UNIVERSIDAD CENTRAL DE VENEZUELA

FACULTAD DE CIENCIAS

ESCUELA DE COMPUTACIÓN

Implantación de plataforma telefónica basada en Asterisk/Kamailio para la interconexión de islas VoIP usando un plan de numeración unificado

Trabajo Especial de Grado presentado ante la Ilustre Universidad Central de Venezuela por el
Br. Sebastian Lenin Sulbaran Rivas (C.I. V-20142794)
Para optar al título de Licenciado en Computación

Tutor: Prof. Rafael A. Angulo R.

Ciudad Universitaria de Caracas, mayo 2017

Universidad Central de Venezuela

Facultad de Ciencias

Escuela de Computación

Implantación de plataforma telefónica basada en Asterisk/Kamailio para la interconexión de islas VoIP usando un plan de numeración unificado

Autor: Sebastian Lenin Sulbaran Rivas

Correo-e: Sebastian.sulbaran@hotmail.com

Tutor: Prof. Rafael Alejandro Angulo

Correo-e: Rafael.angulo@tarma.com.ve

Fecha: Mayo 2017

RESUMEN

La telefonía IP, es una tecnología que ha venido creciendo y tomando auge con el pasar de los últimos años. La adopción de esta tecnología en organizaciones y empresas, les permite a estas establecer comunicación entre sus usuarios a bajos costos y sin la necesidad de invertir en infraestructuras para el soporte exclusivo del tráfico de voz como se hacía anteriormente, permitiendo de esta forma, el aprovechamiento de las redes de datos ya instaladas. La telefonía IP permite que organizaciones puedan comunicarse incluso entre sucursales, pero, ¿qué es necesario para que se puedan seguir aprovechando aún más las bondades de esta tecnología, y permita que distintas organizaciones puedan comunicarse entre sí por medio de la misma?

En el presente trabajo, se plantea la implementación de una plataforma telefónica conformada por el servidor de llamadas Asterisk y el servidor SIP Proxy Kamailio como propuesta de solución a esta interrogante. Además, se diseña y desarrolla una aplicación web básica para el registro de organizaciones en esta plataforma.

En el desarrollo de esta aplicación, se utilizaron, el framework Laravel, así como los beneficios ofrecidos por la arquitectura en tiempo real de Asterisk (Asterisk), permitiendo el registro y establecimiento de enlaces troncales de manera dinámica. Para el desarrollo de ambas fases en las que consiste el presente trabajo, se siguieron los lineamientos establecidos en la metodología Scrum. Esta metodología nos permite realizar múltiples planificaciones en periodos cortos de tiempo permitiendo la constante evaluación que sirva para la toma de decisiones hasta lograr satisfacer los requerimientos planteados.

Como resultado de este trabajo, fueron obtenidos diversos ambientes de prueba donde se evalúan características particulares de los servidores a integrar hasta comprender y lograr dicha integración, así como la generación de la documentación necesaria para crear réplicas de estos ambientes y un listado de requerimientos mínimos necesarios para la participación de una organización que tenga implementada VoIP en la plataforma.

Palabras Clave: VoIP, redes, troncales, Laravel, Scrum, telefonía IP, Plan de Numeración, Asterisk, Kamailio.

UNIVERSIDAD CENTRAL DE VENEZUELA
FACULTAD DE CIENCIAS
ESCUELA DE COMPUTACIÓN

ACTA

Quienes suscriben, miembros del jurado designado por el Consejo de la Escuela de Computación, para examinar el Trabajo Especial de Grado titulado **"Implantación de plataforma telefónica basada en Asterisk/Kamailio para la interconexión de islas VoIP usando un plan de numeración unificado"** y presentado por el Br. **Sebastian Lenin Sulbaran Rivas (C.I. V-20142794)**, a los fines de optar al título de **Licenciado en Computación**, dejamos constancia de lo siguiente:

Leído como fue dicho trabajo, por cada uno de los miembros del jurado, se fijó el día 16 de mayo de 2017, a las 3:00 p.m. para que el autor lo defendiera en forma pública, lo que este hizo en la Sala 1 de la Escuela de Computación, mediante una presentación oral de su contenido. Luego de lo cual respondieron a las preguntas formuladas. Finalizada la defensa pública del Trabajo Especial de Grado, el jurado decidió aprobar con la nota de 20 puntos.

En fe de lo cual se levanta la presente Acta, en Caracas el día 16 de mayo de 2017.

Prof. Rafael A. Angulo R.
Tutor

Prof. Robinson Rivas
Jurado

Prof. Miguel Astor
Jurado

REPUBLICA DE VENEZUELA
Facultad de Ciencias
Escuela de Computación
Universidad Central - Caracas

AGRADECIMIENTOS

La gratitud es un agradable sentimiento el cual es importante desarrollar en cada uno de nosotros para apreciar mejor las cosas, valorarlas y obtener paz. Desde mi llegada a la ciudad de Caracas hasta el día de hoy, he conocido a muchas personas que me han ayudado y acompañado en este largo camino hacia la meta que hoy celebro. Son tantas que me es difícil poder mencionarlas a todas en esta breve sección, sin embargo, de no encontrarse aquí mencionados, sepan que igualmente les estoy agradecido.

Es por ello que le agradezco primeramente a Dios y a la Virgen del Carmen por este de muchos logros por venir. A mi madre, quien amo tanto, por su apoyo y amor incondicional (incluidos los regaños) desde toda la vida y siempre estar ahí en mi formación inicial como persona y profesional y motivarme a crecer e ir siempre hacia adelante, le agradezco de manera infinita y sé que la culminación de este logro, al igual que para mí, le representa mucha felicidad y sacrificio. Agradezco a mi padre ("mi apá") a quien también amo, quien con su esfuerzo incansable nos ha dado muchas oportunidades para seguir luchando por lo que queremos y buscamos, aconsejándome cada vez que hablamos, siendo además un ejemplo a seguir de constancia y perseverancia. Agradezco a mi hermano Omar ("josín") por su apoyo inicial en este camino, por las tantas pizzas de la panadería, los chinos de la cascada, y gratos y no tan gratos momentos que pasamos antes de irte a Mérida. Agradezco a mi hermana Julia ("julinis") quien en esta última etapa representó mucho apoyo y cariño para seguir redactando. A mis hermanos Cristian e Ismael por las echaderas de broma y los buenos momento estos últimos años (¡y que perduren por años venideros!). Agradezco a la señora Vivian quien también me brindó su apoyo y en incontables ocasiones me preparaba mi cafecito en las mañanas antes de salir a la universidad. Agradezco a Rudy, quien me acompaño durante gran parte del camino que hoy celebro. Agradezco mis amigas Yenny, Lyanne, Diana y Rut por su apoyo. Agradezco a mi tía Mirla quien me recibió en su casa y por ese cariño hacia mí. Agradezco a mi tío Guaicaipuro que, aunque no esté aquí físicamente, sé que le habría causado mucha felicidad este logro. Agradezco a mi abuela Teresita quien representa ese cariño de madre, agradezco a todos en la DTIC, en especial a mi segunda madre Adriana y a mis tías adoptivas Delisa, Patricia y Raquel. Agradezco a mis panas a quienes aprecio bastante, Hever y Julio. Agradezco a todos esos profesores que contribuyeron con mi formación como profesional y académica. Agradezco a mi tutor Rafael por esta oportunidad de crecimiento tanto personal como profesional. Agradezco a Mauricio quien sin darse cuenta es un excelente profesor. Agradezco a mi grupo de montaña (Edimar, Jonathan, Luisanna, Leonard, Daniel y Ritzi) quien siempre con su buena vibra me animaban a continuar. Agradezco a Sonia (a quien quiero mucho) por su apoyo estos últimos meses y finalmente agradezco a Emily, quien fue una de mis últimas motivaciones para la culminación de este trabajo.

A todos ustedes, a todos aquellos que pude y no pude mencionar, a todas esas buenas o malas circunstancias que hoy me traen aquí, gracias.

CONTENIDO

INDICE DE FIGURA

INDICE DE TABLAS

INTRODUCCION

La implementación y uso de la telefonía en Venezuela, data del año 1883 [1] cuando se lleva a cabo la instalación de los primeros teléfonos en la ciudad de Caracas, siendo este medio, al igual que en los demás países del mundo, el medio de comunicación preferido hasta la llegada del internet. Posteriormente con el surgir del internet y la Web, surgieron también distintas formas de comunicación basadas en estas tecnologías innovadoras.

Debido a que las infraestructuras de voz, de video y de datos, se encontraban separadas, el mantenimiento y costos que implicaba la implementación y mantenimiento de estas, era elevado. Poco a poco fueron desarrolladas aplicaciones que ofrecían los mismos servicios ofrecidos en las redes de voz y video (las redes separadas), pero utilizando las redes de datos.

Uno de estos servicios integrado a las redes de datos, fue la tecnología de transmisión de voz sobre IP (VoIP). Esta tecnología es joven en consideración con otras, sus primeros desarrollos datan de mediados de los años 90 [4] y su adopción e implementación se ha realizado de manera progresiva. Actualmente son muchas empresas que hacen uso de esta tecnología, pero en el caso de Venezuela, es una tecnología que recién está tomando auge.

Las organizaciones o instituciones que implementan la tecnología VoIP, hacen uso de los sistemas de telefonía tradicional para comunicarse con otras instituciones que, a su vez, también pudiesen implementar esta tecnología, generando así gastos recurrentes por la realización de llamadas usando los sistemas tradicionales.

En algunos casos, podemos encontrar que una organización puede estar conformada por una o más sedes que no necesariamente se encuentran en la misma locación geográfica. Aunque estas sedes pueden comunicarse entre ellas usando VoIP, la comunicación hacia otras instituciones se realiza como se mencionó anteriormente. Al hacer uso de los sistemas de telefonía tradicional, estas instituciones u organizaciones están desaprovechando el beneficio de comunicarse a través del internet usando telefonía IP entre estas "islas", siendo esta una característica bastante favorable que presenta esta tecnología. Donde una "isla VoIP" para efectos de este documento, es cualquier institución u organización que haga uso de la telefonía IP y utilice la PSTN para comunicarse con el exterior.

La penetración en la población del uso y acceso al internet en Venezuela (61.5% [2]), es un factor a tomar en consideración para el futuro desarrollo de la telefonía IP. La alta conectividad que se ha dado en el transcurso de los últimos años, hace que las empresas cuenten casi necesariamente con enlaces hacia el internet para proveer servicios que estos ofrecen o para trabajar. Estos enlaces pueden ser aprovechados para la transmisión de llamadas y el establecimiento de comunicaciones con otras organizaciones o instituciones.

Esta oportunidad de aprovechamiento de los enlaces a internet en el uso de servicios, no solo de datos, sino la posibilidad de interconexión entre las islas VoIP, fue una de las razones para el desarrollo del presente trabajo especial de grado y realizar la implantación de una plataforma de interconexión para llamadas VoIP junto con su interfaz web administrativa.

Esta plataforma crea la posibilidad de conformar una red a gran escala de telefonía IP, haciendo que sea posible la realización de llamadas VoIP entre distintas islas VoIP. La implementación de esta plataforma y el desarrollo de una interfaz web administrativa es el principal objetivo de este trabajo especial de grado.

A continuación, se presenta una breve descripción del contenido abarcado en este trabajo especial de grado:

Capítulo 1: Marco teórico del trabajo especial de grado. En este se incluyen los temas necesarios para el entendimiento del presente trabajo y son tratados los temas de telefonía tradicional, telefonía IP, protocolos utilizados en la telefonía IP, códec, calidad de servicio en redes de telefonía IP, planes de numeración, marco regulatorio de las telecomunicaciones en Venezuela, aplicaciones web y tecnologías utilizadas del lado del cliente y del servidor.

Capítulo 2: Marco metodológico. En este capítulo se presenta y describe la metodología utilizada para el desarrollo e implementación de este trabajo especial de grado (SCRUM).

Capítulo 3: Marco aplicativo. En este capítulo se expone el objetivo principal para el desarrollo de este trabajo especial de grado, así como la definición de los objetivos específicos para la realización del mismo. Es descrito también el planteamiento del problema y la justificación para el desarrollo del presente trabajo. Finalmente se describen las dos fases en las que fue realizado el presente trabajo. Las fases se encuentran divididas en las iteraciones realizadas para lograr la integración entre los servidores Kamailio y Asterisk como primera fase y como segunda fase, se describen las iteraciones para el desarrollo de la aplicación web administrativa de la plataforma integrada

Capítulo 4: Conclusiones y recomendaciones. En este capítulo se realiza un análisis de los resultados obtenidos, así como las conclusiones respectivas conforme a la finalización del presente trabajo.

CAPITULO 1

MARCO TEORICO

1.1 Telefonía tradicional y voz sobre IP

Uno de los derechos por naturaleza e indispensable para los seres humanos es la comunicación. El ser humano desde tiempos remotos ha hecho de la comunicación su manera de expresarse e interactuar tanto con su entorno como con sus iguales.

Este particular derecho innato ha sido incorporado en el desarrollo y avance de nuevas tecnologías de comunicación, por ejemplo, este deseo de comunicación fue el que impulsó el desarrollo para lograr computadoras interconectadas comunicándose entre sí; más tarde, con las mismas bases de comunicación e interconexión de computadoras, surge lo que actualmente conocemos como el Internet (ARPANET en sus inicios). Pero mucho antes de tener computadoras interconectadas, el teléfono fue una de las primeras tecnologías que sirvió para la comunicación masiva, estableciendo redes que interconectaban todos los teléfonos. De esta red de interconexión de teléfonos, surge lo que hoy se conoce como la telefonía tradicional.

La telefonía tradicional tenía como objetivo conectar los teléfonos, sea desde cualquier locación donde estos se encontrarán, con el resto de los teléfonos. Para lograr este objetivo, era necesario la creación de una red que conectara todos y cada uno de los teléfonos existentes (siendo estos analógicos) logrando así que las personas pudieran comunicarse a través de ellos. Cada una de estas redes usualmente era instalada por operadoras que ofrecían este tipo de servicios, la interconexión de estas operadoras conforma la red pública de telefonía tradicional o por sus siglas en ingles PSTN (Public switched Telephone Network).

Se puede definir la PSTN como una red de conmutación por circuitos creada, en principio, para la comunicación entre teléfonos y la transmisión de la voz por dicha red. Siendo esta red conformada por todas las operadoras de telefonía a nivel mundial.

Por medio de esta red los teléfonos pueden establecer conexión y comunicación con otros teléfonos de cualquier parte del mundo que pertenezcan a la PSTN.

De manera general la PSTN está conformada por:

- Teléfonos.
- Líneas de comunicación.
- Centrales telefónicas.

Los teléfonos están vinculados con las operadoras por medio de enlaces dedicados (líneas de comunicación). Por medio de estas líneas de comunicación, se establecen conexiones entre los teléfonos. Las conexiones son creadas al descolgar un teléfono y marcar un número telefónico destino, este destino será notificado y cuando el destinatario descuelgue el teléfono, la conexión será establecida por medio de las líneas de comunicación. Las conexiones forman un circuito dedicado en donde solo los participantes que estén vinculados a ese circuito pueden escuchar la llamada, otorgando un cierto nivel de calidad de servicio para las llamadas. Este circuito no es liberado hasta que una de las partes que mantienen el circuito, abandona la comunicación.

A continuación, describiremos los componentes presentes en la PSTN.

1.2 Componentes de la PSTN

Como toda plataforma de interconexión o red de interconexión, esta posee una topología básica asociada y componentes presentes en esta topología, estos componentes se mencionan a continuación:

- **Operadora local o LEC (Local Exchange Carrier)**: El LEC u operadora local es una entidad o dispositivo (i.e. un *switch*) en donde termina el bucle local y están conectados los teléfonos. Un LEC maneja la señalización, recolección de dígitos, enrutamiento y supervisión de llamadas, configuración de la llamada, entre otras características funcionales.

- **Dispositivos finales**

    - **Teléfonos analógicos:** Un teléfono analógico es un tipo de dispositivo que hace uso de la tecnología analógica para la transmisión de voz sobre las redes analógicas de telefonía. Estos tipos de teléfono son conectados a una central telefónica o a una *Private Branch Exchange* (PBX) y la energía eléctrica es suministrada por la central.

    - **Teléfonos digitales:** Los teléfonos digitales utilizan el mismo cableado para la conexión con la central o con la PBX pero estos transforman la voz analógica en señales digitales.

- **Bucles locales o última milla:** Los bucles locales o enlaces de última milla corresponden a la conexión cableada entre el teléfono, PBX o Key System con la oficina central o la operadora que presta los servicios telefónicos. Estos también indican hasta donde es responsable la operadora que presta los servicios.

- **PBX (Private Branch Exchange):** Cumplen con la misma funcionalidad que las centrales telefónicas, pero a un nivel local. Este tipo de centrales se pueden enlazar con otras oficinas centrales o con la PSTN por medio de enlaces troncales que permitan la comunicación entre ellas.

- **Troncales:** Las troncales telefónicas son conexiones entre PBX o entre Operadoras por donde las llamadas entre sistemas de telefonía son establecidas. Por las líneas troncales pueden circular una o más llamadas simultáneas dependiendo del tipo de enlace con el que sea establecida esta troncal. Algunos ejemplos de troncales digitales son T1, E1, DS0, POTS (Plain Old Telephone Service), BRI (Basic Rate Interface) y PRI (Primary Rate Interface)

- **Key systems:** Los key systems son dispositivos de gestión de llamadas generalmente usados en pequeñas empresas, estos poseen menos características para el control y gestión de llamadas que una PBX.

Podemos ver en la figura 1 cómo estos componentes están interrelacionados.

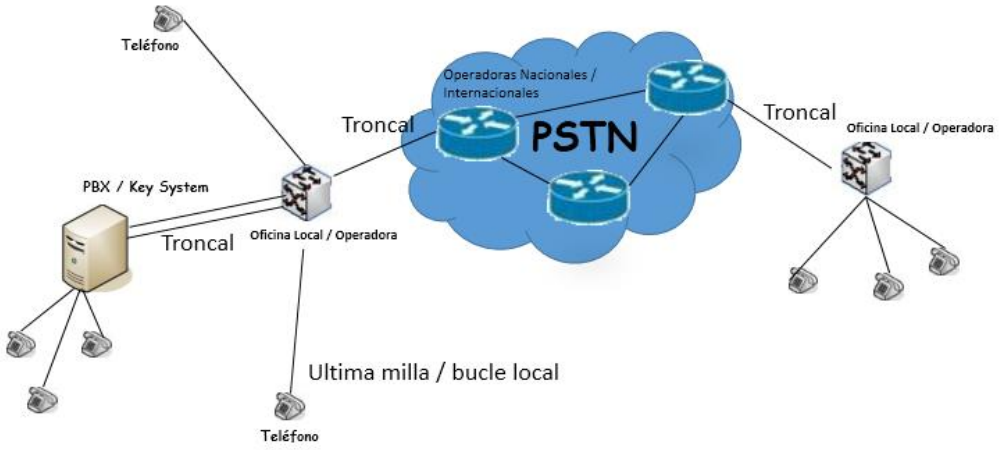

***Figura** 1:* Topología PSTN *1*
Fuente: Autor

Básicamente, cuando expandimos una nube PSTN vemos operadoras conectadas a través de troncales entre ellas.

1.3 ISDN (Red Digital de Servicios Integrados)

La PSTN es la predecesora de las redes actuales, pero esta estaba orientada solo a la transmisión de datos de voz a través de ella. Además de solo transportar datos de voz, esta presentaba problemas frecuentes en las redes analógicas, como problemas en las gestiones de altas cantidades de llamadas y atenuación de la transmisión de la señal. La Red Digital de Servicios Integrados (ISDN) surge para solucionar estos problemas y el aprovechamiento de las redes existentes para proveer otros servicios como la transmisión de datos a través de las líneas telefónicas.

La ISDN es una red conmutada por circuito que sirve de acceso a las redes conmutadas por paquetes al permitir transmitir datos digitalizados sobre los cables telefónicos de cobre siendo esta característica uno de sus principales beneficios. Al ser diseñado para aprovechar las redes de la PSTN, la ISDN posee compatibilidad con los enlaces T1 y E1 existentes, por lo que puede transmitir información a través de ellos. Esta tecnología utiliza al protocolo Q.931 como protocolo de señalización.

Con esta tecnología son definidas dos tipos de interfaces, estas son denominadas PRI (Primary Rate Interface) y BRI (Basic Rate Interface). Estas interfaces utilizan de acuerdo a las recomendaciones ITU-T I.412 y la recomendación ITU-T I.430, canales de 64 Kbps para la transmisión de datos y canales para la señalización de 16 Kbps siendo la interfaz BRI compuesta por 2 canales de 64 Kbps y un canal para señalización y la interfaz PRI compuesta de 23 canales de 64 Kbps y un canal para señalización. La interfaz PRI también posee similitud con un enlace E1 al poder tener 30 canales de 64 Kbps para la transmisión de datos y dos canales para la señalización, de esta forma teniendo la analogía con los enlaces T1/E1.

1.3 Telefonía IP

Podemos encontrar una descripción general en diversas bibliografías de lo que es la voz sobre el protocolo IP, de [4] se define la telefonía sobre el protocolo IP como "la capacidad de transportar la voz como otra aplicación más en una red de datos". Por otro lado [3] define a la telefonía IP como "comúnmente son servicios de telefonía tradicional sobre internet".

De estas definiciones podemos decir entonces que, la telefonía sobre el protocolo IP es una tecnología que nos permite ofrecer servicios de telefonía tradicional y poder transportar la voz digitalizada a través de redes IP e Internet.

El objetivo principal de la telefonía IP tanto como de telefonía tradicional es poder comunicar a las personas. Con la oportunidad de implementar esta solución como una aplicación más en las redes de datos, esta comunicación se puede realizar de maneras atractivas y flexibles, además de agregar controles adicionales y seguimientos. Siendo la telefonía IP una tecnología innovadora.

Para la implementación de sistemas de telefonía IP, es necesario que se cumplan algunos requerimientos, ya que el transporte de la voz requiere de condiciones mínimas para garantizar una buena calidad de servicio y una adecuada prestación de este servicio. Estos requerimientos son tanto físicos (hardware) como requerimientos en los elementos de software.

Por un lado, como requerimiento físico, se debe contar con el hardware (servidor o *appliance*) que soporte (gestione) la cantidad de llamas simultáneas que pueden ocurrir en un momento determinado, así como el procesamiento relacionado al soporte de las llamadas realizadas y servicios relacionados. Se debe contar adicionalmente con una plataforma de red que brinde condiciones mínimas para la prestación de calidad de servicio como el marcado de paquetes, gestión de tráfico y manejo de redes virtuales, o que brinden anchos de banda que permitan la circulación de este tipo de tráfico sin problemas ocasionados por la competencia entre las demás aplicaciones por este recurso.

Por último, existen plataformas y soluciones de software en el mercado que proveen las características deseadas por los usuarios para configurar sus centrales de telefonía IP, entre ellos podemos mencionar software privativo como los ofrecidos por Cisco, Cisco Call Manager y otras opciones de software libre como Asterisk, Elastix y FreePBX. La elección de una plataforma en particular, debe estar basada en las necesidades particulares de la organización donde se desee implementar la solución de telefonía IP.

Bajo el concepto de redes convergentes, la telefonía IP no requiere la instalación de infraestructuras adicionales a las ya instaladas para una red de datos, aunque pueden ser instalados equipos que mejoren la prestación de este servicio. Esto elimina costos adicionales de soporte y mantenimiento de infraestructuras adicionales. La convergencia de redes no es un concepto nuevo. Empresas como Cisco e innovadores como Mark Spencer y Jim Dixon vislumbraron que las distintas redes existentes, video voz y datos convergerían a una sola. Jim Dixon, el creador del proyecto zapata, el cual se convertiría en el precursor de lo que hoy son las interfaces Digium Asterisk Harware Device Interface (DAHDI) [4], también tuvo esta idea, la cual trato de hacer realidad con la creación de interfaces telefónicas incorporadas a una computadora. Estos dispositivos proveerían una interfaz entre la telefonía tradicional y las redes de datos, permitiendo que la voz sea tratada como una aplicación más en las redes conmutadas por paquetes. Al igual que él, muchas otras personas tuvieron esto presente hasta lograr el desarrollo de más y más tecnologías que funcionaran sobre las redes de datos existentes. La convergencia de redes consiste entonces, en que todas las redes puedan ser incorporadas o absorbidas por las redes de datos. Desde el punto de vista económico y de mantenimiento, resulta costoso mantener distintos sistemas por separado.

Los datos de VoIP a diferencia de datos normales, son datos generados en tiempo real. Para el soporte de datos en tiempo real es necesario emplear protocolos y dispositivos que puedan garantizar parámetros relacionados a la calidad de servicio como el retraso, el *jitter* y la pérdida de paquetes. Dichos protocolos de la pila de protocolos TCP/IP serán discutidos en las próximas secciones del presente documento.

1.3.1 Motivación al uso de VoIP

Son muchos los beneficios que cuentan como motivación para el uso de la tecnología VoIP y servicios de telefonía IP. Entre ellos el más destacado es la disminución de gastos recurrentes en llamadas y uso del sistema de telefonía tradicional. La disminución de costos en llamadas suele ser de bastante interés en la mayoría de las organizaciones que implementan este tipo de tecnología.

La reducción de costos en llamadas viene dada por la utilización de los enlaces de datos con los proveedores de Internet para la realización de las llamadas en vez de utilizar los enlaces con la telefonía tradicional. Estas llamadas no acarrean costos adicionales al transitar por las redes de datos y la conexión a internet es un servicio del cual toda organización debe contar. Esta característica es muy útil cuando una organización cuenta con más de una agencia o sede, ya que al momento de realizar llamadas inter agencia estas no generan costos adicionales al usar la tecnología de VoIP.

Aunque las implementaciones de esta tecnología puedan resultar costosas por los dispositivos que se puedan requerir, también existen soluciones basadas en software que brindan la posibilidad de implementarla a bajos costos. Un ejemplo de solución con software y de bajo costo, es la utilización de softphones y servidores de llamadas como Asterisk o Elastix, aunque este último cuenta con las versiones empresarial y libre. El uso de estos componentes nos permitiría realizar llamadas entre las computadoras contando con solo, por ejemplo, un micrófono y unas cornetas o auriculares, sin la necesidad de contar con un teléfono físico para cumplir con esta funcionalidad.

No solo los beneficios económicos toman particular importancia en el uso de esta tecnología. Entre estos beneficios adicionales podemos mencionar, movilidad de los usuarios, servicios de directorios, integración e interacción con otras aplicaciones, personalización de los servicios, implementación de servicios de atención interactiva por voz, servicios de buzón de voz y otros más. Muchos de los sistemas de VoIP, ofrecen amplia flexibilidad al momento de implementar un servicio particular. En sus inicios, la mayoría de estos, solo se encontraban disponibles en las grandes centrales IP corporativas. Plataformas como Asterisk, permiten incluso el desarrollo de módulos que satisfagan necesidades particulares de una organización.

Por ser una tecnología que posee tantas características, su curva de aprendizaje para realizar una adecuada instalación y aprovechamiento de sus bondades, es alta, por lo que es necesaria una inversión de tiempo considerable.

Las comunidades relacionadas a productos de telefonía IP como Asterisk y Elastix son muy activas y aportan un valor agregado al uso de la telefonía IP, aunque también estas soluciones de software poseen sus versiones comerciales (Elastix), las versiones abiertas poseen bastantes funcionalidades y respaldo de las comunidades activas. Otras empresas líderes como Cisco y Avaya también ofrecen soluciones de telefonía IP para empresas e instituciones.

La telefonía IP es una tecnología reciente pero que ha tenido bastante aceptación en las instituciones y sus implementaciones crecen con el transcurrir del tiempo. Con este crecimiento de la tecnología, en donde ya para el año 2013 en Estados Unidos existían aproximadamente 34 millones de usuarios [49], también crecen las demandas de nuevas características que poco a poco son desarrolladas para brindar mejores servicios relacionados a estas tecnologías.

El desarrollo y adición de nuevos componentes para agregar más funcionalidades es, en sistemas de telefonía IP, totalmente viable al ser la voz tratada como una aplicación más dentro de las redes de datos. En la telefonía tradicional, es complicado alcanzar la flexibilidad que los sistemas de telefonía IP ofrecen debido a las limitantes inherentes al sistema, siendo esta última característica una ventaja a favor de la telefonía IP.

Por último, podemos mencionar que, al hacer uso de las redes de datos, la tecnología de VoIP puede ser implementada en redes de datos existentes sin la necesidad de realizar inversión en instalaciones de infraestructuras adicionales, como era necesario en la telefonía analógica.

1.3.2 Arquitectura VoIP

Cualquier arquitectura de telefonía IP, debe contar con los siguientes elementos, aunque algunos de estos elementos pueden considerarse opcionales.

1.3.2.1 Componentes básicos de una arquitectura VoIP

- **Usuarios finales (Softphones o hardphones):** Los nodos finales o también conocidos como usuarios VoIP, conforman todos aquellos dispositivos, softphones y hardphones, instalados en las redes IP que permitan la comunicación y transmisión de voz usando dichas redes.

- **Servidores de llamadas (Call servers):** Los servidores de llamadas son aquellos contra los cuales, los usuarios VoIP se registran y autentican para obtener acceso al sistema de telefonía IP [5]. Pueden implementar diversas funcionalidades para proveer servicios de valor agregado. Los servidores de llamadas pueden estar conectados a otros servidores de llamadas por medio de enlaces troncales.

- **Troncales:** Una troncal es un enlace físico o lógico en un sistema de comunicación diseñado para manejar muchas transmisiones simultáneamente. Dependiendo del sistema una troncal puede transportar señales analógicas o digitales y conecta usualmente dos o más nodos. [6].

- **Servidores de propósito general:** Existen distintas maneras de implementar una arquitectura de telefonía IP y distintos servidores pueden ser instalados para para dar soporte a esta. Servidores como DHCP (Dynamic Host Configuration Protocol), TFTP (Trivial File Transfer Protocol), NTP (Network Time Protocol), DNS (Domain Name System) contribuyen a este fin.

- **Dispositivos intermedios:** Como toda tecnología de comunicación, está también depende de las redes y los dispositivos que llevan a cabo la tarea de interconexión entre los dispositivos. Dispositivos que llevan a cabo la comunicación en capa 2 del modelo de referencia OSI para la interconexión de dispositivos locales y dispositivos de interconexión de capa 3 del modelo de referencia OSI son imprescindibles en las arquitecturas de telefonía IP. Además, estos dispositivos deben ofrecer condiciones mínimas de calidad de servicio para la buena prestación del servicio de telefonía IP.

- **Protocolos VoIP:** Un protocolo designa el conjunto de reglas que rigen el intercambio de información a través de una red de computadoras [7]. En la telefonía IP también encontramos protocolos y entre los que hacen vida en esta tecnología, los podemos clasificar en dos tipos,

los protocolos de señalización y los protocolos de transporte de los datos multimedia. Entre estas dos clasificaciones se encuentran los siguientes protocolos:

- SIP
- SDP
- IAX
- RTP
- RTCP

Los protocolos de señalización serán los encargados de gestionar la lógica para el establecimiento, mantenimiento y finalización de una llamada, mientras que los protocolos de transporte servirán para transportar los datos generados dentro de una comunicación multimedia. Estos protocolos serán descritos en mayor detalle más adelante en el presente documento.

- **Códec:** De igual importancia que los protocolos, los códec son los encargados de codificar y decodificar (codificator / decodificator) la voz de datos analógicos a datos digitales y viceversa. Los códec también pueden agregar compresión de los datos para que estos no consuman excesivo ancho de banda en las redes de comunicación.

  Existen varios estándares para la codificación de la voz. Entre ellos podemos mencionar los siguientes:

  - G.711
  - G.729
  - GSM

Los códec mencionados serán descritos en mayor detalle más adelante en el presente documento. La representación de estos componentes puede apreciarse en la figura 2.

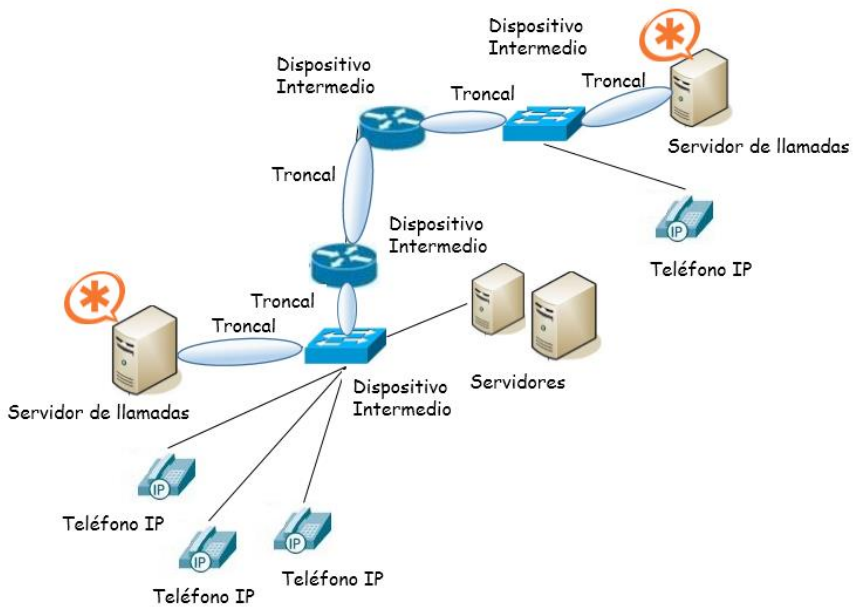

Figura 2: Representación de los elementos presentes en una topología VoIP.
Fuente: Autor.

1.3.3 Funcionamiento de la VoIP

Para entender el funcionamiento de la tecnología VoIP es necesario haber comprendido sus componentes y lo que esta tecnología significa.

El proceso de establecimiento de una llamada en la telefonía IP, se lleva a cabo de manera similar que el establecimiento de una llamada en sistemas de telefonía tradicional. Aunque en esencia, estos procesos de establecimiento de la llamada son muy similares, en la telefonía IP pueden ser añadidas más etapas para poder establecer una conexión con un destino, como por ejemplo autenticación y autorización de usuarios, resolución de nombres, etc.

Al igual que en los sistemas de telefonía tradicional el proceso empieza con un usuario que desea comunicarse con un usuario destino, para ello solo tiene que descolgar el teléfono y marcar la extensión, el número de identificación del usuario destino o un FQDN para iniciar el proceso de comunicación con este usuario. Luego el usuario destino, si existe y está disponible, será notificado que tiene una llamada entrante y al contestar la llamada, la comunicación es establecida hasta la finalización de dicha llamada por parte de alguna de las partes involucradas. De manera general, este proceso es bastante parecido al establecimiento de una llamada en sistemas de telefonía tradicional, pero, aunque sean bastante similares, los aspectos técnicos relacionados a este proceso cambian. En la telefonía IP son realizados procedimientos previos para el establecimiento de una llamada, procedimientos intermedios (durante la llamada) y procedimientos al finalizar la llamada.

Previo a la realización de llamadas, un dispositivo VoIP o usuario VoIP, sea un software de emulación de teléfono o un teléfono VoIP físico, debe obtener información de la red como una dirección IP. Estos usuarios VoIP también pueden ser configurados manualmente, o pueden hacer uso de los servidores mencionados anteriormente para obtener su información. Luego de haber obtenido la información necesaria para comunicarse dentro de la red de datos, este debe hacer uso de protocolos de señalización para registrarse y autenticarse con los servidores de llamadas. Una vez autenticados y registrados en los servidores de llamadas un usuario puede ser localizado por un identificador, este identificador es la extensión. Luego de este proceso, ya el usuario puede realizar llamadas a otros usuarios que se hayan registrado en el servidor de llamadas y es el servidor de llamadas el responsable de generar los tonos de ocupado, de repique y de conexión al sistema.

Durante una llamada VoIP, se realiza un proceso de digitalización de la voz. En este proceso es transformada la señal analógica (voz) a señales digitales. De esta forma los datos pueden ser transmitidos por la red hasta llegar al destino. El proceso de digitalización es llevado a cabo por los codecs. Una vez la señal es recibida en el destino, es responsabilidad del decodificador interpretar esta señal para su posterior conversión a señales analógicas, de esta forma, dicha señal (la voz) pueda ser escuchada e interpretada por el destinatario. La señal recibida no es igual que la generada en un principio ya que en el proceso de conversión analógico/digital se pierden datos de la señal analógica.

Por último, luego de la finalización de una llamada, el servidor de llamadas es informado de esta finalización o cierre de conexión. La finalización de una llamada depende de los protocolos de señalización quienes manejan la lógica para la terminación de una llamada.

El proceso general para el establecimiento de una llamada en telefonía IP puede apreciarse en la figura 3.

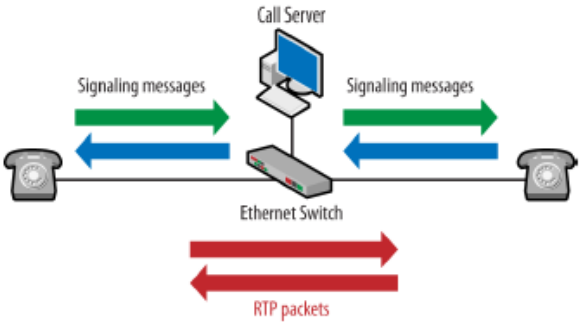

***Figura* 3:** Proceso de establecimiento de una llamada (VoIP)
Fuente: [5]

1.4 Protocolo SIP

El protocolo SIP por sus siglas en inglés Session Initiation Protocol, es un protocolo de señalización, estandarizado por la IETF en su RFC 3261 [8]. Este protocolo es usado para el control de sesiones y trabaja a nivel de capa de aplicación usando los puertos 5060 y 5061 para la versión segura del protocolo (SIPS, SIP Secure) respectivamente.

Este protocolo fue diseñado para la gestión de sesiones, pero para entender más acerca de su funcionamiento, primero debemos entender que es una sesión. Del RFC 3261 extraemos la definición para sesión como "el intercambio de datos entre los entes participantes en la comunicación" [8].

Una sesión puede ser iniciada, mantenida y terminada. Dentro del mantenimiento, una sesión puede ser modificada. SIP, para lograr las funcionalidades para las que fue diseñado hace uso de primitivas y pases de mensajes, estos mensajes poseen una cabecera común, estas cabeceras están compuestas por campos que contienen información acerca del tipo de mensaje, identificación de la sesión, información del remitente y del receptor de los mensajes.

Por la manera que están definidos los mensajes del protocolo y la estructura que estos poseen, este protocolo sugiere el uso de un esquema transaccional basado en solicitudes y respuestas (*request/response*). Este modelo transaccional y el uso de códigos para la identificación de mensajes hacen que se asemeje al protocolo HTML.

Es importante resaltar que para que el protocolo SIP cumpla con sus funciones básicas como lo son la creación mantenimiento y finalización de la sesión, no depende de aplicaciones externas ni de protocolo de transporte en particular, desde este punto de vista, SIP puede emplearse como componente para crear arquitecturas de manejo de sesiones más complejas.

Las características que permiten al protocolo SIP funcionar en conjunto con otros protocolos para dar soporte a arquitecturas multimedia complejas, hacen que este protocolo sea adecuado para formar parte de una arquitectura VoIP. El protocolo SIP trabaja en conjunto con los protocolos SDP (Session Description Protocol), RTP (Real-Time Protocol) y RTCP (Real-Time Control Protocol) para ofrecer una arquitectura de manejo de sesiones multimedia y entre estos datos multimedia se incluyen los datos de voz. De esta forma la creación, modificación y finalización de las sesiones multimedia, caracterización y negociación de parámetros de la sesión multimedia, transporte de los datos multimedia y recepción de datos del estado de la sesión es posible.

De manera general, SIP ofrece las siguientes características:

1.4.1 Características del protocolo SIP

- Administración de la sesión
- Autenticación y Autorización
- Registro de usuarios
- Localización de usuarios
- Disponibilidad de usuarios
- Seguridad

1.4.3 Mensajes de tipo solicitud

Los mensajes de tipo solicitud difieren de los del tipo notificación por su línea de inicio. Están definidos para los mensajes de tipo solicitud un total de seis (6) métodos, los cuales se mencionan a continuación:

- **REGISTER:** Registra a un usuario SIP.
- **INVITE:** Permite invitar un usuario o servicio para participar en una sesión o para modificar parámetros en una sesión ya existente.
- **ACK:** Confirma el establecimiento de una sesión.
- **BYE:** Indica la terminación de una sesión.
- **CANCEL:** Cancela una petición pendiente.
- **OPTIONS:** Solicita información sobre las capacidades de un servidor.

1.4.4 Mensajes de tipo notificación

Las respuestas o notificaciones se diferencian de los requerimientos en su línea inicial. En esta definen solo la versión del protocolo, el código de la notificación y el identificador de la notificación (ejemplo 100 TRYING). Los códigos de respuestas pueden ser clasificados a su vez como sigue:

- **Notificaciones provisionales:** Las notificaciones provisionales están representadas por los códigos de tipo 1xx. Estas notificaciones indican que la solicitud se ha recibido o está siendo procesada.
- **Notificaciones de éxito:** las notificaciones de éxito están representadas por los códigos de tipo 2xx. Estas notificaciones indican que la solicitud ha sido recibida, procesada y aceptada.
- **Notificaciones de redirección:** Las notificaciones de redirección están representadas por los códigos de tipo 3xx. Estas notificaciones indican al cliente SIP que más acciones deben ser realizadas para completar la solicitud.
- **Notificaciones de errores de tipo cliente:** Las notificaciones de error de tipo cliente están representadas por los códigos del tipo 4xx. Estas notificaciones están relacionadas con errores en los requerimientos del cliente SIP y no pueden ser realizadas por los servidores.

- **Notificaciones de error de tipo servidor:** Las notificaciones de error de tipo servidor están representadas por los códigos del tipo 5xx. Estas notificaciones están relacionadas con errores de los servidores SIP al no poder completar un requerimiento válido.

- **Notificaciones de error global:** Las notificaciones de error global están representadas por los códigos del tipo 6xx. Estas notificaciones están relacionadas con errores de solicitudes que no pudieron ser procesadas por ninguno de los servidores SIP.

1.4.5 Campos de cabecera

La cabecera común que poseen ambos tipos de mensajes cuenta con los siguientes campos:

- **Vía:** Este campo contiene la dirección del emisor el cual espera recibir respuesta a sus solicitudes, esto también es hacia donde las respuestas serán enviadas por parte de los receptores de las solicitudes. Además, se le anexa un parámetro "Branch" el cual hace referencia a la transacción, este valor es único.

- **From:** El campo *From* indica la procedencia del mensaje.

- **To:** El campo *TO* muestra hacia quien va dirigido el mensaje (Destinatario).

- **Call-ID:** El campo *Call-ID* contiene un identificador conformado por una cadena de caracteres aleatorios que indica o hace referencia de manera univoca a la sesión. En conjunto con los campos To, From y Call-ID podemos identificar un flujo de sesión entre participantes.

- **Cseq:** Este campo *Cseq (Command Sequence)* posee asociado un numero entero que es incrementado en una unidad cada vez que es generado un nuevo requerimiento o solicitud dentro de la sesión.

- **Contact:** El campo *contact* contiene una dirección SIP o SIP URI. Este campo es útil cuando se desea comunicar directamente con la dirección URI especificada en este campo, o cuando se quieren enviar subsiguientes solicitudes de esta dirección.

- **Max forward:** Este campo contiene un número entero que representa la cantidad máxima de servidores SIP que debe recorrer la solicitud para alcanzar el destino.

- **Content type:** En el RFC 3261, se hace mención a que el protocolo SIP puede servir para transportar un objeto opaco, estos son datos abstractos y son transportados en la sección del cuerpo de un mensaje SIP.

- **Content length:** Este último campo define la longitud en bytes del cuerpo del mensaje SIP. Si no existe cuerpo del mensaje entonces este campo es establecido en cero.

Es importante resaltar que la primera línea de la cabecera SIP contiene un descriptor el cual indica el tipo del mensaje, de esto dependerá la variación del contenido del mensaje. Los parámetros de la cabecera descritos, se encuentran presentes en todos los mensajes SIP (Solicitudes y respuestas).

1.4.6 Componentes SIP

Como en toda arquitectura de protocolos o de servicios, es importante conocer y describir sus componentes para entender cómo funciona o que características ofrece. Los componentes de una arquitectura SIP se describen a continuación:

- **Agentes (User Agent, UA):** Los agentes SIP son todos aquellos dispositivos que puedan ejecutar un servicio SIP para el manejo de sesiones. Estos representan cualquier sistema final. Los agentes pueden ser de dos tipos, agentes de tipo servidor y agentes de tipo usuario, aunque un agente puede cumplir con ambos roles.

- **Agentes usuario cliente (User Agent Client, UAC):** Los agentes de tipo cliente son todos aquellos usuarios que generan solicitudes SIP (request). Estas solicitudes a su vez pueden pasar por uno o más servidores SIP Proxy. Este tipo de agentes son usualmente teléfonos basados en software (Softphone) o teléfonos IP físicos (hardphone).

- **Agente usuario servidor (User Agent Server, UAS):** Un UAS es el encargado de procesar y responder las solicitudes que son creadas por los agentes de tipo cliente.

- **Mensajes SIP:** Los mensajes SIP como se describieron en la sección anterior son el mecanismo con el cual SIP cumple con sus funcionalidades. Para efectos del documento y fines explicativos del funcionamiento básico del protocolo SIP (Establecimiento, mantenimiento y finalización de una sesión) podemos mencionar los siguientes mensajes que son utilizados comúnmente:

    - **INVITE:** Este mensaje SIP es de tipo solicitud. Es generado cuando un usuario desea iniciar una sesión con otro usuario. Este contiene un mensaje de tipo SDP para la caracterización de la sesión multimedia.

    - **TRYING**: Este mensaje es de tipo respuesta. Este mensaje sirve para notificar o dar respuesta al usuario que ha generado una solicitud de tipo INVITE, que dicha solicitud está siendo procesada.

    - **RINGING:** Este mensaje es de tipo respuesta. Cuando un mensaje INVITE llega a su destino, este hace que el UAC receptor genere un mensaje RINGING el cual será enviado a la dirección especificada en el campo FROM del mensaje INVITE recibido. El mensaje RINGING sirve para notificar al llamante que su solicitud ha sido procesada por el destino. Estos mensajes RINGING serán generados por el destino mientras este no conteste la llamada, además, serán enviados de manera periódica y por un lapso de tiempo finito al UAC llamante, generando en este último, tonos de repique.

    - **BYE:** Este mensaje es de tipo respuesta. Dicho mensaje es generado cuando una de las partes involucradas en la sesión no desea seguir comunicándose con las otras partes involucradas; indica cierre de la comunicación.

    - **ACK:** Este mensaje es de tipo respuesta. El mensaje ACK confirma el establecimiento de una sesión ante el arribo de un mensaje 200 OK producido a su vez por una solicitud de tipo INVITE. El mensaje ACK solo sirve para confirmar este tipo de solicitudes y mensajes Re-INVITE

    - **OK:** Este mensaje es de tipo respuesta. El mensaje OK posee un código de mensaje asociado 200. Ante una solicitud de tipo INVITE, un mensaje 200 OK es generado para

confirmar que el destinatario ha aceptado la llamada y a su vez envía embebido un mensaje SDP ofreciendo las características de sesión multimedia que se establecerá. Otro caso donde está presente el mensaje 200 OK es en respuesta de un mensaje BYE.

- **Re-INVITE:** Este mensaje es de tipo solicitud. Los mensajes Re-INVITE son generados cuando algunas de las partes involucradas en una sesión, desea cambiar parámetros en la comunicación o de la sesión multimedia. Uno de estos cambios puede ser redirigir el tráfico multimedia a otro UAC.

- **SIP Proxy:** Un SIP proxy es una entidad intermedia el cual actúa como UAC y UAS. Tiene como principal función el registro de usuarios, realizar la localización de otros UAC por medio de una base de datos para la localización, procesar y redirigir solicitudes basadas en el SIP URI para que un mensaje sea dirigido a su destino. Por poseer la dualidad UAC/UAS, el SIP Proxy también genera solicitudes para resolver a su vez solicitudes entrantes, o en caso de recibir respuestas de error, generar nuevas solicitudes según sea el caso.

- **SIP URI:** Un SIP URI (SIP Uniform Resource Identificator) es una dirección lógica usada por los Agentes SIP para identificar a un usuario o a un servicio.

- **Mensajes SDP:** Estos mensajes forman parte de los mensajes de señalización. Los mensajes SDP (Session Description Protocol) son transportados por el protocolo SIP y tienen como función negociar las características de la sesión multimedia a establecer.

- **Trafico Multimedia:** Una vez establecida una sesión, el protocolo SIP deja en manos de otros protocolos la transferencia de datos multimedia, estos datos multimedia son transportados a través de distintos protocolos diseñados para este fin, tales como RTP o RTSP.

1.4.7 Funcionamiento básico del protocolo SIP

Una vez definidos los componentes básicos de una arquitectura SIP, podemos describir un funcionamiento básico. Para explicar el funcionamiento del protocolo SIP, se usará la topología definida en el ejemplo del RFC 3261. Con este ejemplo se podrá observar el establecimiento, duración y finalización de una sesión multimedia (llamada) entre dos usuarios que desean comunicarse utilizando voz sobre IP (VoIP).

Sea la siguiente topología de ejemplo conocida como el *trapezoide* (figura 4), el usuario A desea establecer comunicación con el usuario B. Supongamos que los Usuarios A y B se encuentran en dominios distintos siendo tarma.com.ve el dominio para el usuario A y el dominio ucv.ve para el usuario B.

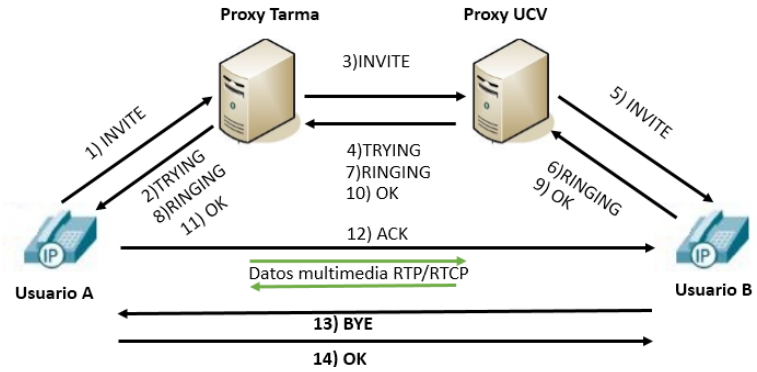

***Figura** 4:* Topología tipo trapezoide.
Fuente: Autor.

Cada dominio posee un servidor SIP proxy encargado de los registros de los usuarios de cada dominio respectivamente. Estos como se explicó anteriormente, son los encargados de procesar las solicitudes para la localización de usuarios y poder establecer comunicación con el o los destinos solicitados.

Cuando el usuario A desea comunicarse con el usuario B, este necesita la dirección SIP o el SIP URI del usuario B. Luego de obtener la dirección SIP del usuario B, el usuario A procede a enviar un mensaje SIP de tipo INVITE, con este mensaje el usuario A busca establecer comunicación con el Usuario B. Dentro de un mensaje INVITE se encuentra embebido un mensaje SDP para negociar parámetros de la sesión multimedia, estos pueden ser aceptados o rechazados por el destinatario. El protocolo SIP hace uso del protocolo SDP para la parametrización de la sesión multimedia ya que por sí solo, "SIP no maneja parámetros para la especificación de la media, códec o tasas de muestreo" [8]

El mensaje INVITE es enviado al SIP Proxy del dominio de A. En respuesta a que el SIP proxy de tarma.com está procesando la solicitud, este envía al usuario A un mensaje TRYING, el cual indica que se está procesando su requerimiento. El SIP proxy del dominio tarma.com.ve, por medio de reglas de enrutamiento o troncales definidas con el servidor SIP Proxy de ucv.ve verifica el SIP URI y reenvía el mensaje INVITE al correspondiente servidor SIP. Esta búsqueda para saber hacia dónde reenviar solicitudes o notificaciones son basadas en el dominio y en el usuario especificado en el SIP URI. Antes de enviar el mensaje INVITE, la cabecera es modificada añadiendo en el campo VIA una entrada adicional, de esta forma cuando sean retornadas las respuestas hacia el servidor este sabrá a quien entregarlas.

El protocolo SIP utiliza los campos de FROM, TO, Call-ID, Cseq y parámetro branch del campo VIA para identificar la correspondencia entre las respuestas y las solicitudes generadas e identificar comunicaciones de manera única.

Luego el INVITE enviado del SIP Proxy tarma.com, es procesado por el SIP Proxy de ucv.ve. Este último, responde al SIP Proxy de Tarma con un mensaje TRYING.

El SIP Proxy UCV busca el SIP URI en su base de datos de localización de usuarios y encuentra coincidencia, de esta forma el SIP Proxy de ucv.ve envía finalmente el INVITE al destino.

La primera fase del intercambio de mensajes puede apreciarse en la figura 5.

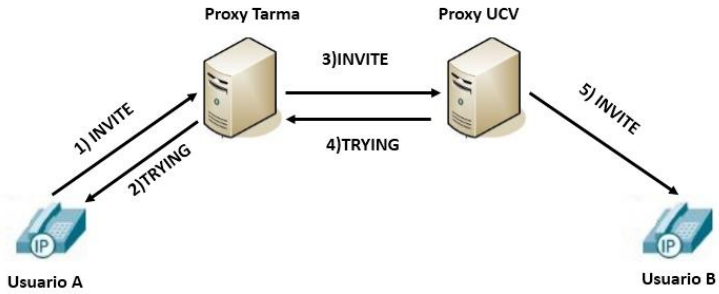

Figura 5***:*** Primera fase del intercambio de mensajes SIP.
Fuente: Autor.

Al arribo del INVITE en el usuario B, causa que se notifique que el usuario A está intentando establecer una conexión con este usuario. El usuario B envía un mensaje 180 RINGING de vuelta en respuesta al mensaje INVITE recibido indicando que se está esperando para que la llamada sea atendida, estos mensajes RINGING de acuerdo al manejo y a las configuraciones de los servidores y dispositivos hacen que se generen tonos de repique. El mensaje RINGING es enviado por el mismo camino utilizado por donde el mensaje INVITE ha transitado, ya que utiliza las entradas en el campo VIA para enviar el mensaje de vuelta. El pase de mensajes queda entonces de la forma establecida en la figura 6.

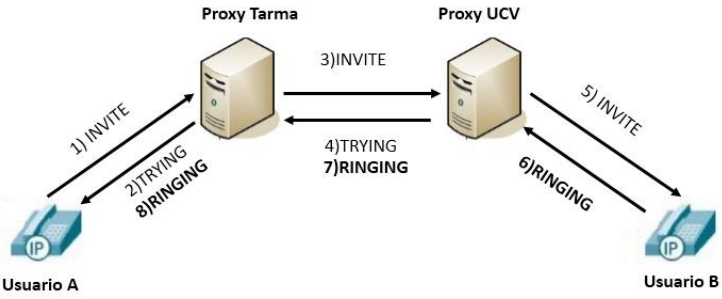

Figura 6***:*** Envío mensaje TRYING.
Fuente: Autor.

Si el usuario B decide aceptar la llamada, se generará un mensaje en respuesta al mensaje INVITE con código 200 OK. Este mensaje a su vez también contendrá un mensaje SDP embebido en el cual, el usuario B define los parámetros de la sesión multimedia que soporta. Este mensaje al igual que el mensaje RINGING es enviado de vuelta usando las direcciones que se encuentran en el campo VIA de la cabecera de mensajes SIP.

Al llegar el mensaje 200 OK al usuario A, se detienen los mensajes RINGING y se indica que la llamada ha sido aceptada por el destinatario. Ver figura 7.

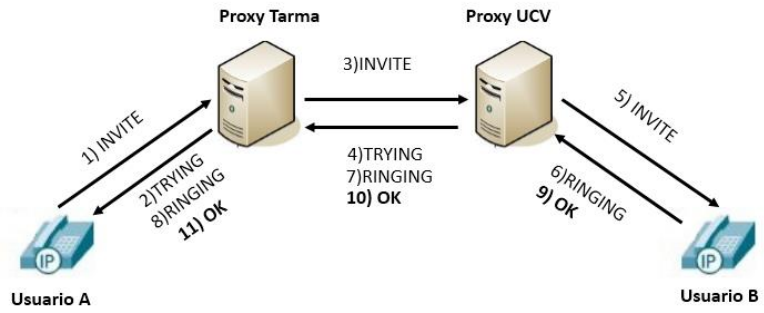

***Figura** 7:* Aceptando la llamada, mensaje 200 OK.
Fuente: Autor.

En confirmación de llegada del mensaje 200 OK, el usuario A responde con un ACK donde además de confirmar el establecimiento de la sesión se confirman los parámetros a usar en la transmisión multimedia (mensaje embebido SDP). A diferencia del mensaje INVITE, el mensaje ACK será transmitido directamente al usuario B sin pasar por los SIP Proxy ya que ambos usuarios conocen la dirección del otro, esta información es encontrada en el campo *Contact* de la cabecera SIP. En este punto los SIP Proxy no son requeridos por lo que estos liberan el flujo de la llamada a los usuarios. Existen configuraciones donde esto último no sucede y la llamada transita igualmente por los SIP proxy, esto puede ser debido a que las llamadas quieran ser grabadas o exista algún tipo de tarificación.

Una vez establecida la sesión, ambos transmitirán los datos multimedia de acuerdo a como lo hayan negociado en los mensajes SDP. Además, estos datos serán transportados por otros protocolos como RTP. Se puede apreciar el estado de la transmisión y pase de mensajes en la figura 8.

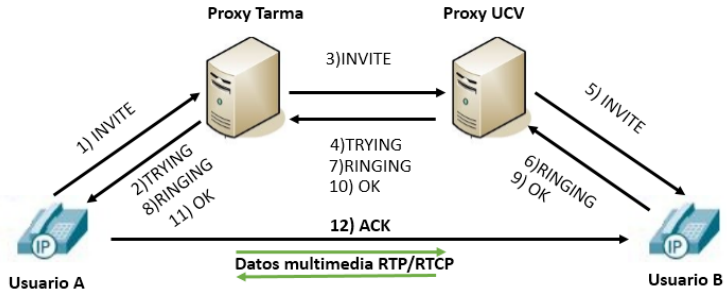

***Figura** 8:* Intercambio completo de los mensajes SIP.
Fuente: Autor.

Como se mencionó anteriormente, una sesión luego de ser establecida, puede ser modificada. SIP logra la modificación de una sesión multimedia basándose en mensajes re-INVITE, de esta forma no es necesario finalizar una sesión y volver a establecerla con los nuevos parámetros. Los mensajes re-INVITE contienen mensajes SDP los cuales, en cualquiera de los casos, aceptados o rechazados los nuevos parámetros, se debe confirmar con un ACK respectivamente, pero esto, no hace que la conexión finalice o se interrumpa.

Luego que los participantes han transmitido los datos multimedia que desean transmitir, para nuestro caso, uno de los usuarios termina la llamada y cuelga el teléfono SIP, se genera un mensaje de tipo BYE. El intercambio final puede verse en la figura 9.

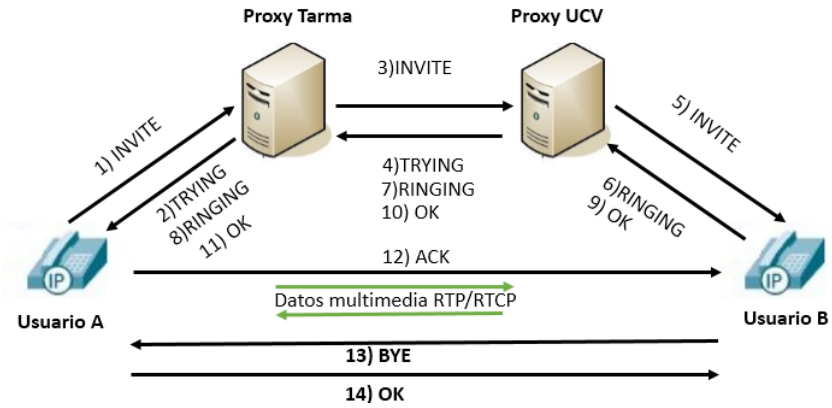

***Figura** 9:* Intercambio completo de mensajes SIP entre los participantes A y B.
Fuente: Autor.

1.5 Protocolo SDP (Session Description Protocol)

El protocolo SDP por sus siglas en inglés Session Description Protocol, es un protocolo estandarizado por la IETF en su RFC 4566 [9]. Tiene como finalidad proveer una estructura estándar para describir y anunciar los parámetros de una sesión multimedia. El protocolo SDP trabaja en conjunto con otros protocolos de señalización como SIP y de transporte como RTP. Los protocolos que hacen uso del estándar SDP, utilizan mensajes que siguen la estructura propuesta por SDP, como consecuencia, las partes involucradas en una sesión multimedia pueden ponerse de acuerdo en la caracterización de la sesión multimedia a establecer.

Un mensaje SDP incluye información relacionada con la temporización de la sesión, características de la sesión y el transporte de datos multimedia.

La información que contienen los mensajes SDP referentes a los datos multimedia y el transporte, especifican: el tipo de datos multimedia a transmitir, el identificador de la sesión, el propósito de la sesión, quien origina la sesión, que protocolo se usará para transportar la media (RTP, UDP, IP, etc), manejo de direcciones y puertos para la transmisión de la media y el formato o codificación de la media. Por otro lado, la información referente a la temporización incluida en los mensajes SDP muestra tiempos de duración, creación y finalización de la sesión.

Los mensajes SDP contienen los parámetros necesarios para describir la sesión multimedia. Estos parámetros pueden ser agrupados en cinco secciones:

- **Versión:** Este parámetro establece la versión del protocolo SDP que está siendo usada, actualmente la definida en el RFC 4566 (versión 0).

- **Nombre de la sesión:** Este parámetro contiene un valor descriptivo para la conexión. Definido con caracteres que siguen la norma ISO 10646 (Unicode).

- **Propietario de la sesión:** En esta sección se definen los parámetros relacionados con el creador o el propietario de la sesión. Para generar estos parámetros la información es tomada del protocolo que transporte el mensaje SDP.

- **Información de la conexión:** En esta sección tenemos parámetros que especifican el tipo de conexión de red (Internet u otras por definir según el RFC 4566 página 11), tipo de la dirección (IPv4 o IPv6) y la dirección de la conexión (Una dirección IP).

- **Descripción del tiempo:** Este tipo de parámetros definen los tiempos desde que la sesión ha sido iniciada o ha sido detenida. Los valores para parámetros de esta sección están expresados en segundos.

