.css">
        <script src="/media/js/jquery.js"></script>
        <script src="/media/bootstrap/js/bootstrap.js"></script>
        <style type="text/css">
                body{ padding-top: 60px; padding-bottom: 40px;}
        </style>
</HEAD>
<BODY>
<div class="navbar navbar-inverse navbar-fixed-top">
        <div class = "navbar-inner">
                <a class="brand">EPN Cloud</a>
                {%if user.is_authenticated %}
                        <p class="navbar-text pull-right">
                                Bienvenido
{{user.get_profile.nombre}}  {{user.get_profile.apellidos}}
                                <a href="{% url vista_logout %}"; text-align:left> Cerrar Sesion</a>
                        </p>
                        <ul class="nav">
                                <li><a href="{% url vista_principal %}"><i class="icon-home icon-
white"></i>Inicio </a></li>
                                <li><a href="{% url vista_panel %}"><i class="icon-cog icon-
white"></i> Panel de Control </a></li>
                                <li><a href="{% url vista_servicios %}"><i class="icon-th-large
icon-white"></i>Servicios </a></li>
                                <li><a href="{% url vista_contacto %}"><i class="icon-comment icon-
white"></i>Contacto </a></li>
                        </ul>
                {%else%}
                <p class="navbar-text pull-right">
                <a href="{% url vista_login %}"; text-align:right><i class="icon-user icon-
white"></i> Iniciar Sesion </a>
                <a href="{% url vista_register %}"; text-align:right><i class="icon-user icon-
white"></i><i class="icon-plus icon-white"></i>Registro</a></p>
                         <ul class="nav">
                <ul class="nav">
                        <li><a href="{% url vista principal %}"><i class="icon-home icon-white"></i>
Inicio </a></li>
                        <li><a href="{% url vista_servicios %}"><i class="icon-th-large icon-
white"></i> Servicios </a></li>
                        <li><a href="{% url vista_contacto %}"><i class="icon-comment icon-
white"></i>Contacto </a></li>
                </ul>
                {% endif %}
        </div>
</div>
<div class="container-fluid">
        <div class= "row-fluid">
                <div class="span12">
                        <div class="hero-unit">
                        {% block content %}
                        {% endblock %}
                        </div>
                </div>
        </div>
</div>
<footer>
Autor: Daniel Nunez E.<br>
<img src="/media/images/django icon.gif">
<!--<img src="/media/images/os.png">-->
</footer>
</BODY>
</HTML>
```

PLANTILLA SERVICIOS_BASE.HTML

```
{% extends "base.html" %}
<HEAD>
        <TITLE>{% block titulo %} {% endblock %}</TITLE>
</HEAD>
{% block content %}
<SECTION>
        {% block servicio %}
        {% endblock %}
</SECTION>
<fieldset>
<legeng><h2>Pricing</h2></legend>
<div class="row">
{%if user.is_authenticated %}
<form action="" method="post">
<div class="span3">
    <div class="well">
                <h2 class="muted">Basico</h2>
                <ul>
                        <li>Acceso Ilimitado</li>
                        <li>3GB de almacenamiento</li>
                        <li>Soporte via página web</li>
                </ul>
                <hr>
                <h3>$145 / mes</h3>
                </hr>
                <form action="." method="POST">{% csrf_token %}
                <p><button class="btn btn-large" name= "basic" value="basic" type="submit"><i
class="icon-ok"></i>Seleccionar</button></p></form>
        </div>
</div>
<div class="span3">
        <div class="well">
                <h2 class="text-warning">Standard</h2>
                <ul>
                        <li>Acceso Ilimitado</li>
                        <li>10GB de almacenamiento</li>
                        <li>Soporte via e-mail</li>
                </ul>
                <hr>
                <h3>$175 / mes</h3>
                </hr>
                <form action="." method="POST">{% csrf_token %}
                <p><button class="btn btn-large" name= "standard" value="standard" type="submit"><i
class="icon-ok"></i>Seleccionar</button></p></form>
        </div>
</div>
<div class="span3">
        <div class="well">
                <h2 class="text-info">Pro</h2>
                <ul>
                        <li>Acceso Ilimitado</li>
                        <li>20GB de almacenamiento</li>
                        <li>Soporte telefónico</li>
                </ul>
                <hr>
                <h3>$280 / mes</h3>
                </hr>
                 <form action="." method="POST">{% csrf_token %}
                <p><button class="btn btn-large" name= "pro" value="pro" type="submit"><i
class="icon-ok"></i>Seleccionar</button></p></form>
        </div>
</div>
{% else %}
<div class="span3">
        <div class="well">
                <h2 class="muted">Basico</h2>
                <ul>
                        <li>Acceso Ilimitado</li>
                        <li>3GB de almacenamiento</li>
                        <li>Soporte via página web</li>
                </ul>
                <hr>
                <h3>$145 / mes</h3>