- **Descripción de la media:** En esta sección, los parámetros definen y describen de manera exacta los datos multimedia. Estos especifican el tipo de datos multimedia, puertos y códecs a utilizar.

1.6 Protocolos de transporte

Una vez estudiados los protocolos que intervienen a nivel de señalización para registrar usuarios, establecer, modificar y terminar sesiones, podemos hablar ahora de protocolos de transporte para datos multimedia en tiempo real.

Algunos protocolos de señalización como por ejemplo el protocolo SIP, no poseen la capacidad de transportar los datos multimedia, ya que sus funcionalidades se encuentran limitadas solo para el establecimiento de sesiones. Estos protocolos de señalización requieren de protocolos de transporte como RTP (Real Time Protocol) para conformar una arquitectura completa y así dar soporte a las comunicaciones multimedia. A continuación, se describirá el protocolo RTP (Real Time Protocol) y los mecanismos de control que utiliza para supervisar el estado de una llamada con el protocolo RTCP (Real Time Control Protocol).

1.6.1 Protocolo RTP (Real-Time Protocol)

El protocolo RTP por sus siglas en inglés (Real Time Protocol), es un protocolo de transporte de datos en tiempo real y datos multimedia. Este protocolo fue especificado originalmente en el RFC 1889 pero este RFC quedo obsoleto por el RFC 3550 [10] donde se actualizan los tipos de algoritmos a usar en los cálculos de tiempo y define perfiles de este protocolo para datos de audio y video. Los datos en tiempo real que transporta el protocolo pueden ser: audio, video, datos de simulaciones y datos de cualquier otra aplicación que requiera de este protocolo como transporte. Este protocolo no está asociado a algún puerto en particular. Generalmente RTP hace uso del protocolo UDP (User Datagram Protocol) como protocolo de transporte y el puerto es seleccionado aleatoriamente. Además, RTP hace uso del protocolo RTCP (Real-Time Control Protocol) para la supervisión de la calidad del servicio en el transcurso de la transmisión de datos y para para comunicar información del protocolo a los participantes. El puerto usado por el protocolo de control RTCP es el puerto impar siguiente al puerto RTP elegido.

1.6.2 Protocolo RTCP (Real-Time Control Protocol)

El protocolo RTCP sirve como complemento del protocolo RTP para el manejo de características relacionadas con la calidad de servicio del flujo RTP. Este protocolo al igual que RTP es definido en el RFC 3550 [10]. El protocolo RTCP consiste en la transmisión periódica de paquetes de control, estos contienen estadísticas de paquetes y tiempos de transmisión. Los paquetes RTCP poseen similitud con los paquetes de datos del protocolo RTP.

Los paquetes de control del protocolo RTCP son entregados en un puerto distinto al usado por RTP para la recepción y entrega de datos. Al igual que RTP, el protocolo RTCP no posee un puerto definido, este se basa en el número de puerto seleccionado por el protocolo RTP siendo el puerto para RTCP el inmediato superior al asignado para la recepción y transmisión de datos.

1.7 Proceso de conversión analógico-digital

El proceso de conversión analógico-digital es el proceso por el cual una señal analógica se transforma a una señal discreta para su transmisión y procesamiento en medios digitales. Así como existe el proceso de conversión de señal analógica a digital, también encontramos en los decodificadores el proceso de transformación de una señal digital a analógica. El proceso de conversión analógico-digital puede apreciarse en la figura 10. El proceso de conversión digital analógica sigue un sentido inverso al mostrado en la imagen 7.1.

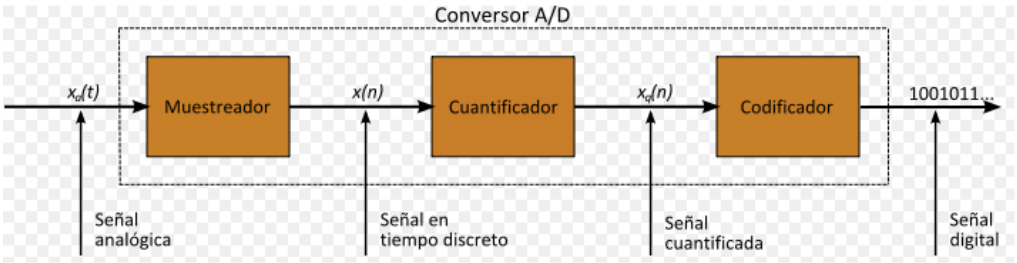

Figura 10*:* Conversión analógico digital.
Fuente: [43]**.**

En ambos procesos se toman en cuenta los siguientes tres parámetros:

- Muestreo
- Cuantificación y
- Codificación

1.7.1 Muestreo

El proceso de muestreo es el proceso en el cual, a intervalos regulares de tiempo (1 segundo), la señal analógica es medida, siendo estos valores de medición, las muestras. Dichos valores o muestras están relacionados con la amplitud de la señal.

Como resultado de este proceso, obtenemos una representación de la señal analógica pero representada en tiempos discretos. Este resultado será el valor de entrada para el proceso de cuantificación.

Un códec, puede utilizar más o menos valores de muestra de la señal analógica para su conversión, siendo este parámetro conocido como el *sampling rate* o la frecuencia de muestreo.

La cantidad de muestras necesarias representar una señal analógica a digital, es calculada usando el teorema de muestreo por Nyquist y Shannon [44]. Este teorema expresa que, si la cantidad de muestras es mayor que el doble de la frecuencia máxima a ser muestreada, esta será completamente reversible partiendo de su representación discreta.

1.7.2 Cuantificación

Luego del proceso de muestreo, es necesario representar las muestras obtenidas en valores discretos para su procesamiento digital. Este proceso es conocido como cuantificación. Cada códec define una cantidad finita para estos valores obtenidos.

En este proceso también se aproximan valores, debido a que las muestras pueden caer en intervalos donde no hay valores definidos, estas se aproximan a valores cercanos, produciendo así una pérdida en la señal.

Para llevar a cabo este proceso de asignación de valores finitos a las muestras son definidos cuatro tipos de cuantificación: La cuantificación uniforme, la cuantificación no Uniforme, la cuantificación logarítmica y la cuantificación vectorial. Luego de la asignación de los valores discretos a las muestras tomadas en el proceso de muestreo sigue el proceso de codificación.

1.7.3 Codificación

Esta es la última fase del proceso de digitalización de la señal analógica. Este proceso recibe como entrada los valores discretos obtenidos en la fase de cuantificación. Estos valores de entrada serán codificados a dígitos binarios los cuales representan la señal digital que será transmitida. Cada valor de entrada debe tener su correspondencia en valor binario, esto asegura que el proceso inverso de reconstrucción de la señal analógica a partir de una señal digital pueda llevarse a cabo. Este proceso de traducción entre las partes (transmisor y receptor) es realizada por los codificadores y decodificadores, o mejor conocidos como los códec.

Para que la señal decodificada sea idéntica, la frecuencia de muestreo debe tender al infinito y esto no es soportado por los sistemas digitales. El proceso de conversión y reconversión analógico, tiene como finalidad reproducir la señal analógica y que esta sea entendida por los humanos.

1.8 Códec

Un códec es cualquier tecnología (hardware o software) que nos permita convertir una señal analógica (En nuestro caso la voz) en su versión digitalizada [11] para su transmisión en medios digitales.

Los códec están presentes en cualquier dispositivo que se comunique en una red de telefonía sobre el protocolo IP.

1.8.1 Características

Los aspectos o características de un códec son los siguientes:

- Consumo de ancho de banda
- Compresión
- Calidad del transporte del sonido
- Costo de procesamiento del códec
- Tasas de muestreo

Cada una de estas características puede poseer valores cuantitativos o cualitativos y pueden ser tomadas a consideración al momento de evaluar la implementación, el uso o el desempeño de un códec sobre otro.

Existen una gran cantidad de códec definidos y estandarizados, algunos de ellos son propietarios y para su uso es necesario la realización de un pago, más sin embargo también existen códec de alta calidad y no propietarios. Una lista con algunos de los códec más utilizados puede apreciarse a continuación en la tabla 1.

Tabla 1

Lista de códec

Nombre	Estandarizado	Descripción	Bit rate (kb/s)	Frecuencia de muestreo (kHz)	Tamaño de la trama(ms)	Observaciones
G.711	ITU-T	Modulación por pulsos codificados (PCM)	64	8	Muestreada	Tiene dos versiones u-law (US, Japan) y a-law (Europa) para muestrear la señal
G.711.1	ITU-T	Modulación por pulsos codificados (PCM)	80-96Kbps	8	Muestreada	Mejora del codec G.711 para abarcar la banda de 50 Hz a 7 Khz. Mas info

G.721	ITU-T	Modulación diferencial adaptativa por pulsos codificados (ADPCM)	32	8	Muestreada	Obsoleta. Se ha transformado en la G.726.
G.722	ITU-T	Codificación de audio en 64 kbit/s	64	16	Muestreada	Divide los 16 Khz en dos bandas cada una usando ADPCM
G.723	ITU-T	Extensión de la norma G.721 a 24 y 40 kbit/s para aplicaciones en circuitos digitales.	24/40	8	Muestreada	Obsoleta por G.726. Es totalmente diferente de G.723.1.
G.726	ITU-T	Modulación diferencial adaptativa por pulsos codificados (ADPCM)	16/24/32/40	8	Muestreada	ADPCM; reemplaza a G.721 y G.723.
G.728	ITU-T	Codificación del audio a 16 kbit/s usando code excited linear prediction (CELP)	16	8	2.5	CELP.
G.729 **	ITU-T	Codificación del audio a 8 kbit/s usando conjugate-structure algebraic-code-excited	8	8	10	Bajo retardo (15 ms)

		linear-prediction (CS-ACELP)				
G.729.1	ITU-T	Codificación del audio a 8 kbit/s usando conjugate-structure algebraic-code-excited linear-prediction (CS-ACELP)	8/12/14/16/18/20/22/24/26/28/30/32	8	10	Ancho de banda desde 50Hz a 7 Khz Mas info
GSM 06.10	ETSI	Regular-Pulse Excitation Long-Term Predictor (RPE-LTP)	13	8	22.5	Usado por la tecnología celular GSM
Speex			8, 16, 32	2.15-24.6 (NB)	30 (NB)	
				4-44.2 (WB)	34 (WB)	
iLBC			8	13.3	30	
DoD CELP	Departamento de Defensa (DoD) Gobierno de USA		4.8		30	
EVRC	3GPP2	Enhanced Variable Rate CODEC	9.6/4.8/1.2	8	20	Se usa en redes CDMA
SILK	Skype	Muestras de audio no comprimido	De 6 a 40 kbit/s	Variable	20	El codec Harmony está basado en SILK

Fuente: [45**].**

Es importante resaltar que la tabla 1 contiene solamente códec para la transformación de la voz. Existen muchos otros códec para el tratamiento de imágenes y video, lo cuales no serán mencionados por salirse del alcance de este documento.

Del total de códec mostrados en la tabla 1, los códec más utilizados en sistemas de telefonía IP podemos mencionar los siguientes tres estándares:

- G.711
- G.729
- GSM

1.8.2 Estándar G.711

Esta recomendación, estandarizada por el ITU-T, codifica la señal analógica a digital basándose en el proceso PCM. Este códec posee una tasa de transferencia de 64Kbps ya que posee un muestreo de 8000 muestras por segundo. Por otro lado, utiliza 8 bits para la cuantificación de los valores obtenidos en el muestreo, dando así el resultado de 64Kbps al multiplicar la cantidad de muestras por segundo y 8 bits asociados a cada muestra. Este estándar es conocido también como PCM (*Pulse Code Modulation*), siguiendo el proceso descrito en la sección de conversión analógica a digital. Este estándar es ampliamente usado tanto para implementaciones de sistemas de telefonía IP como tradicional.

El estándar G.711 posee dos variaciones, u-law y a-law. Aunque estas dos versiones del estándar poseen grandes similitudes, la diferencia más notoria son las áreas geográficas para la implementación de dicho códec. Mientras que el estándar U-law es implementado en Norte América y Japón, el estándar A-law es implementado Europa y el resto del mundo [46]. Este estándar ofrece bastante calidad en la señal de voz, sin embargo, posee un consumo de ancho de banda elevado.

1.8.3 Estándar G.729

El estándar G.729 ofrece una calidad muy similar a la del estándar G.711 pero utilizando un ancho de banda inferior al usado por el estándar g.711. El estándar g.729 utiliza compresión del audio logrando una tasa de 8kbps, haciendo mejor uso de llamadas por enlace de datos. Este códec utiliza *code-excited linear prediction speech coding* (CS-ACELP) Este códec es implementado en algunas plataformas propietarias como Cisco. Una diferencia en desventaja con otros estándares es que el estándar G.729 requiere de licencia para su uso (Sipro Lab Telecom).

1.8.4 Estándar GSM

Este estándar es el utilizado por dispositivos móviles. Sus siglas provienen del acrónimo en inglés *Global System for Mobile Communications* y es un protocolo utilizado en redes celulares y utilizado por teléfonos móviles. El uso de esta tecnología viene dado en *half rate* y *full rate*. Por estar orientado su uso a la telefonía móvil, los códec que esta tecnología utiliza contemplan el ancho de banda y complejidad ya que estos factores pueden repercutir de manera negativa en la duración funcional de los dispositivos móviles (consumo de batería).

Es buena práctica la utilización de este códec cuando se requiere establecer comunicación con sistemas de telefonía móvil desde sistemas de telefonía IP.

GSM de acuerdo al modo de implementación (half o full) varía sus tasas de transmisión, 6.5kbps para *half rate* y 13kbps para *full rate* y son basados en *linear predictive coding* (LPC).

1.9 Calidad de servicio en redes de telefonía IP

Para hablar de calidad de servicio en redes de telefonía IP, es necesario definir primeramente el concepto de calidad de servicio. La calidad de servicio no es más que un conjunto de políticas que permiten hacer un manejo eficiente del tráfico dentro de una red de datos, también puede decirse que la calidad de servicio es la capacidad de dar un buen servicio, y en el caso de las redes de computadoras, la capacidad de la red de proveer estos servicios a un tráfico en particular. Este tráfico puede ser clasificado de acuerdo a prioridades o necesidades particulares del sistema que genera o recibe el tráfico. Una vez clasificado, el tráfico es dividido en distintos flujos. Cada uno de ellos se puede tratar de distintas maneras y es aquí donde radica la importancia de la calidad de servicio en redes de transmisión de datos, ya que dentro de estas clasificaciones o estos flujos de datos podemos identificar y seleccionar flujos de datos que requieran ser tratados distinto de acuerdo a características especiales.

1.9.1 Factores que afectan la calidad de servicio en la VoIP

Al igual que todas las comunicaciones en redes de computadoras, existen elementos que pueden afectar la percepción de la calidad de la comunicación, más aún cuando hablamos de tráfico con requerimientos especiales como lo es el tráfico en tiempo real.

El tráfico en tiempo real es caracterizado por tener bajos márgenes de tolerancia en lo que respecta a los parámetros como perdida de paquetes, retardo y jitter. El tráfico en tiempo real generalmente es conformado por tráfico multimedia, este tráfico generalmente está conformado por datos de video y datos de audio. Para aspectos relacionados con el presente documento se contemplarán los aspectos relacionados con la transmisión de datos multimedia y los parámetros de calidad de servicio que afectan a este tráfico, en particular la transmisión de datos de voz (VoIP).

Algunos de estos factores que afectan la calidad en la comunicación de datos multimedia, son los siguientes:

- Jitter
- Latencia
- Eco
- Pérdida de paquetes
- Ancho de banda

Estos son descritos a continuación:

1.9.2 Jitter

El jitter se define como la variación del tiempo inter arribo de los paquetes a un destino. Este puede ser causado por "congestión de red, perdida de sincronización o por las diferentes rutas seguidas por los paquetes para llegar al destino" [12]. Como consecuencia del jitter, las comunicaciones

multimedia como la voz, pueden verse seriamente afectadas, ya que pueden recibir los paquetes de manera muy rápida o que lleguen con un retraso tal que no puedan ser procesados a tiempo.

Entre las soluciones para combatir este efecto, está la utilización de buffers en los cuales se trata de estabilizar la llegada de los paquetes, implementar políticas de calidad de servicio y gestión de colas para la disminución del jitter y latencia, e incrementar el ancho el ancho de banda en la red.

Como ya se hizo mención, los valores aceptables para que la calidad de servicio en el tráfico de voz no se vea fuertemente afectado de acuerdo a Cisco son valores inferiores a 100ms (milisegundos) para comunicaciones "end-to-end" [13].

1.9.3 Latencia

La latencia o retardo, es la suma total de los tiempos que tarda un paquete en llegar del origen a su destino. Los tiempos que contribuyen al retardo de la llegada de un paquete a su destino son los siguientes: Retardo de propagación, retardo en procesamiento, retardo por encolamiento y el retardo por transmisión. La latencia repercute en la VoIP cuando en una llamada se presenta intermitencias o falta de continuidad del audio.

Los valores tolerables de acuerdo a las bibliografías consultadas [5] son tiempos inferiores a 150 ms (milisegundos).

No existe única solución para resolver los problemas relacionados con la latencia.

1.9.4 Eco

De acuerdo a la bibliografía podemos extraer las siguientes causas y su definición, "El eco se produce por un fenómeno técnico que es la conversión de 2 a 4 hilos de los sistemas telefónicos o por un retorno de la señal que se escucha por los altavoces y se cuela de nuevo por el micrófono. El eco también se suele conocer como reverberación." [14] y se define como "una reflexión retardada de la señal acústica original" [12]

El valor tolerable para el eco de acuerdo a [12] es de 65 ms y con una atenuación de 25 a 30 dB, ya que la llegada del retorno o eco con potencias mayores interfiere en la comunicación entre los participantes y vuelve la llamada bastante molesta.

Para solucionar los problemas relacionados con el eco, pueden hacerse uso de canceladores de eco o supresores de eco. Los supresores de eco convierten una comunicación Full-Duplex en una Half-Duplex alternadamente. De esta forma cuando un segmento de la comunicación este enviando información, el otro no podrá transmitir datos. Con el uso de canceladores de eco, el sistema inspecciona y detecta si en el retorno se encuentran datos repetidos, pero presentando características de atenuación. El problema de los canceladores viene relacionado con el tiempo de procesamiento de la señal pudiendo agregar más retardo en la comunicación.

1.9.5 Pérdida de paquetes

Debido a que las redes por conmutación por paquetes utilizan el modelo de envío de "mejor esfuerzo", y el tráfico multimedia es generalmente transportado sobre el protocolo UDP, los paquetes al perderse no son retransmitidos. Esta propiedad en comunicaciones en tiempo real no presenta problemas, a menos que se sobrepase la cota de tolerancia para la perdida de paquetes para tener una buena comunicación. En el caso de la VoIP siempre podemos pedir que nos repitan lo último que se dijo, por el contrario, esto no es permitido en las transferencias de datos normales como los datos de un archivo o de un archivo ejecutable. Los paquetes pueden perderse por diversas razones como una mala redirección de los mismos, descartado en elementos de interconexión saturados o en enlaces con poco ancho de banda.

El margen de tolerancia [13] debe ser menor al 1% de los paquetes enviados para que no sea afectada la calidad de la llamada. Estos valores pueden depender del códec utilizado.

De igual forma las posibles soluciones a este elemento pueden ser la implementación de dispositivos con anchos de banda reservados para el tráfico de voz o tráfico multimedia, otra solución es no enviar "silencio" en la comunicación.

1.9.6 Ancho de banda

El ancho de banda, como para cualquier otra transmisión de datos, es de gran importancia para el desempeño de una comunicación. El ancho de banda de una red no solo contiene datos de la llamada o datos en tiempo real, podemos encontrar variedades de tráfico como tráfico FTP, HTTP, DNS, DHCP, entre otros. Se debe contar con un ancho de banda mínimo que nos garantice que la comunicación se lleve a cabo de manera satisfactoria.

Para los cálculos del ancho de banda necesarios, debemos tener a consideración factores como el códec a usar y el promedio de llamadas simultaneas en un momento dado. Esto nos ayudara en caso que debamos hacer reserva de recursos para la implementación de soluciones de telefonía IP. Se puede apreciar el consumo de ancho de banda por códec en la tabla anterior (tabla 1).

Los efectos negativos de los factores descritos, deben ser minimizados, esto nos garantizaría un excelente despliegue e implementación de nuestros sistemas de telefonía IP. De acuerdo al ITU-T (Estándares G.729 y G.114) y a empresas como Cisco y Avaya [13], se definen márgenes de tolerancia máxima para los parámetros de retardo, jitter y pérdida de paquete. Estos valores se encuentran en la tabla 2.

Tabla 2

Valores máximos tolerados establecidos por Cisco, Avaya y la ITU-T.

	Cisco	Avaya	ITU-T
Latencia máxima	150ms	80-180ms	150ms
Máximo de paquetes perdidos	Menor al 1%	1%-3%	Menor al 1%

Jitter máximo	Menor a 30ms	Menor a 20ms	Varía de acuerdo a la implementación

Fuente: [13**]**

Que se garanticen estos parámetros es de vital importancia para la implementación de sistemas de telefonía IP. De no mantenerse la calidad de servicio en niveles aceptables, la implementación de esta tecnología no resulta atractiva, a pesar de la reducción de costos que esta pueda brindar. Es por esto que se deben contar con equipos de interconexión o equipos dentro de la red que sean capaces de garantizar la clasificación, el marcado y el despacho de los paquetes de VoIP, de acuerdo a los aspectos básicos para la calidad de servicio.

1.10 Planes de numeración

Cuando se desea implementar un sistema de telefonía tanto tradicional como IP, un aspecto importante a considerar es el uso de esquemas para el direccionamiento y ubicación de los usuarios. De esta forma surge la necesidad de utilizar en la telefonía IP un plan de numeración al igual que el utilizado por la telefonía tradicional.

El concepto del uso de planes de numeración en la telefonía IP parte del principio de utilizar los elementos que ya le son familiares a las personas [16] siendo la analogía para este caso, los planes de numeración de la telefonía tradicional, esto es realizado con la intención de lograr aceptación en la implementación y la facilidad en el aprendizaje y comprensión de estos sistemas. En la telefonía IP con las funcionalidades que proveen servidores de llamadas como Asterisk, el uso de identificadores para los teléfonos no está limitado simplemente a un número telefónico o a una extensión, también podemos incluir el concepto de extensión como un conjunto de caracteres, de esta forma podemos llamar a un teléfono haciendo uso de un usuario en vez de un número telefónico. Aun así, el uso de números como identificativos se sigue manteniendo y posee vital importancia al momento de implementar los sistemas de telefonía IP.

Para sistemas de telefonía IP, la elección de un plan de numeración es totalmente arbitraria. Generalmente se utilizan extensiones o números de 3 a 4 dígitos para identificar a los usuarios [4], pero pueden ser de longitudes arbitrarias y en relación a la cantidad de usuarios.

La elección y asignación arbitraria de números, aunque es una posibilidad, no es recomendable, ya que entre las características de un plan de numeración se encuentra la jerarquización del espacio numérico, siendo un ejemplo para este escenario, la telefonía tradicional.

En Venezuela el plan de numeración para la telefonía tradicional es el plan de numeración nacional asignado por la Comisión Nacional de Telecomunicaciones (CONATEL) de acuerdo a lo establecido en los artículos 111, 112, 113 y 114 en la Ley Orgánica de Telecomunicaciones de la República Bolivariana de Venezuela.

De [17] podemos apreciar el plan de numeración nacional actual para las empresas de telefonía en Venezuela. A nivel internacional, la institución encargada de la gestión del espacio numérico para telefonía es la IANA. En el documento E.164 [18] se encuentra el plan de numeración internacional, siendo asignado el prefijo +58 para Venezuela; por otro lado, en Estados Unidos es la NANPA (North America Numbering Plan Administration) quien a nivel local del país maneja su plan de numeración.

La longitud de un plan de numeración varía, pero es recomendable que un número telefónico este comprendido entre un rango de hasta 9 dígitos, ya que, de acuerdo con estudios relacionados con procesos cognitivos, las personas pueden memorizar aproximadamente entre 5 a 9 dígitos o caracteres, esto se conoce como el número mágico [19].

Un plan de numeración debe poseer también patrones que nos ayuden a identificar conjuntos de usuarios, estos patrones pueden ser implementados en servidores de llamadas para la correcta redirección de las mismas hacia su destino, para detectar llamadas externas e internas y llamadas a enlaces troncales. De igual forma un número, así como sirve para identificar a cada usuario, también identifica una llamada con los números de la fuente y del destino.

1.10.1 Características de un plan de numeración

Un plan de numeración debería poseer las siguientes características:

- Fácil memorización
- Estabilidad y pocas modificaciones en función del tiempo
- Identificación de los usuarios
- Flexibilidad para la expansión y contingencia ante el arribo de nuevos usuarios en grandes cantidades.
- Debe poseer una estructura que permita identificar por códigos las áreas donde se encuentran los usuarios.

1.11 Marco regulatorio de las telecomunicaciones en Venezuela

El marco regulatorio de las telecomunicaciones está comprendido por un conjunto de leyes, las cuales se relacionan entre sí para dar todo el soporte legal y procedimental del manejo, utilización, permiso y explotación de las telecomunicaciones en Venezuela, así como aspectos relacionados con la tarificación de estos servicios.

En Venezuela, actualmente la materia regulatoria para las telecomunicaciones y sobre todo para la telefonía tradicional se encuentra contenida en la actual Ley Orgánica de telecomunicaciones. Adicionalmente, en Venezuela, el ente encargado en materia de regulación en telecomunicaciones es CONATEL (Comisión Nacional de Telecomunicaciones).

1.11.1 Comisión Nacional de Telecomunicaciones (CONATEL)

Actualmente el ente regulador en materia de telecomunicaciones y concesiones es CONATEL. La Comisión Nacional de Telecomunicaciones (CONATEL), fue creada mediante el Decreto Nº 1.826 el 5 de septiembre de 1.991, según Gaceta Oficial Nº 34.801, publicada el 18 de septiembre de 1.991. Esta Comisión en un principio se encontraba adscrita administrativamente al Ministerio de Infraestructura que poseía competencia para la regulación, planificación, promoción, desarrollo y protección de las telecomunicaciones en todo el territorio nacional. Además, CONATEL posee otras atribuciones como administrar, regular y controlar el uso de los recursos limitados utilizados en las telecomunicaciones, otorgar las habilitaciones administrativas y concesiones, así como proponer la aprobación de las tarifas, y fiscalizar y recaudar los recursos de origen tributario, así como fomentar y proteger la libre

competencia en el sector, conjuntamente con la Superintendencia para la Promoción de la Libre Competencia (Pro Competencia). Y es en la Ley Orgánica de Telecomunicaciones aprobada en el 12 de junio del año 2000 quien otorga en materia regulatoria las competencias a esta comisión.

La Ley Orgánica de Telecomunicaciones expresa en su artículo 37 las competencias de la Comisión Nacional de Telecomunicaciones.

1.11.2 Marco regulatorio actual de las telecomunicaciones

En lo que respecta a las primeras implementaciones de redes de telefonía y su uso, estas datan del año 1883 cuando se llevan a cabo la instalación de los primeros teléfonos en la ciudad de caracas y el establecimiento de un acuerdo de conexión entre para el entonces Ministerio de fomento y el proveedor de servicios Intercontinental Telephone Company of New Jersey [1]. En lo que respecta al marco regulatorio de la telefonía tradicional en Venezuela, este se encuentra bastante desarrollado.

En Venezuela, son un conjunto de leyes las que conforman el marco regulatorio de las telecomunicaciones, siendo las más importantes:

- Ley Orgánica de telecomunicaciones.
- Reglamento de la Ley Orgánica de Telecomunicaciones sobre habilitaciones administrativas y concesiones de uso y explotación del espectro radioeléctrico.
- Reglamento de Interconexión.
- Reglamento de apertura de los servicios de telefonía básica, entre otras.

A continuación, se describirán las más importantes en materia de regulación de las telecomunicaciones y con relación a los objetivos del desarrollo de este documento.

1.11.3 Ley Orgánica de Telecomunicaciones

La Ley Orgánica de Telecomunicaciones surge a partir de una seria de anteproyectos y proyectos de ley de telecomunicaciones, siendo aprobada y sancionada finalmente el 12 de junio del año 2000 justo cuando acaba el periodo de concurrencia limitada con la empresa CANTV.

La Ley Orgánica de Telecomunicaciones establece en su artículo 1 que: "Esta ley tiene como objeto establecer el marco legal de regulación general de las telecomunicaciones a fin de garantizar el derecho humano de las personas a la comunicación y la realización de las actividades económicas de las telecomunicaciones necesarias para lograrlo, sin más que las limitaciones que las derivadas de las leyes y de la Constitución de la República.

De esta ley también se desprende que todo operador de telecomunicaciones en Venezuela debe obtener habilitaciones administrativas y concesiones para poder ofrecer ciertos tipos de servicios de telecomunicaciones.

Además, en su artículo 2 establece los objetivos que cumple dicha ley [20]. Estos contemplan aspectos relacionados al establecimiento de redes de telecomunicaciones, la explotación de dichas redes, tarificaciones y la prestación de servicios de telecomunicaciones sobre dichas redes.

1.11.4 Reglamento de la Ley Orgánica de Telecomunicaciones sobre habilitaciones administrativas y concesiones de uso y explotación del espectro radioeléctrico

Este reglamento dispone el marco de las habilitaciones, concesiones y permisos las cuales amparadas bajo la Ley Orgánica de Telecomunicaciones y la Comisión Nacional de Telecomunicaciones son otorgadas. En su primer artículo define : "El presente reglamento tiene por objeto desarrollar el régimen general conforme al cual el Ministerio de Infraestructura o la Comisión Nacional de Telecomunicaciones, según el caso, otorgaran las habilitaciones administrativas para el establecimiento y explotación de redes y para la prestación de servicios de telecomunicaciones y las concesiones de uso y explotación del espectro radioeléctrico, de conformidad con lo previsto en la Ley Orgánica de telecomunicaciones, sus reglamentos y demás normas aplicables"

De este reglamento podemos extraer las siguientes definiciones para habilitación administrativa, atributos y concesión.

- **Atributos:** Actividades y servicios concretos que podrán prestarse bajo el amparo de una habilitación administrativa que otorga derechos y deberes inherentes a la actividad para la cual ha sido habilitado el operador, de conformidad con lo establecido en la Lotel. [20]

- **Habilitación administrativa**: Título que otorga Conatel para el establecimiento y explotación de redes y para la prestación de servicios de telecomunicaciones, a quienes hayan cumplido con los requisitos y condiciones establecidos por la Comisión y la LOTEL. [20]. Las habilitaciones pueden ser de radio difusión, radio difusión sonora y televisión abierta o generales siendo esta última habilitación la que permite la prestación de uno o más servicios o actividades. Dichas actividades están comprendidas por los atributos descritos anteriormente.

- **Concesión:** Acto administrativo unilateral mediante el cual CONATEL otorga o renueva, por tiempo limitado a una persona natural o jurídica, la condición de concesionario para el uso y explotación de una determinada porción del espectro, previo cumplimiento de la Ley [20]. Al igual que las habilitaciones, las concesiones son de tipo general, de radiodifusión y de recursos de orbita/espectro. Siendo las generales aquellas otorgadas por la Comisión Nacional de Telecomunicaciones sobre las porciones de espectro radioeléctrico necesarias para el establecimiento y explotación de redes y para la prestación de servicios amparados bajo las habilitaciones generales que así lo requieran.

1.11.5 Reglamento de interconexión

Este reglamento comprende también parte importante del marco regulatorio de las comunicaciones. Dicho reglamento fue publicado en gaceta oficial a la fecha del 24 de noviembre de 2000, teniendo una reforma parcial y siendo publicado en gaceta oficial para la fecha del 8 de noviembre de 2004. Este reglamento establece en su artículo primero que: "Este reglamento tiene por objeto establecer las normas aplicables a las relaciones que con motivo de la interconexión surjan entre los operadores de redes públicas de telecomunicaciones que presten servicios de telecomunicación y de estos con la Comisión Nacional de Telecomunicaciones" [21].

En este reglamento se contemplas los aspectos necesarios para llevar a cabo una interconexión y los aspectos relacionados a este proceso, estos otros aspectos incluyen contratos, cargos y tarifas de interconexión, derechos y responsabilidades de los prestadores y los usuarios.

En este reglamento se definen los principios bajo los cuales se rige la interconexión. Los principios son:

- Principio de neutralidad.
- Principio de buena fe.
- Principio de transparencia.
- Principio de no discriminación.
- Principio de igualdad de acceso.

1.11.6 Regulación de VoIP

La telefonía IP puede considerarse un servicio que utiliza las redes de comunicaciones actuales para llevar a cabo su funcionalidad, pero desde un punto de vista más general es una aplicación más que hace uso del Internet, y también es considerada una nueva tecnología. Como también se hizo mención en capítulos anteriores la telefonía IP aprovecha las bondades de las redes conmutadas por paquetes y puede ser implementada de manera local en las organizaciones contando con elementos mínimos para su implementación. El uso o la regulación de esta tecnología es llevada y mantenida por cada organización que implementa esta tecnología, ya que una organización o institución que haga uso de esta y extienda esta tecnología haciendo uso de enlaces contratados a proveedores de servicio, no pagara costos o cargos adicionales la circulación de los datos o tráfico que genere esta tecnología.

Siendo así no se ve claramente si esta tecnología deba ser o no ser regulada.

Actualmente el marco regulatorio en lo que respecta a la tecnología de telefonía IP no existe. Pero no solamente es el caso de Venezuela donde no existe regulación sobre esta tecnología. En países como Estados unidos la FCC (*Federal Communications Commission*) apoya la no regulación de esta tecnología. Podemos citar lo siguiente:

"La FCC no regula gran parte del mercado VoIP. De hecho, recientemente en junio de 2012 el comisionado Robert M. McDowell declaró: El gobierno debería resistir la tentación de regular innecesariamente, salir del camino del Internet y permitirle continuar en la difusión de prosperidad y libertad alrededor del mundo. La conectividad del internet, especialmente a través de dispositivos móviles, está mejorando las condiciones humanas como ninguna otra innovación en la historia mundial" [22].

Ahora bien, la manera de cómo se implementa la tecnología de telefonía IP puede incurrir en sanciones legales si estos sistemas de comunicación intentan terminar o dirigir el flujo de una llamada a la telefonía tradicional sin poseer habilitaciones necesarias.

Es bien sabido que, como toda tecnología, estas pueden ser mal utilizadas, siendo la tecnología de VoIP una de ellas. La tecnología de VoIP ha sido usada para realizar "llamadas gratuitas" evadiendo los altos costos de la telefonía tradicional. Usualmente se pueden encontrar implementaciones de estas "soluciones" en escenarios internacionales para la transmisión de llamadas sin pasar por las redes de telefonía tradicional. En ambientes donde son terminadas las llamadas provenientes de la telefonía IP en sistemas de telefonía tradicional, podemos encontrar los siguientes escenarios, donde la principal diferencia entre los escenarios es quien posea o no, habilitaciones administrativas. Debemos recordar que estas habilitaciones son tramitadas por CONATEL y deben poseer atributos de telefonía.

Los operadores que deseen terminar llamadas en sistemas de telefonía tradicional pueden ser de tres tipos, Blancos, grises o negros.

- **Operadores legales:** Son aquellos que cuentan con una habilitación otorgada por la comisión nacional de telecomunicaciones CONATEL, para la terminación de las llamadas generadas en sus sistemas de telefonía IP en los sistemas de telefonía tradicional. Estos cuentan con enlaces y conexiones necesarias para lograr esa funcionalidad.

- **Operadores mixtos**: Estos operadores o proveedores son aquellos que no poseen una habilitación para la terminación de llamadas generadas de sus sistemas de telefonía IP a los sistemas de telefonía tradicional, pero estos operadores contratan o hacen uso de otros proveedores de servicio que tenga esta habilitación, de esta forma los proveedores grises hacen uso de un intermediario "habilitado" para la terminación de las llamadas.

- **Operadores ilegales:** los cuales no cuentan con habilitación y hacen terminar las llamadas provenientes de sus sistemas de telefonía IP hacia los sistemas de telefonía tradicional sin los correspondientes permisos.

1.12 Aplicación web

El termino aplicación web hace referencia a una página web o a un conjunto de páginas web entregadas vía HTTP/HTTPS las cuales hacen uso de un procesamiento de datos, este procesamiento de datos puede realizarse en el servidor (server-side) o del lado del cliente (client-side) para proveerle al usuario una experiencia de tipo aplicación en un explorador web. Las aplicaciones web difieren de las simples páginas web ya que las aplicaciones web incluyen elementos de interacción ejecutables localmente y estados persistentes. [23]

Los datos son presentados al usuario dentro de sus exploradores web como información que es generada automáticamente (en un formato especifico) por la aplicación web desde un servidor web. Las aplicaciones web consultan el servidor del contenido (generalmente un repositorio o una base de datos de contenido) y dinámicamente el contenido es generado para ser servido al cliente. Los documentos o la información son generados en un formato estándar soportado por todos o la mayoría de los exploradores (HTML, XHTML). [24]

En una aplicación web, es el explorador web un elemento importante ya que interpreta y ejecuta los datos recibidos desde la aplicación web mientras muestra las paginas solicitadas y su contenido.

Una ventaja significativa del uso de aplicaciones web es que su funcionamiento no se encuentra ligado directamente al sistema operativo o al explorador web utilizado del lado del cliente y su despliegue es relativamente rápido en su mayoría ya que no requieren (en su mayoría) instalación de software adicional. [24]

Las aplicaciones web están generalmente divididas en secciones lógicas denominadas "tiers" o niveles donde cada nivel tiene asociado un rol. Las aplicaciones web tienden a poseer una estructura multinivel o n-tier, pero generalmente encontramos aplicaciones de 3 niveles. [25]

En una estructura de tres niveles, nuestro primer nivel será el cliente o el usuario y las tecnologías de acceso e interacción hacia nuestra aplicación (Explorador web). Como segundo nivel tenemos los motores de despliegue de nuestra aplicación web en donde las distintas tecnologías como ASP, CGI, JSP/Java, Node.js, PHP, Python, Laravel o Ruby on Rails nos ayudan a cumplir con este

propósito. Por último y no menos importante tenemos el tercer nivel en donde se encuentran las bases de datos de nuestra aplicación. Estos tres niveles son conocidos como nivel de presentación (usuario), nivel de lógica de la aplicación (tecnologías web) y nivel de almacenamiento (bases de datos). [25]

Esto lo podemos apreciar mejor en la figura 11:

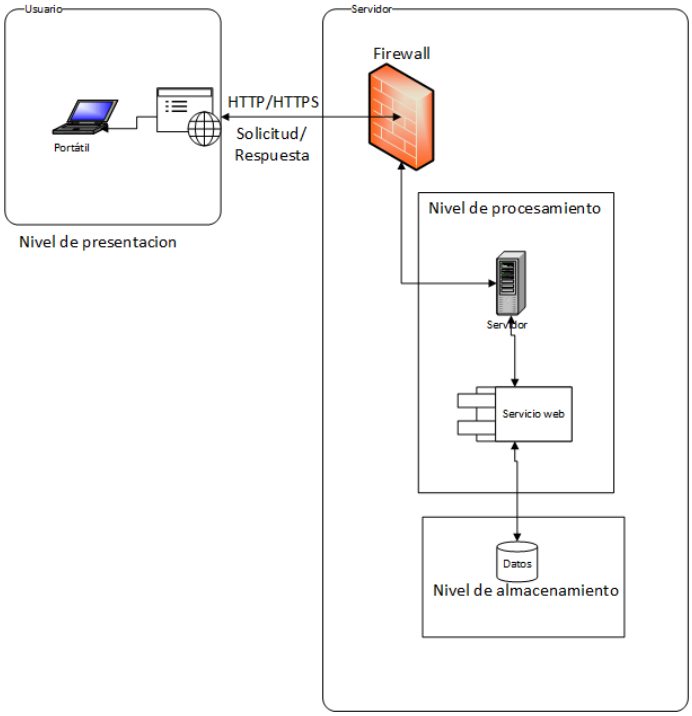

Figura 11***:*** División por capas de una aplicación web.
Fuente: Autor

1.13 Arquitectura cliente/servidor

Desde el punto de vista funcional, se puede definir a la arquitectura cliente-servidor como una arquitectura distribuida que permite a los usuarios finales obtener acceso a la información en forma transparente aún en entornos multiplataforma. Su funcionamiento consiste en que se tiene una máquina cliente, la cual requiere un servicio de una máquina servidor, y ésta realiza la función para la que está programada [47].

En general, un servicio es una abstracción de un recurso computacional en donde un cliente no tiene por qué preocuparse por el cómo el servidor gestiona el mismo y se encarga al mismo tiempo de la petición del usuario, procesamiento y envió de la respuesta a dicha solicitud. El cliente solo tiene que entender que la respuesta está basada sobre una aplicación y protocolo bien conocido, por ejemplo, el contenido y el formato de los datos solicitados por el servicio.

Un servidor puede recibir muchas solicitudes desde diferentes clientes en un periodo corto de tiempo.

Clientes y servidores intercambian mensajes en un patrón de comunicación solicitud-respuesta. En este patrón, el cliente envía una solicitud, y el servidor retorna una respuesta. Para que este proceso pueda llevarse a cabo, las computadoras deben hablar un lenguaje en común y deben seguir las reglas asociadas a dicho lenguaje, de esta forma tanto el cliente como el servidor sabrán que esperar de su

contraparte. El lenguaje y las reglas de comunicación son definidas en un protocolo de comunicación. Estos protocolos operan en la capa de aplicación. Se pueden utilizar convenciones y procesos de abstracción para que un servicio no dependa de una plataforma y que la comunicación se lleve a cabo de una manera sencilla.

Las aplicaciones web se basan generalmente en la arquitectura cliente servidor, además de implementar a su vez una estructura de tres capas como se mencionó anteriormente (figura 12).

Figura 12***:*** Modelo cliente-servidor.
Fuente: Autor.

En términos generales, la capa de presentación proporciona la interfaz necesaria para presentar información y reunir datos. La capa de procesamiento responde a peticiones del usuario para ejecutar una tarea en específico, interactuando con los datos que están almacenados. La capa de almacenamiento representa las fuentes de datos finales y está formada por uno o más gestores de bases de datos que realizan todo el almacenamiento y reciben solicitudes de recuperación de información desde la capa de negocio.

1.13 Tecnologías del lado del cliente

Para el correcto despliegue del producto generado como resultado de la elaboración del presente trabajo especial de grado, es necesario que del lado del cliente se cuente o se soporten las siguientes tecnologías para el acceso a la aplicación web administrativa y a la plataforma de interconexión Kamailio Asterisk.

1.13.1 HyperText Transfer Protocol (HTTP)

Es un protocolo a nivel de aplicación para sistemas de información multimedia distribuidos. Es un protocolo no orientado a estado que puede ser utilizado, entre muchos otros propósitos, para manejar ficheros HTML [48].

HTTP fue desarrollado por el World Wide Web Consortium y la Internet Engineering Task Force, colaboración que culminó en 1999 con la publicación de una serie de RFC, el más importante de ellos es el RFC 2616 que especifica la versión 1.1. HTTP define la sintaxis y la semántica que utilizan los elementos de software de la arquitectura web (clientes, servidores, proxies) para comunicarse. [26]

Entre sus principales propiedades encontradas están las siguientes:

- Posee un esquema de direccionamiento comprensible: utilizando el Universal Resource Identifier (URI), para localizar sitios (URL) o nombres (URN) sobre los que hay que aplicar un método. La forma general de una URL es: servicio://host/fichero.extensión.- Cuenta con una arquitectura cliente servidor: HTTP se asienta en el paradigma solicitud / respuesta. La comunicación se asienta sobre TCP/IP. El puerto por defecto es el 80, pero pueden ser utilizados otros puertos.

- Es un protocolo sin conexión ni estado: luego de que el servidor ha respondido la petición del cliente, la conexión entre ambos se rompe. Además, no se guarda memoria de contexto de la conexión actual para conexiones posteriores.

- Está abierto para nuevos tipos de datos: utiliza tipos MIME (*Multipart Internet Mail Extension*) para la determinación de los tipos de datos que transporta

Cuando un servidor HTTP transmite información de regreso incluye una cabecera que le indica al cliente sobre los tipos de datos que componen el documento. De la gestión de esos datos se encargan las utilidades que tenga el cliente (visor de imágenes, de vídeo, entre otros.)

1.13.2 HTML (Hypertext Markup Language)

Hypertext Markup Language es un lenguaje con el que se define una página web, esta se basa en el uso de etiquetas que sirven para definir el texto y otros elementos que contenga la página. Es fácil de aprender, por lo que cualquier persona, programadora o no pueda usarlo. HTML se ayuda de los formatos CSS para darle un mejor estilo a la página. Actualmente se maneja la última versión HTML 5, el cual contiene un conjunto amplio de tecnologías que permite a los sitios web y a las aplicaciones ser más diversas y de más alcance.

1.13.3 CSS (Cascading Style Sheets)

Cascading Style Sheets, u Hojas de Estilo en Cascada, es un lenguaje o formato utilizado para la presentación de HTML, este se basa en utilización de etiquetas que definen el estilo de porciones de código del HTML; permitiendo tener una apariencia uniforme de la página; como también hacer que el código HTML sea más sencillo de leer, ya los CSS se definen por separado, es intuitivo y fácil de usar. La última versión disponible es CSS 3, que soporta más colores, bordes con degradado, bordes con imágenes, cajas con sombras, sombra para textos y múltiples columnas.

1.13.4 Teléfonos IP

Los usuarios finales para poder establecer comunicación desde y hacia la plataforma, requieren de teléfonos IP. Los teléfonos IP son a veces llamados teléfonos VoIP, teléfonos SIP o teléfonos basados en software. Todos estos son exactamente la misma cosa y están basados en el principio de transmisión de voz sobre Internet, o tecnología VoIP, como es mejor conocido. [27]

Hay varios tipos de teléfonos IP. Entre ellos podemos encontrar dos categorías, los hardphone y los softphone. El primero es un teléfono tangible, físico, con aspecto similar o idéntico al de un teléfono tradicional. Por otro lado, (un softphone es un teléfono que permite realizar llamadas usando redes de datos desde una computadora o un dispositivo inteligente. Como su nombre lo indica, un softphone es un software que actúa como interfaz de telefonía, permitiendo discar números telefónicos además de asumir funciones relacionadas a la telefonía utilizando la pantalla, ratón y teclado para realizarlas. Las

conversaciones utilizando softphone hacen uso de dispositivos como auriculares o micrófono y cornetas [28].

1.14 Tecnologías del lado del servidor

Para el desarrollo del presente trabajo, se utilizaron distintas tecnologías y herramientas de software que permitieron lograr los objetivos planteados. Estas herramientas abarcan desde servidores de llamadas hasta frameworks para el desarrollo de la aplicación web administrativa. A continuación, se listan las herramientas utilizadas:

1.14.1 Asterisk

Asterisk es un aplicativo open source que proporciona funcionalidades de PBX avanzada. Licenciado bajo licencia dual, GNU/GPL y propietaria, Asterisk empezó a desarrollarse el año 1999 de la mano de su creador Mark Spencer y fundador de Digium (principal empresa desarrolladora de Asterisk) quién tras intentar adquirir una solución privada decidió programar su propia centralita por el elevado precio de estas.

Asterisk presenta una arquitectura modular (figura 13) que permite activar y desactivar módulos específicos en función de la necesidad. Estos módulos se dividen en siete categorías:

- Core: Núcleo de Asterisk que posibilita la carga del resto de módulos
- Recursos: Aportan diversas funcionalidades a Asterisk
- Canales: Permiten a Asterisk interactuar con dispositivos de diversas tecnologías (SIP, IAX, DAHDI)
- Aplicaciones y funciones: Proporcionan herramientas para configurar el sistema.
- CDR: Control del registro de llamadas
- Códec: Gracias a estos Asterisk puede actuar como codificador y descodificador de audio y vídeo.
- Formatos: Posibilitan a Asterisk la gestión de ficheros de diversos formatos (wav, mp3, ulaw...)

Asterisk es capaz de trabajar con prácticamente todos los estándares de telefonía tradicional (líneas analógicas, digitales: E1/T1) además de soportar la mayoría de protocolos VoIP (SIP, IAX2, MGCP, Cisco Skinny).

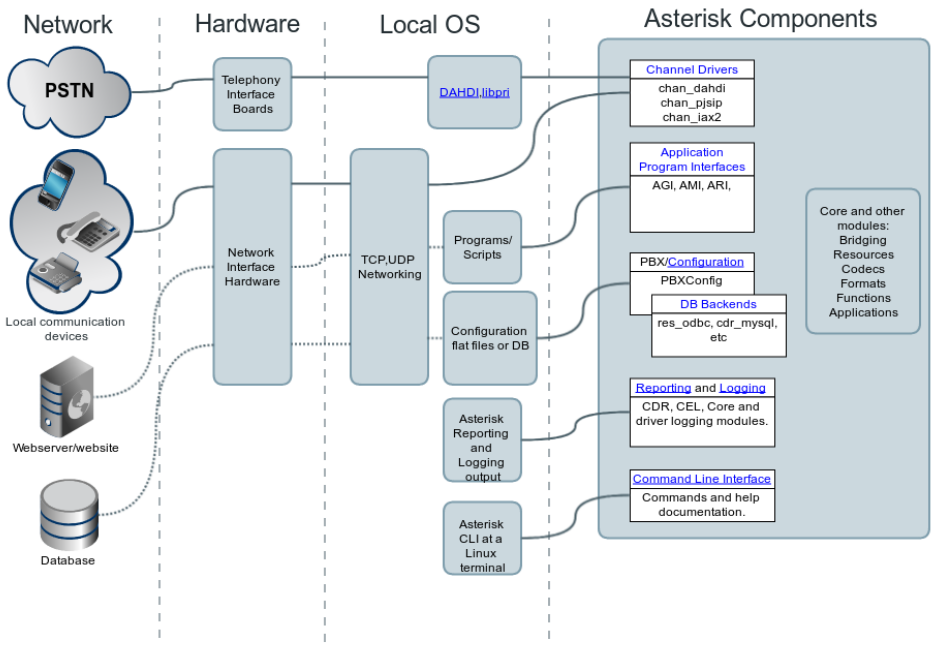

Figura 13***:*** Estructura servidor Asterisk
Fuente: [29]

Asterisk proporciona funcionalidades básicas y avanzadas frente a otras soluciones propietarias, entre estas:

1.14.1.1 Funcionalidades

- Llamada en espera
- Transferencia Identificado de llamada
- Bloqueo de Caller ID
- Timbres distintivos
- Música en espera
- Música en transferencia
- Salas de Conferencia
- Buzón de Voz personal
- Colas de llamada Colas con prioridad
- Registro de llamadas en BD
- Buzón de Voz por Mail
- Pickup de llamadas
- Desvío si ocupado
- Desvío si no responde
- IVR: Interactive Voice Response (Gestión de llamadas a través menús interactivos.)
- LCR: Least Cost Routing, encaminamiento de llamadas por el proveedor más económico.
- AGI: Asterisk Gateway Interface (integración con todo tipo de aplicaciones externas.)
- AMI: Asterisk Management Interface, gestión y control remoto de Asterisk.
- Configuración en BD: usuarios, extensiones ...

Asterisk puede ser instalado sobre sistemas GNU/Linux, BSD y MacOSX. Está disponible en los repositorios de las distribuciones más comunes, sin embargo, su instalación suele realizarse a partir del código fuente. La configuración de Asterisk suele realizarse a través de ficheros de texto.

1.14.3 Kamailio

Kamailio también conocido como SIP Router y anteriormente al año 2008 como OpenSER, es un servidor proxy SIP open source, licenciado bajo GNU/GPL. Se caracteriza por su capacidad de gestionar cientos de llamadas por segundo. Kamailio puede ser utilizado para construir grandes plataformas de servicios de VoIP o para ampliar pasarelas SIP o media servers como Asterisk o FreeSWITCH. La aplicación está escrita en C para plataformas Linux/UNIX y se centra en el rendimiento, la flexibilidad y la seguridad. Además de C, en algunas funcionalidades también se emplean lenguajes como Lua, Perl o Python.

Al igual que Asterisk, Kamailio tiene una arquitectura modular que permite adaptar su configuración en función de las características que se necesiten. Algunas de las características que proporcionan alrededor de 150 módulos disponibles, son las siguientes:

- Diversos protocolos: TCP, UDP y SCTP
- Comunicaciones seguras a través de TLS
- IPV4 y IPV6
- SIMPLE (Session Inititation Protocol for Instant Messaging and Presence Leveraging Extensions)
- ENUM (mapeo de número telefónico)
- Enrutamiento de menor costo
- Balanceo de carga,
- Fail-over
- Autenticación y autorización a través de MySQL, PostgreSQL, Oracle, RADIUS, LDAP
- Monitorización SNMP

Este conjunto de funcionalidades y módulos se encuentran soportados por la manera en la que Kamailio se encuentra estructurado. Esta estructura se encuentra dividida en 2 categorías principales [30]: Core, que proporciona las funcionalidades de más bajo nivel y, los módulos o componentes que proporcionan el resto de funcionalidades.

1.14.4 Laravel

Laravel es un framework de código abierto para desarrollar aplicaciones y servicios web con PHP 5 y PHP 7. Su filosofía es desarrollar código PHP de forma elegante y simple, evitando el "código espagueti". Fue creado en 2011 y tiene una gran influencia de framework como Ruby on Rails, Sinatra y ASP.NET MVC [31]

1.14.4.1 Características [32]:

- **Modular y extensible:** Laravel es modular y extensible. Esto quiere decir que te permite agregar todo lo que necesitas a través de su directorio Packalyst que cuenta con más de 5,500 paquetes.

- **HTTP Routing:** Laravel cuenta con un sistema de enrutamiento rápido y eficiente, similar al que se usa en Ruby on Rails. Este nos permite relacionar las partes de nuestra aplicación con las rutas que ingresa el usuario en el navegador.

- **HTTP Middleware:** Laravel implementa un componente llamado Middleware el cual intercepta, analiza, filtra y procesa las solicitudes HTTP antes de ser redirigidas al módulo correspondiente del procesamiento de dicha solicitud.

- **Caché:** Las aplicaciones desarrolladas en Laravel cuentan con un robusto sistema de caché el cual puede ser ajustado de acuerdo a las necesidades de la aplicación y así, ofrecer la mejor experiencia posible a tus usuarios.

- **Autenticación:** La seguridad es muy importante. Laravel viene listo para implementar autenticación de usuarios de forma nativa e incluye la opción de "recordar" al usuario. Además, te permite incluir parámetros adicionales, lo que nos asegurará, por ejemplo, si se trata de un usuario activo.

- **Tareas automatizadas:** Elixir es una API de Laravel que nos permite definir tareas de Gulp con las que podemos definir el uso pre-procesadores para comprimir nuestro CSS y JavaScript.

- **Encriptación:** Laravel tiene lo necesario para usar seguridad con OpenSSL y cifrado AES-256-CBC.

- **Object-Relational-Map (ORM):** Laravel incluye una capa para manejo de bases de datos que cuenta con un ORM llamado Eloquent. Este, además, funciona perfectamente con PostgreSQL.

- **Unit Testing:** El desarrollo de Unit Testing es una tarea que consume una cantidad considerable de tiempo. Pero nos da la certeza de que nuestra aplicación funcionará sin problemas. Laravel incluye métodos para realizar Unit Testing usando PHPUnit.

CAPITULO 2
MARCO METODOLOGICO

Para el desarrollo del presente trabajo se utilizó la metodología Scrum por la flexibilidad y beneficios que este ofrece. Scrum permite obtener resultados de manera rápida fijando periodos cortos para la entrega de resultados, así como planificaciones por fase frecuentes que nos permiten evaluar y adaptar el desarrollo del producto.

Scrum es un framework para el manejo de proyectos que tienen como fin el desarrollo de productos complejos [33].

De acuerdo a diversas bibliografías podemos decir que, en esencia, Scrum es:

- Dar objetivos claros al equipo
- El equipo se organiza en función al trabajo a realizar
- Periódicamente el equipo prioriza, desarrolla y entrega las funcionalidades más importantes.
- El equipo recibe retroalimentación de individuos que se encuentran fuera del equipo.
- Con la ayuda de la retroalimentación de los integrantes del equipo (internos o externos), el equipo reflexiona constantemente sobre su manera de trabajar, con el objetivo de mejorar.
- La organización completa posee visibilidad sobre el progreso del equipo y del proyecto.
- El equipo y la gerencia se comunican entre sí de manera honesta (preferiblemente cara a cara) tanto en aspecto de progreso como de riesgos.

Scrum tiene sus orígenes en los campos del manejo del conocimiento, los sistemas adaptativos complejos y la teoría de control empírico de procesos. Ha sido influenciado también de patrones observados durante el desarrollo de software y la Teoría de las Limitaciones [33].

Scrum contempla doce principios los cuales se mencionan a continuación [34]:

- Nuestra más alta prioridad es satisfacer al cliente por medio de la entrega continua y temprana de software de valor.

- Recibir cambios en los requerimientos, incluso si se está en una etapa tardía del desarrollo. Los procesos agiles controlan y adoptan los cambios para hacer de estos una ventaja competitiva para el cliente.

- Entregar software funcional frecuentemente, desde un par de semanas hasta un par de meses, con una preferencia hacia las escalas cortas.

- Las personas del negocio deben trabajar en conjunto diariamente a lo largo del desarrollo del proyecto.

- Construir proyectos rodeados de individuos motivados. Otorgarles el ambiente y el soporte que ellos necesitan y confiar en que pueden realicen el trabajo.

- El método más efectivo y eficiente para comunicar información hacia y dentro de un equipo de desarrollo es la conversación cara a cara.

- El software funcional es la primera medida de progreso.

- Los procesos agiles promueven el desarrollo sostenible. Los patrocinadores, desarrolladores y usuarios deberían ser capaces de mantener una paz constante e indefinida.

- La atención continua hacia la excelencia técnica y el buen diseño mejora la Agilidad.
- Simplicidad, el arte de maximizar la cantidad de trabajo no realizado, es esencial.
- Las mejores arquitecturas, requerimientos y diseños emergen desde equipos auto-organizados.
- En intervalos regulares, el equipo refleja maneras de cómo ser más efectivo, por esta razón, el equipo modificara y ajustara su comportamiento.

2.1 Roles en Scrum

Dentro de la metodología SCRUM, Los roles tradicionales y las funciones asociadas a estos se encuentran agrupadas en tres roles definidos [35] dentro de esta metodología:

- ***Product Owner*:** Es el responsable de maximizar el valor del producto, principalmente por la gestion incremental y expresar las expectativas funcionales y de negocio del producto hacia el equipo de desarrollo.
- ***Scrum Master*:** Es responsable de guiar, motivar, enseñar y asistir al equipo Scrum y a su entorno bajo un enfoque y entendimiento pleno de Scrum.
- **Equipo:** Es o son los responsables de gestionar, organizar y hacer todo el trabajo de desarrollo requerido para la entrega incremental del producto durante cada Sprint.

A continuación, se describen y listan las funcionalidades asociadas a cada uno de los roles Scrum:

2.1.1 Product Owner

Las responsabilidades del Product Owner [33] son las siguientes:

- Obtención de una visión compartida.
- Recolección de requerimientos.
- Administración y priorización del Product Backlog.
- Aceptación de software al final de cada iteración.
- Administración del plan de reléase.
- Maximización del retorno de inversión del proyecto.

2.1.2 Scrum Master

Las responsabilidades del ScrumMaster [33] son:

- Aseguramiento de un entorno de trabajo para el equipo, protegido de interferencias y directivas.
- Remoción de impedimentos.
- Fomento del uso y respeto al proceso.
- Extensión del uso de Scrum a lo largo de la organización.

2.1.3 El Equipo

Las responsabilidades del rol Equipo o Miembro de Equipo [33] son:

- Estimación del tamaño de los ítems del backlog.
- Compromiso de entregar incrementos de software con calidad de producción.
- Seguimiento de su propio avance.
- Auto-organización, aunque con la responsabilidad ante el Product Owner de entregar aquello que fue comprometido.

2.2 Otros elementos de la metodología Scrum

La metodología Scrum no solamente está conformada por el equipo, el *product owner* y el scrum master. La metodología Scrum se divide también en fases de trabajo que denominaremos Sprints y un conjunto de reuniones en momentos determinados del desarrollo, así como también artefactos. Estos elementos los definiremos a continuación:

2.2.1 Sprint

El sprint es el latido del ciclo de Scrum. La planificación y la seguidilla de revisión y retrospectiva marcan respectivamente el comienzo y fin del sprint. La longitud del sprint se encuentra fija y jamás se extiende. La gran mayoría de los equipos Scrum eligen una duración de sprint de dos, tres o a lo sumo cuatro semanas. Durante el sprint el equipo lleva a cabo una reunión diaria. Cada reunión en Scrum tiene una duración máxima fijada a priori. Para un sprint de cuatro semanas el tiempo que suele dedicarse a la primer y segunda parte de la reunión planificación, así como también a la revisión y retrospectiva suele fijarse en cuatro horas cada una. Para sprints más cortos deberán ser ajustadas en proporción con la duración del sprint. [33]

2.2.2 Reunión diaria

La reunión diaria (daily meeting) es uno de los tres puntos de inspección y adaptación en Scrum. El equipo se reúne para comunicar y sincronizar su trabajo. Dado que el equipo trabaja de manera colaborativa este momento es esencial para asegurar un progreso continuo y evitar bloqueos. Además, el equipo medirá permanentemente su propio progreso en términos del objetivo del sprint. [33]

2.2.3 Revisión del Sprint

La revisión del sprint es a veces denominada de forma incorrecta como demo. Si bien es cierto que se incluye en ella una demostración de las nuevas funcionalidades que el equipo desarrolló durante el sprint, su principal objetivo es inspeccionar lo entregado por el equipo y obtener retroalimentación de los asistentes a la reunión para poder adaptar el plan para sprints subsiguientes. [33]

2.2.4 Retrospectiva

La retrospectiva es la última reunión del sprint. Sigue inmediatamente después de la revisión y nunca debe ser omitida. Mientras que la revisión está centrada en el producto, la retrospectiva se encuentra enfocada en el proceso, la manera en la que el equipo Scrum trabaja de manera conjunta, incluyendo habilidades técnicas y prácticas y herramientas de desarrollo. Si bien la revisión se encuentra abierta a quien desee asistir, la retrospectiva se restringe a los miembros del equipo Scrum, esto es el Product Owner, los miembros del equipo de desarrollo y el ScrumMaster. Cualquier persona ajena, incluyendo gerentes de cualquier nivel jerárquico, están excluidos, a menos que sean invitados por el equipo. [33]

2.3 Artefactos

Un artefacto en la metodología Scrum representa un objeto o instancia que posee un valor o representa un trabajo. Los artefactos son generados durante el proceso de desarrollo del producto y son diseñados para maximizar la transparencia de la información y del proceso, esto permite que todos tengan el mismo entendimiento del mismo. El artefacto provee además un medio para la inspección y la adaptación en el proceso de desarrollo del producto.

Dentro de la metodología Scrum se define cuatro artefactos:

- Product Backlog
- Sprint Backlog
- Burndown de tareas
- Backlog de impedimentos

2.3.1 Product Backlog

El product backlog es simplemente una lista de elementos que representan trabajo pendiente. Cualquiera puede agregar ítems al backlog, pero sólo el Product Owner tiene el derecho de determinar el orden en el que serán desarrollados por el equipo.

Un buen Product Owner, por supuesto, negociará esto con los stakeholders (asociados de negocio) y el equipo. Los requerimientos son emergentes, lo que significa que no conocemos ni podemos conocer de antemano todas y cada una de las características que queremos que contenga el producto. Es por ello que Product Backlog es un documento viviente, que requiere una constante preparación para mantenerlo actualizado y útil con el paso del tiempo. Se agregarán nuevos elementos; elementos existentes serán desagregados en múltiples elementos más pequeños; algunos elementos serán removidos al darnos cuenta que ya no son necesarios. Más aún, los elementos deben ser estimados para conocer la relación costo beneficio de los mismo, lo que influirá de manera directa en la ubicación que el mismo tendrá en el backlog. [33]

2.3.2 Sprint backlog

La mayoría de los equipos conoce al sprint backlog como el tablero de tareas, que es simplemente una representación física del trabajo al que se han comprometido para lo que resta del sprint. El tablero de tareas es un ejemplo de un Kanban, una palabra japonesa que significa "señal visual". El mismo comunica al equipo y a cualquiera que desee saberlo qué tareas planificó el equipo y su estado actual [33].

2.3.3 Burndown del Sprint

El gráfico de burndown de tareas del sprint está diseñado para ayudar al equipo en la monitorización de su progreso y para ser el indicador principal que informará sobre sus posibilidades de alcanzar su compromiso al finalizar el sprint. El formato clásico requiere que el equipo estime la duración de cada tarea en horas de forma diaria. El burndown deberá completarse de forma tal que grafique cuántas horas de trabajo restan para concluir el sprint. [33]

2.3.4 Backlog de Impedimentos

El backlog de impedimentos es simplemente la lista de situaciones que están impidiendo que el equipo progrese. Estas son situaciones que el ScrumMaster debe quitar del camino en su interminable gesta a través del cual ayudará al equipo para que trabaje de la mejor manera posible. [33]

CAPITULO 3

MARCO APLICATIVO

3.1 Planteamiento del problema

Como pudimos apreciar a lo largo de este documento, la tecnología de VoIP (Voz sobre IP) es una tecnología que ha venido desarrollándose de forma progresiva y positiva, ofreciendo muchos beneficios y servicios de valor agregado, que en redes de telefonía tradicional serían difíciles de lograr. Los servicios de valor agregado son aquellos que ofrecen facilidades o funcionalidades que los diferencian del servicio base (realizar llamadas) y entre ellos podemos mencionar: Servicios interactivos de atención por voz (Interactive Voice Response, IVR), servicios de llamadas en conferencia, servicio de directorio, servicio de buzón de voz, discado a varias extensiones para la localización de un usuario (Servicio Follow me), entre otros

Muchas instituciones en Venezuela han optado por implementar una solución de Voz sobre IP (VoIP) para el aprovechamiento de su infraestructura de red y por los beneficios que la tecnología de VoIP trae consigo como, por ejemplo, uno de los beneficios con mayor atractivo, la disminución de costos en llamadas.

Las organizaciones que hacen uso de la tecnología de VoIP para poder comunicarse con otras organizaciones, hacen uso de los servicios ofrecidos por la telefonía tradicional, desaprovechando el potencial de comunicación que la tecnología VoIP confiere. En general, Los gastos en llamadas vienen dados por la utilización de los sistemas de telefonía tradicional.

Un aspecto importante a considerar, es que la gran mayoría de las organizaciones o instituciones se cuentan con enlaces que proporcionan acceso al Internet, pudiendo este enlace de comunicación ser aprovechado para la transmisión de llamadas y establecimiento de comunicaciones al utilizar la tecnología de VoIP.

Al realizar una interconexión entre estas islas, se crea la posibilidad de conformar una red de telefonía IP a gran escala en la que las organizaciones que se suscriban podrán comunicarse entre ellas mediante el uso de la tecnología VoIP. De lo anterior surgen interrogantes como: ¿Cuáles serían los requerimientos para lograr esta interconexión entre todas estas islas?, ¿Qué elementos serían necesarios a consideran para lograr esta interconexión?

Estas interrogantes conllevan a plantear una solución integral que permita la comunicación entre estas islas, disminuyendo los costos de llamadas entre ellas, la posible incorporación de más entidades a esta red aumentando el uso y aceptación de esta tecnología, así como el posterior desarrollo de servicios de valor agregado en esta plataforma.

En ambientes donde es implementada una solución VoIP, se puede apreciar un enfoque centralizado. Los beneficios que un enfoque centralizado se encuentran relacionados a la administración y gestión del dominio de telefonía IP de una organización. Ahora bien, extrapolar estos beneficios a gran escala representa un reto, entre ellos podemos mencionar la administración de los múltiples usuarios que deseen registrarse en el sistema y hacer uso de los servicios provistos por la plataforma.

Para la comunicación y gestiones relacionadas con este elemento centralizado es necesaria la creación de una interfaz que permita interactuar y generar configuraciones que permitan a las "islas" asociarse y crear enlaces troncales de manera exitosa hacia la plataforma de interconexión, pudiendo ser esta interfaz desarrollada usando tecnologías web.

De lo anterior expuesto, se define entonces:

3.2 Objetivo General

Implementar una plataforma telefonica basada en el servidor de llamadas Asterisk y el servidor SIP proxy Kamailio para la interconexión de islas voz sobre IP contemplando un plan de numeración unificado.

3.3 Objetivos específicos

- Generar las configuraciones necesarias para integrar los servidores Asterisk y Kamailio.
- Realizar pruebas pertinentes para corroborar y verificar el correcto funcionamiento de la plataforma integrada.
- Realizar pruebas de llamadas con distintos escenarios alineados con los objetivos del presente trabajo.
- Generar la documentación necesaria para realizar réplicas de los ambientes de prueba implementados.
- Diseñar plan de marcado unificado para la asignación de códigos de identificación a las islas VoIP empleando características similares al Plan Nacional de Numeración.
- Generar listado de requerimientos para las organizaciones o islas VoIP que deseen suscribirse a la plataforma integrada.
- Realizar el análisis de requerimientos y funcionalidades de la aplicación web administrativa de la plataforma integrada.
- Diseñar y conceptualizar la aplicación web administrativa de la plataforma integrada.
- Desarrollar e implementar los requerimientos y funcionalidades siguiendo las buenas prácticas para el desarrollo de aplicaciones con el uso del framework Laravel.

3.4 Limitaciones y alcance

El presente trabajo constara de la implementación e integración entre los servidores SIP proxy Kamailio y el servidor de llamadas Asterisk para proveer una plataforma que permita la interconexión entre múltiples organizaciones y que las llamadas VoIP puedan transitar por medio de la plataforma. Sin embargo, serán empleadas solamente dos organizaciones suscriptoras para la evaluación de las funcionalidades de la plataforma integrada.

Para la administración, gestión de usuarios y enlaces troncales, se desarrolla una aplicación web basada en el framework Laravel, la cual permite el registro de las organizaciones participantes, gestionar los datos de los suscriptores, generar archivos de configuración para el correcto establecimiento de los enlaces troncales, chequeo del estado de las troncales y del servidor y despliegue de información relacionada a los usuarios y las troncales. El desarrollo fue realizado contemplando el funcionamiento

básico para la implementación de este conjunto de características. Funcionalidades avanzadas relacionadas a las características antes mencionadas no forman parte del alcance del presente proyecto, como el manejo de solicitudes distintas a la de creación de un nuevo enlace troncal o desarrollo front-end con mejores características visuales.

No son contempladas para el alcance de este proyecto, pruebas de estrés en el sistema tanto para la evaluación altas cantidades de llamadas concurrentes entre las organizaciones como en el registro de las mismas. Las pruebas realizadas incluyeron servidores Elastix, FreePBX y Asterisk. No fueron realizadas pruebas con otros servidores de llamadas como Cisco Call Manager o Avaya.

Adicionalmente, la aplicación web permite realizar operaciones básicas en la plataforma remota como la carga y actualización de enlaces troncales que permiten conectar a las organizaciones suscriptoras.

3.5 Justificación e importancia

En Venezuela, la existencia de proveedores de servicios que den soporte a las comunicaciones VoIP es reducida. Estos proveedores de servicios están relacionados con la terminación de llamadas en los sistemas de telefonía tradicional más que con la posibilidad de interconectar organizaciones que hagan uso de esta tecnología. Muchos de estos, adicionalmente a la contratación de sus servicios para la comunicación VoIP, requieren la adquisición de equipos especiales para implementar completamente el servicio de comunicación VoIP. La existencia de una plataforma que permita la interconexión de las organizaciones que hagan uso de la tecnología VoIP para un mejor aprovechamiento de los beneficios que esta confiere, no es evidente.

El presente trabajo surge ante la necesidad de crear una plataforma que nos permita interconectar a las organizaciones que hagan uso de VoIP (islas). Poder implementar esta plataforma, influirá positivamente en las organizaciones participantes, ya que se crea la posibilidad de una reducción de costos en comunicación con las demás organizaciones suscritas. Cada organización interconectada podrá realizar llamadas a otras organizaciones que se hayan suscrito sin la necesidad de utilizar los servicios de telefonía tradicional. Además, dependiendo del número de organizaciones suscritas, se crea la posibilidad de establecer el paso inicial para la conformación de una red a gran escala de telefonía IP.

El desarrollo de la solución con un enfoque centralizado podría servir para la recolección de estadísticas del uso de telefonía IP a nivel nacional, así como el aprovechamiento de las infraestructuras de red existentes. La elección de un esquema no centralizado (distribuido), no sería escalable, ya que este imita a una topología de red de tipo malla completa, esta posee un incremento en enlaces de interconexión a razón de n(n-1)/2 por cada nodo agregado. A su vez, podrían implementarse servicios adicionales como directorio centralizado y la creación de un plan de numeración que de soporte a las necesidades de comunicación e identificación de cada una de estas instituciones. Así como la realización de futuros trabajos relacionados al aprovechamiento de esta nueva plataforma de interconexión.

3.6 Fase de implementación de la plataforma integrada

3.6.1 Aclaratoria Scrum

La presente aclaratoria es válida para ambas fases del desarrollo del proyecto y la elaboración del presente trabajo.

Debido a que solo una persona es integrante del desarrollo del proyecto, algunos de los roles definidos en la metodología SCRUM serán absorbidos o unificados en uno solo. Por lo tanto, el solo integrante en el desarrollo de este proyecto debe cumplir con las siguientes obligaciones:

- Añadir, modificar y eliminar tareas durante el desarrollo del proyecto y la ejecución de los sprints.
- Otorgar los niveles de prioridad a las distintas tareas del backlog o a consideración durante la ejecución del proyecto y la duración de cada sprint.
- Organizar y planificar la ejecución de las tareas durante el desarrollo de cada sprint.
- Identificar y solucionar problemas encontrados durante el desarrollo del proyecto.
- Consultar, sugerir y ayudar en la toma de decisiones en pro del desarrollo del proyecto.

Para el desarrollo del presente trabajo e implantación de la plataforma, la Junta directiva de Tarma consultores asumió el rol de Product Owner. Llevando a cabo reuniones periódicas (Sprint planning y Sprint review) para mostrar su opinión sobre los incrementos generados durante cada Sprint y destacando la importancia de las tareas en el Product Backlog.

3.6.2 Implementación Fase 1 (Integración de la plataforma)

Para la implementación total de este trabajo especial de grado, se procedió a dividir las actividades en dos fases. En la primera fase estaría involucradas todas aquellas actividades relacionadas a Asterisk y a Kamailio y a su integración. Podríamos decir que la primera fase es el despliegue e implantación de la plataforma de llamadas.

Como segunda fase contemplamos todas aquellas actividades relacionadas con el desarrollo del aplicativo web de administración de la plataforma. Este aplicativo administrativo será el encargado del registro y gestión de usuarios que quieran disfrutar de los beneficios ofrecidos por la plataforma integrada, además, desde el aplicativo web se podrán administrar las cuentas SIP que serán agregadas de manera dinámica en el servidor Asterisk y también la asignación de códigos de marcado basados en las propuestas realizadas en el trabajo de seminario y referidos nuevamente en el marco teórico del presente trabajo.

A continuación, iniciaremos la descripción de la primera fase del trabajo.

3.6.3 Análisis de requerimientos y descripción de ambientes de prueba

En esta primera fase de implementación, se realizaron distintas implementaciones de ambientes de prueba, buscando evaluar y verificar las distintas funcionalidades que nos permitieron integrar al servidor SIP Proxy Kamailio y el servidor de llamadas Asterisk, cumpliendo de esta forma con los objetivos para el desarrollo de este trabajo especial de grado.

Siguiendo la metodología Scrum, fue necesario realizar primeras reuniones con el product owner a fin de levantar los puntos importantes, requerimientos y expectativas de la plataforma a implementar. De estas reuniones surgió la planificación de 9 ambientes de prueba, los cuales como se dijo anteriormente, consisten en evaluar funcionalidades de los componentes a integrar. La lista de los ambientes de prueba puede apreciarse en la tabla 3.

Tabla 3
Listado de ambientes de prueba

Ambiente numero	Descripción
1	Instalación y configuración de servidor Asterisk.
2	Instalación y configuración de segundo servidor Asterisk para el establecimiento de enlaces troncales.

3	Configuración e instalación de componentes en servidores Asterisk para implementar la arquitectura en tiempo real Asterisk (ARA)
4	Instalación y configuración de servidor SIP Proxy Kamailio.
5	Integración de Asterisk-Kamailio
6	Kamailio-Asterisk detrás de Firewall y uso de direcciones IP públicas, registro y pruebas de llamada entre usuarios desde usuarios desde el internet hacia usuarios internos o troncales suscritas.
7	Configuración de servidor de llamadas Elastix y central FreePBX con acceso público para la incorporación en la plataforma Kamailio-Asterisk
8	Prueba de configuración y establecimiento de llamadas entre troncales remotas vía plataforma Asterisk Kamailio
9	Pruebas y configuraciones para evaluar la factibilidad de implementación de la transmisión de datos multimedia entre los usuarios sin que estos transiten por Kamailio-Asterisk.

Fuente: Autor.

De esta definición de ambientes de prueba, se destaca la granularidad en la que fueron evaluadas las características y funcionalidades de ambos servidores, desde lo más simple hasta lo más complejo de la arquitectura. Fue utilizado este enfoque, para darle un seguimiento progresivo al desarrollo del proyecto de la mano con SCRUM. Al entender cómo funcionan todas las pequeñas partes que conforman el sistema, podremos tener una mayor comprensión del sistema en general una vez finalizada la integración.

De esta forma entonces se generaron los siguientes diagramas (topologías) basados en los ambientes de prueba:

- La topología definida para el ambiente de prueba número uno puede apreciarse en la figura 15.

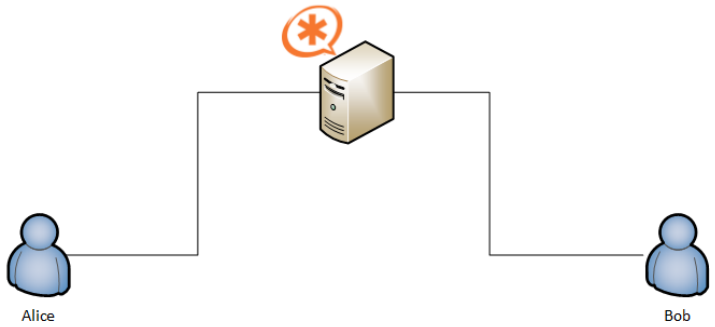

Figura 14***:*** Evaluación de características básicas de servidor Asterisk.
Fuente: Autor

- La topología definida para el ambiente de prueba numero dos puede apreciarse en la figura 16.

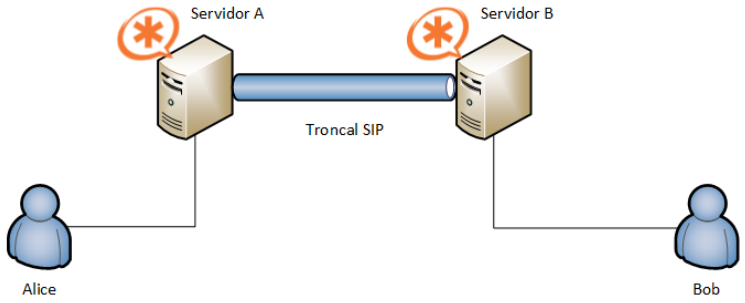

Figura 15***:*** Comunicación vía troncales SIP entre servidores Asterisk.
Fuente: Autor.

- La topología definida para el ambiente de prueba número tres puede apreciarse en la figura 17.

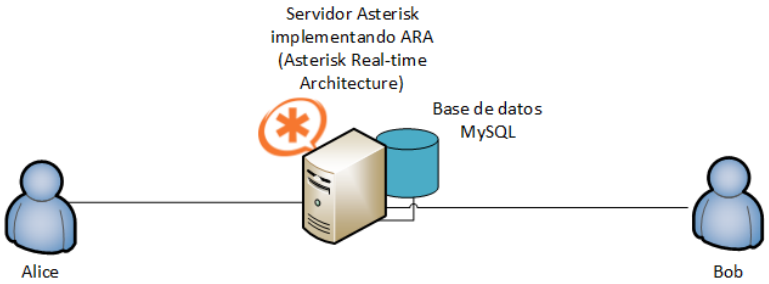

Figura 16***:*** Implantación del Asterisk Real-time Architecture (ARA).
Fuente: Autor.

- La topología definida para el ambiente de prueba número cuatro puede apreciarse en la figura 18.

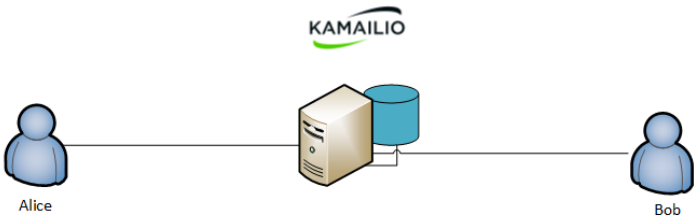

Figura 17: Evaluación de funcionalidades básicas de Kamailio.
Fuente: Autor.

- La topología definida para el ambiente de prueba número cinco puede apreciarse en la figura 19.

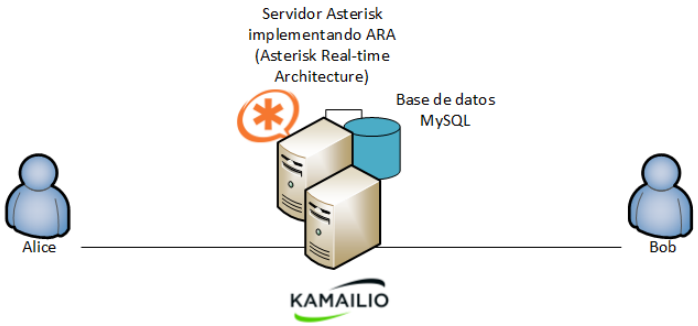

Figura 18***:*** Integración de componentes Kamailio-Asterisk.
Fuente: Autor

- Del ambiente de prueba número 6 se desprenden las siguientes topologías:

  a) Usuario externo en comunicación con usuario detrás del firewall (interno, figura 20).

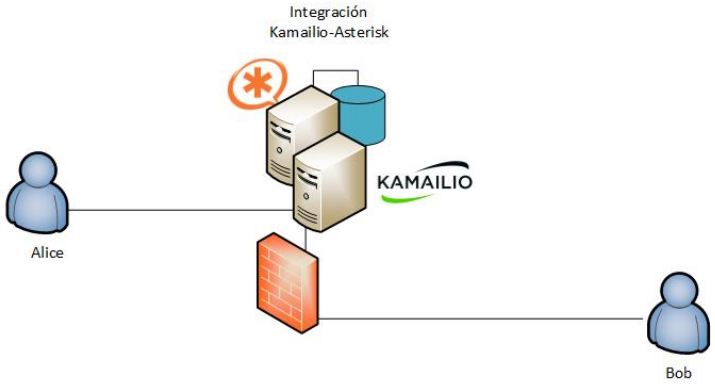

Figura 19*:* Comunicación usuario externo a usuario interno.
Fuente: Autor.

  b) Usuario externo en comunicación con troncal interna (figura 21).

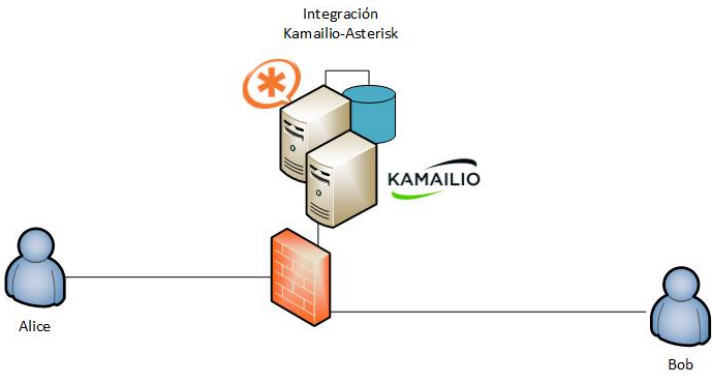

Figura 20: Comunicación usuario externo a troncal interna.
Fuente: Autor.

c) Usuario externo a troncal externa (figura 22).

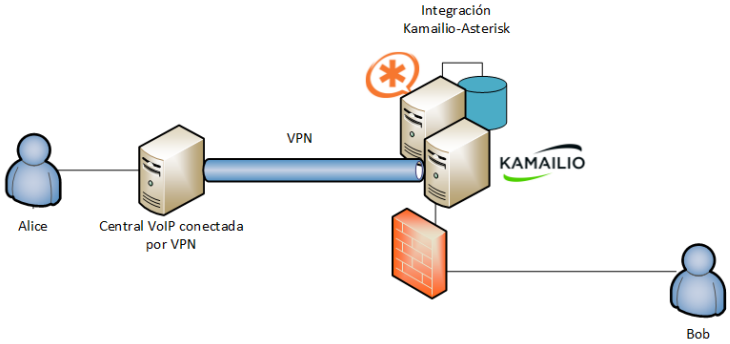

Figura 21***:*** Comunicación Usuario externo hacia troncal externa.
Fuente: Autor

- La topología definida para el ambiente de prueba número ocho puede apreciarse en la figura 23.

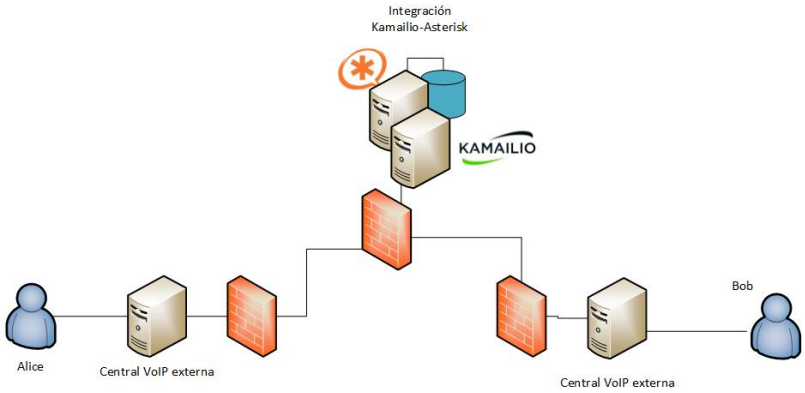

Figura 22***:*** Topología final.
Fuente: Autor.

El ambiente de prueba numero 7 no contiene topología asociada. Y el ambiente de prueba numero 9 mantiene la misma topología del ambiente anterior.

3.6.4 Product backlog

A partir de las especificaciones para el desarrollo de los ambientes de prueba descritos anteriormente, se generó el product backlog para la fase de implementación, teniendo como resultado el listado de actividades que se pueden apreciar en la tabla 4.

Tabla 4
Backlog para la fase de integración.

Sprint ID	ID de tarea	Tarea
1	1	Instalación de servidor Asterisk
	2	Configuración del servidor Asterisk

	3	Configuración del archivo extensions.conf
	4	Configuración del archivo sip.conf
	5	Configuración del archivo asterisk.conf
	6	Configuración del archivo rtp.conf
	7	Configuración del archivo modules.conf
	8	Instalación y configuración de clientes SIP (Softphone)
	9	Pruebas de verificación para ambiente de prueba 1
2	10	Instalación de segundo servidor Asterisk
	11	Configuración de segundo servidor Asterisk
	12	Configuración de archivos sip.conf
	13	Configuración del archivos extensions.conf
	14	Pruebas de verificación para ambiente de prueba 2
3	15	Configurar arquitectura ARA en un servidor Asterisk
	16	Instalación y configuración MySQL
	17	Instalación y configuración de ODBC
	18	Verificación de módulos ODBC activos en Asterisk
	19	Configuración de archivo res_odbc.conf
	20	Configuración de archivo extconfig.conf
	21	Pruebas de verificación para ambiente de prueba 3
4	22	Instalación de servidor SIP Proxy Kamailio
	23	Configurar servidor SIP Proxy Kamailio
	24	Pruebas de verificación para ambiente de prueba 4
5	25	Configurar servidor SIP Proxy Kamailio para la integración con Asterisk
	26	Configuración de usuarios
	27	Pruebas de verificación para ambiente de prueba 5
6	28	Configuración de servidores con direcciones IP públicas y exposición al internet
	29	Configuración de servicios fail2ban e iptables en servidores Kamailio Asterisk públicos
	30	Pruebas de verificación para ambiente de prueba 6 caso 1
	31	Pruebas de verificación para ambiente de prueba 6 caso 2
	32	Pruebas de verificación para ambiente de prueba 6 caso 3
7	33	Configuración de central Elastix en Tarma consultores para exponer servicios VoIP al internet
	34	Configuración de servidor FreePBX en Arkisoft para exponer servicios VoIP hacia la Internet
	35	Configuración de usuarios troncales en los servidores Kamailio Asterisk, Elastix y FreePBX
	36	Pruebas de verificación para ambiente de prueba 9
8	37	Ambiente de prueba adicional: Envió de datos multimedia entre clientes
	38	Pruebas de verificación para ambiente de prueba final

Fuente: Autor.

Para el conjunto de tareas y actividades definidas en la tabla anterior, se realizó la distribución de tareas en Sprints, generando un total de 8 Sprints para el desarrollo de esta fase del presente trabajo. La duración de cada Sprint consta de dos semanas.

3.6.5 Sprint 1

Para el desarrollo del primer Sprint, se contemplaron todas las tareas y actividades necesarias para poder realizar la instalación y configuración de un servidor Asterisk y realizar una llamada entre dos usuarios registrados en este servidor. Para ello fue necesario instalar Asterisk en un equipo virtual con sistema operativo Centos 6.6, realizar las configuraciones de los distintos archivos que requiere Asterisk para su funcionamiento y finalmente realizar la llamada de prueba entre dos usuarios. El listado de las tareas realizadas durante el desarrollo del Sprint 1 pueden apreciarse en la tabla 5.

Tabla 5
Listado de tareas realizadas para el desarrollo del Sprint 1.

Sprint ID	ID de tarea	Tarea
1	1	Instalación de servidor Asterisk
	2	Configuración del servidor Asterisk
	3	Configuración del archivo extensions.conf
	4	Configuración del archivo sip.conf
	5	Configuración del archivo asterisk.conf
	6	Configuración del archivo rtp.conf
	7	Configuración del archivo modules.conf
	8	Instalación y configuración de clientes SIP (Softphone)
	9	Pruebas de verificación para ambiente de prueba 1

Fuente: Autor.

La instalación de Asterisk fue realizada en un ambiente virtualizado utilizando Virtual Box como herramienta de virtualización y se utilizaron las mismas especificaciones para los ambientes virtuales de instalación de servidores Asterisk/Kamailio. El ambiente virtual contó con las siguientes especificaciones:

- Disco duro: 40Gb
- Memoria RAM: 512Mb
- Sistema operativo: Centos 6

Una vez fue instalado el ambiente virtual, se realizó la instalación del servidor Asterisk. Para realizar una instalación de Asterisk ser recomienda fuertemente realizar una actualización de los repositorios del sistema operativo.

La versión instalada de Asterisk es la versión 11.0. Fue necesario descargar el código fuente de esta versión para su posterior instalación utilizando el siguiente comando:

- wget http://downloads.asterisk.org/pub/telephony/asterisk/asterisk-11-current.tar.gz

Una vez descargado, es necesario realizar la descompresión de los archivos de código fuente para Asterisk, para ello empleamos el siguiente comando:

- tar -xzvf asterisk-11-current.tar.gz

Una vez descomprimido, debemos acceder al directorio del código fuente y ejecutar cualquiera de los siguientes dos comandos:

- ./contrib/scripts/install_prerequisits install

- ./contrib/scripts/install_prerequisits install-unpackaged

Estos comandos ejecutan la instalación de componentes requeridos por Asterisk para su instalación y una vez instalados se debe ejecutar la siguiente secuencia de comandos respetando el orden a continuación:

- ./configure
- make all
- make install
- make config

Una vez finalizada la ejecución de los comandos mencionados, solo hizo falta copiar algunas configuraciones de ejemplo para poder trabajar con Asterisk. Los archivos de configuración de muestra utilizados fueron:

- modules.conf.sample
- asterisk.conf.sample
- sip.conf.sample
- extensions.conf.sample

Se inició el servicio de Asterisk con el comando:

- service asterisk start

Para luego, verificar la correcta instalación de Asterisk con el comando

- asterisk -cvv

Con estos pasos, la instalación de Asterisk fue realizada sin inconvenientes.

3.6.5.1 Configuración del servidor Asterisk

Luego de realizar la correcta instalación del servidor Asterisk, procedemos a modificar los siguientes archivos:

- asterisk.conf
- rtp.conf
- modules.conf
- sip.conf
- extensions.conf

3.6.5.2 Modificaciones realizadas al archivo asterisk.conf

El archivo "asterisk.conf" contiene información y parámetros que son aplicables para el servidor en general.

El archivo "asterisk.conf" posee distintas secciones con parámetros relacionados a la descripción de su contexto. Se realizaron las siguientes modificaciones de parámetros dentro del contexto indicado:

En el contexto [options] se modificaron los parámetros los parámetros *run user* y *run group* estableciéndolos en el valor del usuario y del grupo al que pertenece el usuario creado para iniciar el servicio:

- runuser = asterisk

- rungroup = asterisk

El resto de los parámetros fueron dejados con sus valores por defecto.

3.6.5.3 Modificaciones realizadas al archivo rtp.conf

En este archivo se configuraron el rango de inicio y de final de los puertos a elegir para la transmisión de datos multimedia usados por el protocolo RTP. Para ello modificamos los parámetros en la sección de general:

- rtpstart=10000
- rtpend=10100

Este rango se estableció para restringir todo el espectro de puertos abiertos dentro de un firewall restringiendo así la cantidad de puertos a 100.

3.6.5.4 Modificaciones realizadas al archivo modules.conf

En este archivo se configuro solo un único parámetro para la carga automática de los módulos de Asterisk:

- autoload=yes

3.6.5.5 Modificaciones realizadas al archivo sip.conf

Este es uno de los archivos más importantes que fueron configurados durante toda la fase de implementación y para el despliegue de todos los ambientes de prueba. En este archivo de configuración se encuentra todo lo necesario para configurar el comportamiento del protocolo SIP dentro del servidor Asterisk, estos parámetros afectan el comportamiento de este protocolo a nivel general de Asterisk o a nivel de usuario SIP. En este archivo también se definieron los usuarios que se conectan al servidor de llamadas Asterisk. Las configuraciones realizadas a los parámetros de la sección general de este archivo de configuración pueden apreciarse en la tabla 6.

Tabla 6

Parámetros configurados en la sección general.

Nombre del parámetro	Valor
context	public
allowguest	no
udpbindaddr	X.X.X.X:5060
tcpenable	no
transport	udp
srvlookup	yes
disallow	all
allow	ulaw
allow	alaw
allow	gsm
tonezone	ve
alwaysauthreject	yes
directmedia	no

Fuente: Autor.

De la tabla anterior podemos mencionar que el contexto por defecto utilizado por los usuarios es "public", no se permiten llamadas sin autenticación, se especificó la dirección IP y puerto de escucha en el parámetro udpbindaddr, el protocolo UDP es el protocolo seleccionado para el transporte de la señalización SIP, se establecieron los codecs "ulaw", "alaw" y "gsm" como codecs permitidos para la comunicación de los usuarios y el envío directo de los datos multimedia entre usuarios no está permitido haciendo que el servidor Asterisk siempre se mantenga en el medio de la comunicación entre dos usuarios redirigiendo el tráfico multimedia.

Luego de modificar las variables generales de este archivo, procedemos a especificar a los usuarios que se registraran en este ambiente de prueba, Alice y Bob. La configuración dentro de este archivo de configuración para Alice y para Bob se aprecia en la tabla 7.

Tabla 7

Configuración para los usuarios Alice y Bob.

USUARIO SIP	USUARIO SIP
[alice] type=friend username=alice secret=alicesecret host=dynamic allow=all qualify=yes context=prueba	[bob] type=friend username=bob secret=bobsecret host=dynamic allow=all qualify=yes context=prueba

Fuente: Autor.

3.6.5.6 Modificaciones realizadas al archivo extensions.conf

En este archivo de configuración se especificó el plan de marcado de la central telefónica, así como las maneras de localizar a un usuario SIP mediante el uso de una extensión, en donde una extensión es una etiqueta que nos permite asociar un conjunto de instrucciones a ejecutar. Una de estas instrucciones a ejecutar al encontrar coincidencia con una extensión, es la de realizar una llamada a un usuario en particular.

Como la mayoría de los archivos de configuración, el archivo extensions.conf se encuentra estructurado por contextos. Como primer contexto tenemos el contexto general en donde se especifican las variables que afectan a nivel general el plan de marcado. Estas variables para este contexto fueron dejadas sin modificar.

Para este Sprint fue definido el contexto "prueba" y dos extensiones dentro del mismo como se aprecia a continuación:

- [prueba]
- exten => 100,1,Dial(SIP/alice,40)
- exten => 101,1,Dial(SIP/bob,40)

En estas extensiones se podrán enviar llamadas al usuario Alice o al usuario Bob cuando se haga coincidencia con las extensiones 100 y 101 respectivamente.

Una vez configurados estos dos usuarios, fue necesario realizar la instalación y configuración de dos clientes softphone en los clientes para poder realizar las llamadas entre ellos y poder hacer la captura de tráfico y su posterior estudio.

3.6.5.7 Pruebas de verificación para ambiente de prueba 1

Luego de haber instalado y configurado todos los elementos necesarios, se realizó una llamada entre los usuarios Alice y Bob para evaluar las configuraciones realizadas en el servidor de llamadas y las configuraciones realizadas a nivel de usuario SIP en el servidor. Los datos originados por esta llamada, fueron capturados con el analizador de paquetes Wireshark para su estudio.

El objetivo de este estudio del primer ambiente de prueba es poder observar el comportamiento del servidor de llamadas Asterisk en el manejo de la llamada entre estos dos usuarios con la configuración anterior. De esta forma, las configuraciones realizadas evaluadas sirvieron como base para ambientes de pruebas sucesivos y sirvieron como referencias para realizar ajustes en las configuraciones para buscar el comportamiento deseado en la plataforma.

Previo a la realización de la llamada, es necesario en todos los ambientes de prueba, que los usuarios se encuentren registrados en un servidor de llamadas. El proceso de registro por parte de los usuarios de este ambiente de prueba puede apreciarse en la figura 24 en donde se ve el intercambio de mensajes entre el cliente y el servidor de llamadas siguiendo con las especificaciones del RFC 3261. En la figura 25 podemos ver el contenido del mensaje OK que es dirigido desde el servidor al usuario SIP Alice en señal de confirmación. El proceso de registro del usuario Bob es realizado de manera similar.

Figura 23: Diagrama de flujo registro SIP.
Fuente: Autor.

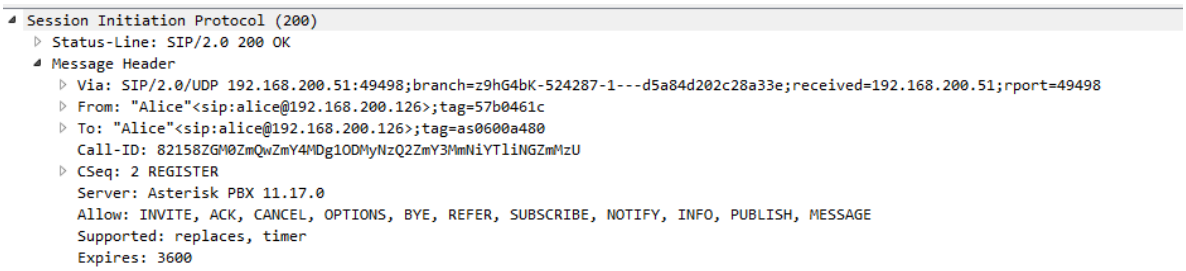

Figura 24***:*** Contenido mensaje OK Asterisk-Alice.
Fuente: Autor.

El registro de ambos usuarios puede apreciarse en el servidor Asterisk al ejecutar el comando "sip show peers", obteniendo como resultado el mostrado en la figura 26.

```
*CLI> sip show peers
Name/username             Host                                  Dyn Forcerport Comedia    ACL Port     Status      Description
alice/alice               192.168.200.51                         D  Auto (No)  No             49498    OK (7 ms)
bob/bob                   192.168.200.62                         D  Auto (No)  No             62942    OK (5 ms)
2 sip peers [Monitored: 2 online, 0 offline Unmonitored: 0 online, 0 offline]
*CLI>
```

Figura 25: Usuarios SIP registrados.
Fuente: Autor.

Luego que ambos usuarios se encontraran registrados y comprobados dentro del servidor Asterisk, se procedió a realizar la llamada entre estos dos usuarios.

El intercambio de mensajes SIP entre los usuarios y el servidor Asterisk, puede apreciarse en la figura 27. Este flujo o intercambio de mensajes SIP que podemos apreciar sigue un flujo normal sin la presencia de mensajes SIP Re-Invite. Esto es debido a la configuración utilizada en este ambiente de prueba al establecer el parámetro "directmedia" en no. Para ambientes donde hay presencia de NAT, se recomienda hacer uso de esta configuración [36].

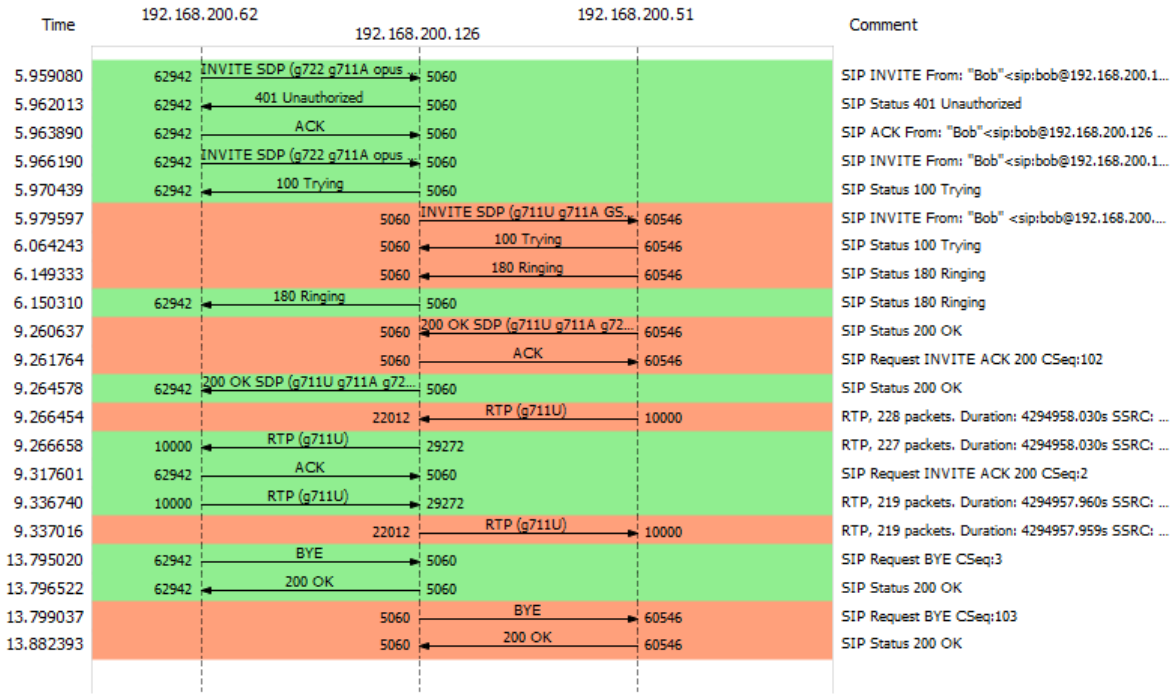

Figura 26*:* Diagrama de flujo de llamada Alice-Bob.
Fuente: Autor.

De la figura anterior podemos observar que el proceso de comunicación se establece en dos etapas. La primera etapa se establece entre el usuario Bob quien es el que inicia la llamada y el servidor Asterisk. Luego, la segunda fase de la comunicación es establecida desde el servidor Asterisk hacia el usuario Alice. Una vez establecida la llamada en ambas secciones de la comunicación, el flujo de datos multimedia es transmitido desde un usuario al servidor, y es el servidor Asterisk es el encargado de retransmitir nuevamente los datos multimedia hacia el usuario destino, actuando como hombre en el medio para la comunicación.

Al finalizar el sprint número uno fue generado un reporte en detalle del funcionamiento y descripción del establecimiento de la llamada para el ambiente de prueba 1. Además, fue evaluado el uso de la configuración por defecto que trae consigo Asterisk en sus archivos de configuración de muestra ayudando a entender aún más el comportamiento para el establecimiento de una llamada entre dos usuarios.

3.6.6 Sprint 2

Para el desarrollo de nuestro segundo Sprint, se contemplaron todas las tareas y actividades relacionadas con la instalación y configuración de un segundo servidor Asterisk para el establecimiento de un enlace troncal entre servidores y realizar una llamada entre distintos dominios de llamada. El listado de las tareas realizadas durante el desarrollo del segundo Sprint puede apreciarse en la tabla 8.

Tabla 8

Listado de tareas realizadas durante el desarrollo del Sprint 2

Sprint ID	ID de tarea	Tarea
2	10	Instalación de segundo servidor Asterisk
	11	Configuración de segundo servidor Asterisk
	12	Configuración de archivos sip.conf
	13	Configuración del archivos extensions.conf
	14	Pruebas de verificación para ambiente de prueba 2

Fuente: Autor.

3.6.6.1 Instalación y configuración de segundo servidor Asterisk

La instalación del segundo servidor Asterisk se realizó repitiendo los pasos de instalación del servidor Asterisk durante el desarrollo del primer Sprint. Luego de verificar la correcta instalación del segundo servidor, fue necesario distribuir un usuario para el servidor 1 y otro para el servidor 2. Esto último siguiendo la topología asociada al ambiente de prueba 2. En este ambiente de prueba fueron utilizados los usuarios 100 y 200, siendo asignados respectivamente al servidor 1 y al servidor Asterisk 2.

Adicionalmente, para el establecimiento del enlace troncal SIP en los servidores, fue necesario la creación de un tercer usuario SIP. Este tercer usuario SIP es el encargado de representar el enlace de tipo troncal entre los servidores. La configuración del usuario SIP troncal puede apreciarse en la tabla 9.

Tabla 9

Configuración de usuario troncal en servidores Asterisk

Configuración usuario troncal en servidor 1	Configuración usuario troncal en servidor 2
[troncal] host=DireccionIPServidor2 username=troncal fromuser=troncal defaultuser=troncal secret=troncalsecret type=friend trunk=yes context=prueba disallow=all allow=gsm qualify=yes deny=0.0.0.0/0.0.0.0 permit=DireccionIPServidor2/255.255.255.255	[troncal] host=DireccionIPServidor1 username=troncal fromuser=troncal defaultuser=troncal secret=troncalsecret type=friend trunk=yes context=prueba disallow=all allow=gsm qualify=yes deny=0.0.0.0/0.0.0.0 permit=DireccionIPServidor1/255.255.255.255

Fuente: Autor.

Fue necesario realizar modificaciones en los archivos extensions.conf de ambos servidores Asterisk a fin de poder dirigir las llamadas a los usuarios respectivos pasando por el enlace troncal. Para ello fueron añadidas las entradas en el archivo extensions.conf como se muestra en la tabla 10.

Tabla 10

Configuraciones para llamar hacia una troncal

Tipo de extensión	Configuración
Extension por patrón de discado	[prueba] exten =>_1XX, 1, NoOP("Se envia la llamada por la Troncal") same => n,Dial(SIP/troncal/${EXTEN},30)

Fuente: Autor.

Fue utilizada una extensión del tipo patrón de discado para redirigir las llamadas por el enlace troncal. De esta forma, todos aquellos números discados que comiencen con el digito 1 coincidirán con esta extensión y la llamada será enviada por la troncal.

3.6.6.2 Pruebas de verificación para ambiente de prueba 2

Luego de haber instalado y configurado todos los elementos necesarios, se realizó una llamada entre los usuarios 100 y 200 para evaluar las configuraciones realizadas en ambos servidores de llamadas. Los datos originados por esta llamada, pueden apreciarse en el diagrama de secuencia de la figura 28. Este diagrama de secuencia corresponde al servidor 1 Asterisk. Como bien se puede apreciar, podemos observar el tráfico proveniente desde el usuario 100 hacia su servidor de registro (.126) este luego redirige el flujo de comunicación al enlace troncal establecido con el servidor 2 (.206). La otra sección de la comunicación sigue un comportamiento similar al observado en la figura 28.

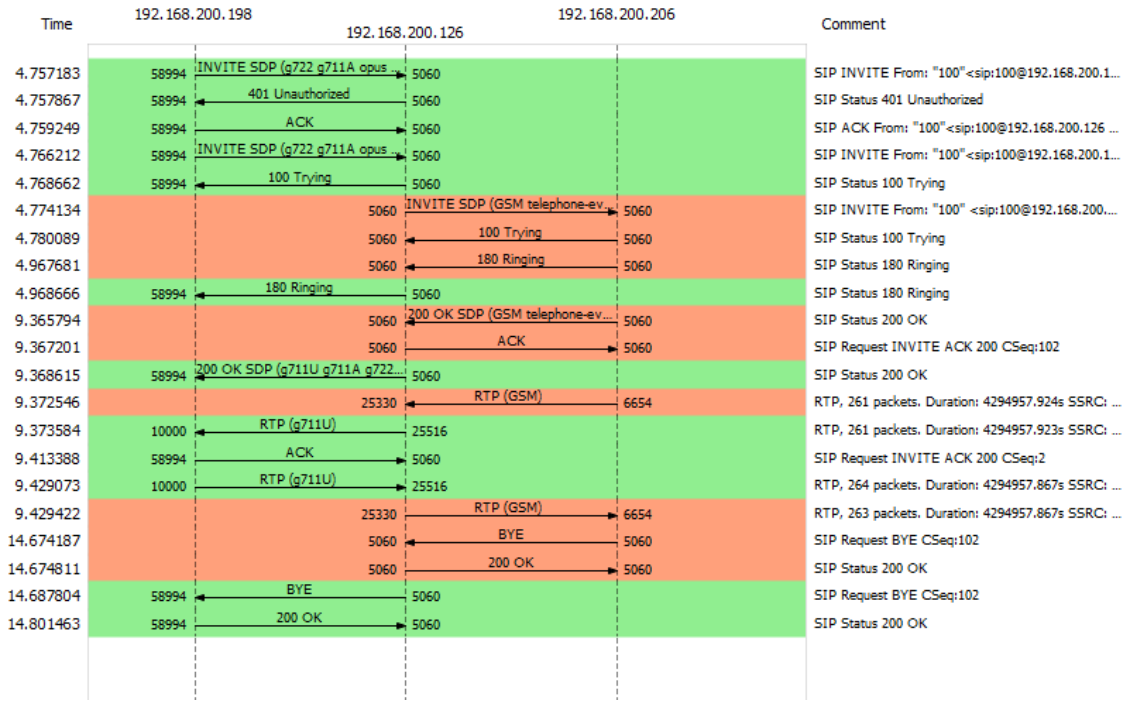

Figura *27:* Diagrama de secuencia entre el usuario 100 y su servidor de registro.

Fuente: Autor.

Ambos flujos de comunicación siguen un flujo normal en el establecimiento y finalización de la comunicación, por lo que la comunicación entre ambos usuarios fue realizada de manera satisfactoria.

3.6.7 Sprint 3

Para el desarrollo de nuestro tercer Sprint, se realizaron todas aquellas tareas o actividades relacionadas a la implementación de la arquitectura en tiempo real ofrecida por Asterisk. Esta arquitectura es conocida como ARA (Asterisk Realtime Architecture) y hace uso de bases de datos para el registro y mantenimiento de información actualizada sobre los usuarios SIP sin la necesidad de reiniciar servicios en este servidor para la incorporación de un nuevo usuario agregado. Existen otras funcionalidades ofrecidas por esta arquitectura, pero estas se encuentran fuera del alcance de este documento. El listado de las tareas realizadas durante el desarrollo de este Sprint, pueden apreciarse en la tabla 11.

Tabla 11
Listado de tareas realizadas durante el desarrollo del Sprint 3

Sprint ID	ID de tarea	Tarea
3	15	Configurar arquitectura ARA en un servidor Asterisk
	16	Instalación y configuración MySQL
	17	Instalación y configuración de ODBC
	18	Verificación de módulos ODBC activos en Asterisk
	19	Configuración de archivo res_odbc.conf
	20	Configuración de archivo extconfig.conf
	21	Pruebas de verificación para ambiente de prueba 3

Fuente: Autor.

3.6.7.1 Configurar arquitectura ARA en un servidor Asterisk

Para la configuración e implementación de la arquitectura en tiempo real ofrecida por Asterisk, fue necesario acondicionar el servidor incluyendo la instalación de una base de datos (MySQL) y el ODBC asociado con esta base de datos.

La instalación del manejador de base de datos MySQL se encuentra en el anexo 1.

Una vez instalado MySQL en nuestro servidor, fue necesario crear una base de datos en la que son almacenados los datos referentes a los usuarios SIP. El nombre de la base de datos es asignado de manera arbitraria, pero a los efectos de este documento, la base de datos fue nombrada "asterisk". La base de datos y el usuario administrador de la base de datos fueron creados con los siguientes comandos:

- CREATE DATABASE asterisk;
- CREATE USER 'asterisk'@'%' IDENTIFIED BY 'contraseña';
- GRANT ALL PRIVILEGES ON asterisk.* TO 'asterisk'@'%';

Luego de la instalación y configuración de la base de datos, fue necesario realizar la instalación de los siguientes componentes, incluidos el ODBC:

- unixODBC
- unixODBC-devel
- libtool-ltdl

- libtool-ltdl-devel
- mysql-connector-odbc

Luego de esta instalación procedemos a consultar y modificar si es necesario los archivos "odbcinst.ini" y "odbc.ini" ambos ubicados en la ruta "/etc/". Estos archivos contienen información relacionada al manejador de base de datos y a la conexión con el ODBC respectivamente. Fue necesario consultarlos y agregar la información presente en la tabla 12 por archivo.

Tabla 12
Información del archive odbcinst.ini

odbcinst.ini	odbc.ini
[MySQL] Description = ODBC for MySQL Driver = /usr/lib/libmyodbc5.so Setup = /usr/lib/libodbcmyS.so Driver64 = /usr/lib64/libmyodbc5.so Setup64 = /usr/lib64/libodbcmyS.so FileUsage = 1	[asterisk-connector] Description = MySQL connection to 'asterisk' database Driver = MySQL Database = asterisk Server = localhost Port = 3306 Socket = /var/lib/mysql/mysql.sock

Fuente: Autor.

Una vez verificados los archivos anteriores, fue necesario ejecutar nuevamente el comando:

- ./configure

Este comando es ejecutado desde el directorio del código fuente de Asterisk, para que este pueda reconocer los componentes de base de datos y ODBC instalados. De esta forma, se activan los módulos en Asterisk relacionados con estos componentes.

Algunos de los módulos que se activan luego de la instalación del ODBC son:

- res_config_mysql
- cdr_odbc
- cdr_adaptive_odbc
- func_db
- func_odbc
- func_realtime
- pbx_realtime
- res_config_odbc
- res_odbc
- res_realtime

Finalmente realizamos nuevamente un "make install".

Posterior a la instalación de los módulos correspondiente, fue necesario indicarle a Asterisk sobre la conexión ODBC, esto es realizado en el archivo "res_odbc.conf" Ubicado en la ruta /etc/asterisk.

El archivo "res_odbc.conf" debe ser creado con la información de la tabla 13.

Tabla 13
Configuración de res_odbc.conf

res_odbc.conf
[asterisk] enabled => yes

```
dsn => asterisk-connector
username => asterisk
password => welcome
pooling => no
limit => 1
pre-connect => yes
```

Fuente: Autor.

En donde [asterisk] es un identificador para la conexión con el ODBC utilizada por Asterisk y los parámetros "username" y "password" son los del usuario de la base de datos para el acceso a los datos de los usuarios SIP.

Luego de configurar este archivo, iniciamos (o reiniciamos Asterisk dependiendo del caso) y accediendo a la consola de Asterisk, verificamos la conexión configurada con el comando:

- odbc show

El resultado de la ejecución del comando anterior debe ser similar al mostrado en la figura 30.

```
odbc show

ODBC DSN Settings
-----------------

  Name:   asterisk
  DSN:    asterisk-connector
    Last connection attempt: 1969-12-31 20:00:00
  Pooled: No
  Connected: Yes
```

Figura 28***:*** Resultado ODBC show.
Fuente: Autor.

Es importante que muestre el "connected: yes". De no mostrar este valor verificamos el usuario y contraseña especificada en el archivo "res_odbc.conf".

Luego de configurar adecuadamente la conexión de Asterisk con la base de datos, debemos especificar los archivos que obtienen información por medio de esta conexión.

Para ello modificamos el archivo "extconfig.conf" ubicado en la ruta "/etc/asterisk". En este archivo de configuración se le indica a Asterisk que archivos tomaran información almacenada en la base de datos formar parte de la configuración del archivo y de que parte de la base de datos tomaran dicha información.

Dentro de este archivo fueron definidos los objetos:

- sippers y sipusers que representan los usuarios SIP y
- sipregs para registros SIP.

La configuración que debe contener el archivo "extconfig.conf" es la especificada en la tabla 14.

Tabla 14

Configuración del archivo extconfig.conf

extconfig.conf
[settings] sipusers => odbc,asterisk,sipusers sippeers => odbc,asterisk,sipusers

sipregs => odbc,asterisk,sipregs

Fuente: Autor.

Luego de realizar esta última configuración, se crearon las tablas en la base de datos asignada para Asterisk. El script SQL para la creación de las tablas y sus distintos atributos, se encuentra en el anexo 2.

Por último es necesario realizar la siguiente modificación de parámetros en el archivo "sip.conf" indicarle que los usuarios SIP serán cargados de manera dinámica. Para ello se configuran los parámetros presentes en la tabla 15.

Tabla 15

Campos activados en el archivo sip.conf

Campo	Valor
rtcachefriends	Yes
rtsavesysname	Yes
rtupdate	Yes
rtautoclear	Yes
ignoreregexpire	Yes

Fuente: Autor.

3.6.7.2 Pruebas de verificación para ambiente de prueba 3

Para la evaluación de las características ofrecidas por ARA, fueron agregados dos usuarios en la base de datos utilizando las instrucciones SQL de la tabla 16.

Tabla 16

Configuración para alicert y para bobrt

Instrucciones para alice	Instrucciones para bob
INSERT INTO sipusers (name, defaultuser, host, type, secret, fromuser, context, qualify, nat, directmedia) VALUES ('alicert', 'alicert', 'dynamic', 'friend', 'alicertsecret', 'alicert', 'prueba', 'yes', 'yes', 'no');	INSERT INTO sipusers (name, defaultuser, host, type, sippasswd, fromuser, context, qualify, nat, directmedia) VALUES ('bobrt', 'bobrt', 'dynamic', 'friend', 'bobrtsecret', 'bobrt', 'prueba', 'yes', 'yes', 'no');
INSERT INTO sipregs(name) VALUES('alicert');	INSERT INTO sipregs(name) VALUES('bobrt');

Fuente: Autor.

Luego de configurados los usuarios en la base de datos, se configuraron los softphone para el registro de ambos usuarios, obteniendo como resultado la carga de los mismos en Asterisk como se puede apreciar en las figuras 31 y 32.

```
    -- Registered SIP 'alicert' at 192.168.200.51:50196

*CLI> sip show peers
Name/username             Host                                  Dyn Forcerport Comedia    ACL Port     Status      Description                      Realtime
alice/alice               (Unspecified)                          D  Auto (No)  Auto (No)       0       OK (7 ms)
alicert/alicert           192.168.200.51                         D  Yes        Yes            50196   OK (9 ms)                                    Cached RT
bob/bob                   (Unspecified)                          D  Auto (No)  Auto (No)       0       UNKNOWN
3 sip peers [Monitored: 2 online, 1 offline Unmonitored: 0 online, 0 offline]
*CLI>
```

Figura 29*:* Registro de Alice RT.
Fuente: Autor

```
    -- Registered SIP 'bobrt' at 192.168.200.42:59454
sip show peers
Name/username             Host                                  Dyn Forcerport Comedia    ACL Port     Status      Description                      Realtime
alice/alice               (Unspecified)                          D  Auto (No)  Auto (No)       0       UNKNOWN
alicert/alicert           192.168.200.51                         D  Yes        Yes            50196   OK (5 ms)                                    Cached RT
bob/bob                   (Unspecified)                          D  Auto (No)  Auto (No)       0       UNKNOWN
bobrt/bobrt               192.168.200.42                         D  Yes        Yes            59454   OK (4 ms)                                    Cached RT
4 sip peers [Monitored: 2 online, 2 offline Unmonitored: 0 online, 0 offline]
*CLI>
```

Figura 30*:* Registro de Bobrt
Fuente: Autor.

De las figuran anteriores se destaca la presencia del valor "cached RT" presente en la columna "Realtime", esto nos indica que los usuarios fueron cargados de manera dinámica (en tiempo real).

Las tablas de registro SIP también fueron poblados como se aprecia en la figura 33 la cual nos muestra el resultado de una consulta realizada.

```
mysql> select name,fullcontact,ipaddr,port,regseconds,defaultuser,useragent,lastms from sipregs;
+---------+-----------------------------------------------------------------------------+---------------+------+------------+-------------+----------------------+
| name    | fullcontact                                                                 | ipaddr        | port | regseconds | defaultuser | useragent            |
+---------+-----------------------------------------------------------------------------+---------------+------+------------+-------------+----------------------+
| alicert | sip:alicert@192.168.200.51:50652^3Brinstance=fda7fb2e119041d2               | 192.168.200.51 | 50652 | 1483737254 | alicert     | X-Lite release 4.9.6 |
| bobrt   | sip:bobrt@192.168.200.42:39180^3Brinstance=bb7aa0381fa8529a^3Btransport=UDP | 192.168.200.42 | 39180 | 1483737258 | bobrt       | Z 3.6.25251 r25476   |
+---------+-----------------------------------------------------------------------------+---------------+------+------------+-------------+----------------------+
```

Figura 31*:* Resultado de consulta a la tabla sipregs.
Fuente: Autor.

Para poder realizar una llamada entre estos dos nuevos usuarios, fue necesario agregar dos entradas adicionales en el archivo extensions.conf. Las entradas adicionales se aprecian en la tabla 17.

Tabla 17
Entradas en el dialplan para alicert y bobrt

Entrada para llamar a Alicert	Entrada para llamar a Bobrt
exten =>102,1,Dial(SIP/alicert,40)	exten =>103,1,Dial(SIP/bobrt,40)

Fuente: Autor.

Finalmente se realizó una llamada entre estos dos usuarios dos usuarios para comprobar las configuraciones realizadas y verificar que fueron poblados los campos necesarios en la base de datos para poder establecer la llamada. Fue realizada una llamada entre ambos usuarios para comprobar la

correcta configuración de los usuarios la cual fue establecida con total normalidad y el intercambio de mensajes siguió un flujo normal.

3.6.8 Sprint 4

Para el desarrollo de nuestro cuarto Sprint, se contemplaron todas las actividades relacionadas a realizar la instalación y configuración del servidor SIP Proxy Kamailio y evaluar las funcionalidades básicas que trae consigo. La lista de tareas realizadas durante el desarrollo de este Sprint puede apreciarse en la tabla 18.

Tabla 18
Listado de tareas realizadas durante el desarrollo del Sprint 4

Sprint ID	ID de tarea	Tarea
4	22	Instalación de servidor SIP Proxy Kamailio
	23	Configurar servidor SIP Proxy Kamailio
	24	Pruebas de verificación para ambiente de prueba 4

Fuente: Autor.

3.6.8.1 Instalación y configuración del servidor Kamailio

Para la instalación del servidor Kamailio, es necesario descargar y actualizar el repositorio de nuestro sistema operativo, para ello utilizamos los comandos:

- wget http://download.opensuse.org/repositories/home:/kamailio:/telephony/CentOS_CentOS-6/home:kamailio:telephony.repo
- mv home:kamailio:telephony.repo /etc/yum.repos.d/
- yum update -y

Luego de actualizar nuestro repositorio, utilizamos el siguiente comando para realizar la instalación de Kamailio:

- yum install -y kamailio kamailio-mysql kamailio-debuginfo kamailio-unixodbc kamailio-utils kamailio-outbound

Kamailio requiere el uso de un manejador de base de datos. Para mantener la uniformidad en el desarrollo de este trabajo, se realizó la instalación de MySQL como sistema manejador de base de datos para uso de Kamailio. Los pasos de instalación de MySQL pueden encontrarse en el anexo 1

Luego de la instalación de estos dos elementos, se procedió a configurar Kamailio, para ello fue necesario modificar el archivo "kamctlrc" ubicado en la ruta "/etc/kamilio/" para descomentar la línea que contenga el parámetro DBENGINE. Este parámetro debe contener el valor "MYSQL" ya que este fue la base de datos que instalamos.