                </hr>
        </div>
</div>
<div class="span3">
```

```
        <div class="well">
                <h2 class="text-warning">Standard</h2>
                <ul>
                        <li>Acceso Ilimitado</li>
                        <li>10GB de almacenamiento</li>
                        <li>Soporte via e-mail</li>
                </ul>
                <hr>
                <h3>$175 / mes</h3>
                </hr>
        </div>
</div>
<div class="span3">
        <div class="well">
                <h2 class="text-info">Pro</h2>
                <ul>
                        <li>Acceso Ilimitado</li>
                        <li>20GB de almacenamiento</li>
                        <li>Soporte telefónico</li>
                </ul>
                <hr>
                <h3>$280 / mes</h3>
                </hr>
        </div>
</div>
<div class="span4">
<div class="controls">
        <hr>
                <form action="." method="POST">
                        {% csrf_token %}
                        <a class="btn btn-success" href="{% url vista login %}"> Iniciar Sesion
</a>
                        <a class="btn btn-primary" href="{% url vista_register %}"> Registrarse
</a>
                </form>
        </hr>
</div></div>
{% endif %}
</div>
</fieldset>
{%endblock%}
```

ANEXO N

COTIZACIÓN DEL SERVICIO DE COLOCACIÓN DE SERVIDORES

Precio de la Solución

telconet
la fibra del Ecuador

Numero de Unidades Us Iniciales: Rack Estandar: 7 Unidades

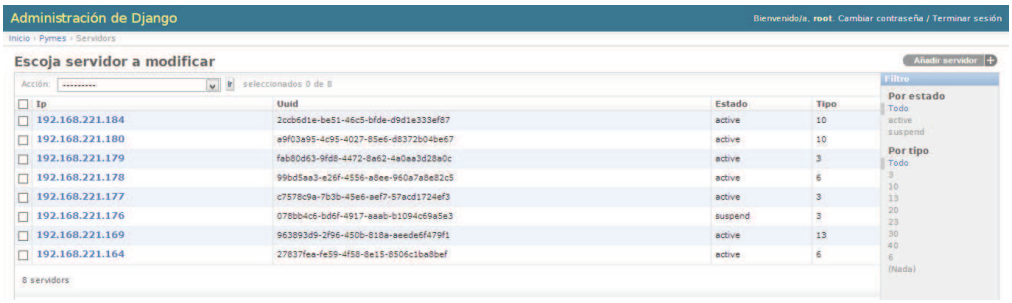
Costo de Alquiler	($150 c/unidad)	$ 1050,00
Costo de 2KVAs (hasta 3 Kvas: $280 c/u) en el Rack		$ 560.00
Costo de conectividad Datos Local 1 Mbps		$ 135.00
Costo de conectividad Internet desde DC 1 Mbps		$ 120.00
Soporte Manos Remotas (Manos Remotas Predefinidas (8 interacciones mensuales de hasta 5 min)		$ 80.00
Total Mensual		**$ 1945.00**
Costo Instalación para 7 Us. Standard (Incluye instalación eléctrica)		$ 300,00
Costo Instalación Conectividad Datos 1 Mbps		$ 100,00
Costo Instalación Conectividad Internet desde DC		$ 100,00
Total Instalación		**$ 500,00**
Fecha de Implementación en **DC /UIO:**	**15 días después de la firma del contrato**	
Tiempo mínimo de contratación:		3 años

Innovación y
Garantía de Calidad

ANEXO O

INFORMACIÓN DE LA BASE DE DATOS GENERADA POR LAS PRUEBAS DE USUARIO

REGISTROS DE LA TABLA SERVIDORES

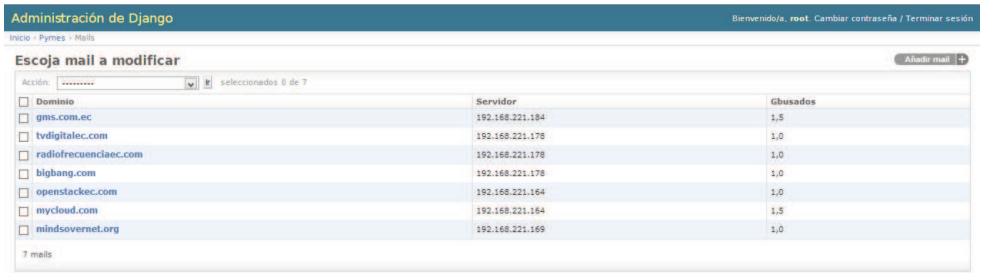


REGISTROS DE LA TABLA CLIENTES

REGISTROS DE LA TABLA FACTURAS

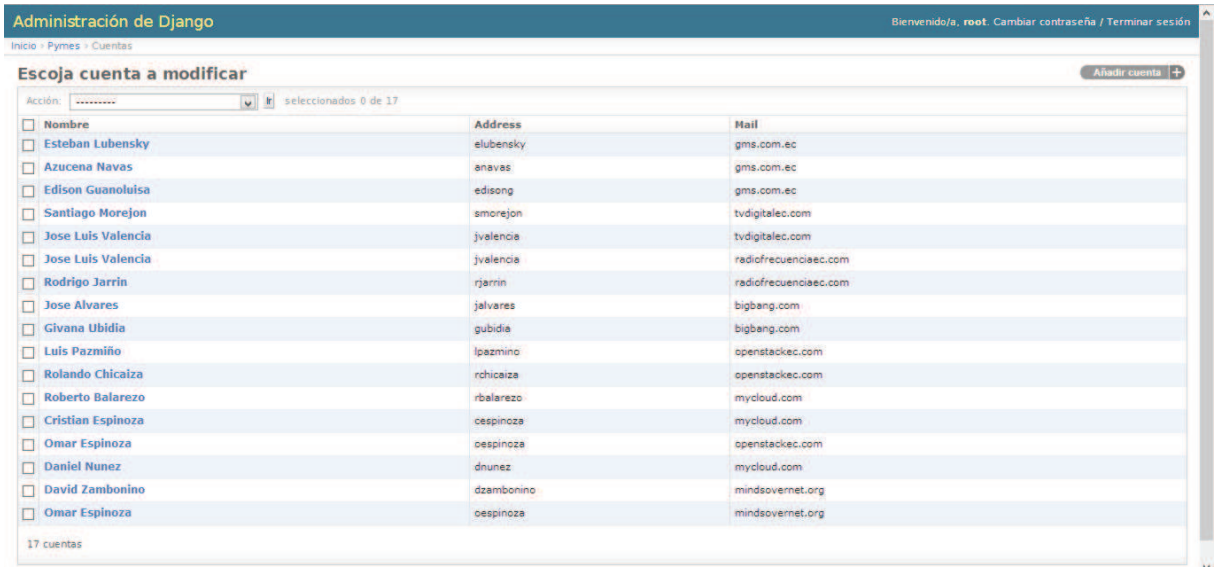
REGISTROS DE LA TABLA MAILS

Estos registros corresponden a los dominios creados por los usuarios.