Luego, se ejecutó el siguiente comando en la ruta "/usr/sbin/":

- kamdbctl créate

El comando anterior genera el esquema de base de datos para Kamailio. Se desplegaron una serie de preguntas en donde fue necesario seleccionar "yes" para cada una de las preguntas realizadas.

Finalmente, el proceso Kamailio es iniciado con el comando:

- service kamailio start

3.6.8.2 Pruebas de verificación para ambiente de prueba 4

Para este ambiente de prueba, se utilizó la configuración por defecto que genera Kamailio durante el proceso de instalación. Este archivo de configuración fue necesario modificarlo en el Sprint donde se realizó la integración entre Asterisk y Kamailio.

Fueron agregados dos usuarios a Kamailio para poder realizar la llamada de evaluación. Para ello se utilizaron los comandos de la tabla 19 respectivamente.

Tabla 19
Configuración de usuarios en Kamailio

Comando para agregar a Alicekam	Comando para agregar a Bobkam
kamctl add alicekam alicekamsecret	kamctl add bobkam bobkamsecret

Fuente: Autor.

Luego de agregar los usuarios, fueron configurados los softphone con los datos de los usuarios SIP. El resultado del registro de estos usuarios puede apreciarse en la figura 35 luego de aplicar el comando "kamctl ul show" el cual muestra los datos de cada usuario registrado.

```
[root@Kamailio4-0 ~]# kamctl ul show
database engine 'MYSQL' loaded
Control engine 'FIFO' loaded
entering fifo_cmd ul_dump
Domain:: location table=512 records=2 max_slot=1
        AOR:: alicekam
                Contact:: sip:alicekam@192.168.200.51:52728;rinstance=36208c7f05
c35ddb Q=
                        Expires:: 3186
                        Callid:: 82158Yjg2NzNlNWY4YTg2M2I1MmMyOThjOWQxNWI5ZWM2ZT
k
                        Cseq:: 1
                        User-agent:: X-Lite release 4.9.6 stamp 82158
                        State:: CS_NEW
                        Flags:: 0
                        Cflag:: 0
                        Socket:: udp:192.168.200.77:5060
                        Methods:: 5087
                        Ruid:: uloc-5873eb99-529-1
                        Reg-Id:: 0
                        Last-Keepalive:: 1483994486
                        Last-Modified:: 1483994486
        AOR:: bobkam
                Contact:: sip:bobkam@192.168.200.42:42265;rinstance=1e60794964e5
6b00;transport=TCP Q=
                        Expires:: 3533
                        Callid:: OTMzODNkN2IyZDg5OGQ1MjM5ZTZjYzkzMzc0OTY2MDM.
                        Cseq:: 1
                        User-agent:: Z 3.6.25251 r25476
                        State:: CS_NEW
                        Flags:: 0
                        Cflag:: 0
                        Socket:: tcp:192.168.200.77:5060
                        Methods:: 5087
                        Ruid:: uloc-5873eb99-530-4
                        Reg-Id:: 0
                        Last-Keepalive:: 1483994833
                        Last-Modified:: 1483994833
```

Figura 32***:*** Resultado del comando "kamctl ul show".
Fuente: Autor.

Luego de realizar la última verificación, se procedió a realizar la llamada de evaluación, obteniendo como resultado el diagrama de secuencia que puede apreciarse en la figura 36.

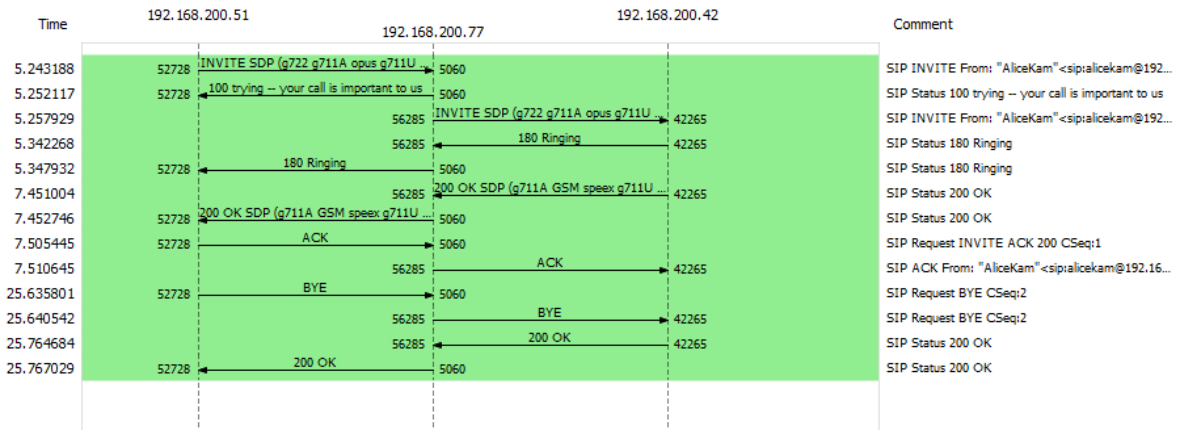

Figura 33***:*** Diagrama de secuencia en llamada usando SIP Proxy Kamailio.
Fuente: Autor.

De la figura 36 podemos decir que el flujo de mensajes SIP, posee un comportamiento normal. Es importante notar que luego del establecimiento inicial de la comunicación no observamos tráfico RTP en la captura de tráfico del servidor Kamailio. Esto nos confirma una de las características presentes en un servidor SIP Proxy mediante la cual este tipo de servidores no actúa como servidor para el manejo de datos multimedia. Para observar el listado de características de Kamailio consulte la sección de este servidor en el marco teórico.

Se realizó la captura de tráfico adicional en uno de los usuarios, dejando ver como el pase de datos multimedia se establece directamente entre los usuarios y no con el servidor Kamailio como sucede generalmente cuando utilizamos Asterisk. El diagrama de secuencia de la llamada desde el usuario Alicekam puede apreciarse en la figura 37.

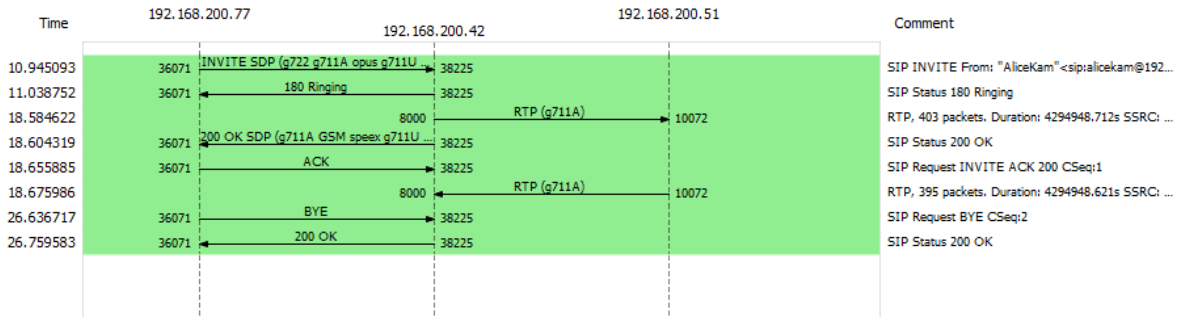

Figura 34: Diagrama de secuencia de la llamada desde Alicekam.
Fuente: Autor.

3.6.9 Sprint 5

Para el desarrollo de nuestro quinto Sprint se contemplaron todas las tareas y actividades relacionadas con la integración entre los servidores Kamailio y Asterisk. En este Sprint, fue necesario configurar nuestro servidor Kamailio y tomando el resultado del Sprint 3 (implementación de ARA) realizar dicha integración. El listado de las tareas realizadas durante el desarrollo de este Sprint puede apreciarse en la tabla 20.

Tabla 20
Lista de tareas realizadas durante el desarrollo del Sprint 5

Sprint ID	ID de tarea	Tarea
5	25	Configurar servidor SIP Proxy Kamailio para la integración con Asterisk
	26	Configuración de usuarios
	27	Pruebas de verificación para ambiente de prueba 5

Fuente: Autor.

Para realizar la integración entre los servidores Kamailio y Asterisk como se mencionó anteriormente, es requisito que se cuente con una instalación del servidor Asterisk empleando ARA. Ya que el servidor Kamailio requiere el acceso a algunos de los componentes presentes en ARA del servidor Asterisk. Se omiten los pasos de la instalación y configuración de ARA en este Sprint ya que los mismos se encuentran bien definidos en el Sprint número 3. Partiendo del resultado obtenido en este Sprint se procede con la configuración del servidor Kamailio para su integración.

3.6.9.1 Configurar servidor SIP Proxy Kamailio para la integración con Asterisk

Para llevar a cabo la integración entre estos dos servidores, es necesario modificar en Kamailio su archivo de configuración, este lo podemos encontrar en la ruta "/etc/kamailio/kamailio.cfg".

Este archivo posee su propia estructura la cual describimos a continuación:

El archivo de configuración de Kamailio (kamailio.cfg) se encuentra dividido en tres grandes secciones: Sección de parámetros globales, sección de configuración de módulos y sección de bloques de enrutamiento.

- **Sección de parámetros globales:** Esta es la primera parte del archivo de configuración. Contiene los parámetros utilizados por el núcleo de Kamailio y parámetros globales personalizados.

- **Sección de configuración de módulos:** Esta es la segunda sección del archivo de configuración. Esta sección contiene las directivas para cargar los módulos y configurar los parámetros relativos a un módulo.

- **Sección de bloques de enrutamiento:** Esta es la última sección del archivo de configuración kamailio.cfg. Típicamente, esta sección es la más grande de todo el archivo de configuración ya que en estos bloques de enrutamiento se contemplan la lógica para redirigir todo el tráfico SIP manejado por Kamailio.

La configuración del archivo "kamailio.cfg" utilizada para lograr la integración se encuentra por completo en el anexo 3. A continuación se describe parte de este archivo de configuración.

El archivo de configuración kamailio.cfg en su sección de parámetros globales, debe tener definidas las directivas resaltadas en el fragmento de código de este archivo figura 38. Estas directivas activan funcionalidades que le permiten a Kamailio interactuar con la base de datos MySQL, realizar autenticación de usuarios, utilizar tablas y mecanismos para que puedan ser localizados usuarios y finalmente habilitar el conjunto de funciones definidas en este archivo que permiten la integración de Kamailio con Asterisk.

```
#!KAMAILIO

#!define WITH_MYSQL
#!define WITH_AUTH
#!define WITH_USRLOCDB
#!define WITH_ASTERISK


#
# Kamailio (OpenSER) SIP Server v4.1 - default configuration script
#     - web: http://www.kamailio.org
```

Figura 35*:* Extracto de código del archivo kamailio.cfg.
Fuente: Archivo kamailio.cfg.

En secciones posteriores de las ilustradas en la figura anterior, podemos encontrar instrucciones del tipo "ifdef" (Figura 39), las cuales evalúan si una directiva ha sido especificada al inicio del archivo de configuración, de ser así, se ejecutarán las instrucciones contenidas en ese bloque de código. De

esta forma es definida la variable DBASTURL cuando la directiva WITH_ASTERISK ha sido definida. Esta variable nos permite definir la URL y los datos de acceso a la base de datos donde se encuentran alojados los datos de usuarios SIP de Asterisk permitiéndonos realizar autenticaciones o registros de usuarios.

```
# *** Value defines - IDs used later in config
#!ifdef WITH_MYSQL
# - database URL - used to connect to database server by modules such
#       as: auth_db, acc, usrloc, a.s.o.
#!ifndef DBURL
#!define DBURL "mysql://kamailiosql:clavesql@localhost/basededatos"
#!endif
#!ifdef WITH_ASTERISK
#!define DBASTURL "mysql://asteriskusuariosql:clavesql@localhost/basededatos"
#!endif
#!endif
#!ifdef WITH_MULTIDOMAIN
# - the value for 'use_domain' parameters
#!define MULTIDOMAIN 1
#!else
#!define MULTIDOMAIN 0
```

Figura 36**:** Definición de valores DBURL y DBASTURL.
Fuente: Archivo kamailio.cfg.

En la figura 40 se muestran resaltados los valores de los parámetros "auto_aliases", "alias", "listen" y "port". El primer parámetro le indica a Kamailio no realizar búsquedas de resolución de nombres, el segundo parámetro define los alias por la cual el servidor Kamailio puede reconocer peticiones dirigidas hacia el servidor. Por ultimo tenemos los parámetros "listen" y "port" en los cuales se define la dirección ip por la cual responderá el servidor Kamailio y el puerto. Se hace uso del subparametro "advertise" para ambientes cuando el Kamailio se coloque detrás de un firewall y exponga sus servicios hacia el internet. El puerto de escucha para Kamailio, está definido con el valor 5060.

```
/* uncomment the next line to disable the auto discovery of local aliases
   based on reverse DNS on IPs (default on) */
auto_aliases=no

/* add local domain aliases */
alias="172.X.X.X:5060"
alias="172.X.X.X:5080"
/* uncomment and configure the following line if you want Kamailio to
   bind on a specific interface/port/proto (default bind on all available) */
listen=udp:172.X.X.X:5060 advertise 66.35.42.243:5060

/* port to listen to
 * - can be specified more than once if needed to listen on many ports */
port=5060

#!ifdef WITH_TLS
enable_tls=yes
#!endif

...
```

Figura 37*:* Valores para los parámetros auto_aliases, alias, listen y port.
Fuente: Autor.

En secciones posteriores de las ilustradas en la figura anterior, podemos encontrar nuevamente instrucciones del tipo "ifdef" (Figura 41), las cuales en este caso definen las variables "asterisk.bindip", "asterisk.bindport", "kamailio.bindip", y "kamailio.bindport" las cuales representan las direcciones y puertos utilizados por los servidores respectivamente.

```
#!ifdef WITH_ASTERISK
asterisk.bindip = "172.18.1.135" desc "Asterisk IP Address"
asterisk.bindport = "5080" desc "Asterisk Port"
kamailio.bindip = "172.18.1.135" desc "Kamailio IP Address"
kamailio.bindport = "5060" desc "Kamailio Port"
#!endif

####### Modules Section ########
...
```

Figura 38**:** Definición de valores para las variables bindip y bindport.
Fuente: Archivo kamailio.cfg.

En la figura 42, se resaltan los módulos que serán cargados una vez iniciado el servidor Kamailio luego de haber especificación las directivas para la integración, así como los demás módulos que son cargados por defecto.

El módulo uac.so provee algunas funcionalidades básicas como el envío de solicitudes SIP, registro con servicios remotos, manipulación del campo "from" de la cabecera SIP y autenticación de clientes SIP. Este último modulo es de gran importancia para la integración entre estos dos servidores [37].

El módulo de "auth_db" especifica que columna es la que contiene el usuario SIP y su contraseña en la base de datos, para ser utilizadas en el proceso de autenticación de peticiones SIP.

Por último, otro modulo importante para realizar la integración es el módulo rr.so. Este agrega información a los mensajes SIP para que estos y sus respuestas puedan circular y sean enviados por medio del servidor SIP Proxy.

```
####### Modules Section ########

# set paths to location of modules (to sources or installation folders)
#!ifdef WITH_SRCPATH
mpath="modules/"
#!else
mpath="/usr/lib64/kamailio/modules/"
#!endif

#!ifdef WITH_MYSQL
loadmodule "db_mysql.so"
#!endif

loadmodule "mi_fifo.so"
loadmodule "kex.so"
loadmodule "corex.so"
loadmodule "tm.so"
loadmodule "tmx.so"
...
loadmodule "cfg_rpc.so"
loadmodule "mi_rpc.so"
loadmodule "acc.so"
loadmodule "outbound.so"

#!ifdef WITH_AUTH
loadmodule "auth.so"
loadmodule "auth_db.so"
#!ifdef WITH_IPAUTH
loadmodule "permissions.so"
#!endif
#!endif

...

#!ifdef WITH_ASTERISK
loadmodule "uac.so"
#!endif


# ----------------- setting module-specific parameters ---------------
...

# ----- rr params -----
# add value to ;lr param to cope with most of the UAs
modparam("rr", "enable_full_lr", 1)
# do not append from tag to the RR (no need for this script)
#!ifdef WITH_ASTERISK
modparam("rr", "append_fromtag", 1)
#!else
modparam("rr", "append_fromtag", 0)
#!endif

...

# ----- auth_db params -----
#!ifdef WITH_AUTH
modparam("auth_db", "calculate_ha1", yes)
modparam("auth_db", "load_credentials", "")


#!ifdef WITH_ASTERISK
modparam("auth_db", "user_column", "name")
modparam("auth_db", "password_column", "sippasswd")
modparam("auth_db", "db_url", DBASTURL)
modparam("auth_db", "version_table", 0)
#!else
modparam("auth_db", "db_url", DBURL)
modparam("auth_db", "password_column", "password")
modparam("auth_db", "use_domain", MULTIDOMAIN)
#!endif

...
```

Figura 39***:*** Configuración de algunos módulos para la integración.
Fuente: Archivo kamailio.cfg.

Finalmente podemos ver en la figura 43 las secciones (resaltadas) que nos fueron de utilidad para realizar la integración. Esta sección del archivo de configuración es la sección de bloques de rutas.

La sección de rutas suele dividirse en 3 secciones más, la primera incluye la ruta principal que procesará toda petición SIP y que tendrá llamadas a las rutas secundarias. La segunda está compuesta por rutas secundarias que son referenciadas desde la ruta principal. La última sección esta compuestas por rutas que sirven para el control frente a posibles fallos producidos al no procesarse de la forma esperadas las anteriores rutinas.

De las rutas resaltadas podemos mencionar:

- La ruta "REGISTRAR" es la encargada de procesar los mensajes SIP de tipo registro. Esta ruta ante la directiva de Asterisk definida, hace envió de estos mensajes SIP a la subruta "REGFWD", en donde esta última reenvía todos estos mensajes de tipo registro hacia el servidor Asterisk. Los mensajes de tipo registro son modificados añadiendo información de contacto del servidor Kamailio para que el servidor Asterisk pueda enviarle las solicitudes a Kamailio y sea este el encargado de procesarlas y enviarlas al usuario correspondiente.
- En la ruta "AUTH" se procesan todas las autenticaciones de los usuarios. Excepto para las peticiones provenientes de Asterisk.
- En la ruta "Location", se verifica el origen de los mensajes para poder redirigirlos adecuadamente. Si son de tipo INVITE son redirigidos al servidor Asterisk, si no serán procesados por Kamailio.
- La ruta "FROMASTERISK" se utiliza como verificación para comprobar si la petición ha sido originada desde Asterisk.

```
# Handle SIP registrations
route[REGISTRAR] {
        if (is_method("REGISTER"))
        {
                if(isflagset(FLT_NATS))
                {
                        setbflag(FLB_NATB);
                        # uncomment next line to do SIP NAT pinging
                        ## setbflag(FLB_NATSIPPING);
                }
                if (!save("location"))
                        sl_reply_error();


        #!ifdef WITH_ASTERISK
        route(REGFWD);
        #!endif
                exit;


        }
}
```

```
# USER location service
route[LOCATION] {

        ...

#!ifdef WITH_ASTERISK
        if(is_method("INVITE") && (!route(FROMASTERISK))) {
                # if new call from out there - send to Asterisk
                # - non-INVITE request are routed directly by Kamailio
                # - traffic from Asterisk is routed also directy by Kamailio
                route(TOASTERISK);
                exit;
        }
#!endif




        ...

        route(RELAY);
        exit;
}
```

```
# Authentication route
route[AUTH] {
#!ifdef WITH_AUTH
#!ifdef WITH_ASTERISK
        # do not auth traffic from Asterisk - trusted!
        if(route(FROMASTERISK))
                return;
#!endif

#!ifdef WITH_IPAUTH
        if((!is_method("REGISTER")) && allow_source_address())
        {
                # source IP allowed
                return;
        }
#!endif

        if (is_method("REGISTER") || from_uri==myself)
        {

#!ifdef WITH_ASTERISK
                if (!auth_check("$fd", "sipusers", "1")) {
#!else
                # authenticate requests
                if (!auth_check("$fd", "subscriber", "1")) {
#!endif
                        auth_challenge("$fd", "0");
                        exit;
                }
                # user authenticated - remove auth header
                if(!is_method("REGISTER|PUBLISH"))
                        consume_credentials();
        }
        # if caller is not local subscriber, then check if it calls
        # a local destination, otherwise deny, not an open relay here
        if (from_uri!=myself && uri!=myself)
        {
                sl_send_reply("403","Not relaying");
                exit;
        }

#!endif
        return;
}
```

```
...

#!ifdef WITH_ASTERISK
# Test if coming from Asterisk
route[FROMASTERISK] {
        if($si==$sel(cfg_get.asterisk.bindip)
                        && $sp==$sel(cfg_get.asterisk.bindport))
                return 1;
        return -1;
}

# Send to Asterisk
route[TOASTERISK] {
        $du = "sip:" + $sel(cfg_get.asterisk.bindip) + ":"
                        + $sel(cfg_get.asterisk.bindport);
        route(RELAY);
        exit;
}

# Forward REGISTER to Asterisk
route[REGFWD] {
        if(!is_method("REGISTER"))
        {
                return;
        }
        $var(rip) = $sel(cfg_get.asterisk.bindip);
        $uac_req(method)="REGISTER";
        $uac_req(ruri)="sip:" + $var(rip) + ":" + $sel(cfg_get.asterisk.bindport);
        $uac_req(furi)="sip:" + $au + "@" + $var(rip);
        $uac_req(turi)="sip:" + $au + "@" + $var(rip);
        $uac_req(hdrs)="Contact: <sip:" + $au + "@"
                                + $sel(cfg_get.kamailio.bindip)
                                + ":" + $sel(cfg_get.kamailio.bindport) + ">\r\n";
        if($sel(contact.expires) != $null)
                $uac_req(hdrs)= $uac_req(hdrs) + "Expires: " + $sel(contact.expires) + "\r\n";
        else
                $uac_req(hdrs)= $uac_req(hdrs) + "Expires: " + $hdr(Expires) + "\r\n";
        uac_req_send();
}
#!endif
```

***Figura* 40: Extracto de código de la sección de rutas del archivo kamailio.cfg.**
Fuente: Archivo kamailio.cfg.

3.6.9.2 Pruebas de verificación para ambiente de prueba 5

Luego de realizar la integración, fueron agregados dos usuarios para realizar la verificación de las funcionalidades ofrecidas por la plataforma integrada y las configuraciones realizadas. Los usuarios SIP agregados son Aliceka y Bobka.

Los usuarios fueron agregados usando sentencias SQL. Las sentencias utilizadas se muestran en la tabla 21 para los usuarios respectivamente.

Tabla 21
Sentencias SQL para agregar a los usuarios Aliceka y Bobka

Sentencia SQL para agregar usuario Aliceka	Sentencia SQL para agregar usuario Bobka
INSERT INTO sipusers (name, defaultuser, host, type, sippasswd, fromuser, context, qualify, nat, directmedia) VALUES ('aliceka', 'alicert', 'dynamic', 'friend', 'alicekasecret', 'aliceka', 'prueba', 'yes', 'yes', 'no');	INSERT INTO sipusers (name, defaultuser, host, type, sippasswd, fromuser, context, qualify, nat, directmedia) VALUES ('bobka', 'bobka', 'dynamic', 'friend', 'bobkasecret', 'bobka', 'prueba', 'yes', 'yes', 'no');
INSERT INTO sipregs(name) VALUES('aliceka');	INSERT INTO sipregs(name) VALUES('bobka');

Fuente: Autor.

Podemos notar como se hizo uso del campo de la base de datos "sippasswd" en vez del campo "secret" usado en el sprint 3. Esto es debido a la especificación efectuada en el módulo "auth_db".

Luego que los usuarios son agregados se procede a registrarlos mediante la configuración de los softphone del lado del usuario. Desde la consola de Asterisk podemos apreciar el siguiente mensaje de registro (ver figura 44).

```
tarmaipbx*CLI>
    -- Registered SIP 'bobka' at 172.18.1.135:5060
       > Saved useragent "kamailio (4.1.9 (x86 64/linux))" for peer bobka

   -- Registered SIP 'aliceka' at 172.18.1.135:5060
      > Saved useragent "kamailio (4.1.9 (x86 64/linux))" for peer aliceka
   -- Registered SIP 'aliceka' at 172.18.1.135:5060
tarmaipbx*CLI>
```

Figura 41***:*** Registro de usuarios desde consola Asterisk.
Fuente: Autor.

Si bien podemos notar que, a pesar de haber configurado los usuarios en el softphone, la imagen muestra que el agente de registro es Kamailio. Ejecutamos un "*sip show peers*" para una vista un poco más detallada de los usuarios (figura 45).

```
tarmaipbx*CLI> sip show peers
Name/username             Host                                    Dyn Forcerport Comedia    ACL Port     Status      Description                      Realtime
aliceka/aliceka,          172.18.1.135                             D  Yes        Yes            5060     OK (96 ms)                                   Cached RT
bobka/bobka               172.18.1.135                             D  Yes        Yes            5060     OK (147 ms)                                  Cached RT
10 sip peers [Monitored: 6 online, 3 offline Unmonitored: 0 online, 1 offline]
tarmaipbx*CLI>
```

Figura 42***:*** Usuarios registrados con Kamailio.
Fuente: Autor.

De las dos figuras anteriores, es importante resaltar, que a pesar que los usuarios se están registrando desde los softphone configurados para los usuarios, la dirección IP con la que se están registrando corresponde a la del servidor Kamailio. Esto es debido a que Kamailio se encarga del proceso de localización y del registro de los usuarios. Esto a su vez coincide con lo establecido en el archivo de configuración kamailio.cfg en la sección de rutas para la ruta REGFWD.

Podemos verificar que es Kamailio quien lleva el registro de los usuarios haciendo una consulta a la base de datos de "sipregs". El resultado de esta consulta lo podemos apreciar en la figura 46.

```
mysql> select * from sipregs where name="bobka" or name="aliceka";
+----+---------+-------------------------------+-------------+------+----------+----------+------------+------------+----------------------+
| id | name    | fullcontact                   | ipaddr      | port | username | regserver | regseconds | defaultuser | useragent            |
+----+---------+-------------------------------+-------------+------+----------+----------+------------+------------+----------------------+
| 25 | aliceka | sip:aliceka@172.18.1.135:5060 | 172.18.1.135 | 5060 |          |          | 1484336794 | alicert     | kamailio (4.1.9 (x86 |
| 26 | bobka   | sip:bobka@172.18.1.135:5060   | 172.18.1.135 | 5060 |          |          | 1484336619 | bobka       | kamailio (4.1.9 (x86 |
+----+---------+-------------------------------+-------------+------+----------+----------+------------+------------+----------------------+
```

Figura 43***:*** Consulta a la tabla "sipregs".
Fuente: Autor.

De la figura anterior, podemos observar en el campo "agent", que el agente para ambos usuarios es el servidor Kamailio esto nos indica que todas las solicitudes pasaran a Kamailio y no directamente a los usuarios sirviendo Kamailio como servidor Proxy.

Para saber la locación de los usuarios es necesario ejecutar el comando "kamctl ul show" en el servidor Kamailio (figura 47). Aquí encontraremos la información detallada de la ubicación de cada usuario registrado. De esta forma Kamailio al recibir los mensajes SIP, posee información hacia donde redirigirlos.

```
[root@tarmaipbx ~]# kamctl ul show
Domain:: location table=512 records=2 max_slot=1
        AOR:: aliceka
                Contact:: sip:aliceka@192.168.200.62:59305;rinstance=edc0a352a812c654;transport=UDP Q=
                        Expires:: 197
                        Callid:: ClmqlB5ceR5a13mFavDUgA..
                        Cseq:: 2
                        User-agent:: Z 3.9.32144 r32121
                        State:: CS_SYNC
                        Flags:: 0
                        Cflag:: 0
                        Socket:: udp:172.18.1.135:5060
                        Methods:: 5087
                        Ruid:: uloc-5818f9a6-5b9-12
                        Reg-Id:: 0
                        Last-Keepalive:: 1484331485
                        Last-Modified:: 1484331485
```

```
AOR:: bobka
        Contact:: sip:bobka@192.168.200.51:60906;rinstance=62f54946df604e99 Q=
                Expires:: 1731
                Callid:: 82158YTRlZDA1YmEwZGQ0ZjQ0ZDY0ZWE2NTIyYjZmMGFkNTc
                Cseq:: 2
                User-agent:: X-Lite release 4.9.6 stamp 82158
                State:: CS_SYNC
                Flags:: 0
                Cflag:: 0
                Socket:: udp:172.18.1.135:5060
                Methods:: 5087
                Ruid:: uloc-5818f9a6-5b8-41
                Reg-Id:: 0
                Last-Keepalive:: 1484333019
                Last-Modified:: 1484333019
```

Figura 44***:*** Resultado del comando "kamctl ul show".
Fuente: Autor.

Una vez verificado el correcto registro para ambos usuarios se realizó una llamada para verificar el correcto funcionamiento de la plataforma. El diagrama de secuencia para la llamada realizada puede apreciarse en la figura 48.

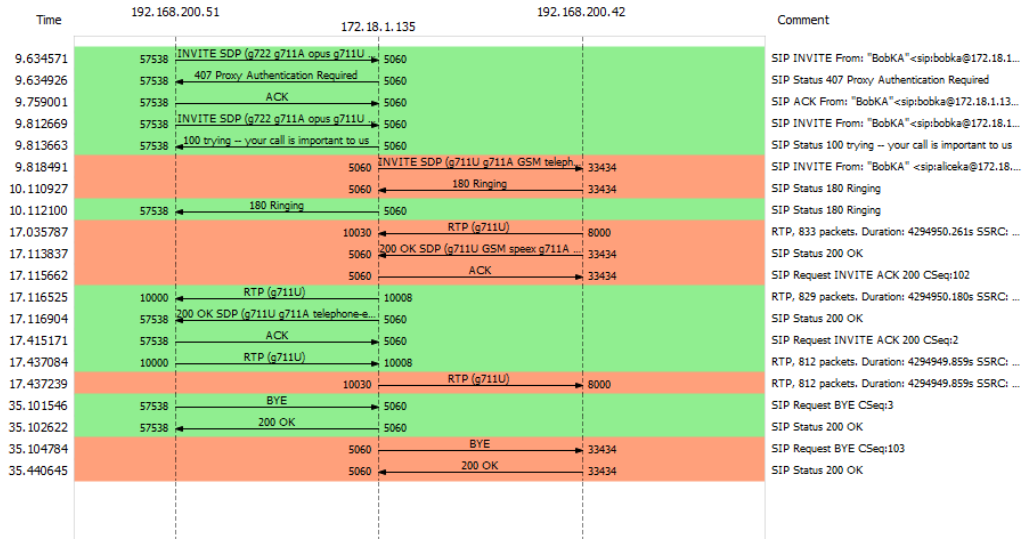

Figura 45***:*** Diagrama de secuencia para la llamada entre los usuarios Aliceka y Bobka.
Fuente: Autor.

De lo anterior se puede apreciar un flujo de mensajes normal para una llamada SIP.

De manera de confirmación para las configuraciones realizadas, se contrastan los dos mensajes invites, el proveniente desde Bobka hacia el servidor y el enviado desde el servidor hacia Aliceka. En el primer invite vemos como este carece del "record-route", ya que esta información es agregada para los paquetes que salen del servidor Kamailio (figura 49). En la figura 50 también se puede apreciar el uso de los parámetros globales establecidos en el archivo de configuración, estos incluyen la dirección IP y el puerto utilizado por Asterisk (campo "From").

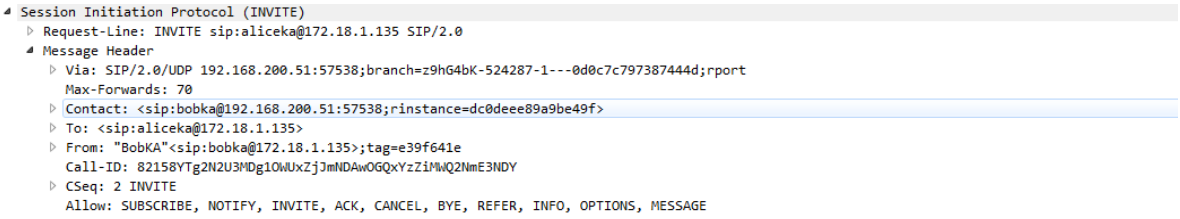

Figura *46:* Mensaje SIP INVITE enviado desde BobKA hacia el servidor Asterisk Kamailio.
Fuente: Autor.

Figura *47:* Mensaje SIP INVITE enviado desde el servidor KAMAILIO ASTERISK a AliceKAM.
Fuente: Autor.

De esta forma se logró realizar la integración de los servidores de manera exitosa al finalizar este Sprint. Fueron realizadas pruebas adicionales para identificar los campos que deben ser necesariamente poblados en la base de datos para poder establecer una comunicación entre usuarios haciendo uso de la plataforma integrada. Estas pruebas arrojaron como resultado la instrucción usada para agregar usuarios en este Sprint. Adicionalmente fueron realizadas modificaciones en el script SQL en el orden en que son especificados los campos para que funcionalidades como deshabilitar codecs y luego activar solo los necesarios pudiesen ser implementadas.

3.6.10 Sprint 6

Para el desarrollo de nuestro sexto Sprint se contemplaron todas aquellas tareas relacionadas con la configuración de la plataforma integrada para su exposición al internet. Entre estas tareas encontramos la configuración del servidor Asterisk y del servidor Kamailio para aceptar solicitudes de redes externas, así como la configuración del servicio IPTables para mitigar posibles brechas de seguridad. Luego de realizar estas configuraciones, fueron evaluados tres escenarios de comunicación entre usuarios. El listado de las tareas realizadas durante el desarrollo del Sprint 6 se pueden apreciar en la tabla 22.

Tabla 22
Listado de actividades realizadas durante el desarrollo del Sprint 6

Sprint ID	ID de tarea	Tarea
6	28	Configuración de servidores con direcciones IP públicas y exposición al internet
	29	Configuración de servicio iptables en servidor Kamailio Asterisk público

	30	Pruebas de verificación para ambiente de prueba 6 caso 1
	31	Pruebas de verificación para ambiente de prueba 6 caso 2
	32	Pruebas de verificación para ambiente de prueba 6 caso 3

Fuente: Autor.

3.6.10.1 Configuración de servidores con direcciones IP públicas y exposición al internet

Para la exposición de los servidores Asterisk y Kamailio fue necesario modificar algunos de los parámetros en los archivos de configuración de ambos servidores.

Las modificaciones en el servidor Asterisk incluyen la modificación de los parámetros "outboundproxy", "domain", "localnet", "externip", "nat" y "media_address" dentro del archivo "sip.conf".

Dentro de este archivo de configuración realizamos las siguientes modificaciones:

- outboundproxy=172.18.1.135:5060

Este parámetro fue configurado para indicarle la dirección IP del servidor Kamailio o de su SIP Proxy. En nuestro caso, la dirección IP del servidor Kamailio y la del servidor Asterisk son las mismas. Los servicios de Asterisk y Kamailio son diferenciados por el puerto utilizado siendo el 5060 utilizado por Kamailio y el 5080 utilizado por Asterisk.

- domain=172.18.1.135

En este parámetro se especifica el dominio local del servidor. Este parámetro también es usado como Realm del servidor.

- localnet=10.0.0.0/255.0.0.0
  localnet=172.18.0.0/255.255.0.0
  localnet=192.168.200.0/255.255.255.0

El parámetro "localnet" puede ser definido más de una vez para especificar las redes locales. Esto es de utilidad cuando se trabaja con redes que hacen uso de NAT y diferenciar las solicitudes provenientes desde una red local o una dirección externa.

- externip=66.35.42.243

En este parámetro se especificó la dirección IP externa que es utilizada para identificar al servidor Asterisk y que permite que el servidor Asterisk sea visible desde Internet.

- nat=auto_force_rport,auto_comedia

El parámetro "rport" permite que la solicitud de un cliente hacia Asterisk, sea respondida hacia la misma dirección IP y puerto origen desde el cual la solicitud es originada (Capa de red). En vez de utilizar los parámetros suministrados mediante el campo VIA de la cabecera SIP. Esto es importante cuando un cliente sabe que se encuentra detrás de NAT.

El parámetro "force_rport" indica a Asterisk que debe enviar el tráfico RTP hacia el mismo puerto donde se origina el tráfico RTP del lado del cliente SIP. [38]

- media_address = 66.35.42.243

Por último, en este parámetro, fue especificada la dirección hacia donde serán enviados los datos multimedia., finalizando así la configuración del lado de Asterisk.

Para la configuración de los parámetros necesarios para la exposición del servidor Kamailio hacia internet, debemos acceder al archivo de configuración "kamailio.cfg". En este archivo fueron modificados los parámetros "alias" y "listen".

- alias="172.18.1.135:5060"
  alias="172.18.1.135:5080"
  alias="66.35.42.243:5060"
  alias="66.35.42.243:5080"

Para el parámetro alias, debemos agregar todas las direcciones IP locales y publicas por las cuales el servicio es identificado. Además, debemos agregar un alias por cada socket ya que Kamailio recibe mensajes SIP con direcciones que también son destinadas al Asterisk.

- listen=udp:172.18.1.135:5060 advertise 66.35.42.243:5060

El ultimo parámetro a configurar es el listen, al cual fue agregado el sub parámetro "advertise". Este sub parámetro, representa la dirección utilizada en el campo VIA de la cabecera SIP que es utilizada por módulos como RR [39].

Luego de realizar las configuraciones para la exposición de servicios hacia internet, se configuró el servicio de IPTables en el servidor donde reside la plataforma integrada.

3.6.10.2 Configuración de servicio iptables en servidores Kamailio y Asterisk públicos

Como bien vimos en el capítulo del marco teórico, Iptables nos Brinda una manera de configurar el filtrado de paquetes en nuestro sistema (herramienta firewall). Generalmente iptables ya viene instalado por defecto en muchas de las distribuciones Linux [40].

Para el correcto funcionamiento de iptables en nuestro servidor, se configuró para permitir el paso de datos VoIP, esto es, permitiendo el tráfico de señalización (SIP) y el tráfico multimedia (RTP).

Para ello se utilizaron los siguientes comandos que pueden apreciarse en la tabla 23.

Tabla 23
Comandos IPtables

Orden	Comando IPTables
1	iptables -P INPUT ACCEPT
2	iptables -P FORWARD ACCEPT
3	iptables -P OUTPUT ACCEPT
4	iptables -A INPUT -s x.x.x.x/xx -j ACCEPT
5	iptables -A INPUT -i lo -j ACCEPT
6	iptables -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT

7	iptables -A INPUT -i eth0 -p udp -m udp --dport 5060 -j ACCEPT
8	iptables -A INPUT -i eth0 -p udp -m udp --dport 10000:10100 -j ACCEPT
9	iptables -A INPUT -j DROP
10	iptables -A OUTPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
11	iptables -A OUTPUT -d x.x.x.x/xx -j ACCEPT
12	iptables -A OUTPUT -i eth0 -p udp -m udp --dport 5060 -j ACCEPT
13	iptables -A OUTPUT -i eth0 -p udp -m udp --dport 10000:10100 -j ACCEPT

Fuente: Autor.

De la anterior tabla podemos mencionar:

- En los primeros tres comandos se establecen las reglas por defecto para las cadenas forward, input y output a permitir. Posterior a esto se definen las reglas para permitir todo el tráfico proveniente de las redes internas, además fueron agregadas reglas que deniegan todo el tráfico proveniente de una red o sub red si los paquetes no coinciden con reglas anteriores.
- El comando numero 5 permite todo el tráfico proveniente desde la interfaz de "loopback".
- En el comando 6 permite todo el tráfico que coincida con los estados de "Established" y "related".
- Los comandos 7 y 8 nos permiten que el tráfico de señalización SIP utilizando el puerto 5060 pueda ingresar sin inconveniente.
- El comando numero 8 especifica el rango de puertos para la transmisión de datos multimedia de la llamada. El rango de puertos utilizados en este comando es el definido en el archivo de configuración "rtp.conf". Luego de esta regla se deniega todo el restante tráfico entrante.
- Luego fueron definidos los comandos para la cadena output que permiten el tráfico de salida si coincide con los estados "established" y "related".
- No se aplicaron restricciones a redes de confianza.
- Se permitió el tráfico del protocolo NTP.

Luego de verificar que las reglas han sido añadidas a nuestro servidor con los comandos iptables fue necesario hacer que estas reglas fueran persistentes. Para ello se utilizó el comando

- Service iptables save

Luego de la configuración realizada para IPTables se ejecutaron los tres escenarios en los que se evaluaron las distintas maneras de comunicación utilizando los servicios expuestos al Internet.

3.6.10.3 Pruebas de verificación para ambiente de prueba 6 caso 1

Como primer escenario, se verifico el correcto funcionamiento en una comunicación entre un usuario registrado externamente y uno registrado internamente. Para ello fue realizada una llamada entre estos dos usuarios. La topología utilizada para este escenario se aprecia en la figura 20 de la sección de análisis de requerimientos.

Este escenario se estudió la viabilidad de realizar llamadas con usuarios registrados desde una dirección IP publica, hacia usuarios registrados dentro del dominio local del servidor. Para ello fueron registrados de manera exitosa dos usuarios (Aliceka y Bobka).

La captura de los datos de la llamada realizada, fue realizada en tres secciones perminitiendo ver el flujo completo de la transmision de datos. Como primera captura podemos ver en la figura 52 el diagrama de secuencia entre el usuario Bobka y la plataforma integrada. De esta imagen podemos apreciar como la comunicación es establecida con la interfaz de la plataforma integrada expuesta al internet. El flujo de datos sigue un comportamiento normal para el establecimiento y finalizacion de una comunicación SIP.

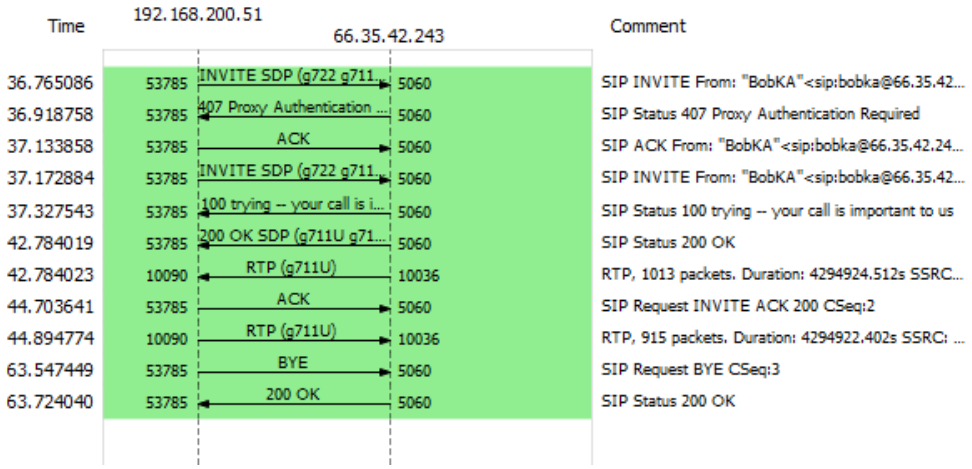

Figura 48***:*** Llamada desde bobka a aliceka.
Fuente: Autor.

La segunda captura de datos fue realizada en el usuario Aliceka. El diagrama de secuencia correspondiente a la captura de datos de la llamada podemos apreciarlo en la figura 53. De esta figura podemos mencionar que la señalizacion SIP posee un comportamiento normal de una comunicación SIP entre dos usuarios, para nuestro caso, la comunicación es establecida entre la plataforma integrada y el usuario Aliceka, haciendo uso de las direcciones internas de la plataforma. Se puede apreciar en la imagen como el flujo RTP de la comunicación, es enviado a la direccion publica de la plataforma integrada. Esto es debido al parametro configurado "media_address" el cual sobre escribe la direccion propuesta en el mensaje SDP de establecimiento de la comunicación.

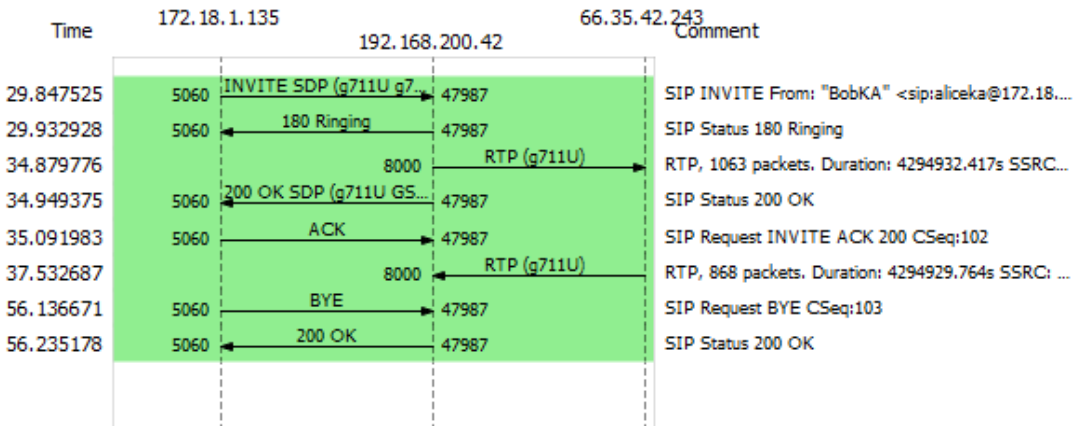

Figura 49***:*** Recepción de la llamada generada por bobka.
Fuente: Autor.

La manera en que esta comunicación es realizada, garantiza que en escenarios mixtos como este y otros escenarios descritos en el presente trabajo puedan realizar sin inconveniente la comunicación entre usuarios.

Finalmente se puede apreciar la comunicación establecida desde el punto de vista de la plataforma integrada en nuestra tercera captura realizada. El diagrama de secuencia de la comunicación entre la plataforma integrada y los usuarios puede apreciarse en la figura 54.

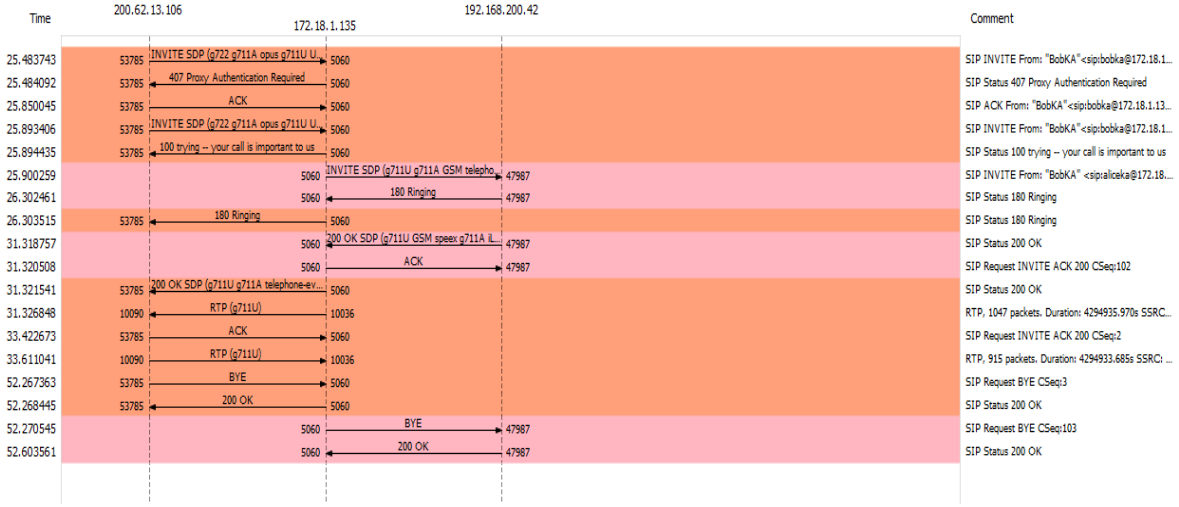

Figura 50***:*** Intercambio de mensajes entre el usuario bobka (externo) y alicka (interno).
Fuente: Autor.

3.6.10.4 Pruebas de verificación para ambiente de prueba 6 caso 2

Como segundo escenario, se verifico el correcto funcionamiento en una comunicación entre dos usuarios registrados externamente a la plataforma integrada. Para ello fue realizada una llamada entre estos dos usuarios. La topología utilizada para este escenario se aprecia en la figura 21 de la sección de análisis de requerimientos.

En este escenario se estudió la viabilidad de realizar llamadas entre usuarios individuales que se suscriben a la plataforma integrada que han sido registrados utilizando direcciones IP públicas. Permitiendo realizar llamadas entre usuarios conectados desde cualquier sitio.

Para ello fueron registrados los usuarios Aliceka y Bobka utilizando los datos públicos del servidor. De esta forma los usuarios fueron registrados desde internet y no a nivel local. Se pudo verificar que los usuarios se registraron correctamente.

Una vez fueron registrados los usuarios desde sus respectivas IP's públicas, se procedió a realizar una llamada entre ellos. La captura de los datos originados por la llamada realizada entre estos usuarios, fue realizada en los tres puntos de la comunicación (los usuarios y la plataforma integrada) permitiéndonos observar con mayor detalle el establecimiento de la llamada.

Como primer punto de captura de datos de muestra la captura del tráfico desde el usuario Bobka quien originó la llamada. El diagrama de secuencia de esta captura puede apreciarse en la figura 56. De este diagrama, podemos decir que la comunicación fue realizada exitosamente y se evidencio un flujo normal de intercambio de mensajes SIP para el establecimiento de la comunicación, así como el flujo de datos RTP entre el usuario Bobka y la plataforma integrada.

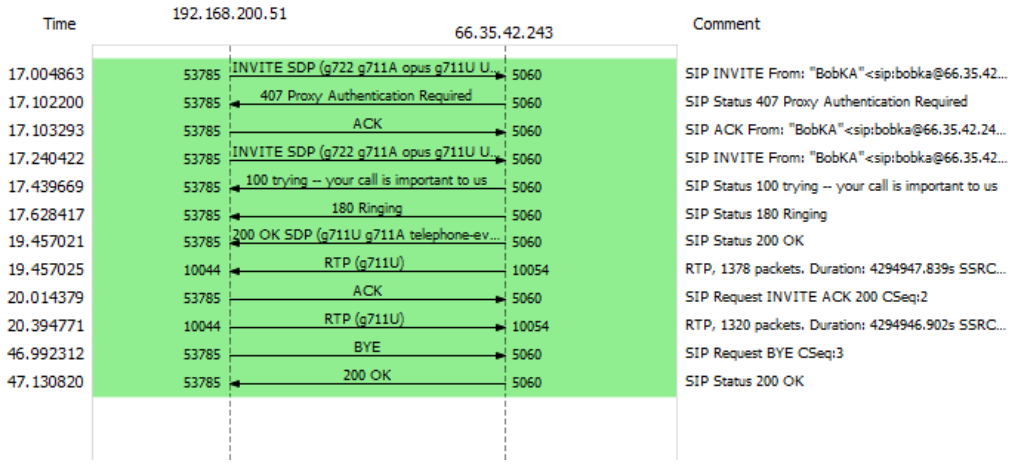

Figura 51**:** Diagrama de flujo de llamada desde bob hacia alice.
Fuente: Autor.

Como segundo punto de captura de datos de la llamada realizada, tenemos la captura de datos en el usuario Aliceka. El diagrama de secuencia asociada a la llamada recibida por el usuario Aliceka podemos apreciarlo en la figura 57. Podemos ver en este diagrama que la comunicación entre la plataforma integrada y el usuario Aliceka se realizó de manera normal y exitosa, desde su establecimiento, pase de datos RTP y finalización de la sesión.

En ambas figuras vemos como el envío y recepción de los datos RTP provienen y van dirigidos a la dirección pública de la plataforma integrada tal y como fue especificado en el parámetro "media_address".

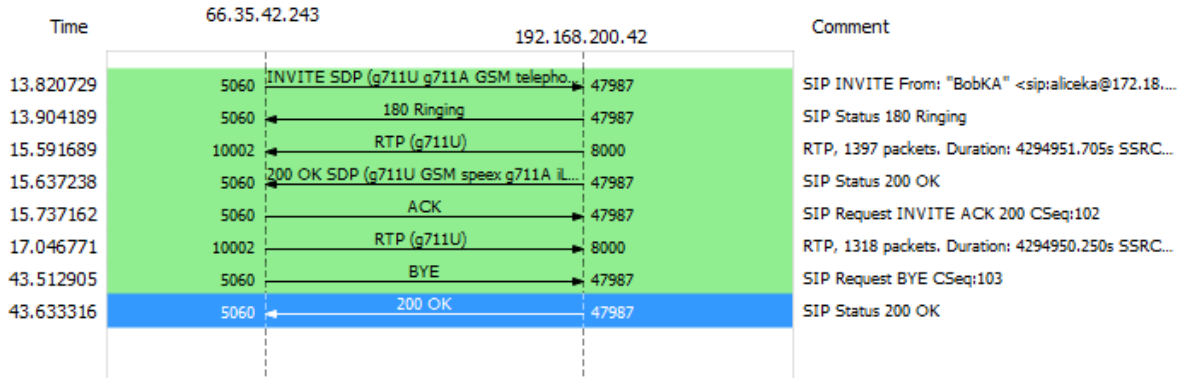

Figura 52***:*** Recepción de llamada Aliceka.
Fuente: Autor.

Por último, podemos apreciar el diagrama de flujo correspondiente a la tercera captura de datos. Esta captura fue realizada en la plataforma integrada permitiendo observar de manera centralizada el comportamiento de la llamada. Del diagrama se pueden apreciar separados por color verde y anaranjado el intercambio de mensajes entre los usuarios Aliceka y Bobka con el servidor respectivamente. Vemos reflejados el intercambio de mensajes para el establecimiento y para la finalización de la sesión. Por la forma en que fue implementado este ambiente de prueba, el tráfico RTP siempre es dirigido a la dirección IP 200.62.13.106. Esta dirección IP fue la utilizada por los usuarios al momento de su registro en el servidor. El trafico RTP es igualmente segmentado y enviado a los respectivos puertos seleccionados para el establecimiento de la comunicación con cada usuario por separado.

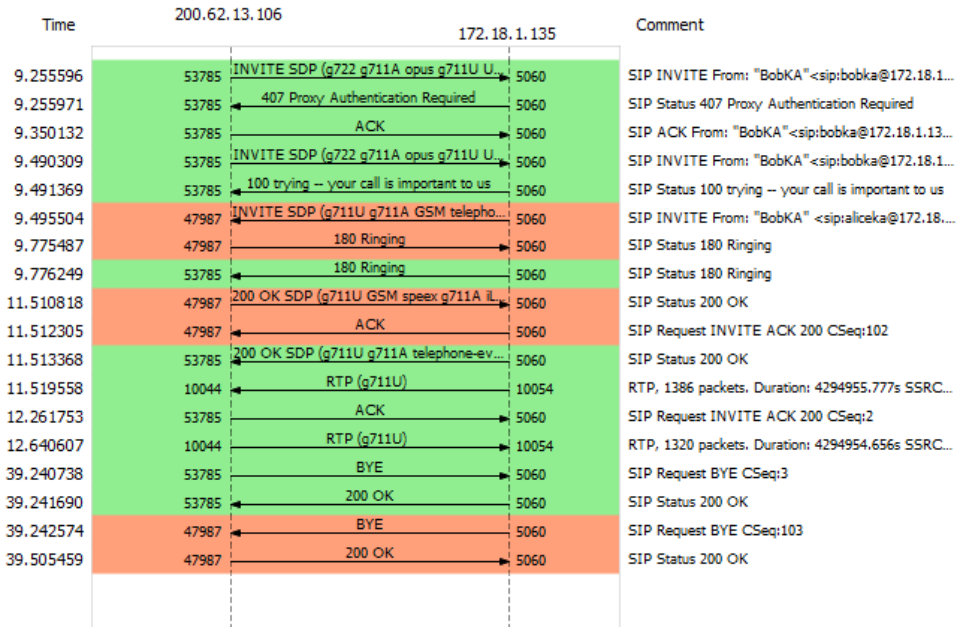

Figura 53***:*** Diagrama de flujo del servidor.
Fuente: Autor.

Los puertos utilizados por cada sesión son mostrados en las figuras 59 y 60.

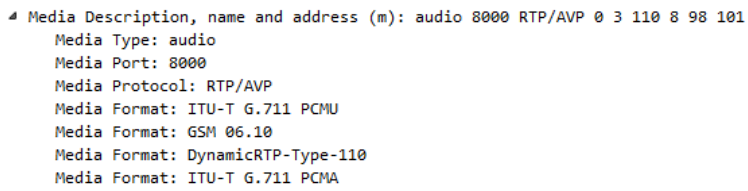

Figura 54: Puertos utilizados por un lado de la comunicación (Alicka-Asterisk).
Fuente: Autor.

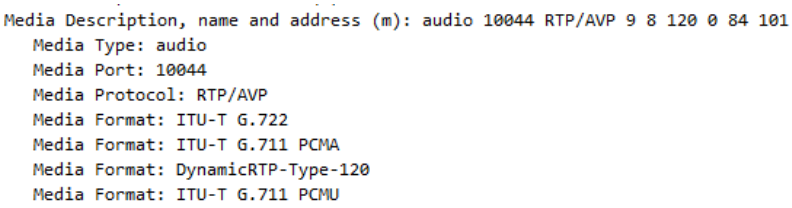

Figura 55*:* Puertos utilizados por un lado de la comunicación (Bobka-Asterisk).
Fuente: Autor.

3.6.10.5 Pruebas de verificación para ambiente de prueba 6 caso 3

Como último escenario, se verifico el correcto funcionamiento en una comunicación entre dos usuarios, uno registrado externamente a la plataforma y un segundo usuario ubicado detrás de un enlace troncal interno con la plataforma integrada. Para ello fue realizada una llamada entre estos dos usuarios. La topología utilizada para este escenario se aprecia en la figura 22 de la sección de análisis de requerimientos.

En este escenario se estudió la viabilidad de realizar llamadas entre usuarios conectados externamente y usuarios registrados en servidores internos asociados a la plataforma por medio de enlaces troncales. Esta topología se acerca a la configuración final del presente trabajo especial de grado.

Para ello fueron registrados los usuarios 110 y Bobka, siendo Bobka el usuario externo y el usuario 110 registrado en el servidor interno.

Una vez comprobados que los usuarios se encontraran correctamente registrados se procedió a realizar una llamada entre ellos. La llamada transita la plataforma integrada, la cual es redirigida hacia el servidor Elastix utilizado como servidor interno y luego desde el servidor Elastix hasta la extensión 110 local a este servidor.

Al igual que en los escenarios anteriores, fueron realizadas tres capturas de tráfico, una en el usuario 110, otra en la plataforma integrada y finalmente en el usuario Bobka. El diagrama de secuencia correspondiente a la captura de tráfico del usuario Bobka puede apreciarse en la figura 61.

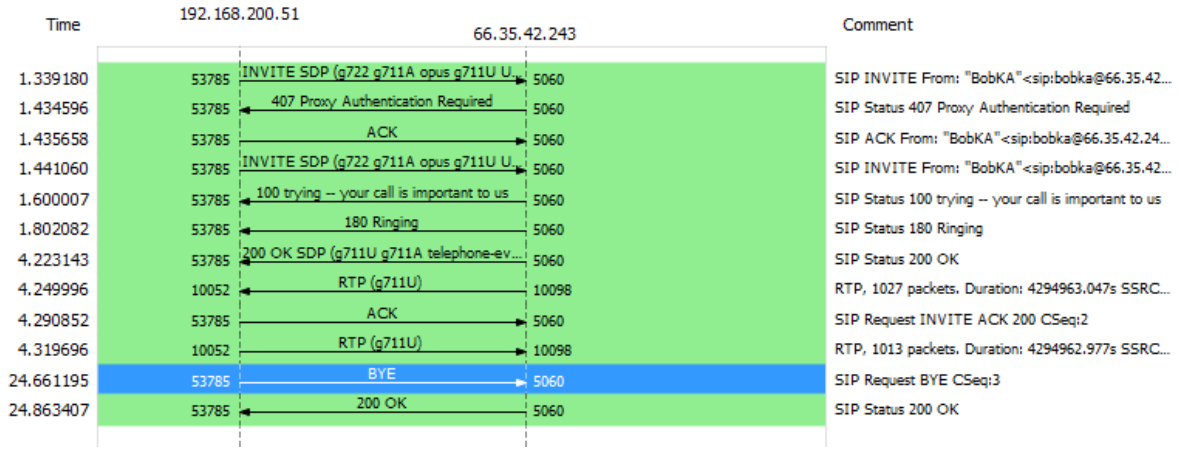

Figura 56***:*** Diagrama de flujo desde Bobka hacia usuario detrás de troncal.
Fuente: Autor.

Del diagrama anterior y al igual que los escenarios anteriores se aprecia un flujo de mensajes normal desde el establecimiento hasta la finalización de la comunicación.

Fue necesario para la implementación de este ambiente de prueba la adición de una entrada en el plan de marcado de la plataforma integrada que nos permitiera alcanzar al usuario 110 utilizando el enlace troncal con el servidor interno.

De esta forma entonces se capturo el tráfico de la plataforma integrada en donde se puede apreciar la comunicación entre esta y el servidor interno. El diagrama de flujo correspondiente a la captura de tráfico realizada puede apreciarse en la figura 62.

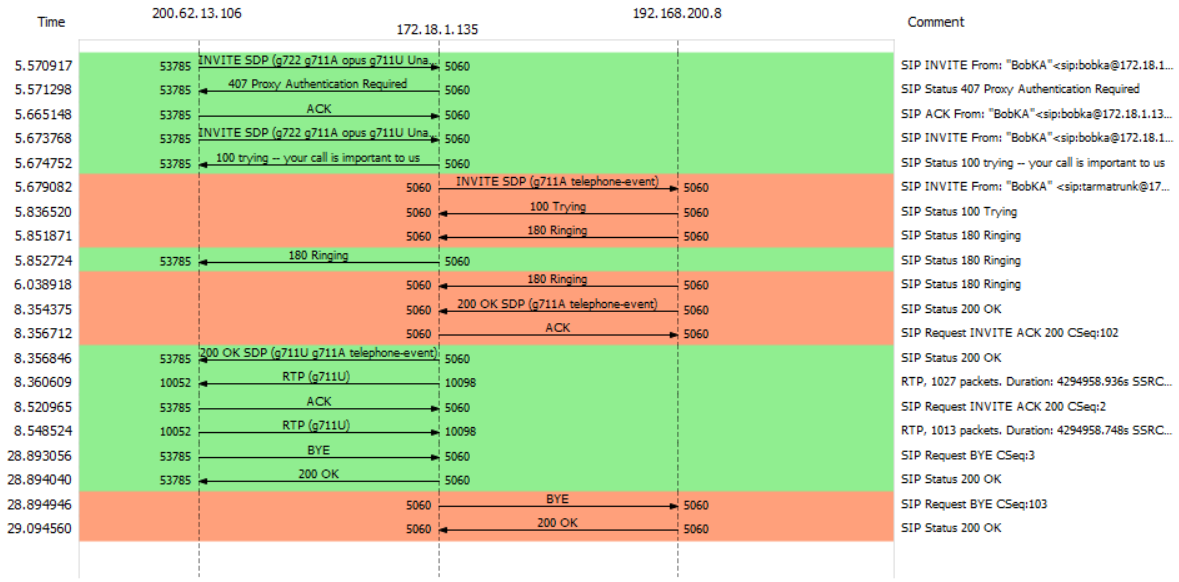

Figura 57***:*** Diagrama de secuencia de captura de tráfico desde la plataforma integrada.
Fuente: Autor.

Del diagrama anterior se puede apreciar el intercambio de mensajes entre el usuario Bobka y la plataforma (color verde), así como los mensajes desde la plataforma hacia el servidor remoto (color anaranjado) siendo el servidor interno aquel que posee la dirección IP 192.168.200.8.

Finalmente observamos el diagrama de secuencia de la recepción de la llamada por parte del usuario 110 (Figura 63). En este diagrama también se observa un establecimiento correcto de la comunicación entre el servidor Elastix y este usuario. Es importante resaltar que por ser el servidor Elastix el servidor de llamadas con el cual fue registrado el usuario 110, este envía el tráfico RTP a dicho servidor y no a la plataforma integrada. Es el servidor Elastix quien envía el tráfico a la dirección IP especificada en el campo "*media_address*".

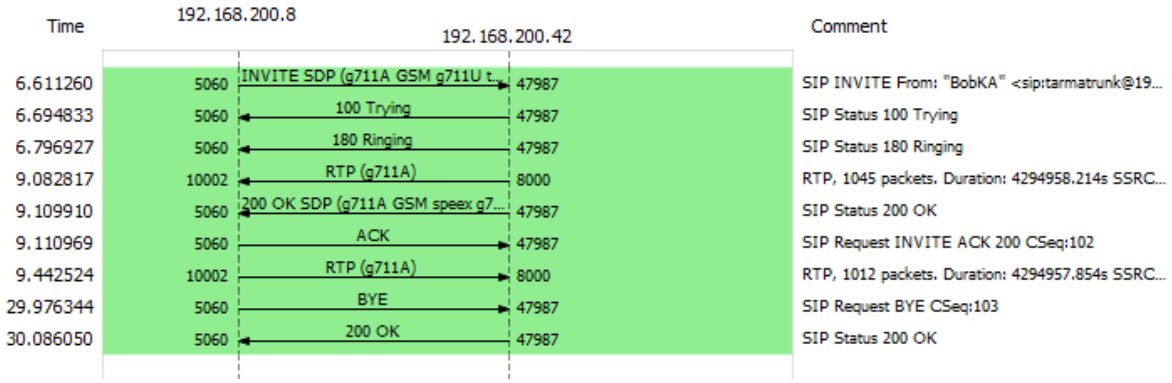

Figura *58:* Recepción de llamada en el usuario 110 desde la troncal con la plataforma integrada.
Fuente: Autor.

Este Sprint evaluó de manera individual cada uno de los posibles escenarios donde puede establecerse una comunicación utilizando la plataforma integrada resultando cada uno de estos escenarios exitosos en sus comunicaciones. Permitiendo así dar cabida a múltiples propuestas que pueden realizarse utilizando la plataforma integrada.

3.6.11 Sprint 7

Para nuestro séptimo Sprint se contemplaron todas aquellas tareas relacionadas a la configuración de los servidores Elastix y FreePBX correspondientes a las empresas Tarma Consultores y Arkisoft para su posterior integración a la plataforma Kamailio Asterisk (Ambiente de prueba 9). El listado de tareas realizadas durante el desarrollo del Sprint número siete se pueden apreciar en la tabla 24.

Tabla 24
Lista de tareas realizadas durante el desarrollo del Sprint número siete

Sprint ID	ID de tarea	Tarea
7	33	Configuración de central Elastix en Tarma consultores para exponer servicios VoIP al internet
	34	Configuración de servidor FreePBX en Arkisoft para exponer servicios VoIP hacia la Internet

	35	Configuración de usuarios troncales en los servidores Kamailio y Asterisk, Elastix y FreePBX
	36	Pruebas de verificación para ambiente de prueba 9

Fuente: Autor.

3.6.11.1 Configuración de central Elastix en Tarma consultores para exponer servicios VoIP al internet

Para realizar una exposición de un servidor Elastix a Internet, es necesario configurar reglas de firewall que permitan el acceso al mismo y sea permitido el tráfico SIP y RTP. Para la ejecución de esta tarea, fue necesario la colaboración del personal de redes de Tarma Consultores.

Una vez implementadas las reglas necesarias para garantizar las llamadas SIP, fueron necesarios configurar los siguientes parámetros en el servidor Elastix en el archivo "sip_nat.conf" con sus respectivos valores mostrados a continuación:

- nat=yes
- externip=<Direccion IP publica>
- localnet=<subredes locales>/<mascara de la subred>
- externrefresh=10

En donde el parámetro "localnet" podrá ser repetido tantas veces como redes locales hayan conectadas a nuestro sistema Elastix.

Luego procedemos a modificar el archivo "rtp.conf" especificando el rango de puertos que deseamos utilizar para el establecimiento de las comunicaciones.

Una vez configurados estos parámetros debemos reiniciar el servicio de Asterisk en nuestro servidor Elastix con la ejecución del comando:

- Asterisk -rx "core restart now"

Una vez realizadas estas configuraciones a nivel de Elastix. Fue necesario realizar configuraciones relacionadas con los componentes fail2ban e IPTables en este servidor a fin de mitigar posibles vulnerabilidades.

3.6.11.2 Configuración de servidor FreePBX en Arkisoft para exponer servicios VoIP hacia la Internet

Para configuración realizada en el servidor FreePBX de la empresa Arkisoft, fue utilizada la interfaz web que este servidor provee. Al igual que Elastix, FreePBX es un software basado en Asterisk por lo que se realizaron las modificaciones de los mismos parámetros.

Una vez dentro de la interfaz web de FreePBX, debemos ir a la sección de configuración avanzada de SIP, en esta se encuentra el apartado de NAT. En esta sección debemos activar la opción en "yes", así como especificar la dirección IP publica por la que son expuestos los servicios hacia el internet. Finalmente especificamos las redes locales que interactúan con el servidor. El resultado de estas configuraciones realizadas puede apreciarse en la figura 64.

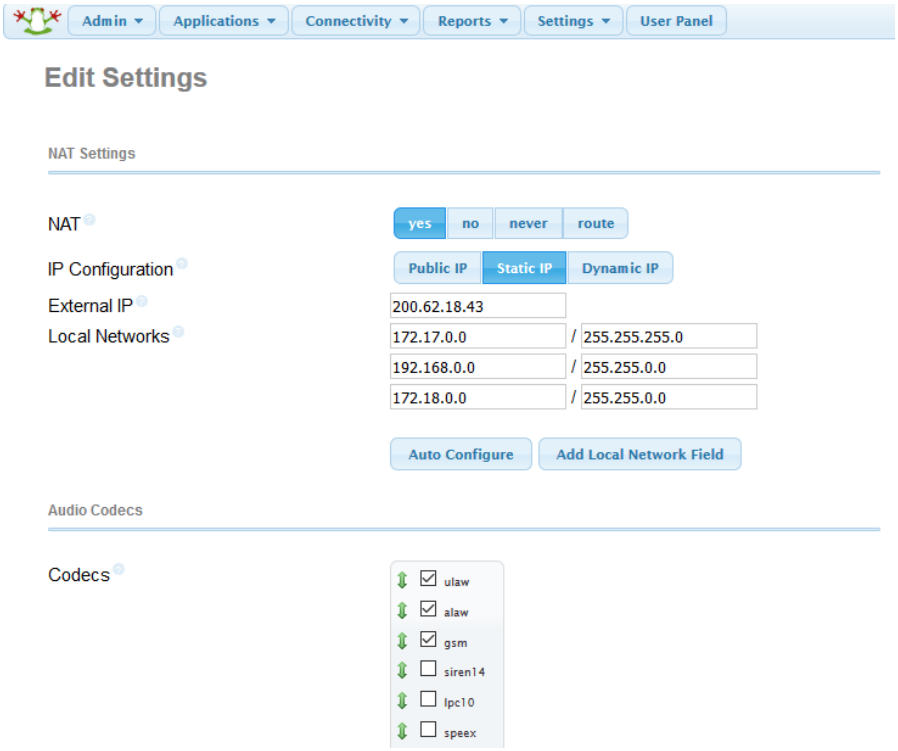

Figura 59***:*** Configuración de parámetros en FreePBX.
Fuente: Autor.

Por ultimo como se aprecia en la figura 65, fue agregado el campo "fromdomain" en la sección de "Other SIP Settings".

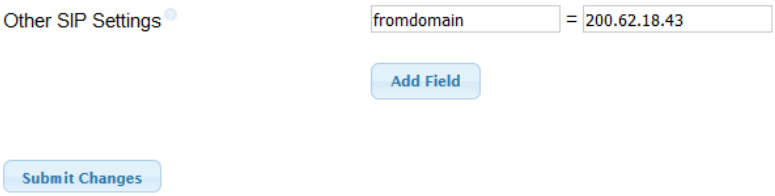

Figura 60***:*** Configurando el parámetro *fromdomain*.
Fuente: Autor.

3.6.11.3 Configuración de usuarios troncales en los servidores Elastix

Para crear los usuarios troncales en Elastix, debemos ir a la sección de "PBX" de la interfaz web y luego a la sección de "Trunks". En esta sección debemos seleccionar crear una troncal de tipo SIP. Una vez seleccionada la opción de creación para la cuenta troncal debemos especificar información referente a esta en la página de detalles de la troncal (ver figura 66).

De la seccion de detalles de la troncal solo nos interesa campo "trunk name" que se encuentra en la primera parte de la pagina y la seccion de "outgoing settings" (ver figura 66).

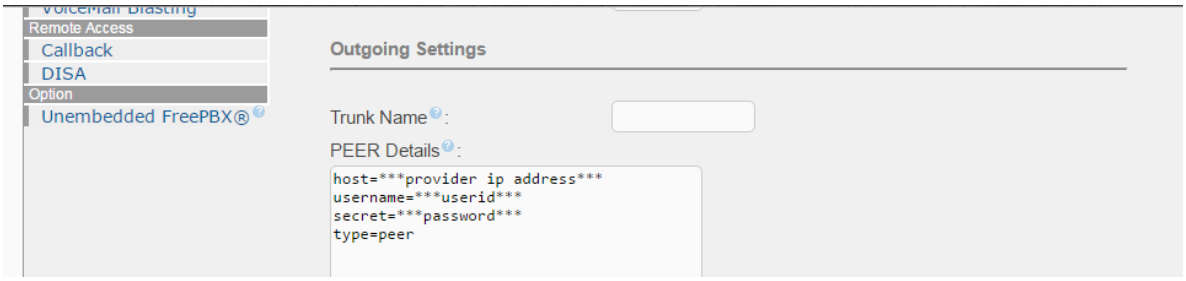

Figura 61***:*** Configurando troncal en Elastix.
Fuente: Autor.

En esta sección especificamos los datos de la troncal como si se tratase del archivo de configuración "sip.conf". De esta forma debemos agregar los siguientes datos especificados en la tabla 25.

Tabla 25
Detalles de configuración

Configuración de parámetros
host=66.35.42.243
username=1146trunk
defaultuser=1146trunk
secret=algunsecreto
directmedia=no
nat=yes
type=friend
trunk=yes
qualify=yes
disallow=all
allow=gsm
context=from-internal

Fuente: Autor.

3.6.11.4 Configuración de usuarios troncales en los servidores FreePBX

De manera similar a la configuración realizada en Elastix se realizó la configuración en FreePBX de la troncal. Debemos dirigirnos a la sección de "conectivity" en el menú superior luego seleccionar la opción "trunk". Al igual que en Elastix nos concentraremos en el nombre de la troncal y en el parámetro "outgoing setting" como se puede apreciar en la figura 67.

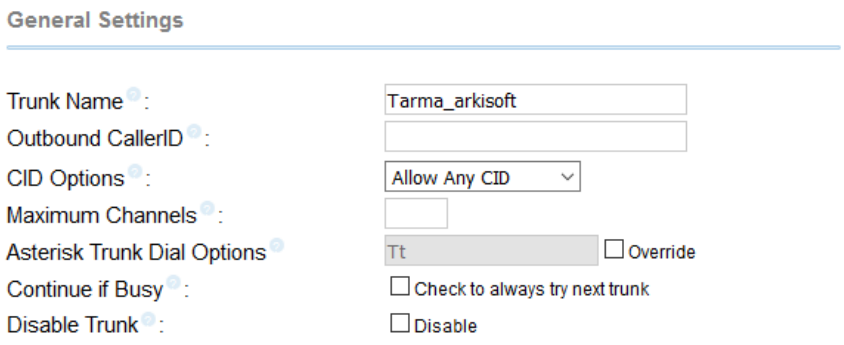

Figura 62*:* Parámetros de configuración para troncal en FreePBX.
Fuente: Autor.

Los datos a ingresar en el recuadro *peer* son los indicados en la tabla 26

Tabla 26.

Datos de usuario para troncal en FreePBX.

Valores para el campo peer details
host=66.35.42.243
username=8661trunk
defaultuser=8661trunk
fromuser=8661trunk
secret=algunsecreto
directmedia=no
type=friend
trunk=yes
context=from-trunk
disallow=all
allow=gsm
qualify=yes
deny=0.0.0.0/0.0.0.0
permit=66.35.42.243/255.255.255.255

Fuente: Autor.

Se puede ver en la figura 68 como debería quedar el recuadro *peer* luego de ingresar los datos.

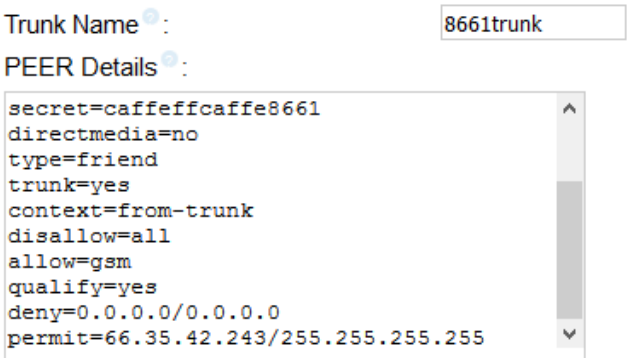

Figura 63*:* Registro de troncal FreePBX.
Fuente: Autor.

3.6.11.5 Configuración de rutas entrantes y salientes

Como requisito adicional para la integración de una organización, es necesaria la creación de una ruta entrante. Esta opción es de total libertad de la organización suscriptora, hacia donde sean dirigidas las llamadas. Esto puede ser, a un conjunto de extensiones, a alguien IVR (Interactive voice Record), a una extensión en particular, etc. Para este ambiente de prueba se utilizó un patrón para poder llamar a las extensiones de Tarma Consultores. Y para el caso de Arkisoft se hizo una ruta entrante en la que se redirige las llamadas discadas (al número suministrado) a un IVR.

Como paso opcional, se pueden configurar rutas salientes. Las rutas salientes solo serán necesarias si un cliente o institución desea realizar llamadas hacia las demás organizaciones conectadas a la plataforma.

3.6.11.6 Configuración de usuarios troncales en la plataforma integrada Kamailio Asterisk

Luego de haber definido y configurado los aspectos necesarios en cada una de las centrales de Tarma y Arkisoft, se procedió a realizar las ultimas configuraciones a la plataforma integrada. Para ello fueron agregados los dos usuarios troncales para Tarma y para Arkisoft. Para lo cual fueron utilizadas las sentencias SQL que pueden apreciarse en la tabla 27.

Tabla 27
Comandos SQL para agregar los usuarios troncales

Sentencia SQL para agregar troncal con Tarma	Sentencia SQL para agregar troncal con Arkisoft
INSERT INTO sipusers (name, ipaddr, defaultuser, host, type, context, fromdomain, fromuser, qualify, defaultip, trunkname, sippasswd, secret, deny, permit, nat, disallow, allow) VALUES("8661trunk", "200.62.18.43", "8661trunk", "200.62.18.43", "friend", "all-trunks", "200.62.18.43", "8661trunk", "yes", "200.62.18.43", "8661trunk", "algunacontraseña", "algunacontraseña", "0.0.0.0/0.0.0.0",	INSERT INTO sipusers (name, ipaddr, defaultuser, host, type, context, fromdomain, fromuser, qualify, defaultip, trunkname, sippasswd, secret, deny, permit, nat, disallow, allow) VALUES("1146trunk", "200.62.13.106", "1146trunk", "200.62.13.107", "friend", "all-trunks", "200.62.13.106", "1146trunk", "yes", "200.62.13.107", "1146trunk", "algunacontraseña", "algunacontraseña", "0.0.0.0/0.0.0.0",

"200.62.18.43/255.255.255.255", "yes", "all", "gsm")	"200.62.13.107/255.255.255.255", "yes", "all","gsm")
INSERT INTO sipregs(name) VALUES('8661trunk ');	INSERT INTO sipregs(name) VALUES('1146trunk');

Fuente: Autor.

Se agregan datos adicionales como los parámetros "deny" y "permit" permitiendo, filtrar conexiones provenientes desde otras direcciones IP que no sean las del cliente.

Una vez agregados los usuarios, se configuraron las entradas dentro del plan de marcado para poder alcanzar ambos destinos. Podemos apreciar la entrada del plan de marcado en la figura 69.

```
exten => _881XX,1, NoOP("Se envia la llamada por la troncal de tarma")
       same => n, NoOp(${EXTEN})
       same => n, Set(extension=${EXTEN})
       same => n,Dial(SIP/1146trunk/${extension},30)
```

Figura 64***:*** Entrada en el plan de marcado de Asterisk para la troncal de Tarma Consultores.
Fuente: Autor.

De la anterior entrada podemos ver que se definió un patrón para poder realizar llamadas hacia Tarma Consultores. Todas aquellas extensiones discadas que inicien con los dígitos '88' seguidos de tres dígitos adicionales que comiencen por el digito '1' es decir desde el numero 100 hasta el 199, serán enviados por el enlace troncal hacia Tarma Consultores.

De la misma forma fue agregada una entrada en el plan de marcado para la troncal hacia Arkisoft. Esta puede apreciarse en la figura 70.

```
exten => 7123224343,1, NoOP("Se envia la llamada por la troncal de arkisoft")
       same => n, NoOp(${EXTEN})
       same => n, Set(extension=${EXTEN})
       same => n,Dial(SIP/8661trunk/${extension},30)
```

Figura 65***:*** Entrada en el plan de marcado de Asterisk para la troncal de Arkisoft.
Fuente: Autor.

De la cual podemos decir que, al marcar el número 7123224343, la llamada será enviada por la troncal hacia Arkisoft para luego ser procesada por la ruta entrante definida en este servidor.

Una vez realizadas estas configuraciones, se procedió a realizar la llamada desde una extensión registrada en el servidor Elastix de Tarma consultores.

Como se puede apreciar en el diagrama de secuencia de la figura 71, la llamada se establece inicialmente desde la extensión 111 hacia el servidor Elastix de Tarma Consultores. De acuerdo con la ruta saliente y el patrón de discado al marcarse el numero 7123224343 esta llamada se envía por la troncal 1146trunk hacia la plataforma integrada Kamailio Asterisk.

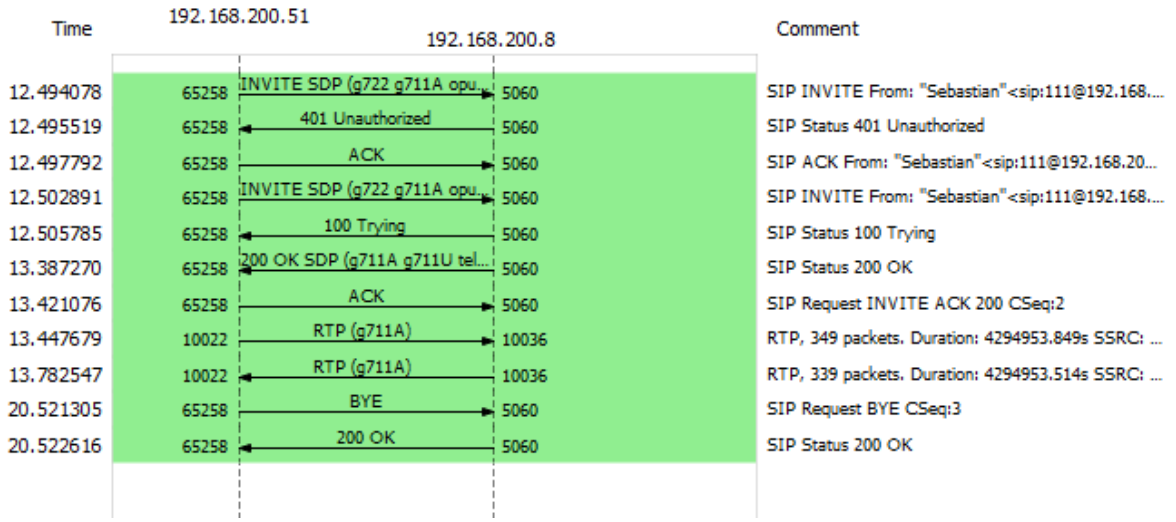

Figura 66***:*** Establecimiento de la llamada desde usuario 111 hacia Arkisoft.
Fuente: Autor.

La dirección destino podemos verla seleccionando el paquete invite y observando el campo to de la cabecera SIP (ver figura 72).

Figura 67***:*** Contenido del mensaje SIP INVITE.
Fuente: Autor.

Luego que la llamada es recibida por el servidor Kamailio Asterisk, este intenta ubicar y hacer coincidir la extensión proveniente desde la troncal 1146trunk con las extensiones definidas dentro del contexto donde están ubicadas ambas troncales. De esta forma coincide el número 7123224343 con la entrada definida anteriormente (ver figura anterior) y redirige la llamada hacia la troncal 8661trunk que es la troncal hacia Arkisoft. El flujo de la llamada desde el servidor Kamailio Asterisk podemos verlo en la figura 73.

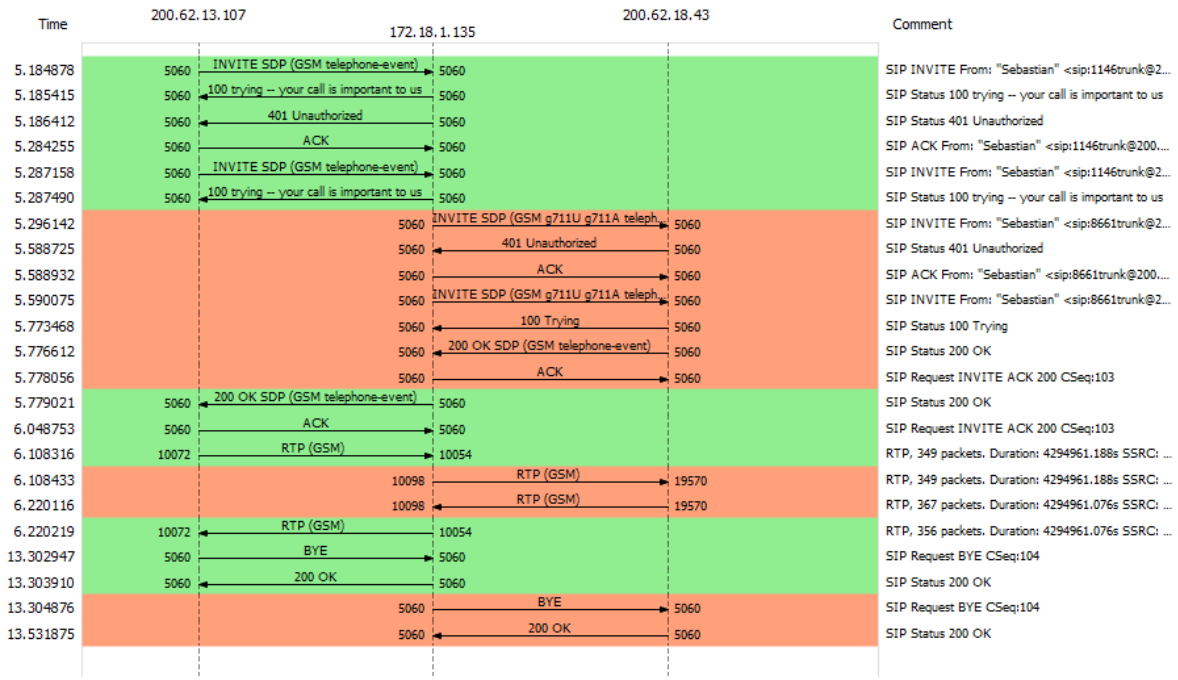

***Figura** 68**:** Comunicación entre plataforma integrada y troncales.*
Fuente: Autor.

Podemos apreciar que la comunicación se estableció de manera satisfactoria. Por último, podemos ver el estado de la comunicación al momento de finalizar la llamada con el diagrama de secuencia obtenido de la captura de paquetes desde el servidor FreePBX de Arkisoft (figura 74).

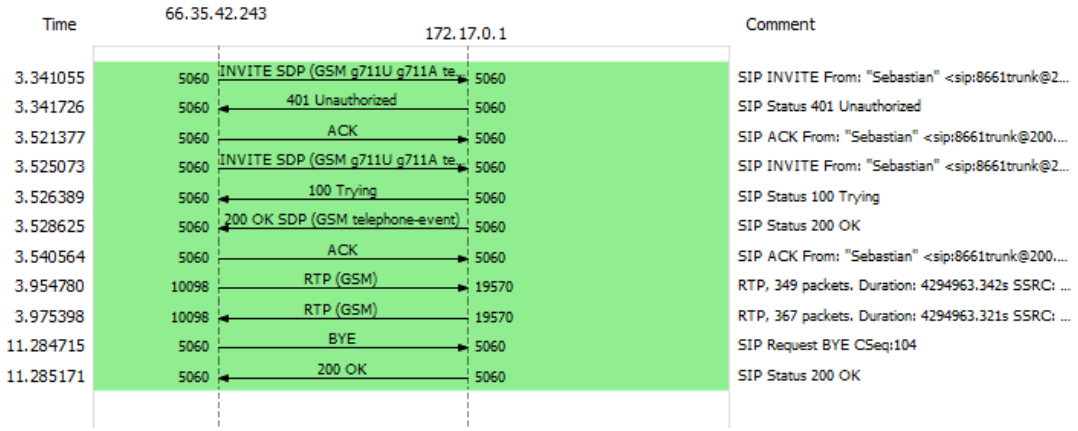

***Figura** 69**:** Diagrama de secuencia de llamada entre plataforma integrada y Arkisoft.*
Fuente: Autor.

Vemos como los mensajes provienen desde la troncal 8661trunk. De la figura 75 podemos observar los detalles del mensaje SIP INVITE para confirmar los datos.

Figura 70***:*** Contenido de mensaje SIP Invite proveniente de la plataforma integrada.
Fuente: Autor.

Vemos como se hace el envío del mensaje invite hacia el usuario SIP 7123224343 en el servidor FreePBX. Este último lo hará coincidir con la ruta entrante definida y finalmente será el IVR quien se comunique con la extensión 111.

De esta forma se ha comprobado el correcto funcionamiento y configuración de la plataforma integrada Kamailio Asterisk para actuar como nodo central para la comunicación y distribución de llamadas entre las mismas, logrando establecer una comunicación efectiva entre las centrales VoIP de Tarma consultores y Arkisoft mediante el establecimiento de enlaces troncales SIP y viajando por internet (uso de direcciones IP públicas).

Luego de finalizar la implementación y verificación de la plataforma en distintos escenarios, se evaluó como caso adicional el ambiente de prueba en donde la plataforma sirve solamente de nodo central para la señalización dejando la transmisión y manejo de datos multimedia entre los usuarios. Para ello fue elaborado nuestro octavo y último Sprint.

3.6.12 Sprint 8

Para nuestro último Sprint se contemplaron todas las tareas relacionadas con la configuración de los usuarios finales a fin de que estos puedan transmitir los datos multimedia (RTP) directamente entre ellos sin tener que circular por la plataforma integrada. La plataforma integrada solo servirá de puente para la señalización del establecimiento y cierre de la comunicación entre estos dos usuarios. El listado de tareas realizadas durante el desarrollo de este último Sprint puede apreciarse en la tabla 28.

Tabla 28
Listado de tareas realizadas durante el último Sprint

Sprint ID	ID de tarea	Tarea
8	37	Ambiente de prueba adicional: Envió de datos multimedia entre clientes
	38	Pruebas de verificación para ambiente de prueba final

Fuente: Autor.

3.6.12.1 Ambiente de prueba adicional: Envió de datos multimedia entre clientes

La motivación para la implementación y evaluación de este ambiente de prueba contempla varias razones. La primera de ellas es la utilización de recursos en la plataforma integrada en presencia de múltiples llamadas concurrentes y los requerimientos de ancho de banda que estas llamadas pueden exigir. Los requerimientos de ancho de banda por las llamadas concurrentes, están relacionados con los códec utilizados en la comunicación. Es por esta razón que el códec GSM fue seleccionado entre todos los demás para las llamadas provenientes desde las troncales.

El códec GSM posee un consumo de ancho de banda muy bajo por llamada (13Kbps base) en contraste con otros códec como G711 que consume 64kbps (base) por llamada [41]. Si bien los códec poseen un consumo de ancho de banda base, la cantidad de ancho de banda total aproximado puede apreciarse en la tabla 29. El ancho de banda total puede variar ligeramente de acuerdo a los protocolos de capas (TCP/IP) inferiores utilizadas.

Tabla 29
Ancho de banda utilizado por distintos códec

Codec	Audio Quality	CPU Resources	Base Size	Total Size (Base + Overhead)
G711	Good	Very Low	64 kbps	95.2 kbps
G722	Very Good	Low	64 kbps	95.2 kbps
GSM	Acceptable	Average	13 kbps	44.2 kbps
G729	Average	High	8 kbps	39.2 kbps

Fuente: [41]

Otro de los beneficios que se pueden mencionar, dependiendo la ubicación de los suscriptores, es que estos pueden encontrarse dentro de un mismo dominio ISP. De encontrarnos en este caso, es ventajoso que el tráfico multimedia circule dentro de la misma red del ISP sin tener que dar saltos adicionales.

Con las anteriores razones descritas se realizaron configuraciones a nivel de registro de usuario para estudiar a posibilidad de la transmisión directa de trafico RTP entre los usuarios. Para ello, fueron definidos los usuarios Alice y Bob, pero esta vez con una configuración particular que permitieron desplegar de manera exitosa el ultimo ambiente de prueba.

Estos usuarios fueron agregados utilizando las sentencias SQL definidas en la tabla 27 con la excepción que fueron modificados los campos "host", "type", "directmedia" y "nat" y fue agregado el parámetro "dtmfmode" dentro de la instrucción *INSERT*. Los valores que poseen estos campos son definidos a continuación.

- Nat = yes
- Host=dynamic
- Type=friend
- Directmedia=yes
- Dtmfmode=info

El parámetro "dtmfmode" nos especifica si los tonos son enviados "inband" o "outband" (utilizando el mismo streaming de audio para enviar tonos o enviándolos por fuere del streaming de audio). Se especifica que los tonos sean enviados utilizando mensajes SIP INFO fuera del streaming del audio.

Fue necesario configurar el parámetro *dtmfmode* con el valor de *info* y el parámetro *directmedia* con el valor *yes* para que el habiente funcionara apropiadamente [36].

Del registro de los usuarios podemos ver que estos han sido registrados utilizando direcciones IP publicas distintas (ver figura 76). por la manera en que fue implementado este ambiente de prueba, verificamos que los usuarios se encuentren registrados en Asterisk utilizando los datos de Kamailio (ver figura 77).

```
[root@tarmaipbx ~]# kamctl ul show
Domain:: location table=512 records=2 max_slot=1
        AOR:: bob
                Contact:: sip:bob@200.8.96.230:50879;rinstance=90b48743f2a8c019 Q=
                        Expires:: 3578
                        Callid:: 82158N2Q0YmU5YTYwNzdlOGUxNTE4N2N1NDMxYWRlZjU0Y2E
                        Cseq:: 2
                        User-agent:: X-Lite release 4.9.6 stamp 82158
                        State:: CS_NEW
                        Flags:: 0
                        Cflag:: 0
                        Socket:: udp:172.18.1.135:5060
        AOR:: alice
                Contact:: sip:alice@200.62.13.106:58952;rinstance=be7711473129407e;transport=UDP Q=
                        Expires:: 3598
                        Callid:: OGVmZTE2Mjg5NjJkYzBlMjg2Y2YwNDExOGU5OGVlNzY.
                        Cseq:: 2
                        User-agent:: Z 3.6.25251 r25476
                        State:: CS_NEW
                        Flags:: 0
                        Cflag:: 0
                        Socket:: udp:172.18.1.135:5060
```

Figura 71***:*** Consulta de usuarios registrados Kamailio.
Fuente: Autor.

Una vez verificados que los usuarios se registraron de manera exitosa, se procedió a realizar la llamada entre Alice y Bob.

La llamada fue realizada con éxito ya que se pudieron escuchar los dos extremos de la comunicación, indicando que no hubo problema alguno con NAT. Se puede apreciar como en la Figura 78 generada a partir de los datos capturados desde el usuario Bob, la comunicación se establece de forma normal.

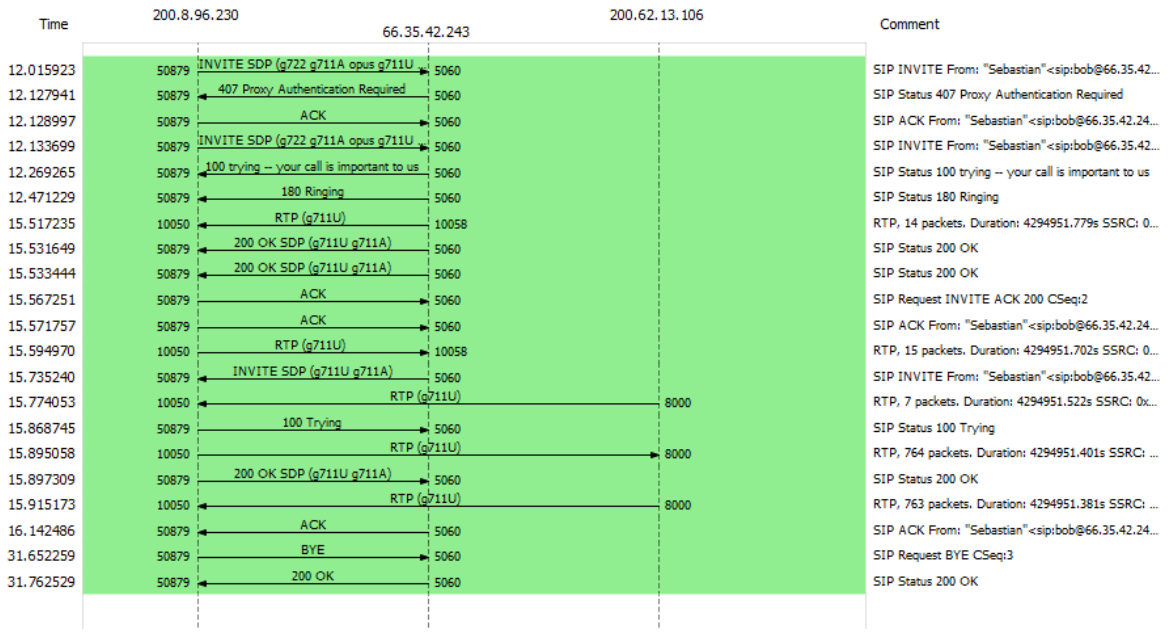

Figura 72***:*** Diagrama de flujo de la llamada desde Bob.
Fuente: Autor.

Luego del intercambio inicial de mensajes SIP, Asterisk envía un mensaje invite. Podemos comprobar que el mensaje invite pertenece al mismo dialogo indicando que es un mensaje SIP Re-Invite.

En la figura 79 vemos resaltados los valores del campo "tag" pertenecientes a los parámetros "from" y "to", recordemos que el campo "tag" es agregado una vez la comunicación es establecida. Este campo es utilizado para verificar que los mensajes pertenecen al mismo dialogo al igual que otros parámetros como el campo "call-id".

```
Session Initiation Protocol (INVITE)
  Request-Line: INVITE sip:333@66.35.42.243 SIP/2.0
  Message Header
    Via: SIP/2.0/UDP 200.8.96.230:50879;branch=z9hG4bK-524287-1---ebd3744edb3c590c;rport
    Max-Forwards: 70
    Contact: <sip:bob@200.8.96.230:50879;rinstance=90b48743f2a8c019>
    To: <sip:333@66.35.42.243>
    From: "Sebastian"<sip:bob@66.35.42.243>;tag=db752203
    Call-ID: 82158NGE1MjY3ZjA0OTU4ODNhZjNhODg2YTY2MDFlOThiYzY
    CSeq: 2 INVITE
    Allow: SUBSCRIBE, NOTIFY, INVITE, ACK, CANCEL, BYE, REFER, INFO, OPTIONS, MESSAGE
    Content-Type: application/sdp
```

Figura 73***:*** Paquete SIP, campo TAG.
Fuente: Autor.

El proceso de RE-INVITE es explicado utilizando solo el diagrama de flujo de la comunicación saliente con Bob.

Vemos como el primer mensaje invite no posee el tag asociado al campo to, este es establecido en el OK enviado por el usuario destino, esto se puede apreciar en la figura 80.

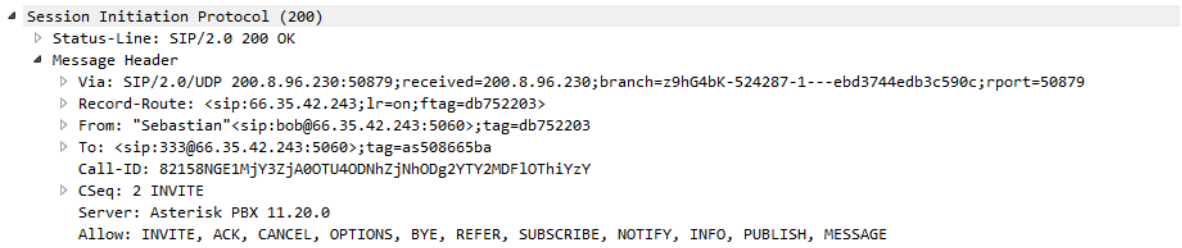

Figura 74***:*** Mensaje SIP OK enviado al usuario Bob.
Fuente: Autor.

Podemos ver de la figura 81 el parámetro "connection information" la dirección IP 66.35.42.243 para el envío de datos multimedia, es decir enviar los datos multimedia a la plataforma integrada. Esto se resalta porque a continuación en la figura 82 observamos el paquete embebido SDP dentro del re-invite como estos parámetros son modificados por Asterisk.

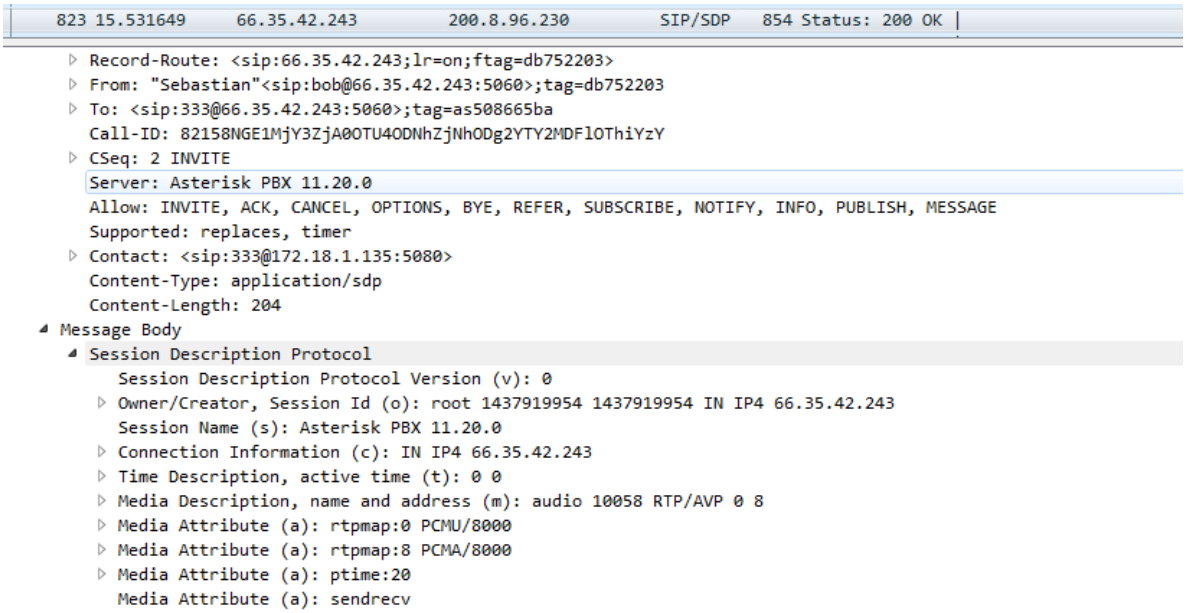

Figura 75***:*** Mensaje SDP.
Fuente: Autor.

Del paquete enviado por Asterisk podemos ver como este invite (re-invite) posee los mismos subparametros "from" y to "tag" asociados a la comunicación y el mismo "call-id". Adicionalmente podemos ver en el analizador de trafico la palabra "in-dialog" en la parte superior derecha asociada con el paquete en cuestión, identificando este mensaje como un mensaje SIP Re-invite. Ahora, dentro de este Re-INVITE es descrito el nuevo mensaje SDP en donde podemos ver que el parámetro "Connection information" posee ahora el valor para 200.62.13.106 indicándole al usuario Bob la nueva dirección donde serán enviados los datos multimedia.

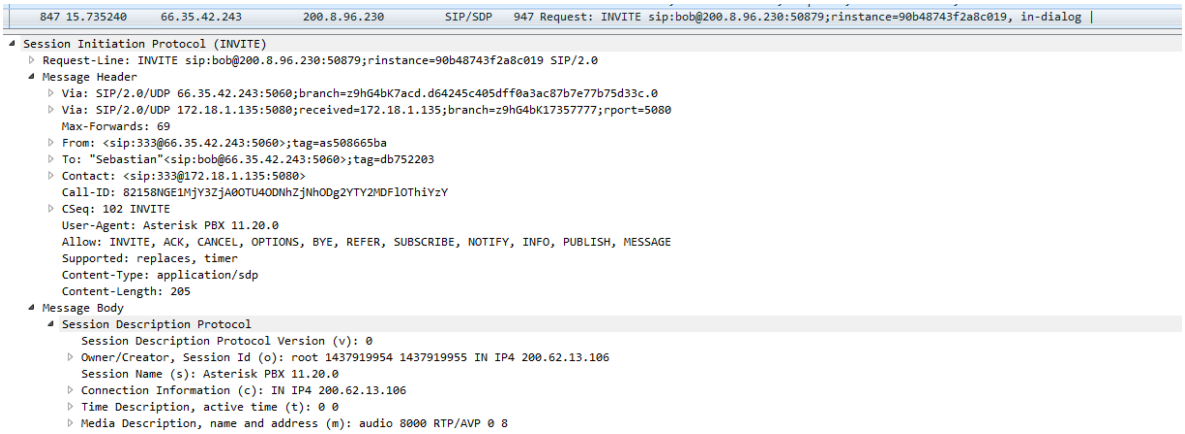

Figura 76***:*** Mensaje SDP Re-Invite.
Fuente: Autor.

En el diagrama de secuencia (figura 83) vemos como este mensaje invite es confirmado y la transmisión de datos multimedia se realiza directamente con el usuario. Finalmente, los mensajes de cierre de sesión se envían a la plataforma integrada.

En el diagrama de secuencia para la llamada de Alice, vemos como llega un primer invite (establecimiento de la segunda sección de la comunicación) y posteriormente se aprecia la recepción de un segundo mensaje SIP INVITE, pero esta vez para la modificación del flujo multimedia.

Al finalizar el diagrama vemos como los mensajes de cierre de sesión son enviados por el Asterisk, terminando con esa sección de la llamada.

En todos los diagramas podemos apreciar el envío de otros mensajes Re-INVITE, pero estos son utilizados para la modificación de otros parámetros como los códec o el muestreo en la comunicación.

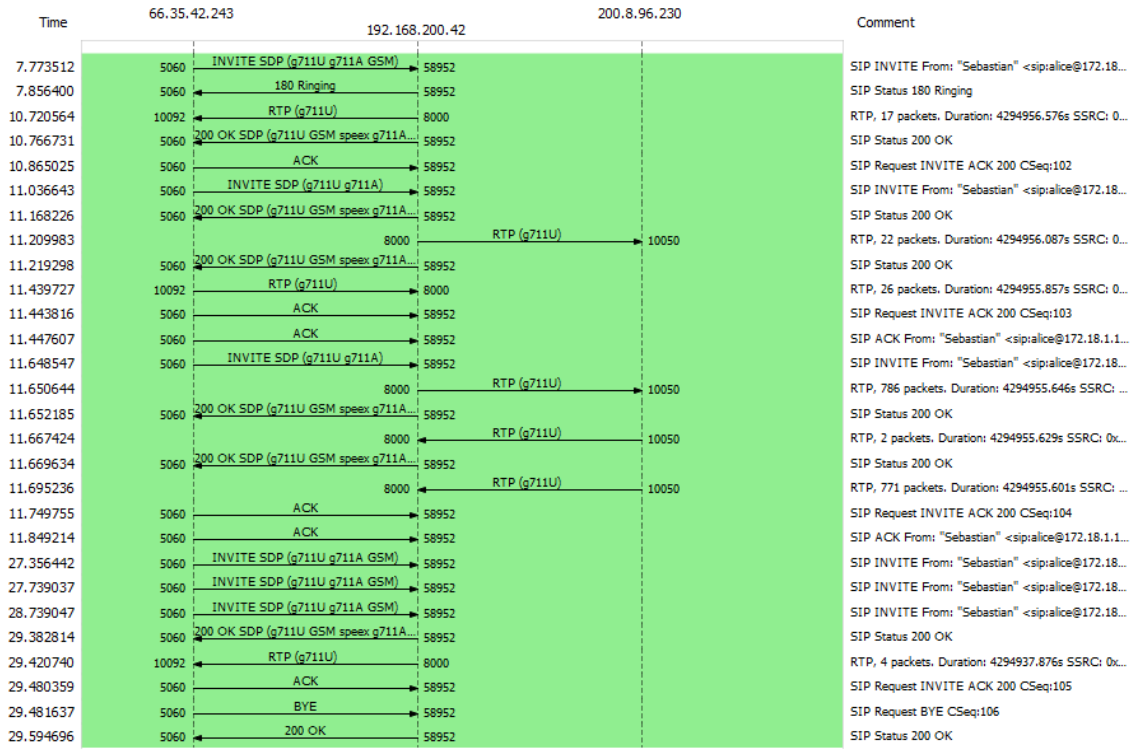

Figura* 77*: Diagrama de flujo de la llamada desde Alice.
Fuente: Autor.

Finalmente podemos apreciar la traza completa de la comunicación desde la plataforma integrada en la figura 84. En este diagrama de secuencia podemos ver como el tráfico RTP no circula por la plataforma integrada.

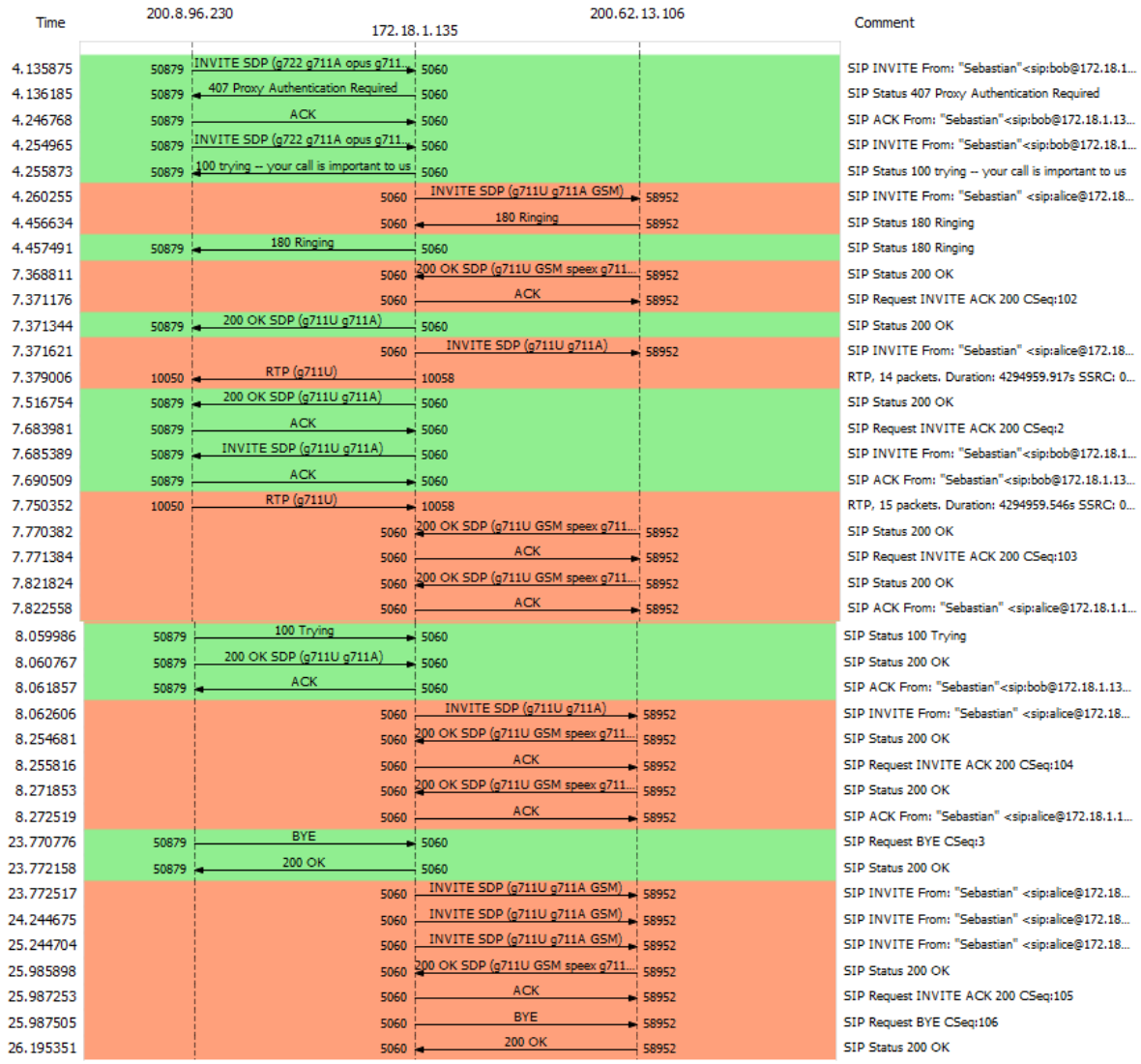

***Figura** 78:* Diagrama de flujo de la llamada entre Alice y Bob desde el servidor.
Fuente: Autor.

3.7 Fase de desarrollo de aplicación web administrativa

Una vez concluida esta fase de integración, se realizó una segunda fase en la que se diseñó y desarrollo la aplicación web administrativa para esta plataforma. Esta aplicación web contó con las funcionalidades necesarias para poder cumplir con los objetivos planteados para el desarrollo del presente trabajo. Algunas de estas funcionalidades incluyen la administración y registro de usuarios, envíos de correos de notificación, operación de carga de datos en distintas bases de datos y generación de datos y archivos de configuración para el establecimiento de los enlaces troncales. Al igual que la fase anterior se utiliza Scrum como metodología de desarrollo realizando a continuación la aclaratoria para el inicio de esta fase bajo esta metodología.

3.7.1 Aclaratoria Scrum

Al igual que para la fase anterior de implantación de la plataforma integrada, se mantienen las mismas condiciones para la fase de desarrollo de la aplicación web administrativa de la plataforma integrada bajo la metodología SCRUM. Estas condiciones son descritas a continuación.

Debido a que solo una persona es integrante del desarrollo del proyecto, algunos de los roles definidos en la metodología SCRUM serán absorbidos o unificados en uno solo. Por lo tanto, el solo integrante en el desarrollo de este proyecto debe cumplir con las siguientes obligaciones:

- Añadir, modificar y eliminar tareas durante el desarrollo del proyecto y la ejecución de los sprints.
- Otorgar los niveles de prioridad a las distintas tareas del backlog o a consideración durante la ejecución del proyecto.
- Organizar y planificar la ejecución de las tareas durante el desarrollo de cada sprint.
- Identificar y solucionar problemas encontrados durante el desarrollo del proyecto.
- Consultar, sugerir y ayudar en la toma de decisiones en pro del desarrollo del proyecto.

Para el desarrollo del presente trabajo y producto, la Junta directiva de Tarma consultores asumió el rol de Product Owner. Llevando a cabo reuniones periódicas (Sprint planning y Sprint review) para mostrar su opinión sobre los incrementos generados durante cada Sprint y destacando la importancia de las tareas en el Product Backlog.

3.7.2 Fase de desarrollo de aplicación web para la administración de funciones de la plataforma integrada Asterisk Kamailio

Para el desarrollo de la segunda fase del presente trabajo, se contemplaron todas aquellas actividades relacionadas al trabajo realizado para el diseño y desarrollo de una aplicación web para la administración de algunas funcionalidades provistas por la plataforma integrada Kamailio Asterisk, así como funcionalidades que den soporte a los usuarios que deseen suscribirse a la plataforma. Esta aplicación web será la encargada de realizar el registro y gestión de los suscriptores que deseen disfrutar de los beneficios ofrecidos por la plataforma y formar parte de la misma. Desde el aplicativo web se pueden administrar las cuentas de usuarios SIP asociados a los suscriptores para el establecimiento de enlaces troncales entre las plataformas VoIP de los suscriptores y la plataforma integrada Kamailio Asterisk. Otra de las características implementadas en la aplicación web, es la asignación de códigos de marcado para la identificación de suscriptores basándose en la propuesta realizada en el trabajo de seminario.

Como todo proceso de desarrollo de software describimos a continuación la fase realizada de análisis de requerimientos.

3.7.3 Análisis de requerimientos

De las primeras reuniones con el Product Owner, se pudieron dar a conocer las funcionalidades generales con la que debería contar la aplicación administrativa. Se planteó en primera instancia la adopción para el desarrollo de la aplicación el uso del framework Laravel para garantizar un ambiente de desarrollo estable.

De la aplicación se requirió que un usuario pudiese acceder a la misma mediante el uso de un navegador web. Como estructura inicial de la aplicación, se describió la creación de una página de inicio donde se pudiera obtener información relacionada al desarrollo del presente trabajo y una posible descripción de las funcionalidades ofrecidas por el sistema.

Fue requerido también que, por medio de la aplicación, un usuario podrá registrarse para disfrutar de los beneficios de interconexión provistos por la plataforma, permitiéndole luego la autenticación por medio de la creación de un perfil y asignación de un rol. El registro incluye la captura de información del usuario y de la organización que se suscriben. Esta información será utilizada para fines de identificación, generación de otras cuentas asociadas y asignación de un código de marcado. Este código, es único y servirá para que cada usuario suscrito pueda recibir llamadas desde la plataforma o enviar llamadas hacia la plataforma.

Es requisito de la aplicación web, permitirle al usuario la consulta y verificación de información y estados de servidores o enlaces troncales o datos relacionados a estas.

Para el almacenamiento de la información de todos los elementos que son creados por la aplicación web para el establecimiento de una conexión desde el suscriptor con la plataforma integrada, se requirió la implementación y uso de una base de datos a fin de realizar procesos de consultas, extracción y carga de información en la aplicación web administrativa.

Esta base de datos es distinta a la utilizada por la plataforma integrada de esta forma se requirió generar un proceso de volcado de información entre bases de datos para la consistencia entre usuarios registrados y usuarios cargados en la plataforma integrada.

Una vez finalizadas las primeras reuniones con el Product Owner se procedio a generar la primera lista de requerimientos para la aplicación web.

3.7.4 Funciones generales de la aplicación web

De forma general pueden agruparse las distintas funcionalidades del sistema y clasificarlas en función del rol que posee el usuario, rol de administrador y rol de usuario.

De lo anterior, se mencionan las siguientes funcionalidades asociadas al rol de administrador:

- Aprobar/consultar/rechazar/eliminar solicitudes
- Agregar/consultar/modificar/eliminar usuarios
- Agregar/consultar/modificar/bloquear/eliminar usuarios SIP
- Reiniciar servicios
- Cargar información hacia la plataforma integrada

Las funcionalidades descritas para el rol de usuario suscriptor se describen a continuación:

- Registro /modificación de información del usuario
- Consulta de cuenta SIP para la creación de enlace troncal desde su plataforma VoIP hacia la plataforma integrada Kamailio Asterisk.
- Creación y consulta de solicitudes y requerimientos a los administradores.
- Consulta de estado de su enlace troncal.

Teniendo a consideración lo anterior descrito, inicialmente fueron propuestas 12 vistas las cuales se mencionan a continuación:

- Registro de usuarios
- Registro de datos de la empresa
- Portal de autenticación
- Solicitud del código de marcado
- Página de inicio
- Portal de inicio para el usuario
- Edición de la información
- Consulta de datos para los archivos de configuración
- Vista de listado de usuarios y números asignados

- Procesamiento de solicitudes
- Estado de las troncales
- Registros exitosos

Finalmente se pudieron definir 6 flujos de trabajo en los que consistiría la aplicación.

3.7.5 Flujos de trabajo

3.7.5.1 Flujo de registro

Para poder iniciar un registro en nuestra plataforma el suscriptor debe ingresar a la aplicación desde su explorador web de preferencia. El flujo de registro inicia al solicitarle el correo electrónico al suscriptor. Por medio del correo electrónico el sistema es capaz de determinar si el usuario ya se encuentra registrado o se encuentra iniciando un nuevo proceso de registro. El registro en la aplicación web por correo es único. Como siguiente paso deben ser solicitados datos personales del usuario y datos de la organización que suscribe.

Como parte del proceso de registro le es solicitado al usuario, ingresar sugerencias de cual código de marcado desea utilizar.

Una vez completada la fase de recolección de datos se envían correos electrónicos al administrador de la plataforma y al usuario registrado indicándole sobre el registro culminado exitosamente. En esta notificación de correo electrónico, el usuario recibe un resumen de los datos registrados.

Adicionalmente, un usuario luego de terminar el registro, le será asignada una cuenta SIP generada automáticamente por la aplicación.

3.7.5.2 Flujo de trabajo para aprobación de solicitud

El proceso de aprobación de solicitudes debe ser realizado por un usuario administrador, el cual debe estar previamente autenticado en la aplicación.

El usuario administrador selecciona una solicitud de nuevo registro que desea para la modificación de su estado (pendiente por aprobar, aprobada o rechazada). Luego de seleccionarla se despliegan las opciones para el cambio de estado. Al modificar el estado de una solicitud se genera el envío de notificaciones vía correo electrónico indicando las modificaciones realizadas.

Una solicitud aprobada de nuevo ingreso a la aplicación implica en un nuevo acceso a la misma por parte de un usuario registrado. Por el contrario, si es rechazada el usuario no tendrá acceso a la aplicación.

3.7.5.3 Flujo de trabajo para la asignación de un código de marcado para una cuenta SIP del sistema

Luego de aprobada una solicitud de nuevo registro, una cuenta SIP es creada y asociada al usuario suscriptor. Una cuenta SIP aprobada es elegible para el proceso de asignación de un código de marcado por parte de un administrador.

El usuario administrador selecciona una cuenta SIP para su modificación. Luego de seleccionarla se despliegan las opciones para la asignación del código de marcado para esta cuenta. Dentro de estas opciones se muestran las sugerencias propuestas al momento del registro del usuario, para su asignación. El administrador selecciona la sugerencia del usuario o puede seleccionar un código de marcado perteneciente al conjunto de códigos de marcado disponibles para asignación. El código de marcado asignado debe ser visible al momento de realizar consultas o desplegar detalles informativos referentes a una cuenta SIP.

Las cuentas SIP que no posean el estado de aprobadas, no serán elegibles para la asignación de un código de marcado.

3.7.5.4 Flujo de trabajo de carga de usuarios aprobados a la plataforma remota Kamailio Asterisk

Todas aquellas cuentas SIP con estado de aprobado dentro del sistema, son elegibles como entrada en el proceso de carga de usuarios hacia la plataforma integrada, siendo este proceso ejecutado por un usuario administrador.

En este proceso se traspasan las cuentas SIP de los usuarios registrados a la base de datos de la plataforma integrada permitiendo así, el uso de esta información por parte de la plataforma.

Al finalizar el proceso de traspaso de usuarios debe ser desplegada una notificación que nos permita saber si el proceso ha concluido de manera satisfactoria.

3.7.5.5 Flujo de trabajo de generación y carga del plan de discado del sistema

Todas aquellas cuentas SIP con un código de marcado asociado y en estado de aprobada, son elegibles como entrada para el proceso de generación de plan de marcado y posterior carga en la plataforma remota, siendo este proceso ejecutado por el administrador.

A partir de las cuentas SIP con las condiciones descritas anteriormente, y con el uso de una plantilla, la generación de un plan de marcado basado en la sintaxis del archivo "extension.conf" es realizada por la aplicación web. Los datos utilizados por la plantilla son tomados de las cuentas SIP.

Luego de completar el proceso de generado del plan de marcado, se inicia un proceso de envío del archivo desde la aplicación web hacia la plataforma integrada de forma tal que este archivo pueda ser utilizado e implementado en la plataforma integrada para proveer las funciones de comunicación y envío de llamadas entre las cuentas SIP de los usuarios registrados.

3.7.5.6 Flujo de trabajo para el suscriptor

El flujo de trabajo del usuario suscriptor es un compendio de actividades mencionadas en los anteriores flujos, las cuales constan de:

Registro de usuario y aprobación de la solicitud de nuevo registro.