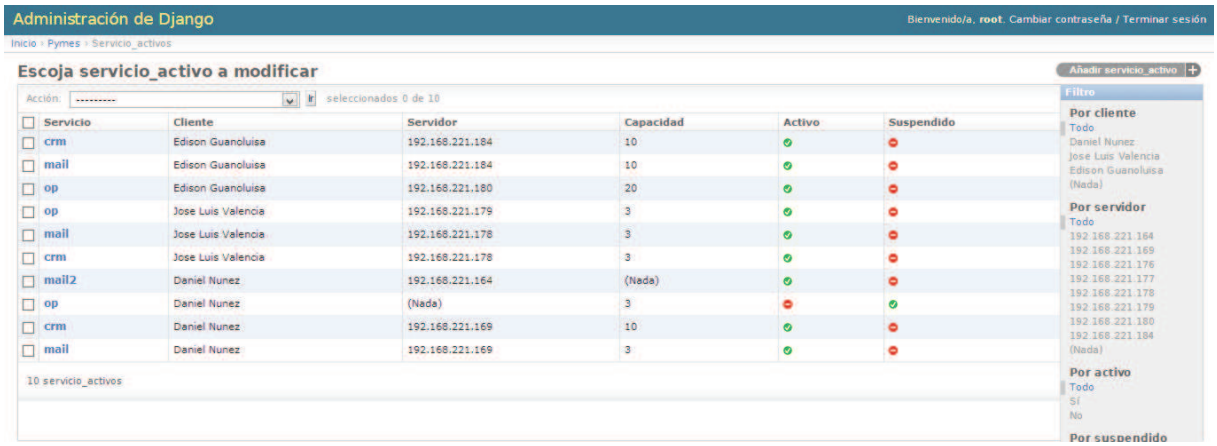
REGISTROS DE LA TABLA CUENTAS

Estos registros corresponden a las cuentas de correo de cada dominio.



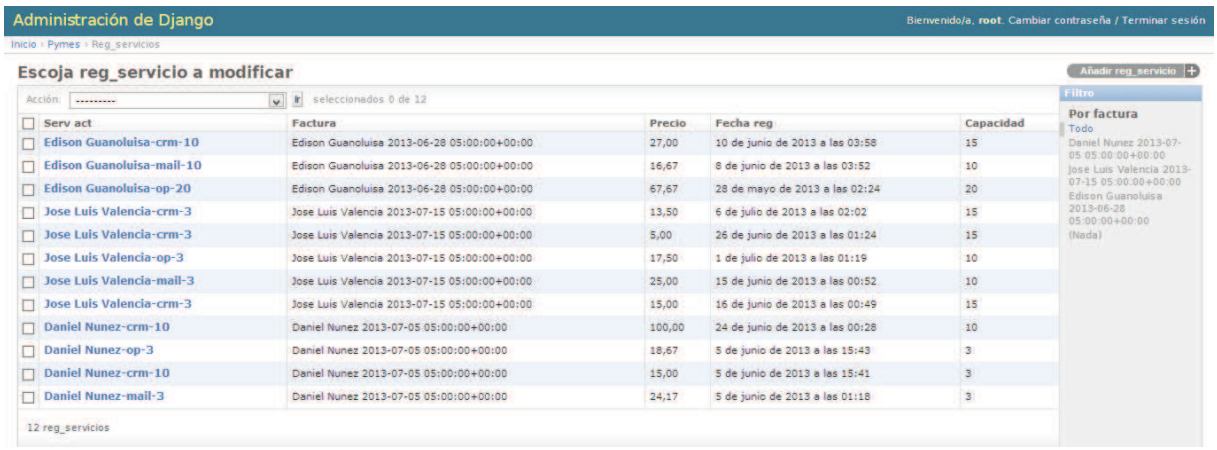
REGISTROS DE LA TABLA SERVICIOS ACTIVOS

Estos registros corresponden al estado de los servicios contratados por los clientes.



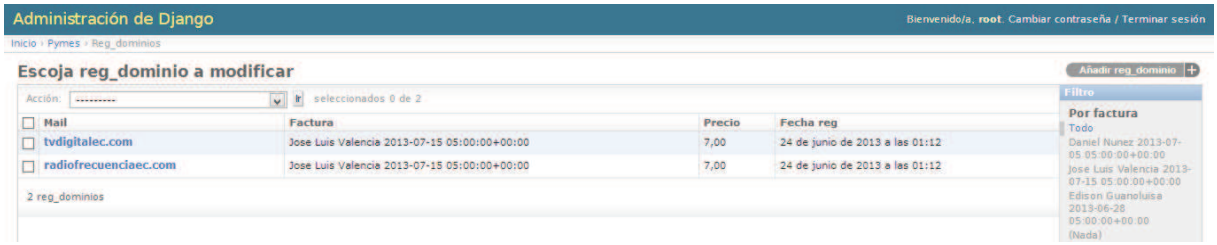
REGISTROS DE LA TABLA REG_SERVICIOS

Estos registros corresponden a los registros de facturación, generados por la creación, suspensión o cancelación de los servicios por parte del cliente.



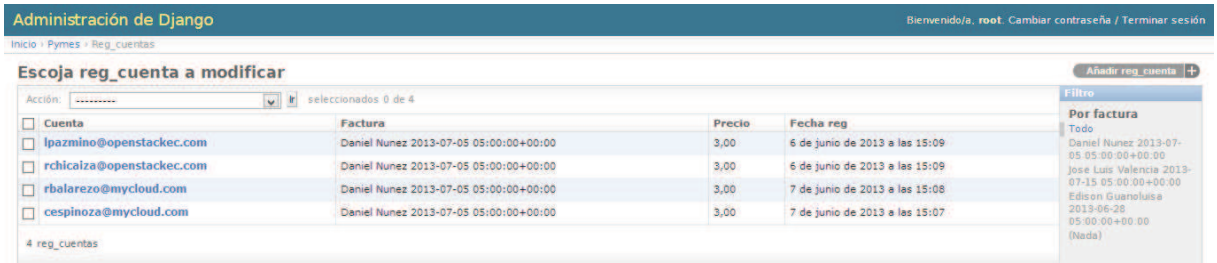
REGISTROS DE LA TABLA REG_DOMINIOS

Estas entradas en la base de datos corresponden a los registros de facturación de los dominios adicionales en el servicio de Correo Electrónico Ilimitado.



REGISTROS DE LA TABLA REG_CUENTAS

Estas entradas en la base de datos corresponden a los registros de facturación de las cuentas creadas en el servicio de Correo Electrónico Limitado.



ANEXO P

PROCESO DE INCERSIÓN DE DISPOSITIVOS Y SENSORES EN EL SOFTWARE DE MONITOREO PRTG

En la pantalla principal de la interfaz web del software PRTG, en la parte superior izquierda se escoge *Aparatos* > *Añadir Grupo* para crear un grupo de dispositivos. A continuación se escoge el grupo "Servidores" y por último se crea el grupo llamado "*cloud*", como se observa en la Figura P-1.

Figura P-1 Creación de grupo de dispositivos en PRTG

En la Figura P-2 se muestra el proceso para agregar un dispositivo al grupo de monitoreo "*cloud*". Para ello, hay que dirigirse a *Aparatos* > *Añadir aparato*, lo que despliega una ventana para seleccionar *"Añadir aparato a grupo existente"*, a continuación se escoge el grupo donde se desea agregar el dispositivo y se hace click en "Continuar". Finalmente se escribe el nombre y la dirección IP del dispositivo.

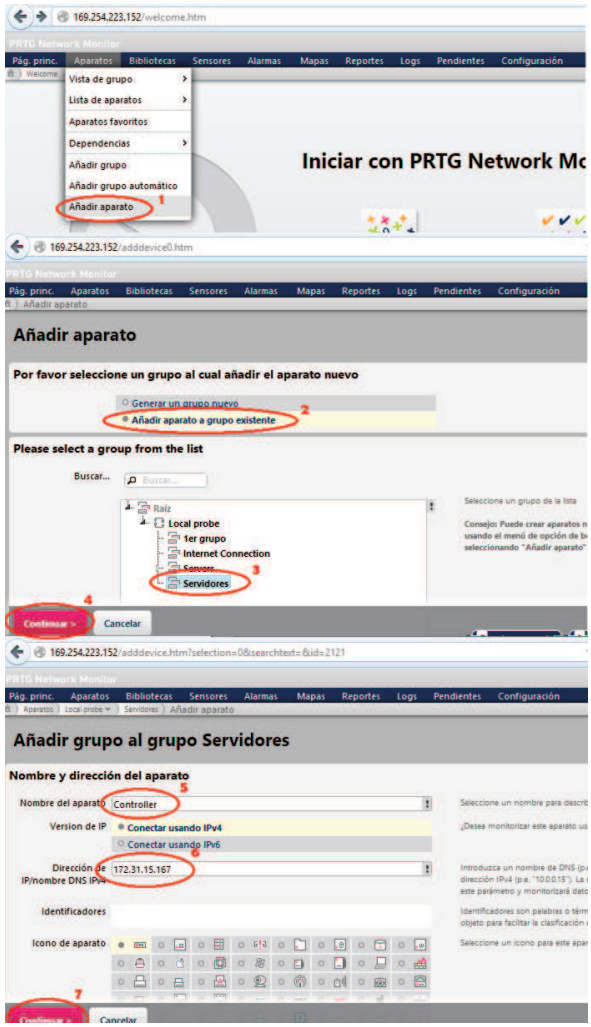


Figura P-2 Creación de un aparato en PRTG

Para monitorizar un aspecto específico de un aparato, se debe agregar un sensor. Para esto, hay que dirigirse a los aparatos y escoger "Añadir sensor", se desplegará una ventana donde aparece una lista predeterminada de sensores ordenados según

el tipo, el sistema operativo y la tecnología de monitorización. Una vez elegido el sensor, se despliega una nueva ventana donde se puede editar el nombre del sensor y el intervalo de escaneo, por defecto es 60 segundos, pero con la finalidad de tener mayor detalle del comportamiento de un dispositivo, se ha usado un intervalo de escaneo de 5 segundos. En la Figura P-3 se muestra todo el proceso para agregar un sensor a un aparato.