Una vez el usuario suscriptor ha pasado por ambos flujos, este podrá acceder a su área de trabajo en las que podrá consultar distintas informaciones relacionadas con su cuenta de usuario y cuenta SIP. Así como acceder a los datos necesarios para el establecimiento del enlace troncal desde el lado del servidor VoIP de su organización.

Un usuario suscriptor tiene permisos de modificación sobre información relativa a su perfil.

3.7.6 Resumen de flujos de trabajo

El siguiente cuadro resume una lista general de las funcionalidades requeridas en la aplicación (tabla 30). Esta lista general luego fue expandida en su nivel de detalle durante la generación del Product Backlog de nuestra aplicación.

Tabla 30

Lista general de funcionalidades de la aplicación web

Nombre de la funcionalidad	Descripción
Registrar usuarios	Se debe proveer las funcionalidades necesarias para realizar el registro de todos los posibles usuarios que muestren interés de suscribirse a la plataforma integrada.
Consultar usuarios	Una vez registrados los usuarios, se debe poder consultar los datos de estos en el sistema y también se debe poder modificar datos referentes a los usuarios.
Crear modulo del administrador	Se requiere de la creación de un perfil administrador y un área de trabajo asociado en la que el administrador pueda acceder, consultar, modificar y ejecutar las tareas definidas para este rol.
Cargar usuarios en la base de datos de la plataforma integrada	Los usuarios registrados y que cumplan con las condiciones, deben poder ser cargados en la base de datos de la plataforma integrada desde la aplicación web.
Generación y carga de plan de marcado	Aquellos usuarios que cuenten con un código de marcado asociado y se encuentren aprobados, deben poder seleccionarse para la generación del plan de marcado en la aplicación web a partir de sus datos y posteriormente enviar este archivo a la plataforma integrada en donde será utilizado por esta.
Generación y asignación de códigos de marcado	Durante el proceso de registro debe poder generarse códigos de marcado sugeridos por los usuarios para su posterior asignación.
Creación de las configuraciones necesarias para el establecimiento de enlaces troncales en la aplicación web	Una vez registrado un usuario, se deben crear automáticamente la información necesaria para el establecimiento de enlaces troncales y plan de marcado a partir de los datos ingresados por el usuario (cuenta SIP).

Fuente: Autor.

De los datos recolectados hasta este momento se generó suficiente información para la elaboración y especificación de los casos de uso de nuestra aplicación web para posteriormente realizar el Product Backlog de nuestra aplicación.

3.7.7 Análisis de requerimientos: Casos de uso

De la sección anterior se obtuvo el listado de las funcionalidades por flujo y una descripción de cada flujo que se desarrolla dentro de la aplicación. A partir de esta información se generó el diagrama de casos de uso. En la figura 85 se puede apreciar el diagrama de casos de uso nivel 1 de nuestra aplicación web.

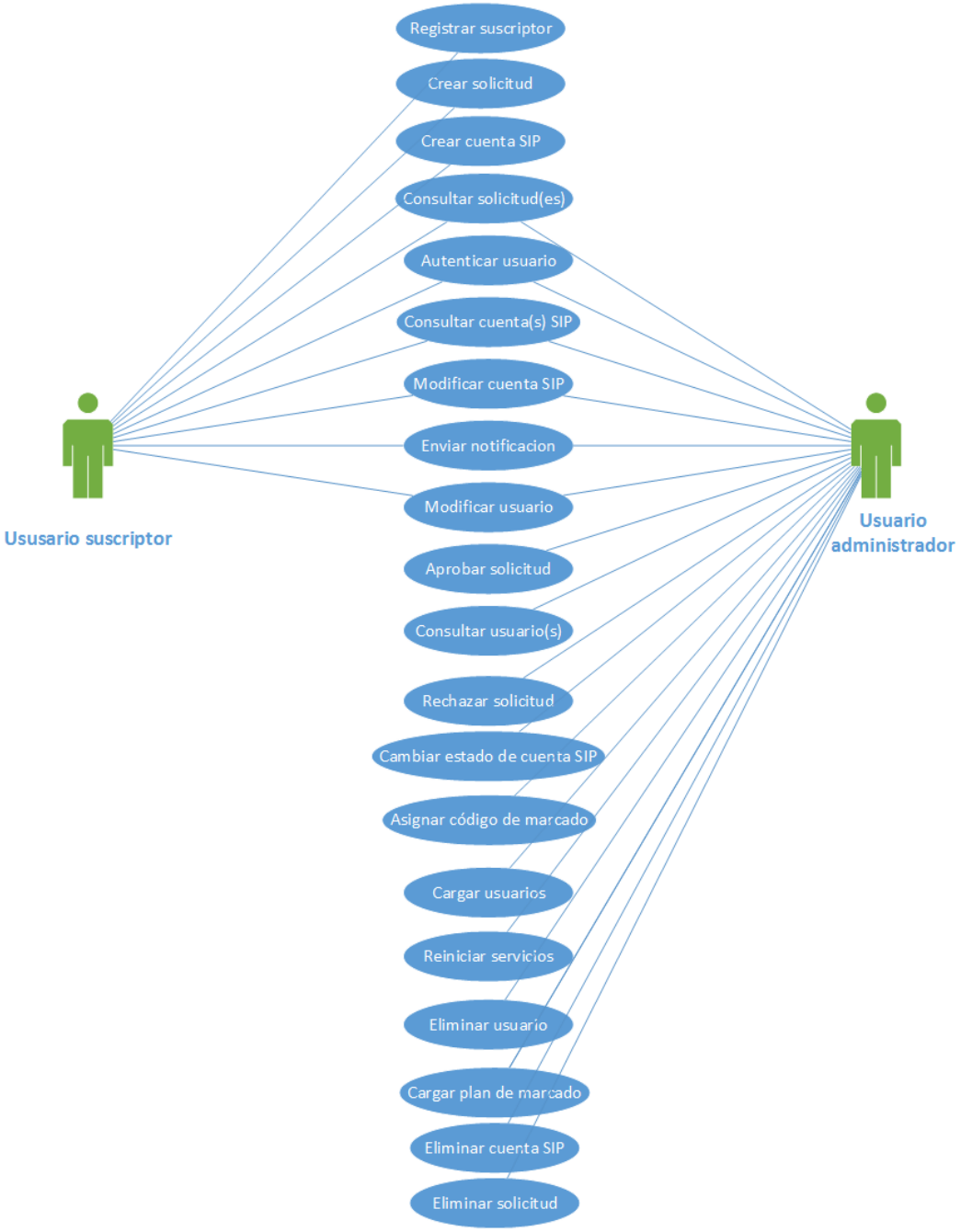

Figura 79***:*** Diagrama de casos de uso nivel 1.
Fuente: Autor.

De la figura anterior se identifican los actores usuario suscriptor y usuario administrador. Se definieron un total de 20 funcionalidades para los actores con sus respectivas asociaciones.

En la siguiente sección se encontrarán las especificaciones para las distintas funcionalidades definidas en nuestros casos de uso.

3.7.8 Especificación de casos de uso

En esta sección se describe cada caso de uso presente en nuestro diagrama de casos de uso nivel uno, generando un total de 20 especificaciones. Estas especificaciones pueden apreciarse en las consecuentes tablas de esta sección. Cada tabla contiene la siguiente información: Nombre del caso de uso, el o los actores que se relacionan con el caso de uso, el propósito del caso de uso, el tipo (principal o secundario), descripción, las referencias cruzadas con el caso de uso ("include" y "exclude"), curso normal de eventos en el cual se describen los pasos realizados por los actores y por el sistema y por ultimo un cuadro para describir si los hubiese, cursos alternativos.

3.7.8.1 Especificación de caso de uso: Registrar suscriptor

En la tabla 31 encontramos la especificación de casos de uso para "Registrar suscriptor".

Tabla 31
Especificación de caso de uso: Registrar suscriptor

Caso de uso	**Registrar suscriptor**
Actores	Usuario suscriptor, usuario administrador
Propósito	Registrar un posible usuario de la plataforma
Tipo	Principal
Descripción	El usuario ingresa en la sección de registros de la aplicación web, ingresa todos los datos requeridos por el sistema para la creación de un nuevo usuario dentro de la plataforma y al finalizar el proceso de registro es enviada una notificación al nuevo usuario y al administrador del aplicativo web.
Referencias Cruzadas	Modificar usuario (extend), Consulta usuario (extend), Crear solicitud (extend), Autenticar usuario (extend), Crear cuenta SIP (extend), Enviar notificación (extend), Eliminar usuario (extend).
Curso Normal de los Eventos	
Acción de los Actores	**Respuesta del Sistema**
1) El caso de uso inicia cuando el usuario accede a la sección de registro de la aplicación web.	
2) El usuario ingresa su correo electrónico para iniciar el proceso de registro.	3) El sistema verifica que el cliente no se encuentre registrado o que haya iniciado previamente un proceso de registro.
4) El usuario ingresa los datos solicitados en el formulario suministrado por el aplicativo web.	5) El sistema notificará al usuario los datos obligatorios que deberá ingresar.
6) El usuario ingresa también un posible número de marcado el cual puede ser asignado por los administradores del sistema.	

7) El usuario finaliza el registro.	8) El sistema le enviara una notificación vía correo electrónico al usuario con los datos suministrados.
	9) El sistema enviara una notificación a los usuarios administradores para procesar el nuevo registro.
Cursos Alternativos	
1) El usuario, si ya se encuentra registrado no podrá realizar un nuevo proceso de registro. **2)** El usuario, si ha iniciado con anterioridad un registro, pero este proceso no es completado, el usuario será redirigido al proceso de registro, pero con la información suministrada anteriormente.	

Fuente: Autor.

3.7.8.2 Especificación de caso de uso: Crear solicitud

En la tabla 32 encontramos la especificación de casos de uso para "Crear solicitud".

Tabla 32

Especificación de caso de uso: Crear solicitud

Caso de uso	**Crear solicitud**
Actores	Usuario suscriptor, usuario administrador
Propósito	Crear solicitudes dentro del sistema por parte del suscriptor que puedan ser atendidas por los administradores de la aplicación.
Tipo	Principal
Descripción	Un usuario luego de registrarse crea una solicitud de tipo "nuevo registro". Esta será procesada por los usuarios administradores a fin de otorgar acceso al usuario suscriptor en el sistema.
Referencias Cruzadas	Consultar solicitud (extend), Autenticar usuario (include), Aprobar solicitud (extend), Enviar notificación (extend), Rechazar solicitud (extend), Eliminar solicitud (extend).
Curso Normal de los Eventos	
Acción de los Actores	**Respuesta del Sistema**
1) El usuario completa exitosamente el proceso de registro en el sistema.	2) El sistema envía una notificación de nuevo registro al usuario suscriptor y al usuario administrador.
	3) Se crea automáticamente una solicitud para la aprobación o rechazo del nuevo usuario.
4) Un usuario administrador puede buscar la solicitud para aprobar o rechazar dicha solicitud	
Cursos Alternativos	
1) Las solicitudes si son rechazadas o aceptadas se enviarán notificaciones.	

Fuente: Autor.

3.7.8.3 Especificación de caso de uso: Crear cuenta SIP

En la tabla 33 encontramos la especificación de casos de uso para "Crear cuenta SIP".

Tabla 33
Especificación de caso de uso: Crear cuenta SIP.

Caso de uso	Crear cuenta SIP
Actores	Usuario suscriptor, usuario administrador
Propósito	Crear cuenta SIP que será utilizada para el establecimiento de enlaces troncales con la plataforma y asociación con códigos de marcado.
Tipo	Principal
Descripción	Un nuevo usuario, luego de finalizar correctamente el proceso de registro, una cuenta SIP es creada automáticamente a partir de los datos suministrados por el usuario. Esta cuenta luego podrá ser consultada modificada o eliminada y se le podrá asociar a las cuentas SIP creadas un código de marcado.
Referencias Cruzadas	Eliminar cuenta SIP (extend), Registrar suscriptor (include), Consultar cuenta SIP (extend), Enviar notificación (extend), Modificar cuenta SIP (extend), Cambiar estado de cuenta SIP (extend), Asignar código de marcado (extend)
Curso Normal de los Eventos	
Acción de los Actores	**Respuesta del Sistema**
1) El usuario completa exitosamente el proceso de registro en el sistema.	2) El sistema envía una notificación de nuevo registro al usuario suscriptor y al usuario administrador.
	3) Se crea automáticamente una cuenta SIP asociada al nuevo usuario.
4) Un usuario administrador puede modificar el estado de una cuenta SIP y asignarle un código de marcado a la cuenta SIP.	
5) El usuario suscriptor puede consultar la información de una cuenta SIP asociada para establecer el enlace troncal hacia la plataforma integrada.	
Cursos Alternativos	
1) Un usuario puede tener más de una cuenta SIP asociada. 2) En caso de no aprobarse una cuenta SIP no podrá asignársele un código de marcado.	

Fuente: Autor.

3.7.8.4 Especificación de caso de uso: Consultar solicitud

En la tabla 34 encontramos la especificación de casos de uso para "Consultar solicitud".

Tabla 34
Especificación de caso de uso: Consultar solicitud

Caso de uso	Consultar solicitud
Actores	Usuario suscriptor, usuario administrador
Propósito	Consultar información relacionada con solicitudes emitidas por un usuario o todos los usuarios del

	sistema dependiendo del rol con que se realicen las consultas.
Tipo	Secundaria
Descripción	Una solicitud, luego de ser creada, puede ser consultada tanto por el usuario que las crea como por parte de los usuarios administradores. Las solicitudes pueden consultarse para observar de manera resumida información que estas contengan.
Referencias Cruzadas	Crear solicitud (include), Autenticar usuario (include).
Curso Normal de los Eventos	
Acción de los Actores	**Respuesta del Sistema**
1) El usuario se autentica en la aplicación web.	2) El sistema valida los datos de autenticación suministrados por el usuario y consulta si es un usuario administrador o suscriptor.
3) El usuario selecciona el apartado de consultas y visualiza la información pertinente a las solicitudes emitidas por el usuario.	
4) El usuario puede filtrar solicitudes por el estado asociado.	
Cursos Alternativos	
1) Si el usuario es de tipo administrador, se desplegarán todas las solicitudes existentes en el sistema. 2) Si el usuario es de tipo suscriptor, el usuario solo podrá ver aquellas solicitudes asociadas con esa cuenta de usuario.	

Fuente: Autor.

3.7.8.5 Especificación de caso de uso: Autenticar usuario

En la tabla 35 encontramos la especificación de casos de uso para "Autenticar usuario".

Tabla 35

Especificación de caso de uso: Autenticar usuario

Caso de uso	Autenticar usuario
Actores	Usuario administrador, usuario suscriptor
Propósito	Validar y verificar que las credenciales suministradas por el usuario coincidan con las almacenadas.
Tipo	Principal
Descripción	El proceso de autenticación es necesario para realizar casi cualquier actividad relacionada con los usuarios, cuentas y consultas. El proceso de autenticación se realiza por medio de la comparación de las credenciales generadas durante el proceso de registro y suministradas por el usuario a autenticar.
Referencias Cruzadas	Registrar usuario (include), modificar usuario (extend), Crear solicitud (extend), Cambiar estado de cuenta SIP (extend), Consultar solicitud (extend), Reiniciar servicios (extend), Asignar código de marcado (extend), Rechazar solicitud (extend), Cargar usuarios (extend), Cargar plan de marcado (extend), Eliminar usuario (extend), Consultar cuenta SIP (extend), Eliminar solicitud (extend), Eliminar cuenta SIP (extend), Aprobar solicitud

	(extend), Consultar usuario (extend), Modificar cuenta SIP (extend).
Curso Normal de los Eventos	
Acción de los Actores	**Respuesta del Sistema**
1) El usuario culmina el proceso de registro exitosamente.	2) Se envía una notificación de nuevo registro al usuario y al administrador.
3) El usuario administrador consulta la solicitud y la aprueba.	4) El usuario es notificado que su solicitud ha sido aprobada.
5) El usuario suscriptor se autentica en la aplicación web y obtiene acceso a su perfil de usuario y a las funcionalidades relacionadas a su rol.	
Cursos Alternativos	
1) Si el usuario no ha sido aprobado, el usuario no podrá acceder a la aplicación web.	

Fuente: Autor.

3.7.8.6 Especificación de caso de uso: Consultar cuenta SIP

En la tabla 36 encontramos la especificación de casos de uso para "Consultar cuenta SIP".

Tabla 36

Especificación de caso de uso: Consultar cuenta SIP

Caso de uso	**Consultar cuenta SIP**
Actores	Usuario suscriptor, usuario administrador
Propósito	Desplegar información relacionada a la cuenta SIP
Tipo	Secundario
Descripción	Luego de autenticarse cualquiera de los dos usuarios puede consultar información sobre la cuenta SIP asociada con el perfil, para el caso del administrador este podrá consultar todas las cuentas del sistema y en el caso de un usuario suscriptor, este solo podrá acceder a la información de la cuenta SIP relacionada con su cuenta de usuario.
Referencias Cruzadas	Crear cuenta SIP (include), autenticar usuario (include)
Curso Normal de los Eventos	
Acción de los Actores	**Respuesta del Sistema**
1) El usuario se autentica en la aplicación web administrativa.	2) El sistema valida y verifica que la información suministrada en la autenticación coincida con la información almacenada.
3) El usuario accede a la sección de consulta de cuentas SIP.	
Cursos Alternativos	
1) Si el usuario es de tipo administrador se desplegarán todas las cuentas SIP del sistema. 2) Si el usuario es de tipo suscriptor solo se desplegará la información de la cuenta o las cuentas SIP relacionada con esa cuenta de usuario.	

Fuente: Autor.

3.7.8.7 Especificación de caso de uso: Aprobar solicitud

En la tabla 37 encontramos la especificación de casos de uso para "Aprobar solicitud".

Tabla 37

Especificación de caso de uso: Aprobar solicitud.

Caso de uso	**Aprobar solicitud**
Actores	Usuario administrador
Propósito	Aceptar solicitudes de nuevos registros por parte de un suscriptor
Tipo	Primaria
Descripción	Luego de haber finalizado el registro de manera exitosa, un usuario suscriptor crea de manera automática una solicitud de nuevo registro. El usuario administrador es notificado de esta y debe verificar la información de la solicitud a fin de aprobar la creación de una cuenta de usuario, una cuenta SIP y el acceso de un suscriptor a la aplicación web.
Referencias Cruzadas	Autenticar usuario (include), crear solicitud (include), enviar notificación (extend), Cargar usuarios (extend), Asignar código de marcado (extend).
Curso Normal de los Eventos	
Acción de los Actores	**Respuesta del Sistema**
1) El usuario finaliza el registro de manera exitosa	2) El usuario administrador y el usuario suscriptor son notificados ante un nuevo registro.
3) El usuario administrador filtra la búsqueda de solicitudes pendientes por aprobar. Selecciona la solicitud. Verifica la información presente en la solicitud. Si todo se encuentra en orden la solicitud es aprobada.	4) El sistema envía una notificación de que la solicitud emitida por el usuario suscriptor ha sido aprobada.
5) El usuario puede autenticarse y acceder a la aplicación web.	
Cursos Alternativos	
1) Si una solicitud de nuevo registro en la aplicación es negada, el usuario no podrá autenticarse y acceder a la aplicación.	

Fuente: Autor.

3.7.8.8 Especificación de caso de uso: Consultar usuario

En la tabla 38 encontramos la especificación de casos de uso para "Consultar usuario".

Tabla 38

Especificación de caso de uso: Consultar usuario

Caso de uso	**Consultar usuario**
Actores	Usuario administrador
Propósito	Desplegar información visual del usuario registrado en el sistema.
Tipo	Secundario
Descripción	Un usuario administrador luego de autenticarse en el sistema, puede realizar consultas generales para apreciar información de los usuarios registrados del sistema (datos, estado, organización, cuenta SIP, etc.)

Referencias Cruzadas	Autenticar usuario (include), registrar suscriptor (include)
Curso Normal de los Eventos	
Acción de los Actores	**Respuesta del Sistema**
1) El usuario administrador se autentica en la aplicación web.	2) El sistema valida y verifica que las credenciales coincidan para otorgar el acceso al perfil de administrador.
3) El usuario administrador accede a la sección de usuarios en donde podrá realizar las respectivas consultas de los usuarios registrados en el sistema.	
Cursos Alternativos	

Fuente: Autor.

3.7.8.9 Especificación de caso de uso: Modificar cuenta SIP

En la tabla 39 encontramos la especificación de casos de uso para "Modificar cuenta SIP".

Tabla 39

Especificación de caso de uso: Modificar cuenta

Caso de uso	**Modificar cuenta SIP**
Actores	Usuario administrador, usuario suscriptor
Propósito	Cambiar valores de una determinada cuenta SIP.
Tipo	Secundaria (Importante)
Descripción	Un usuario autenticado puede modificar algunos valores de la cuenta SIP creada automáticamente por el sistema y asociada con un suscriptor.
Referencias Cruzadas	Crear cuenta SIP (include), Autenticar usuario (include), Cambiar estado de cuenta SIP (extend), Asignar código de marcado (extend).
Curso Normal de los Eventos	
Acción de los Actores	**Respuesta del Sistema**
1) El usuario se autentica en la aplicación web.	2) El sistema valida y verifica que las credenciales coincidan para otorgar el acceso al perfil de administrador.
3) El usuario accede a la sección de cuentas SIP y busca la cuenta que desea modificar.	
4) El usuario ingresa a la vista de modificación de la cuenta SIP donde se desplegará la información de la cuenta SIP y los valores que puede actualizar o modificar.	5) El sistema almacena los cambios e informa si los cambios se han guardado con éxito.
Cursos Alternativos	
1) Un usuario administrador puede asignarle un código de marcado a una cuenta SIP si esta se encentra en estado de aprobada, para ello debe acceder a la sección de modificación de la cuenta SIP en cuestión.	

Fuente: Autor.

3.7.8.10 Especificación de caso de uso: Enviar notificación

En la tabla 40 encontramos la especificación de casos de uso para "Enviar notificación".

Tabla 40
Especificación de caso de uso: Enviar notificación

Caso de uso	**Enviar notificación**
Actores	Usuario administrador, usuario suscriptor.
Propósito	Enviar vía notificaciones al usuario administrador o al usuario suscriptor
Tipo	Secundaria (importante)
Descripción	Las notificaciones son enviadas al finalizar procesos de nuevo registro, aprobación o rechazo de solicitudes, bloqueo de usuarios etc. Estas son enviadas utilizando correo electrónico y son destinadas según sea el caso al usuario suscriptor o al usuario administrador.
Referencias Cruzadas	Aprobar solicitud (extend), rechazar solicitud (extend), registro de suscriptor (extend), crear solicitud (extend).
Curso Normal de los Eventos	
Acción de los Actores	**Respuesta del Sistema**
1) El usuario finaliza uno de los siguientes procesos: Registro exitoso, crear cuenta de usuario, aprobar solicitud, rechazar una solicitud	2) El sistema envía la notificación mediante correo electrónico.
Cursos Alternativos	
1) Otras notificaciones son desplegadas por pantalla al usuario desde la aplicación web.	

Fuente: Autor.

3.7.8.11 Especificación de caso de uso: Modificar usuario

En la tabla 41 encontramos la especificación de casos de uso para "Modificar usuario".

Tabla 41
Especificación de caso de uso: Modificar usuario

Caso de uso	**Modificar usuario**
Actores	Usuario administrador, usuario suscriptor
Propósito	Modificar valores permitidos de la cuenta de acceso a la aplicación web
Tipo	Secundario
Descripción	Un usuario autenticado puede acceder a su perfil de usuario y modificar los valores permitidos entre ellos la contraseña de acceso a la aplicación web.
Referencias Cruzadas	Registrar usuario (include), Autenticar usuario (include)
Curso Normal de los Eventos	
Acción de los Actores	**Respuesta del Sistema**
1) El usuario autenticado accede a la información de su perfil.	
2) El usuario modifica los valores permitidos por el sistema.	3) El sistema guarda los cambios realizados por el usuario en su perfil.
Cursos Alternativos	

1) Un usuario administrador puede modificar los datos de cualquier usuario registrado en el sistema.

Fuente: Autor.

3.7.8.12 Especificación de caso de uso: Rechazar solicitud

En la tabla 42 encontramos la especificación de casos de uso para "Rechazar solicitud".

Tabla 42

Especificación de caso de uso: Rechazar solicitud

Caso de uso	**Rechazar solicitud**
Actores	Usuario administrador
Propósito	Rechazar solicitudes de nuevos registros por parte de un suscriptor.
Tipo	Secundaria (importante)
Descripción	Un usuario administrador luego de autenticarse puede consultar y rechazar una solicitud de nuevo registro.
Referencias Cruzadas	Crear solicitud (include), enviar notificación (extend), autenticar usuario (include)
Curso Normal de los Eventos	
Acción de los Actores	**Respuesta del Sistema**
	1) El sistema notifica a los usuarios administradores sobre las solicitudes pendientes.
2) Un usuario administrador se autentica en la aplicación web.	3) El sistema valida y verifica que las credenciales coincidan para otorgar el acceso al perfil de administrador.
4) El usuario administrador selecciona una solicitud pendiente por aprobar para procesarla.	
5) El usuario administrador rechaza la solicitud.	6) El sistema envía una notificación al usuario suscriptor de solicitud rechazada.
Cursos Alternativos	
1) Rechazar solicitudes de tipo nuevo registro implica que el usuario suscriptor propietario de la solicitud no tendrá acceso a la aplicación web.	

Fuente: Autor.

3.7.8.13 Especificación de caso de uso: Cambiar estado de cuenta SIP

En la tabla 43 encontramos la especificación de casos de uso para "Cambiar estado de cuenta SIP".

Tabla 43

Especificación de caso de uso: Cambiar estado de cuenta SIP

Caso de uso	**Cambiar estado de cuenta SIP**
Actores	Usuario Administrador
Propósito	Cambiar el estado asociado a una cuenta SIP en particular (bloqueada, aprobada, en proceso, rechazada)

Tipo	Secundaria
Descripción	Las cuentas SIP, aunque asociadas a las cuentas de usuarios suscriptores, pueden poseer estados distintos a las cuentas de suscriptores. Una cuenta SIP puede tener alguno de los siguientes estados: Aprobada, en proceso, rechazada o bloqueada.
Referencias Cruzadas	Autenticar usuario (include), Crear cuenta SIP (include), Cargar usuarios (extend), Asignar código de marcado (include).
Curso Normal de los Eventos	
Acción de los Actores	**Respuesta del Sistema**
1) Un usuario administrador se autentica en la aplicación web.	2) El sistema valida y verifica que las credenciales coincidan para otorgar el acceso al perfil de administrador.
3) El usuario administrador selecciona la sección de cuentas SIP y selecciona alguna de su interés.	
4) El usuario administrador modifica el estado asociado a la cuenta SIP de algún usuario suscriptor.	5) El sistema basado en el cambio de estado realiza las modificaciones respectivas para cuenta SIP del usuario suscriptor.
Cursos Alternativos	
1) Si la cuenta SIP no posee un estado de aprobada, la cuenta no será elegible para asignársele un código de marcado.	

Fuente: Autor.

3.7.8.14 Especificación de caso de uso: Asignar código de marcado

En la tabla 44 encontramos la especificación de casos de uso para "Asignar código de marcado".

Tabla 44

Especificación de caso de uso: Asignar código de marcado

Caso de uso	**Asignar código de marcado**
Actores	Usuario administrador
Propósito	Proporcionarle a una cuenta SIP un código de marcado para realizar y recibir llamadas desde y hacia la plataforma integrada
Tipo	Primaria
Descripción	Luego de ser aprobada una cuenta SIP es candidata para la asignación de un código de marcado, este puede ser proporcionado por el usuario o asignado arbitrariamente por el administrador.
Referencias Cruzadas	Autenticar usuario (include), Aprobar solicitud (include), Cambiar estado de cuenta SIP (include), Cargar plan de marcado (extend).
Curso Normal de los Eventos	
Acción de los Actores	**Respuesta del Sistema**
1) Un usuario administrador se autentica en la aplicación web.	2) El sistema valida y verifica que las credenciales coincidan para otorgar el acceso al perfil de administrador.
3) El usuario selecciona la cuenta SIP con estado de aprobado.	
4) Se desplegará la sugerencia del código de marcado del usuario además de	5) El sistema guarda los cambios realizados y emite una solicitud para indicar que el

desplegar la lista de todos los números de marcado disponibles en el sistema. El administrador selecciona uno y guarda.	proceso de ha realizado de manera satisfactoria.
Cursos Alternativos	
1) Si una cuenta SIP no se encuentra aprobada no se le podrá asignar un código.	

Fuente: Autor.

3.7.8.15 Especificación de caso de uso: Cargar usuarios

En la tabla 45 encontramos la especificación de casos de uso para "Cargar usuarios".

Tabla 45
Especificación de caso de uso: Cargar usuarios

Caso de uso	**Cargar usuarios**
Actores	Usuario administrador
Propósito	Volcar los usuarios registrados que cumplan con las condiciones adecuadas desde la aplicación a la plataforma integrada.
Tipo	Principal
Descripción	Todas aquellas cuentas SIP que posean el estado de aprobado sirven como entrada para el proceso de volcado de estas cuentas hacia la base de datos de la plataforma integrada, de forma tal que estas cuentas creadas en la plataforma puedan ser utilizadas para establecer los enlaces troncales con las plataformas VoIP de los suscriptores.
Referencias Cruzadas	Cambiar estado de la cuenta SIP (include), Aprobar solicitud (include), Autenticar usuario (include), Registrar suscriptor (include).
Curso Normal de los Eventos	
Acción de los Actores	**Respuesta del Sistema**
1) Un usuario administrador se autentica en la aplicación web.	2) El sistema valida y verifica que las credenciales coincidan para otorgar el acceso al perfil de administrador.
3) El administrador inicia el proceso de volcado desde la aplicación web.	4) El sistema se encarga de seleccionar todas las cuentas SIP que cumplan con el estado de aprobada para crear las nuevas entradas en la base de datos de la plataforma integrada. Luego, muestra una notificación al usuario si se completó el proceso exitosamente.
Cursos Alternativos	
1) Este caso de uso puede ser integrado con cargar plan de marcado para ejecutar ambas actividades de una sola vez y de manera secuencial.	

Fuente: Autor.

3.7.8.16 Especificación de caso de uso: Cargar plan de marcado

En la tabla 46 encontramos la especificación de casos de uso para "Cargar plan de marcado".

Tabla 46

Especificación de caso de uso: Cargar plan de marcado

Caso de uso	Cargar plan de marcado
Actores	Administrador
Propósito	Generar y cargar el plan de marcado que utilizara la plataforma integrada para redirigir llamadas.
Tipo	Principal
Descripción	La aplicación web, a partir de los usuarios aprobados y que posean código de marcado asociado, genera un plan de marcado utilizando la sintaxis utilizada por Asterisk para extensión.conf y luego enviarlo a la plataforma integrada de forma que este pueda ser utilizado redirigir las llamadas entre los suscriptores.
Referencias Cruzadas	Autenticar usuario (include), asignar código de marcado (include)
Curso Normal de los Eventos	
Acción de los Actores	**Respuesta del Sistema**
1) Un usuario administrador se autentica en la aplicación web.	2) El sistema valida y verifica que las credenciales coincidan para otorgar el acceso al perfil de administrador.
3) El administrador inicia el proceso de generación del plan de marcado y envío hacia la plataforma integrada	4) El sistema a partir de las cuentas que tengan asignado un código de marcado, generara un plan de marcado basado en el archivo de configuración extensión.conf de Asterisk y luego envía el archivo a la plataforma integrada.
Cursos Alternativos	
1) Este caso de uso puede ser integrado con cargar plan de marcado para ejecutar ambas actividades de una sola vez y de manera secuencial.	

Fuente: Autor.

3.7.8.17 Especificación de caso de uso: Eliminar usuario

En la tabla 47 encontramos la especificación de casos de uso para "Eliminar usuario".

Tabla 47

Especificación de caso de uso: Eliminar usuario

Caso de uso	Eliminar usuario
Actores	Usuario administrador
Propósito	Eliminar de la base de datos al usuario suscriptor
Tipo	Secundaria
Descripción	Un usuario puede eliminar de manera permanente a un usuario.
Referencias Cruzadas	Autenticar usuario (include), registrar suscriptor (include)
Curso Normal de los Eventos	
Acción de los Actores	**Respuesta del Sistema**
1) Un usuario administrador se autentica en la aplicación web.	2) El sistema valida y verifica que las credenciales coincidan para otorgar el acceso al perfil de administrador.
3) El usuario selecciona al usuario de que desea eliminar de la aplicación web y selecciona la opción para ello. Se	4) El sistema eliminara el registro de la base de datos del sistema y le notificara al usuario

mostrará un mensaje de confirmación antes de borrar al usuario.	que el registro ha sido eliminado satisfactoriamente.
Cursos Alternativos	

Fuente: Autor.

3.7.8.18 Especificación de caso de uso: Eliminar cuenta SIP

En la tabla 48 encontramos la especificación de casos de uso para "Eliminar cuenta SIP".

Tabla 48

Especificación de caso de uso: Eliminar cuenta SIP

Caso de uso	Eliminar cuenta SIP
Actores	Usuario administrador
Propósito	Eliminar de la base de datos una cuenta SIP
Tipo	Secundaria
Descripción	Un usuario administrador puede eliminar una cuenta SIP del sistema si esta no cumple con los requerimientos.
Referencias Cruzadas	Autenticar usuario (include), crear cuenta SIP (include)
Curso Normal de los Eventos	
Acción de los Actores	**Respuesta del Sistema**
1) Un usuario administrador se autentica en la aplicación web.	2) El sistema valida y verifica que las credenciales coincidan para otorgar el acceso al perfil de administrador.
3) El usuario selecciona la cuenta SIP que desea eliminar de la aplicación web y selecciona la opción para ello. Se mostrará un mensaje de confirmación antes de borrar esta cuenta SIP.	4) El sistema eliminara el registro de la base de datos del sistema y le notificara al usuario que el registro ha sido eliminado satisfactoriamente.
Cursos Alternativos	

Fuente: Autor.

3.7.8.19 Especificación de caso de uso: Eliminar solicitud

En la tabla 49 encontramos la especificación de casos de uso para "Eliminar solicitud".

Tabla 49

Especificación de caso de uso: Eliminar solicitud

Caso de uso	Eliminar solicitud
Actores	Usuario administrador
Propósito	Eliminar de la base de datos la solicitud generada por un usuario suscriptor
Tipo	Secundaria
Descripción	Un usuario administrador puede eliminar una solicitud del sistema si esta no cumple con los requerimientos.

Referencias Cruzadas	Autenticar usuario (include), crear solicitud (include)
Curso Normal de los Eventos	
Acción de los Actores	**Respuesta del Sistema**
1) Un usuario administrador se autentica en la aplicación web.	2) El sistema valida y verifica que las credenciales coincidan para otorgar el acceso al perfil de administrador.
3) El usuario selecciona al usuario de que desea eliminar de la aplicación web y selecciona la opción para ello. Se mostrará un mensaje de confirmación antes de borrar al usuario.	4) El sistema eliminara el registro de la base de datos del sistema y le notificara al usuario que el registro ha sido eliminado satisfactoriamente.
Cursos Alternativos	

Fuente: Autor.

3.7.8.20 Especificación de caso de uso: Reiniciar servicios

En la tabla 50 encontramos la especificación de casos de uso para "Reiniciar servicios".

Tabla 50
Especificación de caso de uso: Reiniciar servicios

Caso de uso	**Reiniciar servicios**
Actores	Usuario administrador
Propósito	Reiniciar los servicios de Asterisk en la plataforma integrada
Tipo	Primaria
Descripción	Luego de cargar el plan de marcado y los usuarios, es necesario realizar el reinicio o recarga de ciertos módulos presentes en Asterisk.
Referencias Cruzadas	Autenticar usuario (include)
Curso Normal de los Eventos	
Acción de los Actores	**Respuesta del Sistema**
1) Un usuario administrador se autentica en la aplicación web.	2) El sistema valida y verifica que las credenciales coincidan para otorgar el acceso al perfil de administrador.
3) El usuario administrador ejecuta la función desde la aplicación web.	4) El sistema será el encargado de realizar el envío de los comandos para reiniciar los componentes necesarios en la plataforma integrada. Este le informara al usuario administrador que fueron reiniciados los servicios exitosamente.
Cursos Alternativos	

Fuente: Autor.

Con este último caso de uso se concluye la sección de especificación de casos de uso.

3.7.9 Diagrama de secuencia

Fue necesario incluir entre la documentación de la aplicación un diagrama de secuencia donde se muestre el pase de mensajes entre actores del sistema, aplicación y plataforma integrada de tal forma que puedan identificarse más claramente las funcionalidades de la aplicación y su interacción con

los elementos externos (actores y plataforma integrada). El diagrama de actividades puede apreciarse en la figura 6.

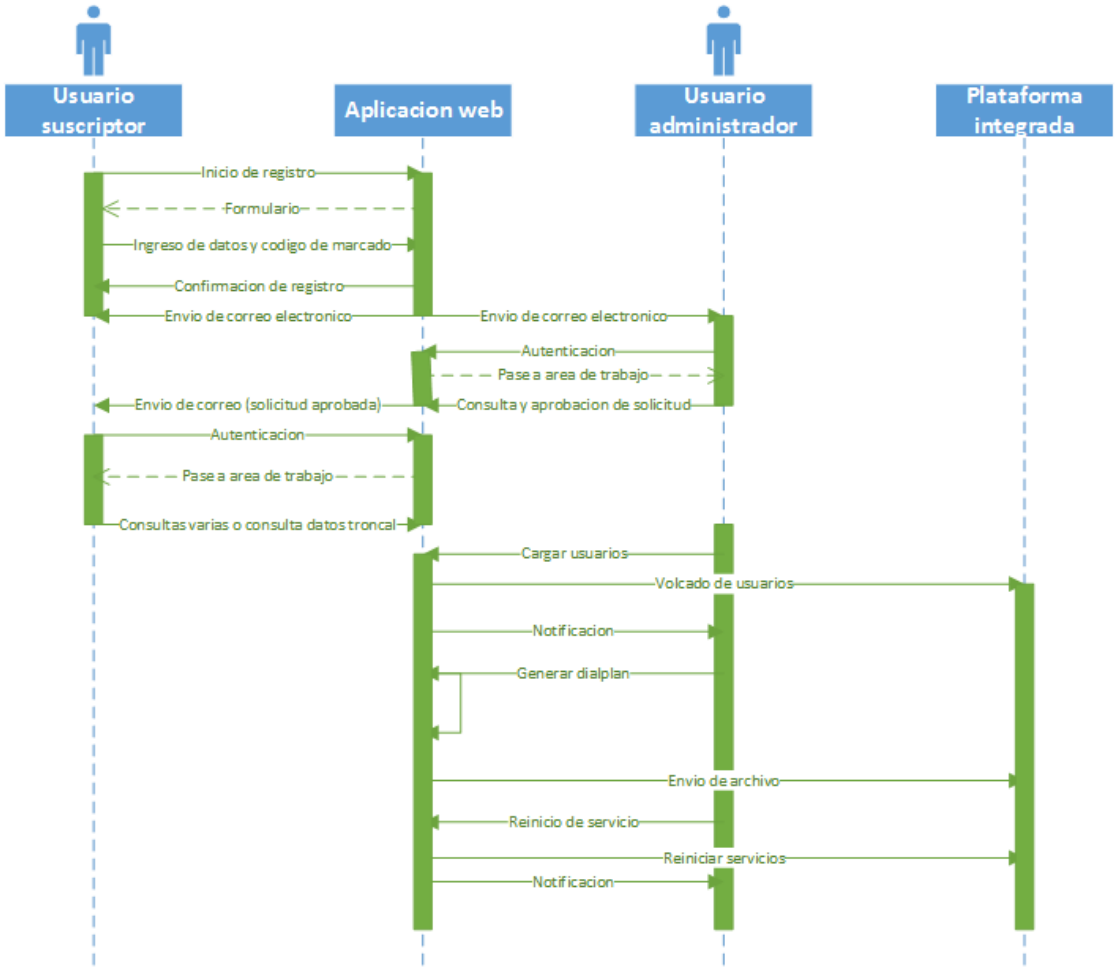

Figura 80***:*** Diagrama de secuencia general.
Fuente: Autor.

Luego de la finalización de los eventos ilustrados en la figura 86, el usuario suscriptor debe contar con un enlace establecido entre la plataforma y su servidor VoIP (de no existir complicaciones por firewalls presentes en la arquitectura).

En la siguiente sección, se especifica el product backlog que resulto del desarrollo realizado para el presente trabajo.

3.7.10 Product Backlog

De la anterior etapa de análisis de requerimientos, se extrajeron la mayor cantidad de detalles y funcionalidades de tal forma que las tareas o funcionalidades fueron en la medida de lo posible descompuestas en micro tareas para una asignación más atómica a los Sprint backlog durante los Sprint

planning. Del proceso anterior se obtuvo como resultado el product backlog que fue utilizado para el desarrollo de la aplicación web.

En la tabla 51 se podrá encontrar el listado de tareas que describen el product backlog y la asignación de estas a los distintos sprint que fueron necesarios para implementar para generar nuestra aplicación.

Tabla 51
Product backlog y sprint backlog

Sprint	Identificador	Nombre
1	1	Instalación de ambiente de desarrollo
	2	Configuración del ambiente de desarrollo
	3	Creación de la base de datos para el proyecto y usuario administrador de la base de datos
	4	Instalación de Laravel Collection
	5	Diseño de diagrama Entidad-Relación
	6	Especificación de estados asociados con usuarios, cuentas SIP y solicitudes
2	7	Implementación del diagrama Entidad-Relación
	8	Generación e implantación de archivos migration en Laravel para implementar el diagrama entidad-relación
	9	Generación de archivos model en Laravel para el acceso a los datos de las entidades y especificar sus relaciones
	10	Verificación de acceso a los datos y obtención de los mismos
	11	Crear los controladores necesarios para las distintas funcionalidades o entidades
3	12	Definición de la paleta de colores a utilizar en el diseño de las vistas
	13	Generación del layout para las vistas con Blade
	14	Definir rutas de acceso para la página de inicio y para el módulo de registro
	15	Creación de página de inicio
4	16	Desarrollo de vista para manejar los registros de suscriptores
	17	Generar vista de solicitud de correo electrónico para iniciar proceso de registro
	18	Generar vista para el registro de datos de usuario
	19	Generar vista para el registro de datos organizacionales
	20	Generar vista para el registro de la dirección de la organización
	21	Generar vista para la especificación del código de macado
	22	Generar vista de proceso de registro culminado con éxito
5	23	Desarrollar lógica de generación de sugerencias de código de marcado

	24	Crear función para la gestión del manejo de registro paginado del suscriptor
	25	Realizar las respectivas validaciones para el registro de suscriptores o modificación de información de las diferentes vistas
	26	Crear función de creación de organización
	27	Crear función de creación de dirección
	28	Crear función de creación de cuenta SIP
	29	Crear función de creación de nueva solicitud
6	30	Configurar Laravel para el envío de correos electrónicos
	31	Funcionalidad para enviar correo electrónico y nuevo registro
	32	Generar layout para el envío de correos electrónicos
	33	Crear función de envió de correo electrónico
7	34	Implementación del módulo de autenticación en Laravel para los usuarios
	35	Generar vista de solicitud de datos de autenticación
	36	Aplicación del middleware AUTH para los componentes que requieran autenticación
8	37	Generar vista de área de trabajo para usuario
	38	Definir rutas de acceso para las áreas de trabajo y secciones relacionadas
	39	Generar vista de área de trabajo para administrador
	40	Generar vista de administración de solicitudes
	41	Generar vista de administración de troncales
	42	Generar vista de administración de usuarios
9	43	Paginar resultados en las vistas de consulta
	44	Función de consulta de usuarios
	45	Función de consulta de cuentas SIP
	46	Función de consulta de solicitudes
	47	Función de eliminar usuarios
	48	Función para eliminar solicitudes
	49	Función para eliminar cuentas SIP
10	50	Generar vista de modificación de solicitud
	51	Generar vista de modificación de usuario
	52	Generar vista de modificación de cuenta SIP
	53	Realizar las respectivas validaciones para el registro de suscriptores o modificación de información de las diferentes vistas
11	54	Función resumen de solicitudes
	55	Función resumen de usuarios
	56	Generar vista de detalles de cuenta SIP
	57	Función de asignación de código de marcado
	58	Función de cambio de código de marcado

	59	Función cambio de estado de una cuenta SIP
12	60	Instalación del componente Laravel collective remote
	61	Configuración del perfil remote para el acceso a la plataforma remota
	62	Configuración de segundo perfil en Laravel para el acceso a la base de datos de la plataforma integrada
	63	Modificar base de datos de la plataforma integrada para el funcionamiento con Eloquent
	64	Modificar el archivo de configuración extensions.conf para agregar de manera dinámica el plan de marcado generado en la aplicación web
	65	Función Información del servidor
	66	Función estado de la troncal del usuario
	67	Crear función para el volcado de datos de usuario en la base de datos de la plataforma remota
	68	Crear función de generación de plan de marcado
	69	Crear función para el envío del archivo que contiene el código de marcado a la plataforma integrada
	70	Crear función para el reinicio de los servicios de la plataforma integrada.

Fuente: Autor.

El product backlog tomó un total de 12 Sprints para culminar totalmente con las especificaciones- Las demostraciones realizadas al product owner, fueron hechas en cada Sprint review donde se podían apreciar los incrementos del producto para ese determinado Sprint. Cada Sprint tuvo una duración de 2 semanas. A continuación, se describen las actividades realizadas por sprint.

3.7.11 Sprint 1

Para el desarrollo del Sprint 1 se contemplaron las tareas relacionadas con la instalación y adecuación de la plataforma de desarrollo y algunos otros aspectos en cuanto a diseño de la aplicación web como es el caso del diagrama entidad relación utilizado para el almacenamiento persistente de los datos de la aplicación. El listado de las tareas realizadas durante el desarrollo del Sprint 1 se pueden apreciar en la tabla 52.

Tabla 52

Tareas asignadas al Sprint 1

Sprint	Identificador	Nombre
1	1	Instalación de ambiente de desarrollo
	2	Configuración del ambiente de desarrollo
	3	Creación de la base de datos para el proyecto y usuario administrador de la base de datos
	4	Instalación de Laravel Collection
	5	Diseño de diagrama Entidad-Relación

	6	Especificación de estados asociados con usuarios, cuentas SIP y solicitudes

Fuente: Autor.

El despliegue del ambiente de desarrollo consistió en la instalación del paquete de software XAMPP (Miltiplataforma, Apache, MariaDB, PHP y Perl). Este paquete de software nos suministra casi todo lo necesario para el establecimiento de un ambiente local para el desarrollo ya que nos provee el servidor web apache y la base de datos MySQL. Fue necesario la instalación de Composer, un manejador de dependencias PHP para la posterior instalación del framework Laravel, en donde realizamos finalmente el desarrollo de la aplicación.

En Laravel se realizaron las configuraciones de lenguajes necesarias para desplegar mensajes de error y notificación para el usuario en español (utilizado por el módulo de validaciones de Laravel). Además, se realizó la configuración de las credenciales para el acceso a la base de datos que utiliza nuestra aplicación.

Estas configuraciones fueron realizadas en el archivo "App.php" en el directorio "/Config/" de nuestro directorio de proyecto en Laravel.

Fue necesario configurar las variables de entorno en el archivo ".env" ubicado en la raíz de nuestro proyecto. Las variables quedaron de la manera que se puede apreciar en la figura 87, indicando valores de usuario, contraseña, base de datos a utilizar y dirección de la base de datos.

```
6   DB_CONNECTION=mysql
7   DB_HOST=127.0.0.1
8   DB_PORT=3306
9   DB_DATABASE=teg
10  DB_USERNAME=teg
11  DB_PASSWORD=teg
```

Figura 81*:* Extracto de código archivo ".env"
Fuente: Archivo ".env"

Se utilizaron los datos definidos en las variables de entorno del archivo ".env" para crear el usuario, contraseña y base de datos a nivel local.

Como ultima instalación de componentes para el ambiente de desarrollo para este Sprint se realizó la instalación de Laravel collective, siendo este un conjunto de herramientas que nos fue de utilidad a lo largo del desarrollo de nuestra aplicación.

3.7.11.1 Diseño de diagrama Entidad-Relación

Como parte de los requerimientos, fue necesario definir el diagrama entidad relación para la representación de la base de datos que da soporte a nuestra aplicación. Este diagrama contempló un total de 9 entidades de las cuales 2 (SIPRegs y SIPUsers) son representaciones de las tablas creadas y utilizadas en la plataforma integrada. En nuestro diagrama podemos observar la entidad Form sin relación con demás tablas, esto es debido a que esta tabla es usada exclusivamente para llevar el registro de usuarios. Esta última entidad nos sirve como tabla auxiliar para el posterior traspaso de datos desde sus registros a entidades por separado. El resto del diagrama puede observarse en la figura 88.

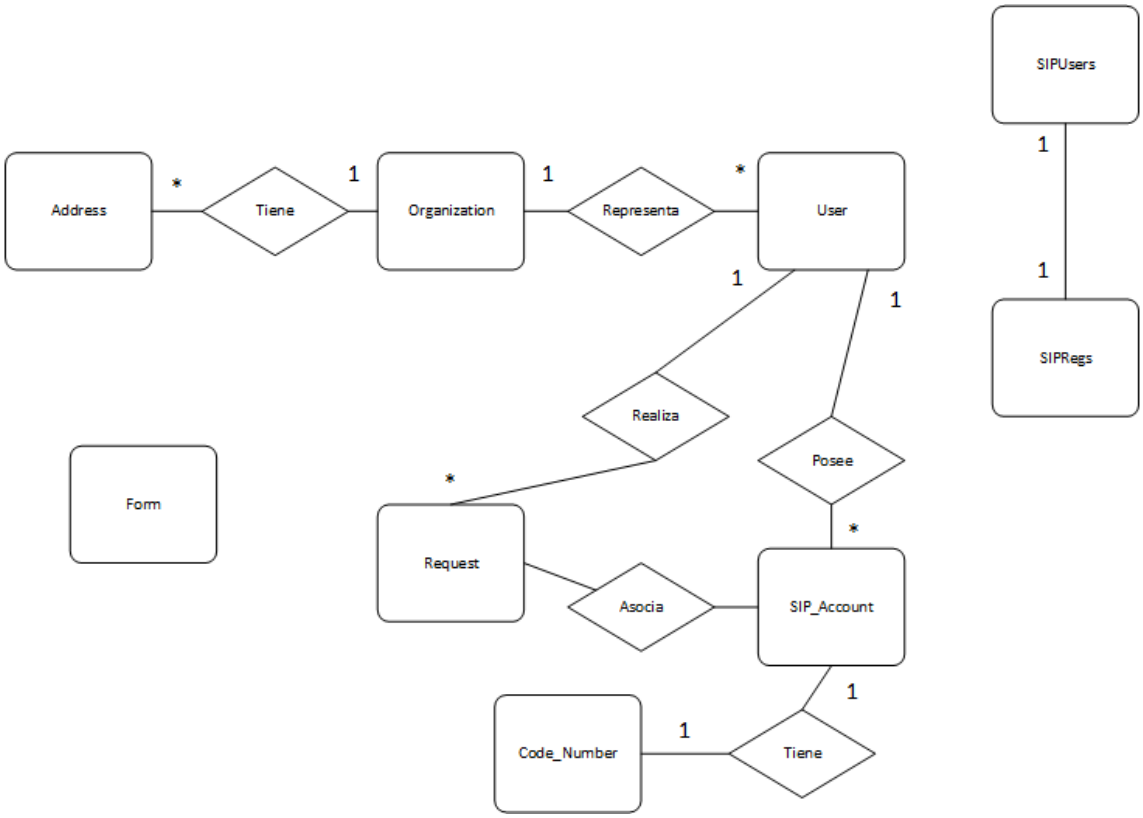

Figura 82***:*** Diagrama entidad-relación.
Fuente: Autor.

Para cada una de las entidades a excepción de las entidades SIPRegs y SIPUsers, se especifican sus respectivos diccionarios de datos.

3.7.11.2 Diccionario de datos entidad dirección

El diccionario de datos para la entidad dirección puede apreciarse en la tabla 53.

Tabla 53
Diccionario de datos para la entidad dirección

Nombre del atributo	Tipo del atributo	Restricción	Descripción de uso
Id	int(10) unsigned	PRI	Identificador único por usuario
created_at	timestamp		Fecha y hora de creación del registro
updated_at	timestamp		Fecha y hora de la última modificación del registro

country	varchar(255)		Campo de almacenamiento del país
state_region	varchar(255)		Campo para el almacenamiento del estado o región
city	varchar(255)		Campo para el almacenamiento de la ciudad
avenue	varchar(255)		Campo para el almacenamiento de la avenida
street	varchar(255)		Campo de almacenamiento de la calle
building	int(10) unsigned		Valor que identifica tipos de edificaciones
reference	varchar(255)		Campo que almacena una referencia u otras informaciones adicionales a la dirección de la organización
organization_id	int(10) unsigned	MUL	Llave foránea a la tabla organización

Fuente: Autor.

3.7.11.3 Diccionario de datos entidad código de marcado

El diccionario de datos para la entidad código de marcado puede apreciarse en la tabla 54.

Tabla 54

Diccionario de datos para la entidad Código de marcado

Nombre del atributo	Tipo del atributo	Restricción	Descripción de uso
id	int(10) unsigned	PRI	Identificador único por código de marcado
created_at	timestamp		Fecha y hora de creación del registro
updated_at	timestamp		Fecha y hora de la última modificación del registro
prefix_number	varchar(255)		Campo de almacenamiento del código de marcado asignado a una cuenta SIP

assigned	tinyint(1)		Valor booleano para saber si un código de marcado ya ha sido asignado o está siendo usado por una cuenta SIP
sip_account_id	int(10) unsigned	MUL	Llave foránea a la tabla cuenta SIP

Fuente: Autor.

3.7.11.4 Diccionario de datos entidad formulario

El diccionario de datos para la entidad formulario puede apreciarse en la tabla 55.

Tabla 55

Diccionario de datos para la entidad Formulario

Nombre del atributo	Tipo del atributo	Restricción	Descripción de uso
id	int(10) unsigned	PRI	Identificador único por registro de usuario
created_at	timestamp		Fecha y hora de la última modificación del registro
updated_at	timestamp		Fecha y hora de creación del registro
form_status	int(10) unsigned		Valor que indica cual es el estado del formulario
form_step	int(10) unsigned		Del formulario paginado cual fue el último actualizado
email	varchar(255)	UNI	Valor que identifica unívocamente a cada registro de usuario y cada usuario una vez terminado el proceso de registro. Este valor es comparado con el campo email de la tabla usuarios para saber si un usuario ya ha sido registrado
usr_name	varchar(255)		Campo que almacena el nombre del usuario suscriptor
usr_surname	varchar(255)		Campo que almacena el apellido del usuario suscriptor
usr_code_phone	varchar(4)		Campo que almacena el código de área del teléfono del usuario suscriptor
usr_phone	varchar(8)		Campo que almacena el teléfono del usuario suscriptor
usr_password	varchar(255)		Campo que almacena la contraseña suministrada por el usuario suscriptor

usr_role	int(10) unsigned		Valor que diferencia a tipos de usuario (administrador o usuario normal)
org_name	varchar(255)		Campo que almacena el nombre de la organización
org_rif	varchar(255)		Campo que almacena el RIF de la organización
org_code_phone	varchar(4)		Campo que almacena el código de área del teléfono de la organización
org_phone	varchar(8)		Campo que almacena el teléfono de la organización
org_voip_tech	varchar(255)		Campo que almacena el tipo de tecnología VoIP utilizada por la organización
org_country	varchar(255)		Campo que almacena el país origen de la organización
org_state_region	varchar(255)		Campo que almacena el estado o la región origen de la organización
org_city	varchar(255)		Campo que almacena la ciudad origen de la organización
org_avenue	varchar(255)		Campo que almacena el nombre de la avenida en la que se encuentra la organización
org_street	varchar(255)		Campo que almacena el nombre de la calle donde se encuentra ubicada la organización
org_building	int(10) unsigned		Valor que representa el tipo de edificación donde se encuentra ubicada la organización
org_reference	varchar(255)		Campo que almacena otras referencias para ubicar la organización
ip_address	varchar(255)		Campo que almacena la dirección IP publica que debe poseer la organización a suscribir a fin de generar los datos necesarios para el establecimiento de un enlace troncal con la plataforma
usr_did	varchar(255)		Campo que almacena el código de marcado que desea ser asignado por parte del usuario

Fuente: Autor.

3.7.11.5 Diccionario de datos entidad organización

El diccionario de datos para la entidad organización puede apreciarse en la tabla 56.

Tabla 56

Diccionario de datos para la entidad Organización

Nombre del atributo	Tipo del atributo	Restricción	Descripción de uso
id	int(10) unsigned	PRI	Identificador único por organización
created_at	timestamp		Fecha y hora de creación del registro
updated_at	timestamp		Fecha y hora de la última modificación del registro
name	varchar(255)		Campo que almacena el nombre de la organización
rif	varchar(255)		Campo que almacena el RIF de la organización
code_phone	varchar(4)		Campo que almacena el código de área del teléfono de la organización
phone	varchar(8)		Campo que almacena el teléfono de la organización
voip_tech	varchar(255)		Campo que almacena la tecnología VoIP utilizada por la organización

Fuente: Autor.

3.7.11.6 Diccionario de datos entidad solicitud

El diccionario de datos para la entidad solicitud puede apreciarse en la tabla 57.

Tabla 57

Diccionario de datos para la entidad Solicitud

Nombre del atributo	Tipo del atributo	Restricción	Descripción de uso
id	int(10) unsigned	PRI	Identificador único por solicitud
created_at	timestamp		Fecha y hora de creación del registro
updated_at	timestamp		Fecha y hora de la última modificación del registro
type	int(10) unsigned		Valor entero que permite clasificar e identificar tipos de solicitudes
status	enum('Aprobada','Rechazada','En progreso')		Campo que almacena el estado en que se encuentra actualmente una solicitud, este puede ser en proceso, aceptada o rechazada
description	text		Campo que almacena una descripción acerca de la solicitud generada permitiendo ver detalles sobre la misma
user_id	int(10) unsigned	MUL	Clave foránea a la tabla Usuario

Fuente: Autor.

3.7.11.7 Diccionario de datos entidad cuenta SIP

El diccionario de datos para la entidad cuenta SIP puede apreciarse en la tabla 58.

Tabla 58

Diccionario de datos para la entidad Cuenta SIP

Nombre del atributo	Tipo del atributo	Restricción	Descripción de uso
id	int(10) unsigned	PRI	Identificador único por cuenta SIP
created_at	timestamp		Fecha y hora de creación del registro
updated_at	timestamp		Fecha y hora de la última modificación del registro
name	varchar(255)		Campo que almacena el nombre de la troncal y usuario SIP
default_ username	varchar(255)		Campo que almacena el usuario SIP por defecto
from_domain	varchar(255)		Campo que almacena el dominio del servidor VoIP de la organización suscriptora, este campo es la dirección IP publica suministrada por la misma
from_user	varchar(255)		Campo que almacena el usuario SIP origen
host	varchar(255)		Campo que almacena la dirección IP publica suministrada por el usuario suscriptor al momento del registro
host_type	enum('friend', 'peer', 'user')		Campo que almacena el tipo de cuenta SIP. El valor por defecto es friend
password	varchar(255)		Campo que almacena la contraseña de la cuenta SIP troncal del usuario suscriptor
context	varchar(255)		Campo que almacena el contexto donde será ubicada la cuenta SIP troncal del usuario suscriptor
qualify	varchar(40)		Campo que almacena el valor de yes o no para monitorear desde la plataforma integrada si la cuenta SIP troncal del suscriptor se encuentra activa.
disallow_codec	varchar(255)		Campo que almacena el valor all. Esto deshabilita el uso de todos los codecs para una cuenta SIP troncal del usuario suscriptor para luego permitir solamente el codec que utilizara esta troncal.
allow_codec	varchar(255)		Campo que almacena el valor "gsm" que es el codec a utilizar por la cuenta SIP troncal del usuario suscriptor para comunicarse con la plataforma integrada
ip_address	varchar(255)		Campo que almacena la dirección IP publica suministrada por el usuario suscriptor al momento del registro
default_ip	varchar(255)		Campo que almacena la dirección IP publica suministrada por el usuario suscriptor al momento del registro, este campo será utilizado por Asterisk en la plataforma integrada como dirección IP por defecto para enviar mensajes SIP o llamadas.

trunk_name	varchar(255)		Campo que almacena el nombre de la cuenta SIP troncal del usuario suscriptor
denny_ip	varchar(255)		Campo que almacena la o las direcciones IP que serán restringidas por la plataforma integrada
permit_ip	varchar(255)		Campo que almacena la o las direcciones IP que serán permitidas por la plataforma integrada
user_id	int(10) unsigned	MUL	Clave foránea a la tabla usuario
request_id	int(10) unsigned	MUL	Clave foránea a la tabla solicitud
accStatus	int(10) unsigned		Valor que representa el estado actual de una cuenta SIP. Para efectos administrativos
nes	int(10) unsigned		Valor que indica si la cuenta ha sido editada, es nueva o ya se encuentra almacenada en la plataforma remota
description	varchar(40)		Campo que almacena una breve descripción sobre la cuenta SIP troncal del usuario suscriptor
nat	varchar(255)		Campo que almacena el valor de yes. Utilizado por la plataforma integrada para determinar si un usuario se encuentra detrás de NAT.

Fuente: Autor.

3.7.11.8 Diccionario de datos entidad usuario

El diccionario de datos para la entidad usuario puede apreciarse en la tabla 59.

Tabla 59

Diccionario de datos para la entidad Usuario

Nombre del atributo	Tipo del atributo	Restricción	Descripción de uso
id	int(10) unsigned	PRI	Identificador único por usuario
name	varchar(255)		Fecha y hora de creación del registro
surname	varchar(255)		Fecha y hora de la última modificación del registro
email	varchar(255)	UNI	Campo que almacena el correo electrónico del usuario, este campo debe ser único entre todos los usuarios ya que es usado para fines de autenticación y registro
code_phone	varchar(4)		Campo que almacena el código de área del teléfono del usuario suscriptor
phone	varchar(8)		Campo que almacena el teléfono del usuario suscriptor
password	varchar(255)		Campo que almacena la contraseña de acceso a la aplicación web del usuario suscriptor
role	int(10) unsigned		Valor que diferencia a tipos de usuario (administrador o usuario normal)

remember_token	varchar(100)		Valor usado por Laravel para saber cuándo un usuario ha sido autenticado dentro de nuestra aplicación web
created_at	timestamp		Fecha y hora de creación del registro
updated_at	timestamp		Fecha y hora de la última modificación del registro
organization_id	int(10) unsigned	MUL	Clave foránea a la tabla organización
code	int(11)		Código único generado por la aplicación para identificar unívocamente al usuario.