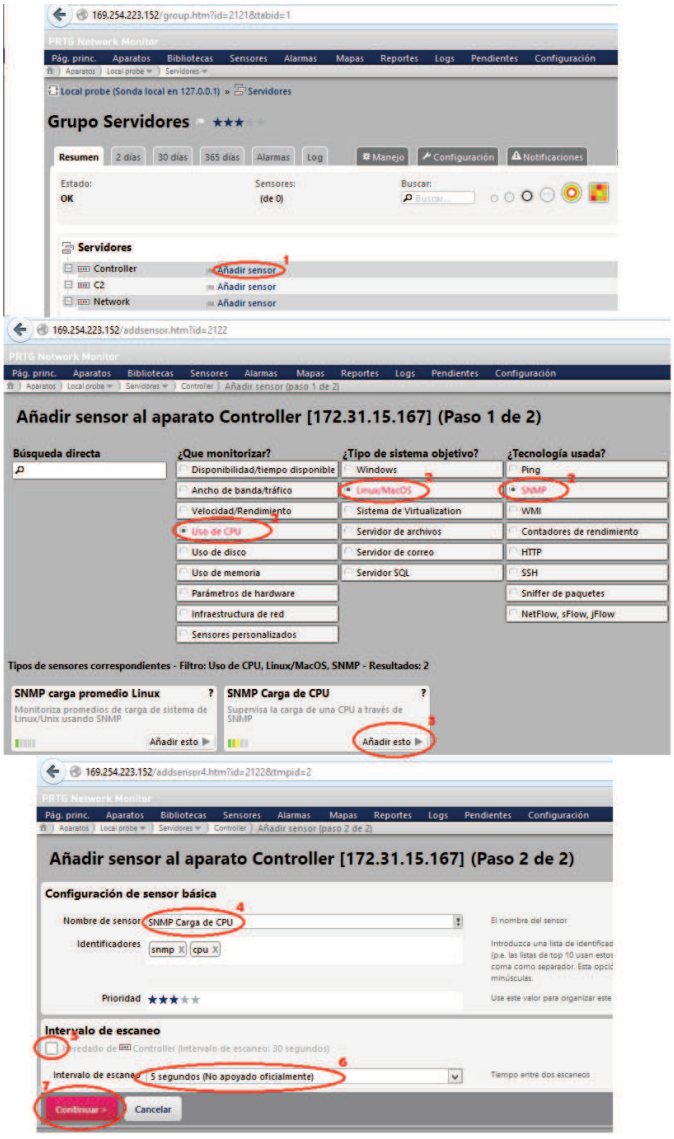


Figura P-3 Creación de un sensor en PRTG

ANEXO Q

CUADRO DE PREVISIÓN DE VENTAS MENSUALES Y CONSUMO DE RECURSOS

Recursos						Mes																																
Servicio	Precio($)	Costo($)	Ram(MB)	Disco(GB)	Internet(kbps)	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
CRM 15	45	1,95	256	15	6,1	2	1	1	1	2	1	3	3	4	2	3	4	4	7	7	8	8	9	10	12	13	15	15	18	20	21	21	21	21	32	32	32	32	49	49	64
CRM 50	150	3,61	512	50	20,2	1	1	2	2	2	2	2	2	2	3	2	2	3	4	4	5	5	6	7	8	9	10	12	14	13	13	13	13	13	13	13	30	33	40	48	52
CRM 100	300	6,72	768	100	40,5	1	2	1	2	2	3	3	3	3	2	4	4	4	4	5	7	6	7	8	9	10	11	14	15	15	15	17	17	17	17	17	17	17	41	41	53
Mail1 10	25	8,35	256	10	60,7	1	2	1	1	2	2	2	2	2	3	2	2	2	3	5	6	7	8	9	11	13	12	13	14	15	16	18	21	21	26	32	35	39	43	49	55
Mail1 20	40	19,01	512	20	151,7	1	1	2	2	1	2	1	1	1	2	3	4	3	4	5	5	6	7	9	9	10	12	12	14	14	15	18	18	22	22	28	30	30	38	42	46
Mail1 50	65	55,31	768	50	455,1			1	2	2	2	2	3	2	3	2	4	4	4	3	4	4	5	6	6	8	8	10	11	14	14	14	15	17	19	22	27	29	32	39	42
Mail 2	3	0.31<x<0.45	256<x<4006	10<x<1000	115<x<1100	25	32	39	54	75	93	96	122	134	149	163	168	173	179	185	197	215	219	228	245	267	279	294	315	334	355	353	384	417	454	494	511	556	585	609	625
OM 10	35	12,17	256	10	93,3	2	2	1	1	2	3	2	3	3	4	3	3	4	4	4	5	5	6	7	8	10	11	13	15	16	16	21	21	21	21	34	36	39	44	47	51
OM 20	70	24,93	512	20	202,2	1	2	2	2	1	2	3	2	2	2	2	3	3	4	3	4	6	6	5	7	6	8	10	11	12	12	14	14	14	21	21	21	27	30	33	36
OM 50	140	56,67	768	50	466,7		1	1	2	2	1	2	1	2	2	3	4	3	2	4	3	4	5	5	6	6	7	8	10	10	12	12	12	12	12	12	21	21	21	30