Fuente: Autor.

Finalmente, para el primer Sprint se especificaron los distintos estados que pueden tener asociado un usuario (tabla 60), una cuenta SIP (tabla 61) y una solicitud (tabla 62).

Tabla 60

Estados asociados al atributo role

Entidad:	**Usuario**	
	Campo	**Estado**
	role	Administrador
	role	Usuario pendiente por aprobar
	role	Usuario aprobado
	role	Usuario rechazado
	role	Usuario bloqueado

Fuente: Autor.

Tabla 61

Estados asociados al atributo NES y AccStatus

Entidad:	**Cuenta SIP**	
	Campo	**Estado**
	AccStatus	Cuenta SIP pendiente por aprobar
	AccStatus	Cuenta SIP aprobada
	AccStatus	Cuenta SIP rechazada
	AccStatus	Cuenta SIP con código de marcado asociado
	AccStatus	Cuenta SIP bloqueada
	NES	Cuenta SIP nueva
	NES	Cuenta SIP editada
	NES	Cuenta SIP guardada

Fuente: Autor.

Tabla 62

Estados asociados al atributo Type

Entidad:	Solicitud	
	Type	Solicitud de nuevo registro

Fuente: Autor.

Al finalizar el Sprint se realizaron las pruebas respectivas para verificar el acceso a la base de datos desde consola MySQL y se discutió en el sprint review los diagramas generados siendo estos los definitivos para el desarrollo de la aplicación.

3.7.12 Sprint 2

Para el desarrollo de nuestro sprint numero dos se crearon los archivos necesarios para la implementación del diagrama entidad relación. Para ello se utilizaron las migraciones, una de las características que trae consigo Laravel. Esto nos permite definir dentro de Laravel las tablas y sus atributos. Adicionalmente se crearon los modelos que están asociados a las tablas de la base de datos para el acceso a los datos. Por último, se definen los controladores en los que serán los encargados de realizar la comunicación con los modelos e interactuar con las vistas de nuestra aplicación y contener la lógica necesaria para cumplir con las funcionalidades relacionadas con ciertos componentes de la aplicación. Un listado de las tareas realizadas durante el desarrollo de este Sprint se puede apreciar en la tabla 63.

Tabla 63

Listado de actividades realizadas durante el desarrollo del Sprint 2

Sprint	Identificador	Nombre
2	7	Implementación del diagrama Entidad-Relación
	8	Generación e implantación de archivos migration en Laravel para implementar el diagrama entidad-relación
	9	Generación de archivos model en Laravel para el acceso a los datos de las entidades y especificar sus relaciones
	10	Verificación de acceso a los datos y obtención de los mismos
	11	Crear los controladores necesarios para las distintas funcionalidades o entidades
	35	Generar vista de solicitud de datos de autenticación
	36	Aplicación del middleware AUTH para los componentes que requieran autenticación

Fuente: Autor.

Se utilizaron las configuraciones realizadas en el Sprint anterior para realizar la implementación de las migraciones en Laravel. Estas usan las credenciales definidas en el archivo ".env". Las migraciones contienen código siguiendo la sintaxis definida para el módulo "Schema" de Laravel para crear y manipular tablas en la base de datos. Se puede apreciar un fragmento de código de una de las migraciones en la figura 89.

```php
class CreateCodeNumbersTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('code_numbers', function (Blueprint $table) {
            $table->increments('id');
            $table->timestamps();
            $table->string('prefix_number');
            $table->boolean('assigned');
            //relaciones con las tablas user y codigo de marcado
            $table->integer('sip_account_id')->unsigned()->index();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::drop('code_numbers');
    }
}
```

Figura 83***:*** Fragmento de código de la migración para la tabla código de marcado.
Fuente: Autor.

Por cada entidad se creó una migración a fin de poder implementar el diagrama entidad relación.

Laravel además de las migraciones, también define los modelos. Estos representan una interfaz para interactuar con las tablas de una base de datos y se definen las relaciones entre entidades. Fueron creados por cada entidad un modelo. Un fragmento de código de un modelo puede apreciarse en la figura 90.

```php
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class code_Number extends Model
{
    //
    protected $table = 'code_numbers';

    public function sip_account(){

        return $this->BelongsTo('App\sip_Account');
    }


}
```

Figura 84***:*** Fragmento de código del modelo para la entidad código de marcado.
Fuente: Autor.

En la figura 90 se puede apreciar cómo se invoca al método BelongsTo() indicando que esta tabla pertenece al modelo (entidad) cuenta SIP (relación 1 a 1 de acuerdo al diagrama entidad relación). Estas relaciones son definidas dentro de funciones que son llamas posteriormente para consultar y extraer datos de una entidad particular y sus relaciones con otras entidades.

Para la finalización del segundo Sprint se generaron los controladores necesarios para darle soporte a las funcionalidades que fueron necesarias implementar. Cada funcionalidad tiene relación con algún modulo o proceso, los cuales están representados por los controladores.

Los controladores alojan toda la lógica de las funcionalidades y nos ayudan en el procesamiento de solicitudes provenientes desde las vistas. Se puede observar un fragmento de código (figura 91) de como luce un controlador en Laravel.

```
class serverController extends Controller
{
    /*
        Route::post('/admin/servidor/cargar_usuarios', 'serverController@loadUsers');
        Route::post('/admin/servidor/cargar_dialplan', 'serverController@loadDialplan');
        Route::post('/admin/servidor/cargar_servicio', 'serverController@reloadServer');
    */
        public static function loadUsers(){

            //contadores
            $news=0;
            $edited=0;
            $bloqued=0;
            $deleted=0;

            $accounts = sip_Account::all();

            foreach ( $accounts as $account ){
                //Aprobada?
```

Figura 85***:*** Fragmento de código de controlador Servidor.
Fuente: Autor.

3.7.13 Sprint 3

Para nuestro tercer sprint, se definieron aspectos básicos de diseño como la paleta de colores que posee la nuestra aplicación web. Además, se diseñó la plantilla general de las vistas de la aplicación con el motor de plantillas Blade. Por último, para poder acceder a los distintos módulos o vistas fue necesario la especificación de las rutas de acceso a la aplicación. El resultado de este Sprint fue el despliegue de la página de inicio de la aplicación web administrativa. El listado de tareas asignado para este sprint se puede apreciar en la tabla 64.

Tabla 64
Listado de tareas realizadas durante el desarrollo del Sprint 3

Sprint	Identificador	Nombre
3	12	Definición de la paleta de colores a utilizar en el diseño de las vistas
	13	Generación del layout para las vistas con Blade
	14	Definir rutas de acceso para la página de inicio y para el módulo de registro
	15	Creación de página de inicio

Fuente: Autor.

El Sprint numero 3 estuvo enfocado en su gran parte para el diseño para las vistas que se implementaron en la aplicación. Para ello fue definida la siguiente paleta de colores que se puede apreciar en la figura 92.

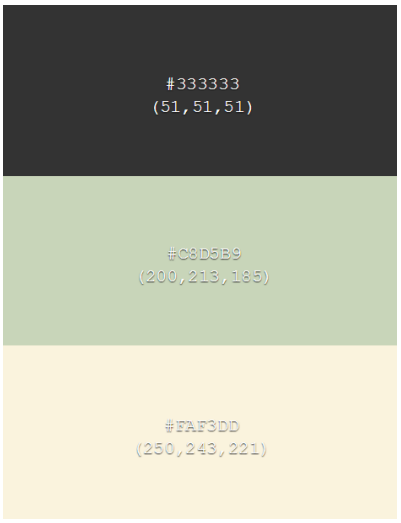

Figura 86***:*** Paleta de colores para las vistas.
Fuente: Autor

De los cuales los colores oscuros fueron utilizados para implementar la barra de navegación, el color verde fue utilizado como fondo y el color crema claro para resaltar el contenido de las vistas.

Al generar la plantilla de las vistas en conjunto con la paleta de colores, las vistas quedaron de la forma como se aprecia en la figura 93 (plantilla utilizada para todas las vistas de la aplicación). Se cuentan con cuatro secciones: Banner con la imagen del muro de nuestra Facultad de Ciencias de la Universidad Central de Venezuela, una barra de navegación, un menú lateral y el área para el contenido de la página.

Figura 87***:*** Layout para la aplicación web.
Fuente: Autor

Parte del desarrollo de las vistas es realizado utilizando el motor de plantillas Blade. Un fragmento de código de la página que sirve como plantilla puede apreciarse en la figura 94.

```
<div class="col-md-8 container blu rounded-body-bottom" id="contenedor">

    @yield('header')

    @yield('content')

    @yield('footer')

</div>
```

***Figura** 88**:** Fragmento de código de la plantilla para las vistas de la aplicación.*
Fuente: Autor.

Para garantizar el acceso al módulo de registro, fueron definidas las rutas de acceso al mismo en el archivo de rutas que trae consigo Laravel. Las rutas quedaron de la siguiente manera:

- http://localhost/registro
- http://localhost/step/{numero}

En donde el primer URL nos redirige al segmento inicial del registro y el segundo URL nos redirige a los pasos a completar del registro.

Por último, se desarrolló la página de inicio para la aplicación web, obteniendo como resultado la página que se puede apreciar en la figura 95. La misma se caracteriza por tener un diseño muy sencillo en donde se muestra un poco de información relacionada con el presente trabajo especial de grado.

***Figura** 89**:** Página de inicio de la aplicación.*
Fuente: Autor.

Al finalizar el sprint se especificaron las demás rutas relacionadas con el módulo de registro y se comprobó el funcionamiento del despleguéde información a través del llamado a las vistas por medio de controladores, así como la validación en la base de datos luego de ejecutadas las migraciones e implementados los modelos.

3.7.14 Sprint 4

Para nuestro cuarto Sprint, se generaron las vistas necesarias para el registro paginado de nuestra aplicación web. La lista de tareas realizadas durante el desarrollo del Sprint número cuatro, se pueden apreciar en la tabla 65.

Tabla 65

Lista de tareas realizadas en el Sprint 4

Sprint	Identificador	Nombre
4	16	Desarrollo de vista para manejar los registros de suscriptores
	17	Generar vista de solicitud de correo electrónico para iniciar proceso de registro
	18	Generar vista para el registro de datos de usuario
	19	Generar vista para el registro de datos organizacionales
	20	Generar vista para el registro de la dirección de la organización
	21	Generar vista para la especificación del código de macado
	22	Generar vista de proceso de registro culminado con éxito

Fuente: Autor.

Este Sprint fue desarrollado con el objetivo de generar las vistas del módulo de registro de usuario. Para el manejo de registros de usuarios, se optó por utilizar un formulario paginado el cual está conformado por 5 vistas. De las cuales la primera es la solicitud de correo electrónico (figura 96). El atributo correo electrónico es utilizado como identificador único para el registro.

Figura 90***:*** Vista de solicitud de dirección de correo electrónico.
Fuente: Autor.

El formulario de solicitud de datos personales de usuario, fue la segunda vista generada en este Sprint. En ella se solicitan los siguientes datos: Nombre, apellido, código de área del teléfono del usuario seguido de su número telefónico, la dirección de correo electrónico se mantiene inmodificable y es obtenida de la vista anterior y, por último, la contraseña para autenticarse en nuestra plataforma. La vista generada puede apreciarse en la figura 97.

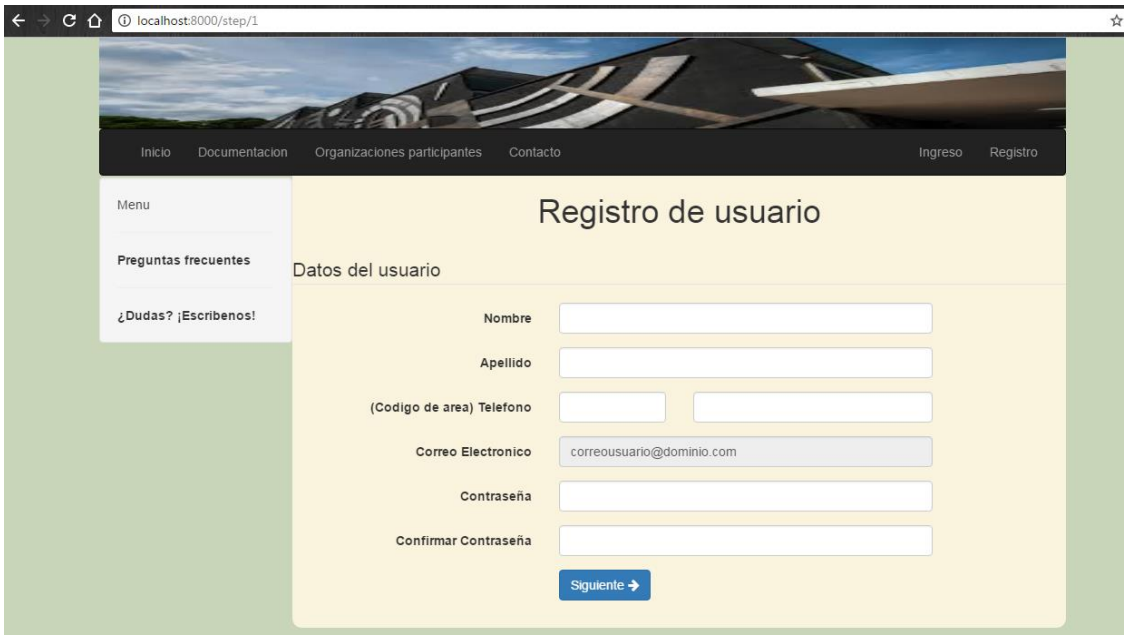

Figura 91***:*** Vista de solicitud de los datos de usuario para su registro.
Fuente: Autor.

El formulario para la solicitud de los datos relacionados con la organización fue la tercera vista generada en este Sprint. En esta, se solicitan los siguientes datos: Nombre de la organización, RIF de la organización, código de área del teléfono de la organización, número de teléfono de la organización, tecnología VoIP utilizada por la organización suscriptora y dirección IP. La vista generada puede apreciarse en la figura 98.

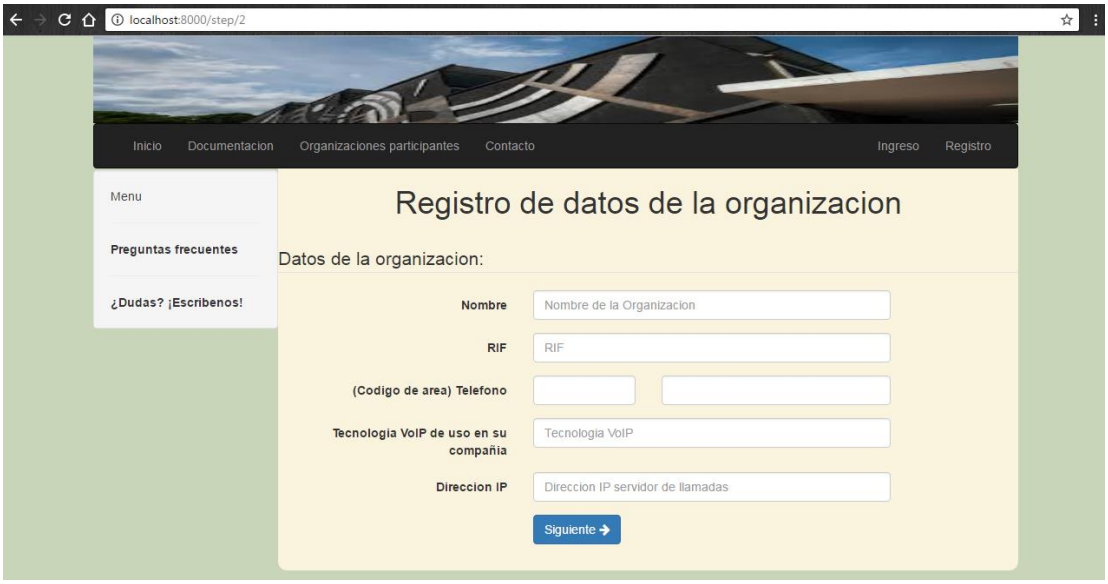

Figura 92***:*** Vista de solicitud de datos de la organización.
Fuente: Autor.

El formulario de solicitud de datos relacionados a la ubicación de la organización fue la tercera vista generada en este Sprint. En esta, se solicitan los siguientes datos: País, estado, ciudad, avenida, calle, tipo de inmueble y otras referencias. La vista generada puede apreciarse en la figura 99.

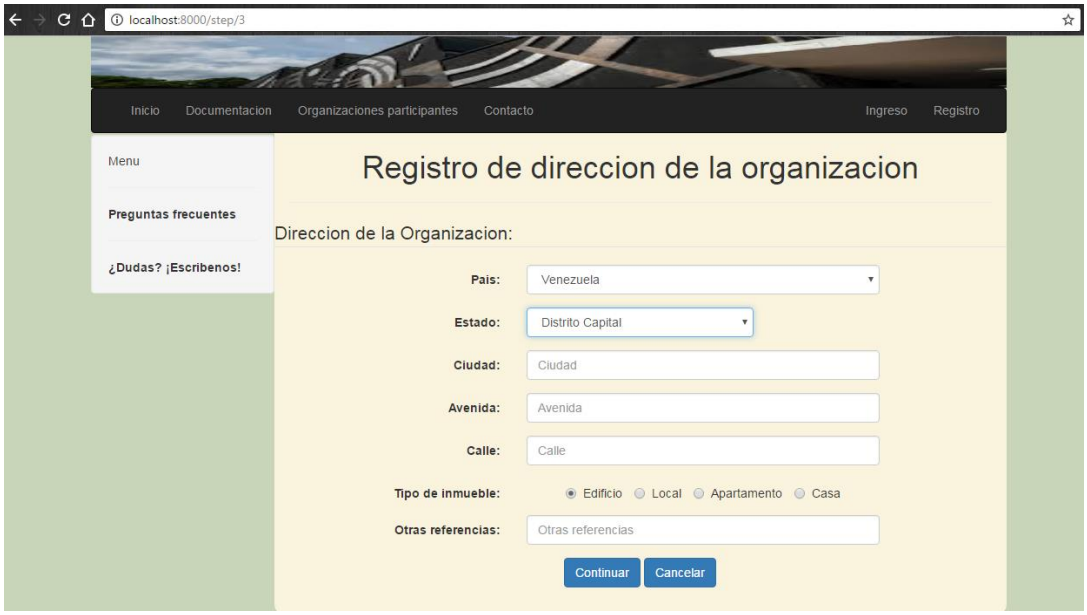

Figura 93***:*** Vista de solicitud de datos de ubicación de la organización a suscribir.
Fuente: Autor.

Finalmente fueron desarrolladas las vistas para la especificación de código de marcado y la de registro exitoso para el usuario siendo esta última mostrada al finalizar exitosamente el registro (figura 101). En la vista de especificación del código de marcado (Figura 100), se le solicita al usuario que ingrese un código de marcado el cual será utilizado por la plataforma para redirigir llamadas. Se

muestran adicionalmente tres sugerencias en un cuadro informativo para la selección de código de marcado.

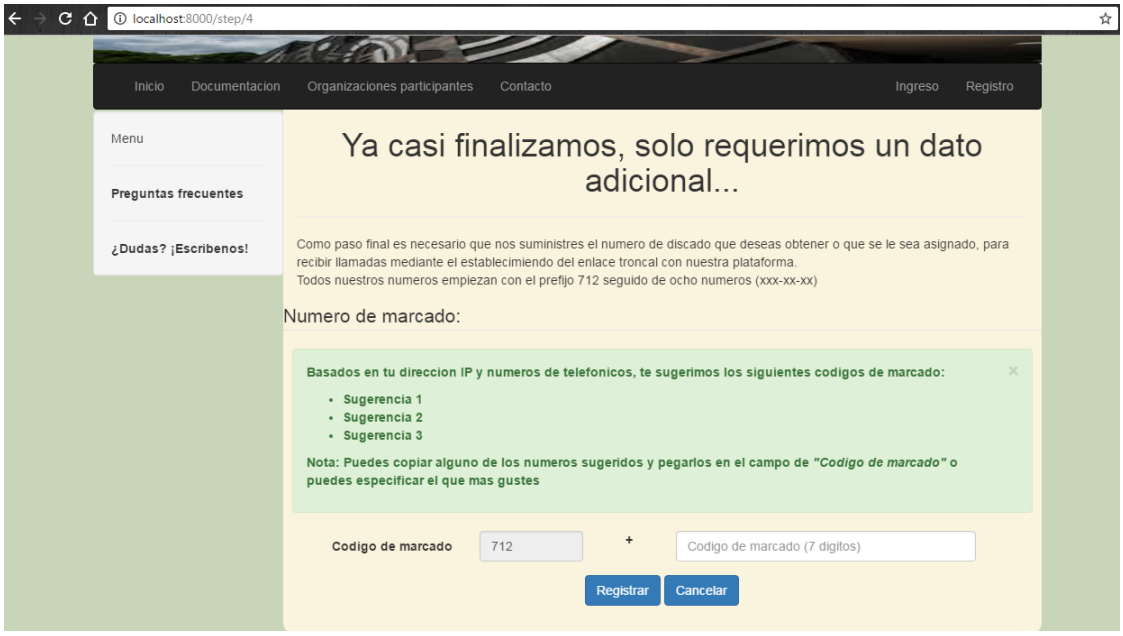

Figura 94***:*** Vista de solicitud de código de marcado.
Fuente: Autor.

Figura 95***:*** Vista de finalización exitosa del registro de usuario.
Fuente: Autor.

Al finalizar este Sprint, fueron realizadas las validaciones de campo desde el HTML y se diseñó la lógica necesaria para manejar el formulario paginado. Se estableció el correo electrónico como identificador único para registros y a cada estado le fue asignado un código en relación a su asociación con el plan de marcado nacional, obteniendo de esta forma información para generar códigos de área.

3.7.15 Sprint 5

En nuestro quinto Sprint, se desarrollaron las funcionalidades asociadas con las vistas y el proceso de registro de un nuevo suscriptor. Se desarrolló la función de generación aleatoria de códigos de marcado para las sugerencias que se le muestran al usuario al momento de llegar a este paso del registro y se establecieron las funciones para crear las instancias resultantes del registro de un usuario. El listado de las tareas realizadas en este Sprint, pueden apreciarse en la tabla 66.

Tabla 66
Listado de tareas realizadas en el Sprint 5

Sprint	Identificador	Nombre
5	23	Desarrollar lógica de generación de sugerencias de código de marcado
	24	Crear función para la gestión del manejo de registro paginado del suscriptor
	25	Realizar las respectivas validaciones para el registro de suscriptores o modificación de información de las diferentes vistas
	26	Crear función de creación de organización
	27	Crear función de creación de dirección
	28	Crear función de creación de cuenta SIP
	29	Crear función de creación de nueva solicitud

Fuente: Autor.

Para generar las sugerencias al usuario al momento de ingresar el código de marcado, se toman como entradas la dirección IP de la organización, el teléfono del usuario y el teléfono de la organización. De la propuesta de trabajo especial de grado se obtiene que se generaran códigos de marcado que inicien con el digito 7.

Luego siguen dos números correspondientes al número de área. Finalmente, el código de marcado es compuesto por 7 dígitos que componen el identificador de usuario quedando de la forma

- 7AA-XXXXXXX

Las sugerencias contienen códigos de marcado únicos.

Una vez finalizada la función de generación de sugerencias, se desarrolló la función para el manejo del formulario paginado. Para el manejo del formulario paginado fue necesario trabajar con la entidad formulario. Esta nos permite almacenar datos a medida que el usuario complete pasos del formulario. Donde luego de finalizar el proceso de registro, serán extraídos los datos para generar entradas en las entidades dirección, organización, usuario y cuenta SIP.

Durante el desarrollo del presente Sprint, fue necesario realizar la implementación de validaciones de las entradas de datos provenientes del registro, esto fue realizado con el módulo de validaciones de Laravel. En la figura 102 podemos un fragmento de código de las validaciones realizadas al registro de información del usuario.

```
$this->validate($request, [

'password' => 'required|min:6|confirmed',
'code_phone' => 'required|digits:4',
'name' => 'required|max:255',
'surname'=>'required|max:255',
'phone' => 'required|digits:7' ]);
```

Figura 96***:*** Validación de los campos del formulario de datos de usuario.
Fuente: Autor.

Este mismo proceso de validación se aplicó para las distintas vistas donde se incluyen modificaciones de datos.

De las validaciones obtenemos como resultado que los datos cumplan con las condiciones establecidas. De esta forma se generaron las funciones para crear las distintas instancias dentro de la base de datos con datos validados. Dentro de los controladores relacionados a las entidades se pueden apreciar las funciones create() donde de crean las instancias para cada entidad. Un fragmento de código de la función créate para la entidad cuenta SIP puede apreciarse en la figura 103.

```
public static function create($user,$code){

  $sip= new sip_Account;
  $string =$code.'trunk';

  $sip->name=$string;
  $sip->default_username=$string;
  $sip->from_domain=$user->ip_address;
  $sip->from_user=$string;
  $sip->host=$user->ip_address;
  $sip->host_type="friend";

  $sip->password="********".$code;

  $sip->context="*********";

  $sip->qualify="yes";

  $sip->nat="yes";
  $sip->ip_address=$user->ip_address;
  $sip->default_ip=$user->ip_address;
  $sip->trunk_name=$string;

  $sip->denny_ip="";

  $sip->permit_ip=$user->ip_address.'/255.255.255.255';
  $sip->accStatus='En proceso';
  $sip->nes='Nuevo';
  $sip->description='troncal de '.str_replace(' ','',$user->org_name);
  $sip->allow_codec='gsm';
  $sip->save();
  return $sip;

 }
```

Figura 97***:*** Función de creación de nueva instancia en la entidad cuenta SIP.
Fuente: Autor.

Al finalizar el presente Sprint se realizaron las pruebas respectivas para verificar que las validaciones fueron correctamente implementadas y que el proceso de registro se ejecutaba correctamente.

3.7.16 Sprint 6

En el Sprint número seis se configuro Laravel para el envío de correo electrónico. Estos nos fueron de utilidad para el envío de notificaciones y poder cumplir con esta funcionalidad de la aplicación. La lista de tareas realizadas durante el desarrollo de este Sprint se puede apreciar en la tabla 67.

Tabla 67
Lista de tareas realizadas para el Sprint 6

Sprint	Identificador	Nombre
6	30	Configurar Laravel para el envío de correos electrónicos
	31	Funcionalidad para enviar correo electrónico y nuevo registro
	32	Generar layout para el envío de correos electrónicos
	33	Crear función de envió de correo electrónico

Fuente: Autor.

Para la implementación de las funciones de envío de correo electrónico utilizando Laravel, se realizaron configuraciones en el archivo "mail.php" del directorio "Config/" de nuestro proyecto. El archivo de configuración de correo electrónico requirió que se establecieran las variables de entorno relacionadas con el correo en el archivo ".env" y que pueden ser apreciadas en la figura 104.

```
MAIL_HOST=smtp.gmail.com
MAIL_PORT=587
MAIL_USERNAME=n******@gmail.com
MAIL_PASSWORD=**************
MAIL_ENCRYPTION=tls
```

Figura *98:* Configuración de variables de entorno para el correo.
Fuente: Autor.

Luego de establecer la conexión para el envío de correo electrónico, se desarrolló la función de envío de notificaciones al usuario suscriptor y al usuario administrador mediante correo electrónico. Las notificaciones son enviadas ante nuevos registros, solicitudes aprobadas o solicitudes rechazadas y notificaciones de confirmación. El extracto de código de la función de envío de correo electrónico puede apreciarse en la figura 105.

```php
public static function send(Form $req, Request $request){

    $dest = $req->email;
    $from= 'n*****@gmail.com';
    $emailTitle='Registro exitoso en nuestra plataforma VoIP!';
    $username = $req->usr_name;

    //primer parametro es la vista que se le pasa como formato y lo que le llega al destinatario
    //segundo parametro es variables y valores pasados a la vista
    //como tercer parametro tenemos el clousure que especifica algunos parametros del correo
    Mail::send('correo_usuario', ['usr'=>$req], function ($message) use ($dest, $from,  $emailTitle)
    {

        $message->from( 'n*****@gmail.com' , 'AdminTEG')->subject($emailTitle);

        $message->to($dest);

        //$message->cc($address);

    });

    return view('registroexitoso')->with('usr',$username);
}
```

Figura 99*:* Función para el envío de correos electrónicos de nuestra aplicación web.
Fuente: Autor.

Al finalizar el Sprint se probaron las funciones de envío de correo para verificar su correcto funcionamiento.

3.7.17 Sprint 7

En nuestro Sprint número siete se implementó una de las funcionalidades más importantes en nuestra aplicación web, el autenticado de usuarios. Para ello se utilizaron clases auxiliares definidas en Laravel para logarlo de la manera más fácil. Se realizaron modificaciones y sobre escritura de métodos predefinidos en Laravel para lograr los objetivos deseados en cuanto a comportamiento de la aplicación. La lista de tareas realizadas durante el desarrollo de este Sprint puede apreciarse en la tabla 68.

Tabla 68
Listado de tareas realizadas para el Sprint 7

Sprint	Identificador	Nombre
7	34	Implementación del módulo de autenticación en Laravel para los usuarios
	35	Generar vista de solicitud de datos de autenticación
	36	Aplicación del middleware AUTH para los componentes que requieran autenticación

Fuente: Autor.

Para la implementación de una autenticación básica de usuario, fue necesario utilizar el módulo de autenticación de usuario AUTH que trae consigo Laravel. Para este módulo se definen un conjunto de funciones en los que también se encuentra la funcionalidad de registro, estos fueros adecuados para abarcar las necesidades del proyecto. El módulo "auth" trabaja directamente con la instancia "Usuario".

Las rutas definidas para este módulo pueden apreciarse en la figura 106. Estas se mantuvieron inmodificables dejando sus valores por defecto.

```
|        | GET|HEAD | datos-troncal |        | Closure                                                 | web          |
|        | GET|HEAD | home          |        | App\Http\Controllers\HomeController@index               | web,auth     |
|        | GET|HEAD | login         |        | App\Http\Controllers\Auth\AuthController@showLoginForm  | web,guest    |
|        | POST     | login         |        | App\Http\Controllers\Auth\AuthController@login          | web,guest    |
|        | GET|HEAD | logout        |        | App\Http\Controllers\Auth\AuthController@logout         | web          |
|        | GET|HEAD | newlog        |        | Closure                                                 | web          |
```

Figura 100***:*** Rutas por defecto para el módulo de autenticación.
Fuente: Autor.

El módulo de autenticación de Laravel, aparte de los controladores y funcionalidades asociados, trae consigo vistas en donde se muestra un pequeño cuadro para autenticar. Este fue modificado para adaptarlo al layout de la aplicación. Esta vista se puede apreciar en la figura 107.

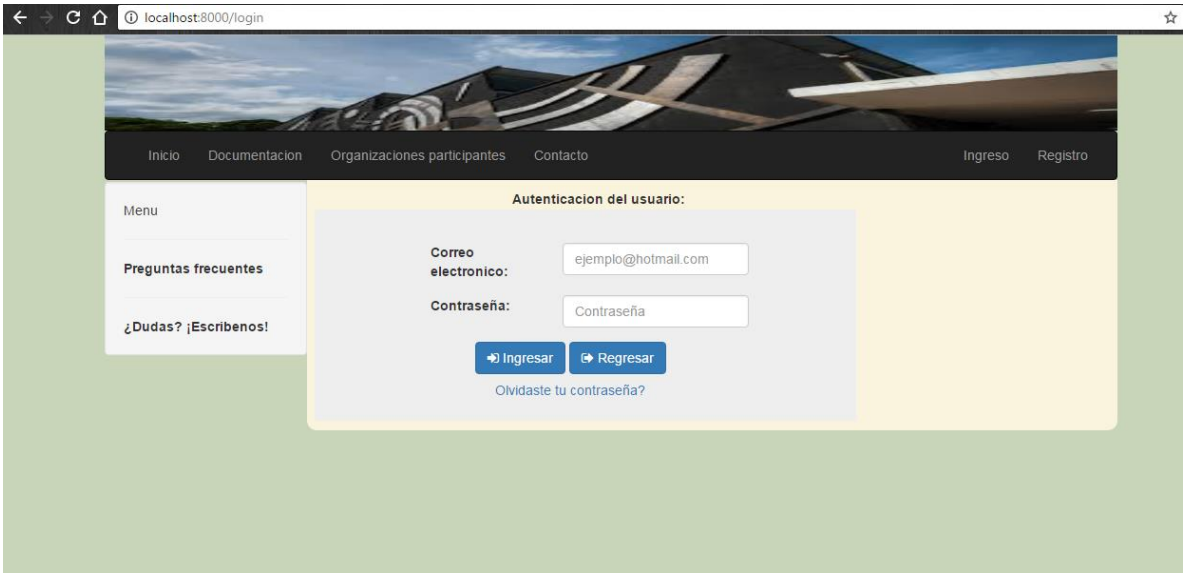

Figura 101***:*** Vista de solicitud de datos para la autenticación.
Fuente: Autor.

Finalmente fue necesario aplicar el middleware de autenticación a los controladores que contuvieran funciones que requirieran de autenticación para proceder con la ejecución de actividades. Un fragmento de código para la aplicación del middleware de autenticación en un controlador puede apreciarse en la figura 108.

```php
public function __construct()
{
    $this->middleware('auth');
}
```

Figura 102***:*** Aplicando el middleware de autenticación a un controlador.
Fuente: Autor.

Se realizaron las pruebas en relación al módulo de autenticación, al final de este Sprint, evaluando y verificando el correcto funcionamiento de este módulo para la aplicación.

3.7.18 Sprint 8

Para nuestro Sprint 8, se desarrollaron las distintas vistas relacionadas al área de trabajo del usuario administrador y para el usuario suscriptor, una de las más importantes en el desarrollo de nuestra aplicación, así como las rutas de acceso a las mismas. El listado de las tareas realizadas durante el desarrollo del Sprint 8 se puede apreciar en la tabla 69.

Tabla 69

Listado de tareas realizadas del Sprint 8

Sprint	Identificador	Nombre
8	37	Generar vista de área de trabajo para usuario
	38	Definir rutas de acceso para las áreas de trabajo y secciones relacionadas
	39	Generar vista de área de trabajo para administrador
	40	Generar vista de administración de solicitudes
	41	Generar vista de administración de troncales
	42	Generar vista de administración de usuarios

Fuente: Autor.

Se definieron en este Sprint las rutas de acceso a las respectivas áreas de trabajo para el usuario administrador y para el usuario suscriptor. Estas rutas son de la forma:

- http://localhost/admin/
- http://localhost/profile/
- http://localhost/home

En donde la primera ruta abarca todas las vistas y funciones del administrador. La segunda ruta define los accesos a usuarios en particular y por último los perfiles de administrador y usuario tienen su área de trabajo bajo la ruta home.

El área de trabajo para el usuario suscriptor puede apreciarse en la figura 109. en donde podemos notar que cuenta con cuatro secciones. La primera contiene material informativo, la segunda sección datos de la organización, la tercera el estado de las solicitudes del usuario y la cuarta se aprecia el estado de la troncal del usuario.

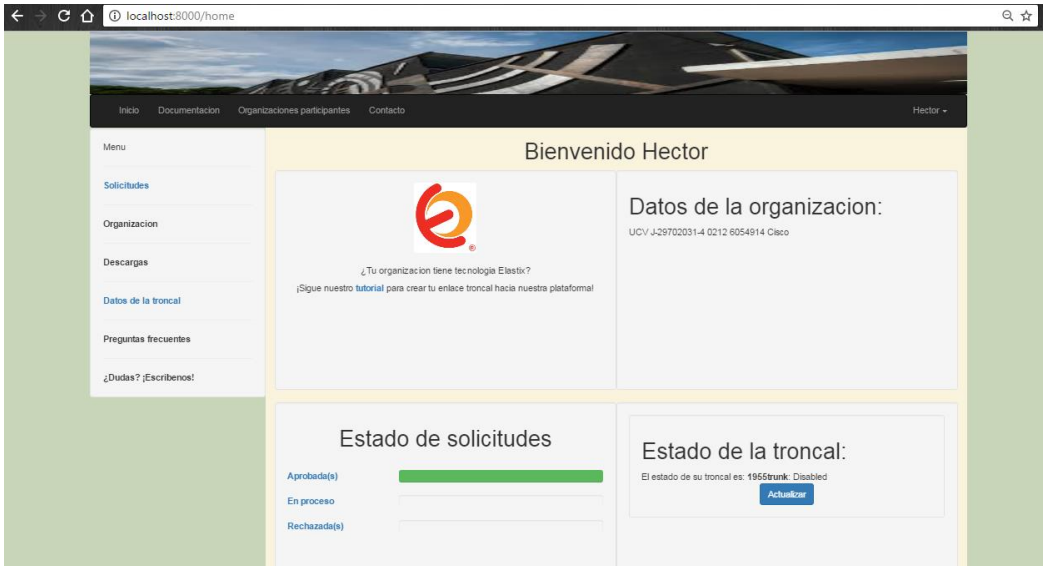

Figura 103***:*** Área de trabajo del usuario suscriptor.
Fuente: Autor.

De manera similar fue desarrollada la vista para el área de trabajo del administrador (ver figura 110). Esta cuenta con tres secciones donde se despliega la siguiente información: Sección de usuarios donde se muestra un cuadro resumen donde se muestran los usuarios totales del sistema. Otra sección de resumen para las solicitudes totales del sistema y una sección para desplegar la información relacionada con el servidor y funciones relacionadas con este (reinicio de servicios, carga de usuarios, etc.).

Figura 104***:*** Ambiente de trabajo para el usuario administrador.
Fuente: Autor.

Para este Sprint también fueron desarrolladas las vistas de administración de solicitudes, usuarios y cuentas SIP.

En la vista para administración de solicitudes, se despliega información relacionada a estas como ID de la solicitud, tipo de solicitud (tipo 1 nuevo registro), usuario que la genera, correo electrónico del usuario que la genera, descripción de la solicitud, cuenta SIP asociada a la solicitud y su estado (ver figura 111). Desde esta vista es posible acceder a las vistas de modificación de una solicitud.

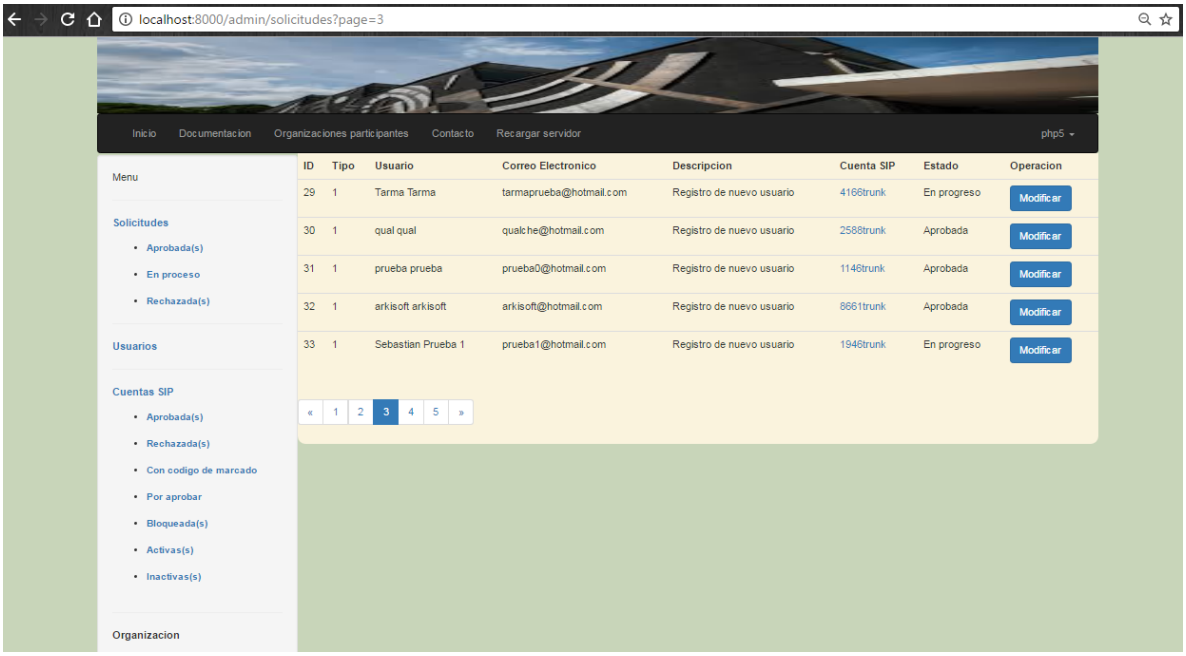

Figura 105***:*** Vista de sección de administración de solicitudes.
Fuente: Autor.

En la vista para la administración de cuentas SIP se despliega la siguiente información: ID de la cuenta SIP, el nombre de la troncal o cuenta SIP, la dirección IP de la cuenta SIP, el usuario al que pertenece la cuenta SIP, si posee un código de marcado asociado y cuál es su estado actual (ver figura 112). desde esta vista se puede acceder a las vistas de modificación de una cuenta SIP en particular. En estas vistas de modificación podemos realizar la asignación de códigos de marcado.

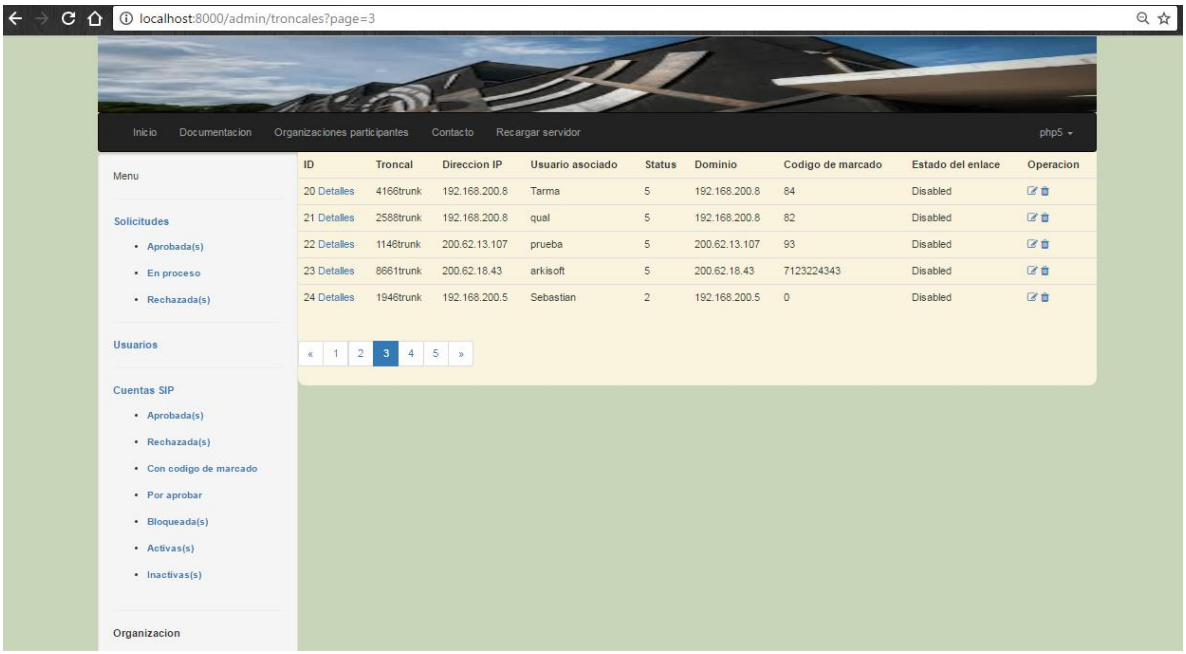

Figura 106***:*** Vista de sección de administración de cuentas SIP.
Fuente: Autor.

Por último, se definió la vista para la administración de usuarios. En esta vista se despliega la siguiente información: Id del usuario, nombre y apellido del usuario, correo electrónico, organización, numero de cuentas SIP que posee y un resumen de las solicitudes del usuario donde A es aprobadas, R es rechazadas y P en proceso (ver figura 113). Desde esta vista es posible acceder a las vistas de modificación de un usuario en particular.

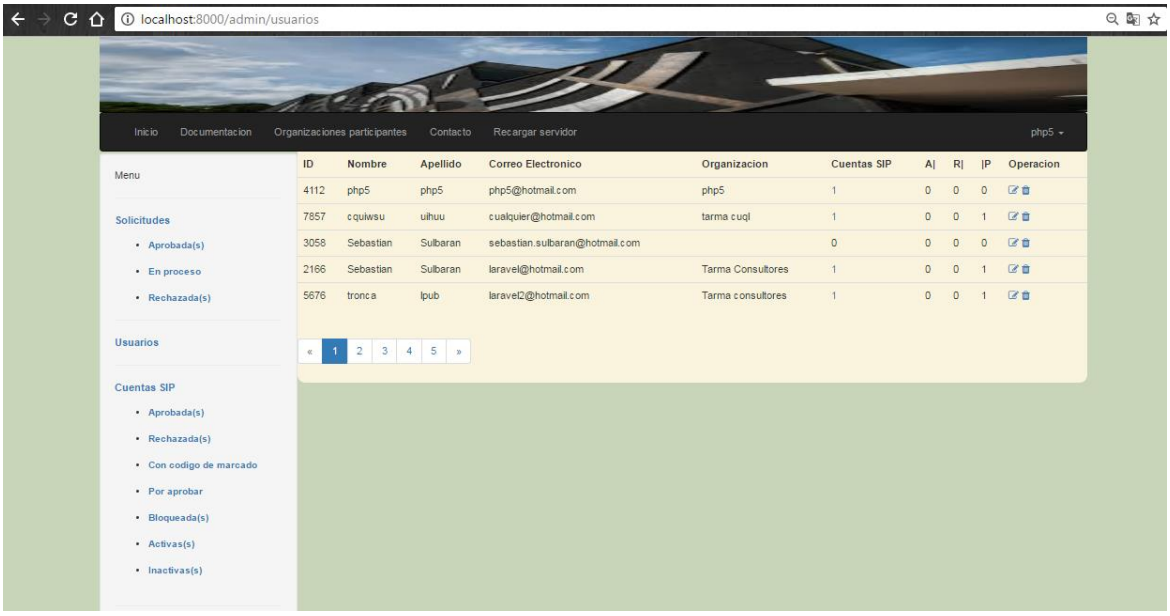

Figura 107***:*** Vista de sección de administración de usuarios.
Fuente: Autor.

3.7.19 Sprint 9

Para nuestro noveno Sprint se implementaron funcionalidades en las páginas de consulta y administración como la paginación, y se realizaron las respectivas funciones de consulta para poblar las vistas con información. Además, se implementaron las funciones para eliminar usuarios, cuentas SIP y solicitudes. La lista completa de las tareas realizadas durante el desarrollo del sprint, pueden apreciarse en la tabla 70.

Tabla 70
Lista de tareas realizadas en el Sprint 9

Sprint	Identificador	Nombre
9	43	Paginar resultados en las vistas de consulta
	44	Función de consulta de usuarios
	45	Función de consulta de cuentas SIP
	46	Función de consulta de solicitudes
	47	Función de eliminar usuarios
	48	Función para eliminar solicitudes
	49	Función para eliminar cuentas SIP

Fuente: Autor.

Las actividades realizadas durante el desarrollo de este Sprint estuvieron enfocadas en poblar las vistas generadas en el Sprint anterior. Por ello se hizo uso de las utilidades ofrecidas por Laravel para el paginado de resultados obtenidos de las consultas realizadas a los modelos. El paginado es una funcionalidad utilizada frecuentemente en página de consultas.

Las consultas fueron realizadas a los modelos empleando las funciones de consulta definidas para los modelos y Eloquent. De esta forma pudimos generar las consultas para obtener los usuarios (figura 114), cuentas SIP (figura 115) y solicitudes (figura 116). Las imágenes referidas contienen fragmentos de código en el que se muestran las consultas realizadas.

```
public static function displayUsers(){
    if(Auth::user()->role==1){
      $users= \App\User::paginate(5);

      return view('teg.adminuserview', compact('users'));
    }
    return view('accessDenied');//fin if
}
```

Figura 108***:*** Función de consulta de usuarios.
Fuente: Autor.

```
public static function displayTrunks(){
        if(Auth::user()->role==1){
          $trunks= \App\User::with('sip_account.code_number')->paginate(5);

          return view('teg.admintrunkview', compact('trunks','response'));
        }
        return view('accessDenied');//fin if

    //dd($response);

    //$trunks= \App\User::with('sip_account.code_number')->get();

}
```

Figura 109***:*** Función de consulta de cuentas SIP
Fuente: Autor.

```
public static function displayRequests(){

        if(Auth::user()->role==1){

         $issues= \App\User::with('requests.sipaccs')->paginate(5);

         return view('teg.adminisuuerequest', compact('issues'));
        }
        return view('accessDenied');//fin if


}
```

Figura 110***:*** Función de consulta de solicitudes.
Fuente: Autor.

Luego para finalizar el Sprint fueron definidas las funciones para la eliminación de usuarios troncales y solicitudes. Estas son similares al fragmento de código mostrado en la figura 117.

```
public static function delete($id){
  $user = User::find($id);

  $user->delete();
}
```

Figura 111***:*** Función para eliminar usuario.
Fuente: Autor.

Al finalizar el Sprint fueron realizadas las pruebas para comprobar el correcto funcionamiento de las características desarrolladas.

3.7.20 Sprint 10

En nuestro decimo Sprint se generaron las vistas de modificación para los usuarios, para las cuentas SIP y para las solicitudes. Estas a su vez por almacenar datos y actualizar información en las entidades y modelos, fue necesario establecer nuevamente las validaciones necesarias utilizando el

componente de Laravel para validaciones Validator. La lista completa de las tareas realizadas durante el desarrollo del décimo Sprint pueden apreciarse en la tabla 71.

Tabla 71
Lista de tareas realizadas en el Sprint 10

Sprint	Identificador	Nombre
10	50	Generar vista de modificación de solicitud
	51	Generar vista de modificación de usuario
	52	Generar vista de modificación de cuenta SIP
	53	Realizar las respectivas validaciones para el registro de suscriptores o modificación de información de las diferentes vistas

Fuente: Autor.

Para este Sprint fueron generadas las vistas de modificación de solicitud, usuario y cuenta SIP.

En la vista de modificación de usuario se muestra información detallada de la cuenta además de los campos que pueden ser modificados (ver figura 118).

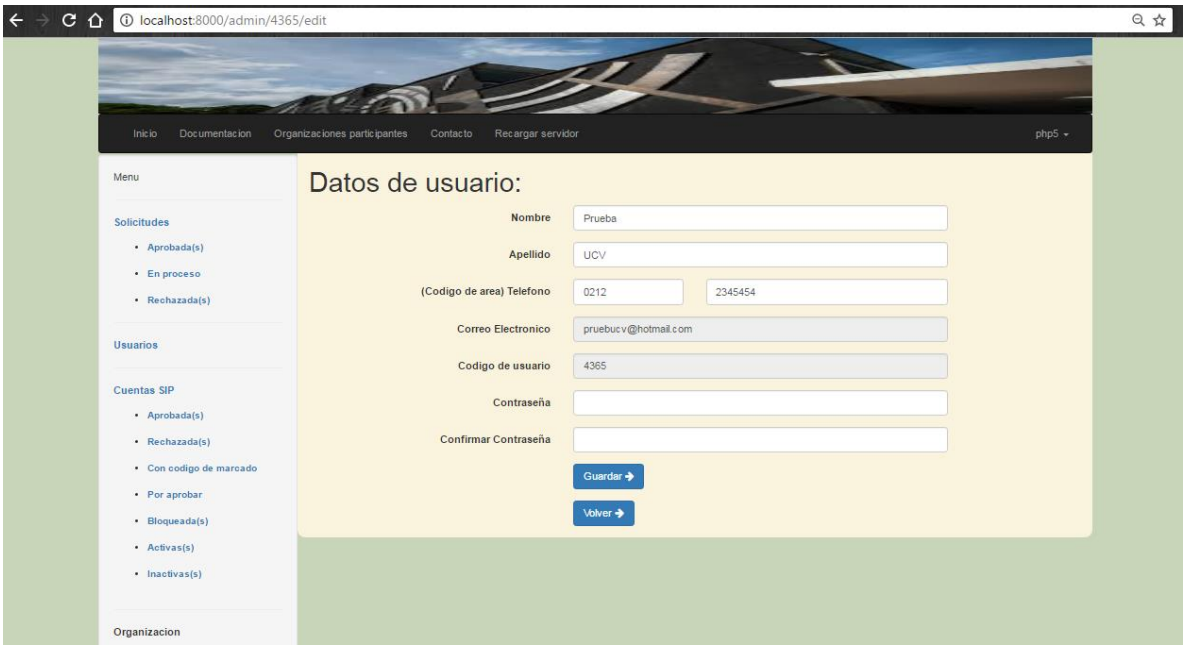

Figura 112**:** Vista de modificación de datos de usuario.
Fuente: Autor.

De la misma forma, la vista de modificación de solicitud muestra en detalle la información contenida en una solicitud, a diferencia que esta solo puede cambiar su estado (ver figura 119).

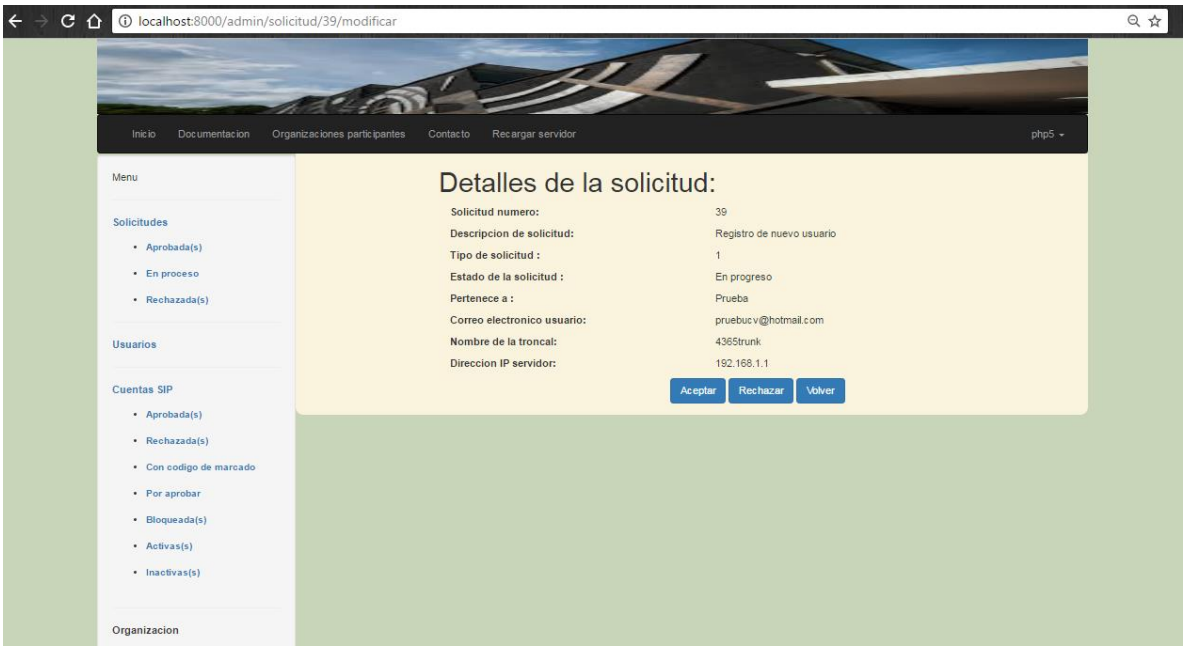

Figura 113***:*** Vista de modificación de una solicitud.
Fuente: Autor.

La ultima vista de modificación desarrollada fue la de las cuentas SIP. En esta se pueden cambiar solo los parámetros habilitados para modificación. Entre ellos, se encuentra el campo de estado que nos permite cambiarle el estado a una cuenta SIP. Si una cuenta SIP se encuentra en estado de aprobado, será desplegada una sección adicional en donde podremos asignarle a una cuenta SIP aprobada un código de marcado (ver figura 120) donde es mostrada la sugerencia proporcionada por el usuario al momento del registro. De resto, la vista de modificación de una cuenta SIP puede verse como en la figura 121.

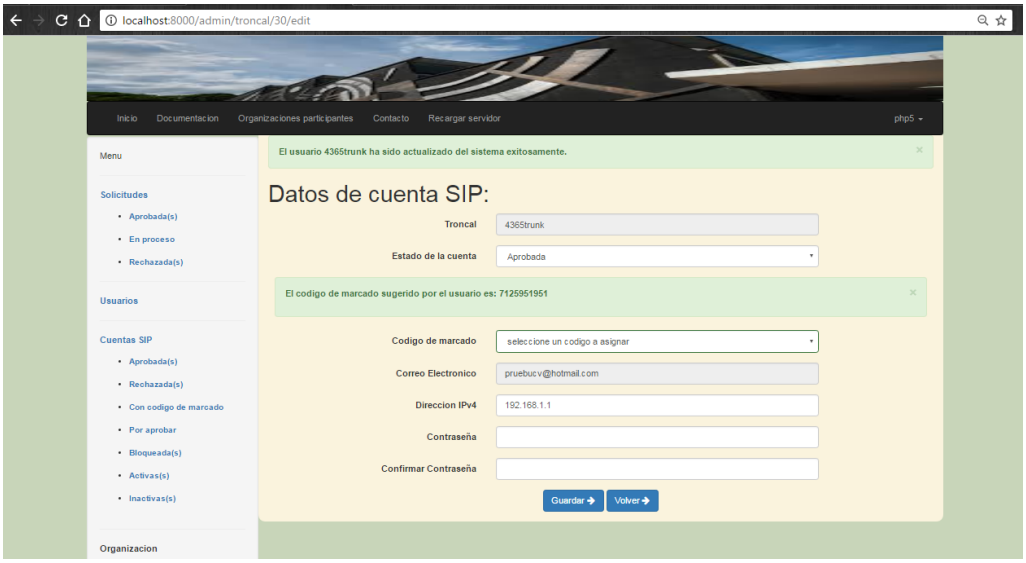

Figura 114***:*** Vista de modificación de cuenta SIP para la asignación del código de marcado.

Fuente: Autor.

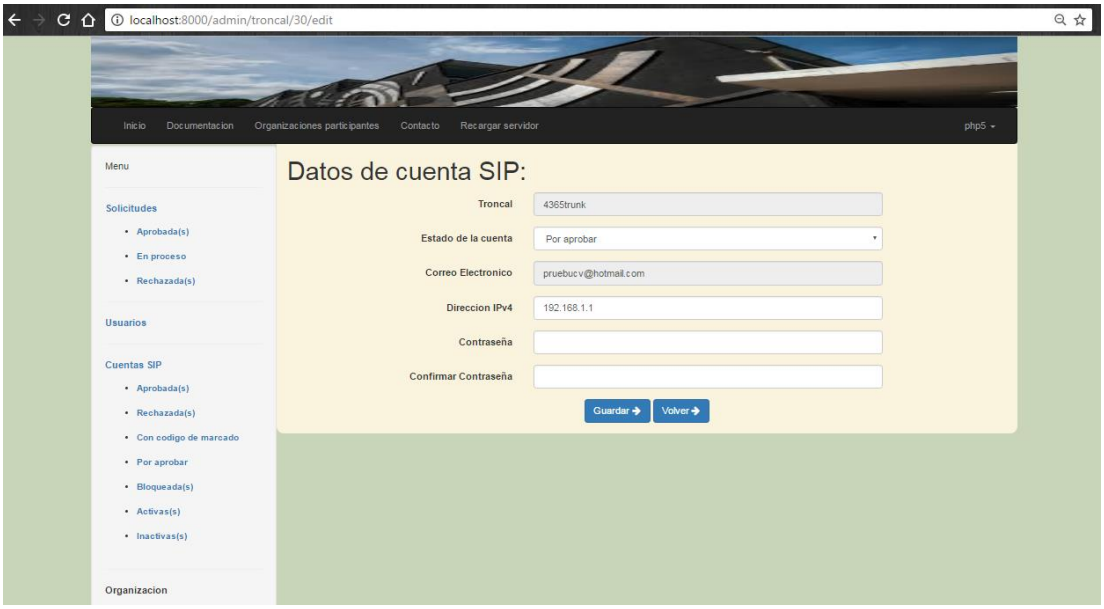

Figura 115***:*** Vista de modificación de cuenta SIP para la asignación del código de marcado.
Fuente: Autor.

Al igual que en Sprints anteriores, se dejó para final del sprint la realización de las validaciones por ingreso de datos del usuario desde los formularios de modificación y se hicieron las respectivas pruebas verificar que los datos ingresados cumplen con las condiciones para su almacenamiento.

3.7.21 Sprint 11

En nuestro Sprint 11 se desarrollaron otro conjunto de funcionalidades de la aplicación web como para las áreas de trabajo del usuario administrador y usuario suscriptor (resumen de solicitudes, usuarios y despliegue de datos de la organización). Se generó también la vista de detalles de la cuenta SIP de un usuario, así como funcionalidades para el manejo y asignación de los códigos de marcado y el cambio de estado de una cuenta SIP. El listado de las tareas realizadas durante el desarrollo del sprint 11 pueden apreciarse en la tabla 72.

Tabla 72
Lista de tareas realizadas para el Sprint 11

Sprint	Identificador	Nombre
11	54	Función resumen de solicitudes
	55	Función resumen de usuarios
	56	Generar vista de detalles de cuenta SIP
	57	Función de asignación de código de marcado
	58	Función de cambio de código de marcado
	59	Función cambio de estado de una cuenta SIP

Fuente: Autor.

La sección de resumen de solicitudes y resumen de usuarios fue construida utilizando Bootstrap para mostrar las barras que representan las proporciones de los usuarios o las solicitudes según su estado. Para generar el volumen de cada barra, se realizaron consultas a los modelos para extraer las cantidades de incidencias de una entidad con determinado estado. Las consultas para determinar las cantidades por solicitud, se aprecia en el extracto de código de la figura 122.

```php
public static function totalRequest(){
    //'Aprobada','Rechazada','En progreso'
    $req = array();
    $count=0;
    $count = \App\Request::where('status','Aprobada')->count();
    array_push($req,$count);
    $count = \App\Request::where('status','En progreso')->count();
    array_push($req,$count);
    $count = \App\Request::where('status','Rechazada')->count();
    array_push($req,$count);
    //se retorna un arreglo donde en su primera posicion se encuentra el total de solicitudes Aprobadas
    //su segunda posicion indica el numero de solicitudes en progreso y su tercera posicion las rechazadas

    return $req;
}
```

Figura 116***:*** Consulta de cantidades.
Fuente: Autor.