				Ingreso mensual($)		820	1331	1247	1797	1855	2169	2403	2371	2527	2497	3089	3529	3549	3847	4400	5286	5385	6197	6854	7810	8591	9537	11287	12675	12997	13450	14529	14762	15151	16502	17662	20813	23413	33515	35867	42905
				Costo de Producción($)		101,3	197,4	250,6	374,8	360,7	371,4	426,8	422,3	429,9	523,2	539,9	754,9	696,9	704,1	781,8	849,3	983,0	1148,2	1252,7	1408,5	1574,5	1741,5	2024,2	2315,9	2534,5	2684,7	2881,9	2973,6	3172,2	3534,0	4036,9	4468,4	5335,7	6053,0	6715,2	7770,9
				Gasto variable($)		15,0	139,5	139,5	264,0	265,5	268,5	391,5	391,5	393,0	516,0	517,5	940,5	646,5	655,5	781,5	792,0	1073,0	1344,0	1180,5	1315,5	1569,0	1996,5	2117,0	2518,5	2475,0	3150,0	3030,5	3265,5	3952,5	4622,0	7729,0	5637,0	5791,5	6602,0	7512,5	8174,0
				Recursos Consumidos																																				
				RAM(MB)		3840	6144	6656	8960	8704	9728	10752	10752	11008	12032	13568	17152	17133	17590	19456	22784	24576	28672	32000	36352	40448	45056	52480	59648	62976	65536	71424	72960	76544	85760	96000	110848	125696	162304	177664	208128
				Almacenamiento(GB)		256	423	425	629	629	718	789	786	804	817	986	1192	1183	1280	1451	1759	1784	2090	2347	2651	2962	3275	3899	4389	4564	4714	5083	5171	5359	5824	6294	7488	8389	11861	12847	15316
				Internet(kbps)		749,9	1535,1	1989,0	2996,7	2866,6	2936,2	3381,2	3339,7	3393,7	4167,7	4286,2	6038,2	5548,6	5553,2	6160,9	6649,2	7746,8	9053,2	9857,6	11067,8	12369,0	13685,4	15921,4	18225,7	19996,8	21212,3	22734,7	23465,9	25083,1	27891,3	31865,6	35208,8	42252,3	47450,9	52728,9	60951,1
				IPs máximo		10	13	13	16	17	19	21	21	22	24	25	31	31	37	41	48	52	60	67	77	86	95	108	123	130	135	149	153	159	184	212	241	268	339	370	430