De esta forma, el componente del área de trabajo puede llenar las barras de estado con valores puntuales. De la misma forma es realizado el pase de datos para el resumen de usuarios.

Otra de las vistas generadas en este Sprint fue la de detalles de cuenta SIP. En esta vista que puede apreciarse en la figura 123, se muestra información relacionada al contenido de los atributos de una cuenta SIP. De esta vista podemos ir a la vista de modificación de cuenta SIP. Además, se generó la vista para que un usuario pudiese copiar y pegar en sus centrales Elastix/Asterisk/FreePBX la configuración del usuario troncal generado por el sistema (Figura 124). El usuario debe cambiar los valores indicados para su modificación (e.g. contexto).

Figura 117***:*** Detalles de una cuenta SIP.
Fuente: Autor.

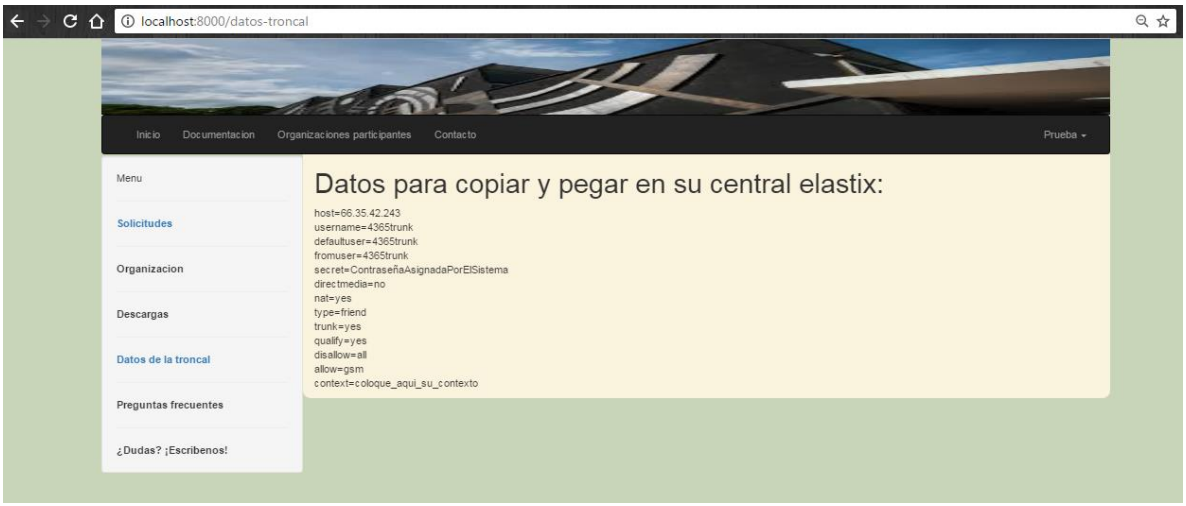

Figura 118***:*** Datos de la troncal para crear usuario troncal del lado del cliente
Fuente: Autor.

Dentro de este Sprint, también fueron desarrolladas las funciones para la asignación de un código de marcado a una cuenta SIP y para el cambio de estado de una cuenta SIP. Ambas se encuentran relacionadas ya que una cuenta SIP necesariamente una cuenta SIP debe estar en estado de aprobada para poder ser elegible a la asignación de un código de marcado. La función de asignación de código de marcado puede apreciarse en la figura 125. En esta podemos ver como se verifica que una cuenta SIP no tenga un código de marcado asociado antes, luego es actualizada la relación entre estas dos entidades y finalmente la cuenta SIP es cambiada de estado (Cuenta SIP con código de marcado).

```
if(isset($request->code_number) && !(isset($request->did)) ){
  if($request->code_number!='-'){

  $code=\App\code_Number::find($request->code_number);
  $code->assigned=1;
  $trunk->accStatus=5;
  $trunk->code_number()->save($code);
  $trunk->update();
  }
```

Figura 119***:*** Fragmento de código, función de asignación de código de marcado.
Fuente: Autor.

Al finalizar este Sprint fueron realizadas las pruebas para verificar la consistencia entre las asignaciones y cambios de estados de las cuentas SIP.

3.7.22 Sprint 12

Para nuestro Sprint número 11 se implementaron las funcionalidades para el acceso remoto desde la aplicación web con la plataforma integrada, pudiendo así de esta forma implementar las funcionalidades relacionadas con la carga de usuarios en la base de datos de Asterisk y el envío del plan de marcado para su adición a Asterisk, así como el reinicio de servicios en Asterisk para que los cambios

puedan ser tomados en cuenta. Un listado de las tareas realizadas durante el desarrollo de este sprint puede apreciarse en la tabla 73.

Tabla 73
Lista de tareas realizadas en el Sprint número 11

Sprint	Identificador	Nombre
12	60	Instalación del componente Laravel collective remote
	61	Configuración del perfil remote para el acceso a la plataforma remota
	62	Configuración de segundo perfil en Laravel para el acceso a la base de datos de la plataforma integrada
	63	Modificar base de datos de la plataforma integrada para el funcionamiento con Eloquent
	64	Modificar el archivo de configuración extensions.conf para agregar de manera dinámica el plan de marcado generado en la aplicación web
	65	Función Información del servidor
	66	Función estado de la troncal del usuario
	67	Crear función para el volcado de datos de usuario en la base de datos de la plataforma remota
	68	Crear función de generación de plan de marcado
	69	Crear función para el envío del archivo que contiene el código de marcado a la plataforma integrada
	70	Crear función para el reinicio de los servicios de la plataforma integrada.

Fuente: Autor.

Para poder implementar las funcionalidades de acceso remoto, fue necesario instalar y configurar el componente Laravel Collective Remote.

La configuración del perfil de acceso remote fue realizada en "/Config/remote.php" y fue necesario agregar los siguientes datos ilustrados en la figura 126.

```
'connections' => [
    'production' => [
        'host'      => 'Direccion IP plataforma',
        'username'  => 'UsuarioRemoto',
        'password'  => 'ContraseñaUsuarioRemoto',
        'key'       => '',
        'keytext'   => '',
        'keyphrase' => '',
        'agent'     => 'xterm',
        'timeout'   => 10,
    ],
```

Figura 120*:* Configuración del perfil remoto.
Fuente: Autor.

Luego de la configuración del perfil remoto fue necesario configurar el segundo perfil de acceso a la base de datos. Esta conexión es realizada hacia la base de datos de la plataforma integrada (figura 127).

```
'asterisk' => [
    'driver' => 'mysql',
    'host' => env('DB_HOST2', 'localhost'),
    'port' => env('DB_PORT2', '3306'),
    'database' => env('DB_DATABASE2', 'forge'),
    'username' => env('DB_USERNAME2', 'forge'),
    'password' => env('DB_PASSWORD2', ''),
    'charset' => 'utf8',
    'collation' => 'utf8_unicode_ci',
    'prefix' => '',
    'strict' => false,
    'engine' => null,
],
```

Figura 121***:*** Configuración de perfil numero dos para acceso a la base de datos en Asterisk.
Fuente: Autor.

Fueron especificados los datos correspondientes a las variables de entorno para la segunda conexión en el archivo ".env".

Finalmente se crearon los modelos para las tablas "sipregs" y "sipusers" de la base de datos remota y se especificaron sus parámetros de conexión a utilizar y tabla asociada. Esto puede apreciarse en el fragmento de código de la figura 128, respectivamente para ambos modelos.

```php
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class sipRegs extends Model
{
    protected $connection='asterisk';
    protected $table='sipregs';
}
```

```php
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class sipUsers extends Model
{
    protected $connection='asterisk';
    protected $table='sipusers';
}
```

Figura 122***:*** Modelos sipregs y sipusers.
Fuente: Autor.

Luego de haber configurado lo necesario para interactuar con las tablas remotas en Laravel, fue necesario realizar una modificación a las tablas "sipregs" y "sipusers" al agregarle los atributos "updated_at" y "created_at" los cuales son utilizados por Eloquent al momento de crear registros o actualizarlos.

Adicionalmente se tuvo que agregar una entrada adicional en el archivo "extensions.conf" de Asterisk para la inclusión del archivo generado automáticamente por la aplicación web y que Asterisk pudiese utilizarlo. Esta modificación puede apreciarse en la figura 129.

```
; You can include other config files, use the #include command
; (without the ';'). Note that this is different from the "include" command
; that includes contexts within other contexts. The #include command works
; in all asterisk configuration files.
;#include "filename.conf"
;#include <filename.conf>
;#include filename.conf
#include "dialplan-custom"
```

Figura 123***:*** Modificación del archivo extensions.conf.
Fuente: Autor.

Luego de haber realizado las últimas modificaciones se procedió a desarrollar las distintas funciones relacionadas con la plataforma remota en la obtención de datos, carga de datos, carga de usuarios, consulta de servicios y reinicio de servicios. A excepción de la carga de usuarios las demás funciones están basadas en SSH (Laravel Collective Remote).

La función de estado del servidor, consiste en evaluar mediante la ejecución per medio del módulo remoto de Laravel un estado del servicio en el servidor donde esta alojada la plataforma integrada (service asterisk status). Por otro lado, la función de estado de troncales se basa en el procesamiento del resultado de aplicar el comando "asterisk -rx 'sip show peers'" usando el módulo de conexión remota de Asterisk. Con el resultado de este último comando se puede realizar una búsqueda por usuario para saber si la troncal asociada a ese usuario se encuentra en un estado activo.

De la misma manera en que son consultados datos y el estado del servicio, es realizado el reinicio de los servicios en la plataforma remota mediante la ejecución del comando 'asterisk -rx "core reload"'

Ya para finalizar el ultimo Sprint se definieron entonces las funciones de carga de usuarios, generar plan de marcado y carga del archivo de plan de marcado.

La función de carga de usuarios consiste en realizar una consulta general de todos los usuarios registrados y que cumplan con la condición de aprobados. Todos estos usuarios serán entonces traspasados desde la base de datos de la aplicación hasta la base de datos de la plataforma integrada. Un fragmento del código de esta función puede apreciarse en la figura 130. A medida que se van consultando usuarios se hacen validaciones contra el campo "AccStatus" y "NES" para verificar que estos ya no se encuentren almacenados en la base de datos remota, estén en estado de aprobados o se encuentren editados. En caso contrario no se realizará la carga de los usuarios.

```php
public static function loadUsers(){

    //contadores
    $news=0;
    $edited=0;
    $bloqued=0;
    $deleted=0;

    $accounts = sip_Account::all();

    foreach ( $accounts as $account ){
        //Aprobada?
        if ($account->accStatus == 3 || $account->accStatus == 5  ){
            //Nueva?
            if($account->nes == 1){
                $news=$news+1;
                $asteriskUser = new sipUsers;
                $asterisReg = new sipRegs;

                $asteriskUser->name=$account->name;
                $asteriskUser->ipaddr=$account->ip_address;
```

Figura 124***:*** Fragmento de código función de carga de usuarios.
Fuente: Autor.

En el proceso de carga de usuarios, son agregados los datos respectivos en las tablas "sipusers" y "sipregs" de Asterisk con los datos almacenados en la aplicación web. Por último, aquellas cuentas en estado de rechazadas son eliminadas de la plataforma remota.

Casi para finalizar podemos decir que la función de generación de plan de marcado, utiliza la siguiente plantilla

- [contexto]
- exten => $codigo_Marcado,1, NoOP("Se envia la llamada por la $nombre_Troncal")
- same => n, NoOp(${EXTEN})
- same => n, Set(extension=${EXTEN})
- same => n,Dial(SIP/$troncal/${extension},30)

De la anterior plantilla podemos decir:

- El contexto solo será añadido una sola vez y tendrá el nombre donde se encontrarán las distintas troncales con códigos de marcado asociado.
- Por cada cuenta SIP en el sistema que tenga asociado el estado de "con código de marcado asociado", se generara una entrada con las instrucciones.
- Se suplantan las variables "$codigo_Marcado", "$nombre_Troncal" y "$troncal", por datos provenientes de la consulta de usuarios que cumplen con la condición de tener código de marcado.
- Las variables "${EXTEN}" y "${extension}" son variables definidas en Asterisk.
- El resto de la estructura nos despliega información como la aplicación "NoOP" y finalmente el Dial() indica por cual troncal y a que extensión llamar dada la extensión de la cuenta SIP.

Finalmente, una vez creado el archivo que contiene todas las extensiones de todas las cuentas SIP, el archivo es enviado empleando una funcionalidad de Laravel remote. Se hace uso de la siguiente función para realizar el envío del archivo:

- SSH::put($localFile, $remotePath);

De esta forma el archivo generado es cargado en la locación del directorio de Asterisk para que este pueda ser cargado desde el plan de marcado de Asterisk.

Luego de generar todas estas funcionalidades, se realizaron pruebas de registros con usuarios reales logrando la verificación de las funcionalidades aquí descritas y logrando establecer la comunicación mediante el registro de un usuario entre un servidor VoIP de una organización suscriptora y la plataforma integrada.

RESULTADOS, CONCLUSIONES Y RECOMENDACIONES

Luego de la culminación de la fase de integración y de la fase de desarrollo, podemos mencionar los siguientes resultados:

- En cada uno de los ambientes fue posible establecer comunicación entre los usuarios de prueba registrados respectivamente por caso de estudio. Esto puede evidenciarse por medio de los diagramas de secuencia de la llamada representados en las figuras 26, 27, 33, 34, 45,50, 53, 57, 68 y 78. Estas pruebas están asociadas a las topologías de red definidas en la sección 3.6.3 de análisis de requerimientos para la fase de integración.

- Se pudo evaluar gradualmente el conjunto de funcionalidades necesarias para realizar la integración entre los componentes que conforman la plataforma telefónica por medio de la implementación de los ambientes de prueba definidos en la sección 3.6.3.

- Cada uno de los ambientes de prueba fue documentado con alto grado de detalle. Esta documentación ayuda a replicar cada ambiente de prueba por separado en caso que se desee realizar un estudio posterior.

- Por cada ambiente de prueba realizado, fueron corregidos detalles en las configuraciones de los servidores que nos permitieron realizar la implementación de la plataforma. La configuración resultante por cada servidor puede apreciarse en el anexo 3 y 4.

- Fueron identificados los requerimientos que deben cumplir las instituciones u organizaciones que quieran formar parte de la plataforma integrada y establecer la correcta comunicación con la misma. Los cuales son mencionados a continuación:

    - Creación y asignación de usuarios SIP con sus credenciales para el establecimiento de los enlaces troncales entre los suscriptores y la plataforma integrada.
    - Definición de los rangos específicos de puertos asociados al establecimiento de una comunicación entre suscriptores y la plataforma.
    - Configuración de servicios adicionales para mitigar posibles vulnerabilidades relativas al protocolo SIP como los servicios Fail2Ban e Iptables.
    - Creación de rutas entrantes salientes para establecer distintos modos de comunicación desde y hacia la plataforma.

- Fue diseñado y establecido un plan de numeración con características similares al Plan Nacional de Numeración de la República Bolivariana de Venezuela a fin de poder asignarles identificadores únicos a las organizaciones que deseen formar parte de la plataforma.

- Fue diseñada, desarrollada e implementada la aplicación web que permite el registro y administración de la información relativa a aquellos usuarios que representan a las organizaciones. De esta forma son generados los archivos de configuración y datos necesarios para el correcto establecimiento de los enlaces troncales hacia y desde la plataforma integrada.

Los resultados descritos, van en concordancia con los objetivos planteados al inicio del desarrollo del presente trabajo. Decimos entonces que estos fueron cumplidos exitosamente.

En los múltiples ambientes de prueba evaluados en los que nuestra plataforma integrada puede funcionar, fueron contempladas comunicaciones entre dos usuarios cualesquiera conectados a nivel local o desde internet, hasta la interconexión de organizaciones de tipo empresarial. Ofreciendo así, una amplia visión del alcance que este trabajo puede tener para la realización de trabajos futuros, además de dar a conocer la importancia de este trabajo como paso inicial para el establecimiento de una red a gran escala de interconexión de organizaciones que utilicen la tecnología VoIP. La adición de servicios de valor agregado como la grabación de llamadas, realizar llamadas internacionales o buzones para la mensajería de voz son algunos de los trabajos futuros a realizar.

El uso del servidor SIP Proxy Kamailio, nos permite poder implementar en trabajos futuros, arquitecturas de alta disponibilidad (escalabilidad), incluyendo la posibilidad de desarrollar e implementar nuevas funcionalidades y servicios que puedan ser requeridos por la plataforma como el manejo e integracion de WebRTC con Kamailio. Por la forma en la que funciona Kamailio, este ofrece bastante libertad para el desarrollo de nuevas funcionalidades. También se puede hacer uso de los múltiples módulos ya desarrollados en Kamailio para la cumplir con necesidades emergentes.

Por último, si bien en este trabajo fue desarrollada una interfaz web administrativa para la plataforma, esta puede mejorarse considerablemente a nivel de front-end así como en la incorporación de nuevas funcionalidades que se acoplen, continúen y expandan los objetivos del proyecto. Algunos de estos nuevos desarrollos pueden incluir, por ejemplo, mejoras al módulo de gestión de solicitudes, o la visualización de estadísticas relacionadas a las troncales de la plataforma desde el panel de administración o el desarrollo de un módulo para el manejo de incidentes por parte de los usuarios.

REFERENCIAS BIBLIOGRAFICAS

- [1] Zaidman, N. (s.f). Telecommunications in Venezuela. Recuperado de http://www.vii.org/papers/vene.htm
- [2] South America Internet and Facebook Users - Population 2016 Statistics. (2016). Recuperado de http://www.internetworldstats.com/south.htm
- [3] What is VOIP. (2016) Recuperado de https://www.voip-info.org/wiki/view/What+is+VOIP
- [4] Russell, B. Leif, M y Van Meggelen, J. (2013) Asterisk- The Definitive Guide. USA, O'Reilly Media, Inc.
- [5] Hartpence, B. (2013). Packet Guide to Voice over IP. USA, O'Reilly Media, Inc.
- [6] Rouse, M. (2016). What is trunk? – Definition. Recuperado de http://searchnetworking.techtarget.com/definition/trunk
- [7] Protocolos de red. (2016) Recuperado de https://es.wikipedia.org/wiki/Anexo:Protocolos_de_red
- [8] J. Rosenberg. (2002). SIP: Session Initiation Protocol. RFC 3261. Recuperado de https://www.ietf.org/rfc/rfc3261.txt
- [9] V. Jacobson. (2006). SDP: Session Description Protocol. RFC 4566. Recuperado de https://tools.ietf.org/html/rfc4566
- [10] H. Schulzrinne. (2003). RTP: A Transport Protocol for Real-Time Applications. RFC 3550. Recuperado de https://www.ietf.org/rfc/rfc3550.txt
- [11] Códecs. (2013). Definición de Códec. Recuperado de http://www.voip-info.org/wiki/view/Codecs
- [12] QoS-Calidad de Servicio para VoIP. (s.f). Recuperado de http://elastixtech.com/qos-calidad-de-servicio-para-voip/
- [13] Quality of Service for Voice over IP. (2001). Recuperado de http://www.cisco.com/c/en/us/td/docs/ios/solutions_docs/qos_solutions/QoSVoIP/QoSVoIP.html
- [14] QoS Quality Of sevice VoIP. (s.f). Eco. Recuperado de http://www.voipforo.com/QoS/QoS_Eco.php
- [15] Recommendation G.729. (2008). Descargado de https://www.itu.int/rec/T-REC-G.729-201206-I/en, ITU-T G.729
- [16] Pesquera, A. (2002). Aprendizaje por Analogía. Recuperado de http://www.sindominio.net/~apm/articulos/ia/x939.html
- [17] PLAN NACIONAL DE NUMERACIÓN PARA TELEFONÍA Y RADIOCOMUNICACIONES MÓVILES TERRESTRES. (18 de agosto de 2016). Gaceta Oficial No. 40.969. Caracas.
- [18] INTERNATIONAL TELECOMMUNICATION UNION. (2005). LIST OF ITU-T RECOMMENDATION E.164 ASSIGNED COUNTRY CODES. Descargado de https://www.itu.int/itudoc/itu-t/ob-lists/icc/e164_763.pdf
-
- [19] Fernandez, G. (2004). La psicología cognitiva. Madrid. Recuperado de http://www.gsi.dit.upm.es/~gfer/ssii/rcsi/rcsisu12.html

- [20] LEY ORGÁNICA DE TELECOMUNICACIONES. (7 de febrero de 2011). Gaceta Oficial Nº 39.610. Caracas
- [21] REGLAMENTO DE INTERCONEXIÓN. (08 de noviembre de 2004). Gaceta Oficial Nº 5735. Caracas.
- [22] VoIP and FCC Regulation. (2013). Recuperado de https://voipweb.wordpress.com/2013/06/11/voip-and-fcc-regulation/
- [23] World Wide Web Consortium. Web Application. (2010). Mobile Web Application Best Practices. Recuperado de https://www.w3.org/TR/mwabp/#webapp-defined
- [24] Web Applications. (s.f). Recuperado de http://www.acunetix.com/websitesecurity/web-applications/
- [25] Wikipedia. (2016). Web application. Recuperado de https://en.wikipedia.org/wiki/Web_application
- [26] Wikipedia. (2016). Hypertext Transfer Protocol Recuperado de https://es.wikipedia.org/wiki/Hypertext_Transfer_Protocol
- [27] Teléfonos IP. (s.f). Recuperado de http://www.3cx.es/voip-sip/telefonos-ip/
- [28] What Exactly is a Softphone. (s.f). Recuperado de https://www.virtualpbx.com/blog/features/what-exactly-is-a-softphone/
- [29] Asterisk Architecture, The Big Picture. (2014). Recuperado de https://wiki.asterisk.org/wiki/display/AST/Asterisk+Architecture%2C+The+Big+Picture
- [30] Mierla, D. Kamailio SIP Server v3.2.0 Development Guide. Recuperado de http://www.asipto.com/pub/kamailio-devel-guide/
- [31] Wikipedia. (2016). Laravel. Recuperado de https://es.wikipedia.org/wiki/Laravel
- [32] Anton, C. (2015). Recuperado de https://platzi.com/blog/laravel-framework-php/
- [33] Hundermark, P. (2009) Un mejor Scrum. Ciudad del Cabo.
- [34] Beck, K. (2011). Principles behind the Agile Manifesto. Recuperado de http://agilemanifesto.org/principles.html
- [35] Scrum Glossary. (s.f). Recuperado de https://www.scrum.org/Resources/Scrum-Glossary
- [36] Asterisk SIP canreinvite. (2015). Recuperado de http://www.voip-info.org/wiki/view/Asterisk+sip+canreinvite
- [37] Modroiu, R. (2010). Recuperado de https://www.kamailio.org/docs/modules/devel/modules/uac.html
- [38] Asterisk SIP NAT. (2015). Recuperado de http://www.voip-info.org/wiki/view/Asterisk+sip+nat
- [39] Kamailio SIP Server v4.0.x (stable): Core Cookbook. (2015). Recuperado de https://www.kamailio.org/wiki/cookbooks/4.0.x/core#advertised_address
- [40] How to Set Up a Basic Iptables Firewall on Centos 6. (2013). Recuperado de https://www.digitalocean.com/community/tutorials/how-to-set-up-a-basic-iptables-firewall-on-centos-6

- [41] Voice Over IP - Per Call Bandwidth Consumption. (2016). Recuperado de http://www.cisco.com/c/en/us/support/docs/voice/voice-quality/7934-bwidth-consume.html

- [42] Psyllos, A. (2012). Bandwidth Utilisation for VoIP Calls. Recuperado de https://www.3cx.com/blog/docs/bandwidth-utilised-for-voip/

- [43] Wikipedia. (2016). Proceso de conversión analógico hacia digital. Recuperado de http://es.wikipedia.org/wiki/Conversi%C3%B3n_anal%C3%B3gica-digital

- [44] Wikipedia. (2016). Teoría de muestreo. Recuperado de http://es.wikipedia.org/wiki/Teorema_de_muestreo_de_Nyquist-Shannon

- [45] Tabla resumen de códec. (s.f). Recuperado de http://www.voipforo.com/codec/codecs.php

- [46] Funcionamiento de un códec - G711. (s.f). Recuperado de http://www.voipforo.com/codec/codec-g711--ley.php#G711

- [47] Márquez. (2004). Implementación de un reconocedor de voz gratuito al sistema de ayuda a invidentes Dos-Vox en español. Recuperado de http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/marquez_a_bm/capitulo5.pdf

- [48] Vegas. (2002). Creación de un Portal Web Docente. Recuperado de http://www.infor.uva.es/~jvegas/cursos/buendia/pordocente/

- [49] Statista (2013). Countries by number of Voice over Internet Protocol (VoIP) subscribers in 1Q 2013. Recuperado de https://www.statista.com/statistics/236824/number-of-voip-subscribers-by-leading-countries/

ANEXO 1

Instalación de MySQL versión Community para sistema operativo CentOS 6

Si bien queremos realizar la instalación con la versión más reciente de MySQL, debemos actualizar nuestros repositorios con los suministrados desde la página web de MySQL. El comando para realizar dicha actualización de repositorio es el siguiente:

- yum install http://dev.mysql.com/downloads/file/?id=465605

Aunque este proceso es opcional ya que con la versión de repositorios con la que viene CentOS 6 por defecto nos sirve para realizar la implementación del ARA (Asterisk Real-timeArchitecture).

Una vez actualizado o no el repositorio mediante YUM, procedemos a instalar MySQL con el siguiente comando:

- yum install mysql-server -y

Se utiliza también el manejador de paquetes YUM para la instalación de este manejador de base de datos. Se realiza la instalación de un servidor mysql y de un cliente mysql.

Una vez instalado debemos iniciar el servicio con el comando:

- service mysqld start

Luego de iniciar el servicio de mysql, utilizamos el siguiente comando para que al iniciar el sistema operativo, mysql inicie como servicio del sistema operativo y inicie automáticamente

- chkconfig mysqld on

Luego de dejar el servicio como persistente, ejecutamos el script de instalación segura de mysql, este script se encuentra en la ubicación:

- /usr/bin/mysql_secure_installation

El script nos hara una serie de preguntas que nos ayudaran a configurar de una manera sencilla nuestro mysql. Las preguntas realizadas por el script son las siguientes:

- Set root password? [Y/n] Y
- Remove anonymous users? [Y/n] Y
- Disallow root login remotely? [Y/n] N
- Remove test database and access to it? [Y/n] Y
- Reload privilege tables now? [Y/n] Y

En donde luego de decir que si al establecer la contraseña del administrador, se solicitara el ingreso de dicha contraseña y luego su confirmación para poder establecer de manera adecuada la contraseña para el administrador. Deberemos seleccionar que se eliminen los usuarios anónimos y eliminar la base de datos de prueba. Para la correcta implementación de otras funcionalidades (de prueba) se establece que se permita el acceso remoto de la cuenta del administrador, por último, reiniciamos los privilegios en la base de datos.

Verificamos que la instalación de nuestro SMBD se ha realizado de manera exitosa utilizando los comandos

- Mysql --version

Y luego verificamos la correcta configuración por medio del script accediendo con el usuario administrador a la base de datos con el comando, con el cual se nos solicitara la contraseña establecida durante el proceso de configuración y luego podremos acceder a la base de datos:

- Mysql -u root -p

ANEXO 2

Scripts MySQL para tabla de registro SIP y tabla para usuario SIP

Los siguientes scripts son para crear las tablas sipregs y sipusers respectivamente. Estas tablas serán usadas por Asterisk en su arquitectura ARA para el registro de datos de ubicación del usuario y datos del registro del usuario SIP.

```
CREATE TABLE `sipregs` (
  `id` INT(11) NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(80) NOT NULL DEFAULT '',
  `fullcontact` VARCHAR(80) NOT NULL DEFAULT '',
  `ipaddr` VARCHAR(45) DEFAULT NULL,
  `port` mediumint(5) UNSIGNED NOT NULL DEFAULT '0',
  `username` VARCHAR(80) NOT NULL DEFAULT '',
  `regserver` VARCHAR(100) DEFAULT NULL,
  `regseconds` INT(11) NOT NULL DEFAULT '0',
  `defaultuser` VARCHAR(80) NOT NULL DEFAULT '',
  `useragent` VARCHAR(20) DEFAULT NULL,
  `lastms` INT(11) DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `name` (`name`)
);
CREATE TABLE `sipusers` (
    `id` INT(11) NOT NULL AUTO_INCREMENT,
     `name` VARCHAR(10) NOT NULL,
     `ipaddr` VARCHAR(15) DEFAULT NULL,
     `port` INT(5) DEFAULT NULL,
     `regseconds` INT(11) DEFAULT NULL,
     `defaultuser` VARCHAR(10) DEFAULT NULL,
     `fullcontact` VARCHAR(35) DEFAULT NULL,
     `regserver` VARCHAR(20) DEFAULT NULL,
     `useragent` VARCHAR(20) DEFAULT NULL,
     `lastms` INT(11) DEFAULT NULL,
     `host` VARCHAR(40) DEFAULT NULL,
     `type` enum('friend','user','peer') DEFAULT NULL,
     `context` VARCHAR(40) DEFAULT NULL,
     `permit` VARCHAR(40) DEFAULT NULL,
     `deny` VARCHAR(40) DEFAULT NULL,
     `secret` VARCHAR(40) DEFAULT NULL,
     `md5secret` VARCHAR(40) DEFAULT NULL,
     `remotesecret` VARCHAR(40) DEFAULT NULL,
     `transport` enum('udp','tcp','udp,tcp','tcp,udp') DEFAULT NULL,
     `dtmfmode` enum('rfc2833','info','shortinfo','inband','auto') DEFAULT NULL,
     `directmedia` enum('yes','no','nonat','update') DEFAULT NULL,
     `nat` enum('yes','no','never','route') DEFAULT NULL,
     `callgroup` VARCHAR(40) DEFAULT NULL,
     `pickupgroup` VARCHAR(40) DEFAULT NULL,
     `language` VARCHAR(40) DEFAULT NULL,
     `disallow` VARCHAR(40) DEFAULT NULL,
     `allow` VARCHAR(40) DEFAULT NULL,
     `insecure` VARCHAR(40) DEFAULT NULL,
```

```
`trustrpid` enum('yes','no') DEFAULT NULL,
`progressinband` enum('yes','no','never') DEFAULT NULL,
`promiscredir` enum('yes','no') DEFAULT NULL,
`useclientcode` enum('yes','no') DEFAULT NULL,
`accountcode` VARCHAR(40) DEFAULT NULL,
`setvar` VARCHAR(40) DEFAULT NULL,
`callerid` VARCHAR(40) DEFAULT NULL,
`amaflags` VARCHAR(40) DEFAULT NULL,
`callcounter` enum('yes','no') DEFAULT NULL,
`busylevel` INT(11) DEFAULT NULL,
`allowoverlap` enum('yes','no') DEFAULT NULL,
`allowsubscribe` enum('yes','no') DEFAULT NULL,
`videosupport` enum('yes','no') DEFAULT NULL,
`maxcallbitrate` INT(11) DEFAULT NULL,
`rfc2833compensate` enum('yes','no') DEFAULT NULL,
`mailbox` VARCHAR(40) DEFAULT NULL,
`session-timers` enum('accept','refuse','originate') DEFAULT NULL,
`session-expires` INT(11) DEFAULT NULL,
`session-minse` INT(11) DEFAULT NULL,
`session-refresher` enum('uac','uas') DEFAULT NULL,
`t38pt_usertpsource` VARCHAR(40) DEFAULT NULL,
`regexten` VARCHAR(40) DEFAULT NULL,
`fromdomain` VARCHAR(40) DEFAULT NULL,
`fromuser` VARCHAR(40) DEFAULT NULL,
`qualify` VARCHAR(40) DEFAULT NULL,
`defaultip` VARCHAR(40) DEFAULT NULL,
`rtptimeout` INT(11) DEFAULT NULL,
`rtpholdtimeout` INT(11) DEFAULT NULL,
`sendrpid` enum('yes','no') DEFAULT NULL,
`outboundproxy` VARCHAR(40) DEFAULT NULL,
`callbackextension` VARCHAR(40) DEFAULT NULL,
`timert1` INT(11) DEFAULT NULL,
`timerb` INT(11) DEFAULT NULL,
`qualifyfreq` INT(11) DEFAULT NULL,
`constantssrc` enum('yes','no') DEFAULT NULL,
`contactpermit` VARCHAR(40) DEFAULT NULL,
`contactdeny` VARCHAR(40) DEFAULT NULL,
`usereqphone` enum('yes','no') DEFAULT NULL,
`textsupport` enum('yes','no') DEFAULT NULL,
`faxdetect` enum('yes','no') DEFAULT NULL,
`buggymwi` enum('yes','no') DEFAULT NULL,
`auth` VARCHAR(40) DEFAULT NULL,
`fullname` VARCHAR(40) DEFAULT NULL,
`trunkname` VARCHAR(40) DEFAULT NULL,
`cid_number` VARCHAR(40) DEFAULT NULL,
`callingpres` enum('allowed_not_screened','allowed_passed_screen','allowed_failed_screen'
              ,'allowed','prohib_not_screened','prohib_passed_screen','prohib_failed_screen'
              ,'prohib') DEFAULT NULL,
`mohinterpret` VARCHAR(40) DEFAULT NULL,
`mohsuggest` VARCHAR(40) DEFAULT NULL,
`parkinglot` VARCHAR(40) DEFAULT NULL,
`hasvoicemail` enum('yes','no') DEFAULT NULL,
`subscribemwi` enum('yes','no') DEFAULT NULL,
`vmexten` VARCHAR(40) DEFAULT NULL,
```

```
    `autoframing` enum('yes','no') DEFAULT NULL,
    `rtpkeepalive` INT(11) DEFAULT NULL,
    `call-limit` INT(11) DEFAULT NULL,
    `g726nonstandard` enum('yes','no') DEFAULT NULL,
    `ignoresdpversion` enum('yes','no') DEFAULT NULL,
    `allowtransfer` enum('yes','no') DEFAULT NULL,
    `dynamic` enum('yes','no') DEFAULT NULL,
    `sippasswd` VARCHAR(80) DEFAULT NULL,
    PRIMARY KEY (`id`),
    UNIQUE KEY `name` (`name`),
    KEY `ipaddr` (`ipaddr`,`port`),
    KEY `host` (`host`,`port`)
) ENGINE=MyISAM;
```

ANEXO 3

Archivo de configuración Kamailio.cfg

```
#!KAMAILIO
#!define WITH_MYSQL
#!define WITH_AUTH
#!define WITH_USRLOCDB
#!define WITH_ASTERISK

#!ifdef ACCDB_COMMENT
  ALTER TABLE acc ADD COLUMN src_user VARCHAR(64) NOT NULL DEFAULT '';
  ALTER TABLE acc ADD COLUMN src_domain VARCHAR(128) NOT NULL DEFAULT '';
  ALTER TABLE acc ADD COLUMN src_ip varchar(64) NOT NULL default '';
  ALTER TABLE acc ADD COLUMN dst_ouser VARCHAR(64) NOT NULL DEFAULT '';
  ALTER TABLE acc ADD COLUMN dst_user VARCHAR(64) NOT NULL DEFAULT '';
  ALTER TABLE acc ADD COLUMN dst_domain VARCHAR(128) NOT NULL DEFAULT '';
  ALTER TABLE missed_calls ADD COLUMN src_user VARCHAR(64) NOT NULL DEFAULT '';
  ALTER TABLE missed_calls ADD COLUMN src_domain VARCHAR(128) NOT NULL DEFAULT '';
  ALTER TABLE missed_calls ADD COLUMN src_ip varchar(64) NOT NULL default '';
  ALTER TABLE missed_calls ADD COLUMN dst_ouser VARCHAR(64) NOT NULL DEFAULT '';
  ALTER TABLE missed_calls ADD COLUMN dst_user VARCHAR(64) NOT NULL DEFAULT '';
  ALTER TABLE missed_calls ADD COLUMN dst_domain VARCHAR(128) NOT NULL DEFAULT '';
#!endif
####### Include Local Config If Exists #########
import_file "kamailio-local.cfg"
####### Defined Values #########
# *** Value defines - IDs used later in config
#!ifdef WITH_MYSQL
# - database URL - used to connect to database server by modules such
#       as: auth_db, acc, usrloc, a.s.o.
#!ifndef DBURL
#!define DBURL "mysql://kamailio:contraseña@localhost/kamailio"
#!endif
#!ifdef WITH_ASTERISK
#!define DBASTURL "mysql://asterisk:contraseña@localhost/asterisk"
#!endif
#!endif
#!ifdef WITH_MULTIDOMAIN
# - the value for 'use_domain' parameters
#!define MULTIDOMAIN 1
#!else
#!define MULTIDOMAIN 0
#!endif

# - flags
#   FLT_ - per transaction (message) flags
#       FLB_ - per branch flags
#!define FLT_ACC 1
#!define FLT_ACCMISSED 2
#!define FLT_ACCFAILED 3
#!define FLT_NATS 5

#!define FLB_NATB 6
#!define FLB_NATSIPPING 7
####### Global Parameters #########
### LOG Levels: 3=DBG, 2=INFO, 1=NOTICE, 0=WARN, -1=ERR
#!ifdef WITH_DEBUG
debug=4
log_stderror=yes
#!else
debug=2
log_stderror=no
#!endif

memdbg=5
memlog=5

log_facility=LOG_LOCAL0

fork=yes
children=4

/* uncomment the next line to disable TCP (default on) */
#disable_tcp=yes

/* uncomment the next line to disable the auto discovery of local aliases
   based on reverse DNS on IPs (default on) */
```

```
auto_aliases=no

/* add local domain aliases */
alias="172.X.X.135:5060"
alias="172.X.X.135:5080"
alias="66.X.X.243:5060"
alias="66.X.X.243:5080"
/* uncomment and configure the following line if you want Kamailio to
   bind on a specific interface/port/proto (default bind on all available) */
listen=udp:172.18.1.135:5060 advertise 66.35.42.243:5060
/* port to listen to
 * - can be specified more than once if needed to listen on many ports */
port=5060
#!ifdef WITH_TLS
enable_tls=yes
#!endif

# life time of TCP connection when there is no traffic
# - a bit higher than registration expires to cope with UA behind NAT
tcp_connection_lifetime=3605

####### Custom Parameters #########
# These parameters can be modified runtime via RPC interface
# - see the documentation of 'cfg_rpc' module.
#
# Format: group.id = value 'desc' description
# Access: $sel(cfg_get.group.id) or @cfg_get.group.id
#

#!ifdef WITH_PSTN
# PSTN GW Routing
#
# - pstn.gw_ip: valid IP or hostname as string value, example:
# pstn.gw_ip = "10.0.0.101" desc "My PSTN GW Address"
#
# - by default is empty to avoid misrouting
pstn.gw_ip = "" desc "PSTN GW Address"
pstn.gw_port = "" desc "PSTN GW Port"
#!endif

#!ifdef WITH_VOICEMAIL
# VoiceMail Routing on offline, busy or no answer
#
# - by default Voicemail server IP is empty to avoid misrouting
voicemail.srv_ip = "" desc "VoiceMail IP Address"
voicemail.srv_port = "5060" desc "VoiceMail Port"
#!endif

#!ifdef WITH_ASTERISK
asterisk.bindip = "172.18.1.135" desc "Asterisk IP Address"
asterisk.bindport = "5080" desc "Asterisk Port"
kamailio.bindip = "172.18.1.135" desc "Kamailio IP Address"
kamailio.bindport = "5060" desc "Kamailio Port"
#!endif

####### Modules Section ########

# set paths to location of modules (to sources or installation folders)
#!ifdef WITH_SRCPATH
mpath="modules/"
#!else
mpath="/usr/lib64/kamailio/modules/"
#!endif

#!ifdef WITH_MYSQL
loadmodule "db_mysql.so"
#!endif

loadmodule "mi_fifo.so"
loadmodule "kex.so"
loadmodule "corex.so"
loadmodule "tm.so"
loadmodule "tmx.so"
loadmodule "sl.so"
loadmodule "rr.so"
loadmodule "pv.so"
loadmodule "maxfwd.so"
loadmodule "usrloc.so"
loadmodule "registrar.so"
loadmodule "textops.so"
loadmodule "siputils.so"
```

```
loadmodule "xlog.so"
loadmodule "sanity.so"
loadmodule "ctl.so"
loadmodule "cfg_rpc.so"
loadmodule "mi_rpc.so"
loadmodule "acc.so"
loadmodule "outbound.so"

#!ifdef WITH_AUTH
loadmodule "auth.so"
loadmodule "auth_db.so"
#!ifdef WITH_IPAUTH
loadmodule "permissions.so"
#!endif
#!endif

#!ifdef WITH_ALIASDB
loadmodule "alias_db.so"
#!endif

#!ifdef WITH_SPEEDDIAL
loadmodule "speeddial.so"
#!endif

#!ifdef WITH_MULTIDOMAIN
loadmodule "domain.so"
#!endif

#!ifdef WITH_PRESENCE
loadmodule "presence.so"
loadmodule "presence_xml.so"
#!endif

#!ifdef WITH_NAT
loadmodule "nathelper.so"
loadmodule "rtpproxy.so"
#!endif

#!ifdef WITH_TLS
loadmodule "tls.so"
#!endif

#!ifdef WITH_ANTIFLOOD
loadmodule "htable.so"
loadmodule "pike.so"
#!endif

#!ifdef WITH_XMLRPC
loadmodule "xmlrpc.so"
#!endif

#!ifdef WITH_DEBUG
loadmodule "debugger.so"
#!endif

#!ifdef WITH_ASTERISK
loadmodule "uac.so"
#!endif

# ----------------- setting module-specific parameters ---------------
# ----- mi_fifo params -----
modparam("mi_fifo", "fifo_name", "/tmp/kamailio_fifo")


#-----------------------ctl module-------------------------
modparam("ctl", "user", "root")


# ----- tm params -----
# auto-discard branches from previous serial forking leg
modparam("tm", "failure_reply_mode", 3)
# default retransmission timeout: 30sec
modparam("tm", "fr_timer", 30000)
# default invite retransmission timeout after 1xx: 120sec
modparam("tm", "fr_inv_timer", 120000)
modparam("tm", "local_ack_mode", 1)

# ----- rr params -----
# add value to ;lr param to cope with most of the UAs
modparam("rr", "enable_full_lr", 1)
# do not append from tag to the RR (no need for this script)
```

```
#!ifdef WITH_ASTERISK
modparam("rr", "append_fromtag", 1)
#!else
modparam("rr", "append_fromtag", 0)
#!endif

# ----- registrar params -----
modparam("registrar", "method_filtering", 1)
/* uncomment the next line to disable parallel forking via location */
# modparam("registrar", "append_branches", 0)
/* uncomment the next line not to allow more than 10 contacts per AOR */
#modparam("registrar", "max_contacts", 10)
# max value for expires of registrations
modparam("registrar", "max_expires", 3600)
# set it to 1 to enable GRUU
modparam("registrar", "gruu_enabled", 0)


# ----- acc params -----
/* what special events should be accounted ? */
modparam("acc", "early_media", 0)
modparam("acc", "report_ack", 1)
modparam("acc", "report_cancels", 0)
/* by default ww do not adjust the direct of the sequential requests.
   if you enable this parameter, be sure the enable "append_fromtag"
   in "rr" module */
modparam("acc", "detect_direction", 0)
/* account triggers (flags) */
modparam("acc", "log_flag", FLT_ACC)
modparam("acc", "log_missed_flag", FLT_ACCMISSED)
modparam("acc", "log_extra",
         "src_user=$fU;src_domain=$fd;src_ip=$si;"
         "dst_ouser=$tU;dst_user=$rU;dst_domain=$rd")
modparam("acc", "failed_transaction_flag", FLT_ACCFAILED)
/* enhanced DB accounting */
#!ifdef WITH_ACCDB
modparam("acc", "db_flag", FLT_ACC)
modparam("acc", "db_missed_flag", FLT_ACCMISSED)
modparam("acc", "db_url", DBURL)
modparam("acc", "db_extra",
         "src_user=$fU;src_domain=$fd;src_ip=$si;"
         "dst_ouser=$tU;dst_user=$rU;dst_domain=$rd")
#!endif


# ----- usrloc params -----
/* enable DB persistency for location entries */
#!ifdef WITH_USRLOCDB
modparam("usrloc", "db_url", DBURL)
modparam("usrloc", "db_mode", 2)
modparam("usrloc", "use_domain", MULTIDOMAIN)
#!endif


# ----- auth_db params -----
#!ifdef WITH_AUTH
modparam("auth_db", "calculate_ha1", yes)
modparam("auth_db", "load_credentials", "")


#!ifdef WITH_ASTERISK
modparam("auth_db", "user_column", "name")
modparam("auth_db", "password_column", "sippasswd")
modparam("auth_db", "db_url", DBASTURL)
modparam("auth_db", "version_table", 0)
#!else
modparam("auth_db", "db_url", DBURL)
modparam("auth_db", "password_column", "password")
modparam("auth_db", "use_domain", MULTIDOMAIN)
#!endif
# ----- permissions params -----
#!ifdef WITH_IPAUTH
modparam("permissions", "db_url", DBURL)
modparam("permissions", "db_mode", 1)
#!endif
#!endif

# ----- alias_db params -----
#!ifdef WITH_ALIASDB
modparam("alias_db", "db_url", DBURL)
modparam("alias_db", "use_domain", MULTIDOMAIN)
```

```
#!endif

# ----- speeddial params -----
#!ifdef WITH_SPEEDDIAL
modparam("speeddial", "db_url", DBURL)
modparam("speeddial", "use_domain", MULTIDOMAIN)
#!endif

# ----- domain params -----
#!ifdef WITH_MULTIDOMAIN
modparam("domain", "db_url", DBURL)
# register callback to match myself condition with domains list
modparam("domain", "register_myself", 1)
#!endif

#!ifdef WITH_PRESENCE
# ----- presence params -----
modparam("presence", "db_url", DBURL)
# ----- presence_xml params -----
modparam("presence_xml", "db_url", DBURL)
modparam("presence_xml", "force_active", 1)
#!endif

#!ifdef WITH_NAT
# ----- rtpproxy params -----
modparam("rtpproxy", "rtpproxy_sock", "udp:127.0.0.1:7722")

# ----- nathelper params -----
modparam("nathelper", "natping_interval", 30)
modparam("nathelper", "ping_nated_only", 1)
modparam("nathelper", "sipping_bflag", FLB_NATSIPPING)
modparam("nathelper", "sipping_from", "sip:pinger@kamailio.org")

# params needed for NAT traversal in other modules
modparam("nathelper|registrar", "received_avp", "$avp(RECEIVED)")
modparam("usrloc", "nat_bflag", FLB_NATB)
#!endif

#!ifdef WITH_TLS
# ----- tls params -----
modparam("tls", "config", "//etc/kamailio/tls.cfg")
#!endif

#!ifdef WITH_ANTIFLOOD
# ----- pike params -----
modparam("pike", "sampling_time_unit", 2)
modparam("pike", "reqs_density_per_unit", 16)
modparam("pike", "remove_latency", 4)

# ----- htable params -----
# ip ban htable with autoexpire after 5 minutes
modparam("htable", "htable", "ipban=>size=8;autoexpire=300;")
#!endif

#!ifdef WITH_XMLRPC
# ----- xmlrpc params -----
modparam("xmlrpc", "route", "XMLRPC");
modparam("xmlrpc", "url_match", "^/RPC")
#!endif

#!ifdef WITH_DEBUG
# ----- debugger params -----
modparam("debugger", "cfgtrace", 1)
#!endif

####### Routing Logic ########

# Main SIP request routing logic
# - processing of any incoming SIP request starts with this route
# - note: this is the same as route { ... }
request_route {

        # per request initial checks
        route(REQINIT);

        # NAT detection
        route(NATDETECT);

        # CANCEL processing
        if (is_method("CANCEL"))
        {
```

```
                if (t_check_trans()) {
                        route(RELAY);
                }
                exit;
        }

        # handle requests within SIP dialogs
        route(WITHINDLG);
        ### only initial requests (no To tag)

        t_check_trans();

        # authentication
        route(AUTH);

        # record routing for dialog forming requests (in case they are routed)
        # - remove preloaded route headers
        remove_hf("Route");
        if (is_method("INVITE|SUBSCRIBE"))
                record_route();

        # account only INVITEs
        if (is_method("INVITE|ACK"))
        {
                setflag(FLT_ACC); # do accounting
        }

        # dispatch requests to foreign domains
        route(SIPOUT);

        ### requests for my local domains

        # handle presence related requests
        route(PRESENCE);

        # handle registrations
        route(REGISTRAR);

        if ($rU==$null)
        {
                # request with no Username in RURI
                sl_send_reply("484","Address Incomplete");
                exit;
        }

        # dispatch destinations to PSTN
        route(PSTN);

        # user location service
        route(LOCATION);
}


route[RELAY] {

        # enable additional event routes for forwarded requests
        # - serial forking, RTP relaying handling, a.s.o.
        if (is_method("INVITE|BYE|SUBSCRIBE|UPDATE")) {
                if(!t_is_set("branch_route")){ t_on_branch("MANAGE_BRANCH");}
        }
        if (is_method("INVITE|SUBSCRIBE|UPDATE")) {
                if(!t_is_set("onreply_route")) t_on_reply("MANAGE_REPLY");
        }
        if (is_method("INVITE")) {
                if(!t_is_set("failure_route")) t_on_failure("MANAGE_FAILURE");
        }

        if (!t_relay()) {
                sl_reply_error();
        }
        exit;
}

# Per SIP request initial checks
route[REQINIT] {
#!ifdef WITH_ANTIFLOOD
        # flood dection from same IP and traffic ban for a while
        # be sure you exclude checking trusted peers, such as pstn gateways
        # - local host excluded (e.g., loop to self)
        if(src_ip!=myself)
        {
```

```
                if($sht(ipban=>$si)!=$null)
                {
                        # ip is already blocked
                        xdbg("request from blocked IP - $rm from $fu (IP:$si:$sp)\n");
                        exit;
                }
                if (!pike_check_req())
                {
                        xlog("L_ALERT","ALERT: pike blocking $rm from $fu (IP:$si:$sp)\n");
                        $sht(ipban=>$si) = 1;
                        exit;
                }
        }
#!endif

        if (!mf_process_maxfwd_header("10")) {
                sl_send_reply("483","Too Many Hops");
                exit;
        }

        if(!sanity_check("1511", "7"))
        {
                xlog("Malformed SIP message from $si:$sp\n");
                exit;
        }
}

# Handle requests within SIP dialogs
route[WITHINDLG] {
        if (has_totag()) {
                # sequential request withing a dialog should
                # take the path determined by record-routing
                if (loose_route()) {
                        route(DLGURI);
                        if (is_method("BYE")) {
                                setflag(FLT_ACC); # do accounting ...
                                setflag(FLT_ACCFAILED); # ... even if the transaction fails
                        }
                        else if ( is_method("ACK") ) {
                                # ACK is forwarded statelessy
                                record_route();
                                route(NATMANAGE);
                        }
                        else if ( is_method("NOTIFY") ) {
                                # Add Record-Route for in-dialog NOTIFY as per RFC 6665.
                                record_route();
                        }
                        route(RELAY);
                } else {
                        if (is_method("SUBSCRIBE") && uri == myself) {
                                # in-dialog subscribe requests
                                route(PRESENCE);
                                exit;
                        }
                        if ( is_method("ACK") ) {
                                if ( t_check_trans() ) {
                                        # no loose-route, but stateful ACK;
                                        # must be an ACK after a 487
                                        # or e.g. 404 from upstream server
                                        route(RELAY);
                                        exit;
                                } else {
                                        # ACK without matching transaction ... ignore and discard
                                        exit;
                                }
                        }
                        sl_send_reply("404","Not here");
                }
                exit;
        }
}

# Handle SIP registrations
route[REGISTRAR] {
        if (is_method("REGISTER"))
        {
                if(isflagset(FLT_NATS))
                {
                        setbflag(FLB_NATB);
                        # uncomment next line to do SIP NAT pinging
                        ## setbflag(FLB_NATSIPPING);
```

```
                }
                if (!save("location"))
                        sl_reply_error();


        #!ifdef WITH_ASTERISK
        route(REGFWD);
        #!endif
                exit;


        }
}

# USER location service
route[LOCATION] {

#!ifdef WITH_SPEEDDIAL
        # search for short dialing - 2-digit extension
        if($rU=~"^[0-9][0-9]$")
                if(sd_lookup("speed_dial"))
                        route(SIPOUT);
#!endif

#!ifdef WITH_ALIASDB
        # search in DB-based aliases
        if(alias_db_lookup("dbaliases"))
                route(SIPOUT);
#!endif

#!ifdef WITH_ASTERISK
        if(is_method("INVITE") && (!route(FROMASTERISK))) {
                # if new call from out there - send to Asterisk
                # - non-INVITE request are routed directly by Kamailio
                # - traffic from Asterisk is routed also directy by Kamailio
                route(TOASTERISK);
                exit;
        }
#!endif

        $avp(oexten) = $rU;
        if (!lookup("location")) {
                $var(rc) = $rc;
                route(TOVOICEMAIL);
                t_newtran();
                switch ($var(rc)) {
                        case -1:
                        case -3:
                                send_reply("404", "Not Found");
                                exit;
                        case -2:
                                send_reply("405", "Method Not Allowed");
                                exit;
                }
        }

        # when routing via usrloc, log the missed calls also
        if (is_method("INVITE"))
        {
                setflag(FLT_ACCMISSED);
        }

        route(RELAY);
        exit;
}

# Presence server route
route[PRESENCE] {
        if(!is_method("PUBLISH|SUBSCRIBE"))
                return;

        if(is_method("SUBSCRIBE") && $hdr(Event)=="message-summary") {
                route(TOVOICEMAIL);
                # returns here if no voicemail server is configured
                sl_send_reply("404", "No voicemail service");
                exit;
        }

#!ifdef WITH_PRESENCE
        if (!t_newtran())
        {
```

```
                sl_reply_error();
                exit;
        }

        if(is_method("PUBLISH"))
        {
                handle_publish();
                t_release();
        } else if(is_method("SUBSCRIBE")) {
                handle_subscribe();
                t_release();
        }
        exit;
#!endif

        # if presence enabled, this part will not be executed
        if (is_method("PUBLISH") || $rU==$null)
        {
                sl_send_reply("404", "Not here");
                exit;
        }
        return;
}

# Authentication route
route[AUTH] {
#!ifdef WITH_AUTH

#!ifdef WITH_ASTERISK
        # do not auth traffic from Asterisk - trusted!
        if(route(FROMASTERISK))
                return;
#!endif

#!ifdef WITH_IPAUTH
        if((!is_method("REGISTER")) && allow_source_address())
        {
                # source IP allowed
                return;
        }
#!endif

        if (is_method("REGISTER") || from_uri==myself)
        {


#!ifdef WITH_ASTERISK
                if (!auth_check("$fd", "sipusers", "1")) {
#!else

                # authenticate requests
                if (!auth_check("$fd", "subscriber", "1")) {
#!endif
                        auth_challenge("$fd", "0");
                        exit;
                }
                # user authenticated - remove auth header
                if(!is_method("REGISTER|PUBLISH"))
                        consume_credentials();
        }
        # if caller is not local subscriber, then check if it calls
        # a local destination, otherwise deny, not an open relay here
        if (from_uri!=myself && uri!=myself)
        {
                sl_send_reply("403","Not relaying");
                exit;
        }

#!endif
        return;
}

# Caller NAT detection route
route[NATDETECT] {
#!ifdef WITH_NAT
        force_rport();
        if (nat_uac_test("19")) {
                if (is_method("REGISTER")) {
                        fix_nated_register();
                } else {
                        if(is_first_hop())
```

```
                                set_contact_alias();
                }
                setflag(FLT_NATS);
        }
#!endif
        return;
}

# RTPProxy control
route[NATMANAGE] {
#!ifdef WITH_NAT
        if (is_request()) {
                if(has_totag()) {
                        if(check_route_param("nat=yes")) {
                                setbflag(FLB_NATB);
                        }
                }
        }
        if (!(isflagset(FLT_NATS) || isbflagset(FLB_NATB)))
                return;

        rtpproxy_manage("co");

        if (is_request()) {
                if (!has_totag()) {
                        if(t_is_branch_route()) {
                                add_rr_param(";nat=yes");
                        }
                }
        }
        if (is_reply()) {
                if(isbflagset(FLB_NATB)) {
                        if(is_first_hop())
                                set_contact_alias();
                }
        }
#!endif
        return;
}

# URI update for dialog requests
route[DLGURI] {
#!ifdef WITH_NAT
        if(!isdsturiset()) {
                handle_ruri_alias();
        }
#!endif
        return;
}

# Routing to foreign domains
route[SIPOUT] {
        if (!uri==myself)
        {
                append_hf("P-hint: outbound\r\n");
                route(RELAY);
        }
}

# PSTN GW routing
route[PSTN] {
#!ifdef WITH_PSTN
        # check if PSTN GW IP is defined
        if (strempty($sel(cfg_get.pstn.gw_ip))) {
                xlog("SCRIPT: PSTN rotuing enabled but pstn.gw_ip not defined\n");
                return;
        }

        # route to PSTN dialed numbers starting with '+' or '00'
        #     (international format)
        # - update the condition to match your dialing rules for PSTN routing
        if(!($rU=~"^(\+|00)[1-9][0-9]{3,20}$"))
                return;

        # only local users allowed to call
        if(from_uri!=myself) {
                sl_send_reply("403", "Not Allowed");
                exit;
        }

        if (strempty($sel(cfg_get.pstn.gw_port))) {
```

```
                $ru = "sip:" + $rU + "@" + $sel(cfg_get.pstn.gw_ip);
        } else {
                $ru = "sip:" + $rU + "@" + $sel(cfg_get.pstn.gw_ip) + ":"
                                        + $sel(cfg_get.pstn.gw_port);
        }

        route(RELAY);
        exit;
#!endif

        return;
}

# XMLRPC routing
#!ifdef WITH_XMLRPC
route[XMLRPC] {
        # allow XMLRPC from localhost
        if ((method=="POST" || method=="GET")
                        && (src_ip==127.0.0.1)) {
                # close connection only for xmlrpclib user agents (there is a bug in
                # xmlrpclib: it waits for EOF before interpreting the response).
                if ($hdr(User-Agent) =~ "xmlrpclib")
                        set_reply_close();
                set_reply_no_connect();
                dispatch_rpc();
                exit;
        }
        send_reply("403", "Forbidden");
        exit;
}
#!endif

# route to voicemail server
route[TOVOICEMAIL] {
#!ifdef WITH_VOICEMAIL
        if(!is_method("INVITE|SUBSCRIBE"))
                return;

        # check if VoiceMail server IP is defined
        if (strempty($sel(cfg_get.voicemail.srv_ip))) {
                xlog("SCRIPT: VoiceMail rotuing enabled but IP not defined\n");
                return;
        }
        if(is_method("INVITE")) {
                if($avp(oexten)==$null)
                        return;
                $ru = "sip:" + $avp(oexten) + "@" + $sel(cfg_get.voicemail.srv_ip)
                                + ":" + $sel(cfg_get.voicemail.srv_port);
        } else {
                if($rU==$null)
                        return;
                $ru = "sip:" + $rU + "@" + $sel(cfg_get.voicemail.srv_ip)
                                + ":" + $sel(cfg_get.voicemail.srv_port);
        }
        route(RELAY);
        exit;
#!endif

        return;
}

# manage outgoing branches
branch_route[MANAGE_BRANCH] {
        xdbg("new branch [$T_branch_idx] to $ru\n");
        route(NATMANAGE);
}

# manage incoming replies
onreply_route[MANAGE_REPLY] {
        xdbg("incoming reply\n");
        if(status=~"[12][0-9][0-9]")
                route(NATMANAGE);
}

# manage failure routing cases
failure_route[MANAGE_FAILURE] {
        route(NATMANAGE);

        if (t_is_canceled()) {
                exit;
        }
```

```
#!ifdef WITH_BLOCK3XX
        # block call redirect based on 3xx replies.
        if (t_check_status("3[0-9][0-9]")) {
                t_reply("404","Not found");
                exit;
        }
#!endif

#!ifdef WITH_VOICEMAIL
        # serial forking
        # - route to voicemail on busy or no answer (timeout)
        if (t_check_status("486|408")) {
                $du = $null;
                route(TOVOICEMAIL);
                exit;
        }
#!endif
}

#!ifdef WITH_ASTERISK
# Test if coming from Asterisk
route[FROMASTERISK] {
        if($si==$sel(cfg_get.asterisk.bindip)
                        && $sp==$sel(cfg_get.asterisk.bindport))
                return 1;
        return -1;
}

# Send to Asterisk
route[TOASTERISK] {
        $du = "sip:" + $sel(cfg_get.asterisk.bindip) + ":"
                        + $sel(cfg_get.asterisk.bindport);
        route(RELAY);
        exit;
}

# Forward REGISTER to Asterisk
route[REGFWD] {
        if(!is_method("REGISTER"))
        {
                return;
        }
        $var(rip) = $sel(cfg_get.asterisk.bindip);
        $uac_req(method)="REGISTER";
        $uac_req(ruri)="sip:" + $var(rip) + ":" + $sel(cfg_get.asterisk.bindport);
        $uac_req(furi)="sip:" + $au + "@" + $var(rip);
        $uac_req(turi)="sip:" + $au + "@" + $var(rip);
        $uac_req(hdrs)="Contact: <sip:" + $au + "@"
                                + $sel(cfg_get.kamailio.bindip)
                                + ":" + $sel(cfg_get.kamailio.bindport) + ">\r\n";
        if($sel(contact.expires) != $null)
                $uac_req(hdrs)= $uac_req(hdrs) + "Expires: " + $sel(contact.expires) + "\r\n";
        else
                $uac_req(hdrs)= $uac_req(hdrs) + "Expires: " + $hdr(Expires) + "\r\n";
        uac_req_send();
}
#!endif
```

ANEXO 4

Archivos de configuración para servidor Asterisk

sip.conf

```
[general]
context=public
allowguest=no
allowoverlap=no
udpbindaddr=0.0.0.0:5080
tcpenable=no
tcpbindaddr=0.0.0.0:5080
transport=udp
srvlookup=yes
disallow=all
allow=gsm
tonezone=ve
dtmfmode = rfc2833
alwaysauthreject = yes
outboundproxy=172.18.1.135:5060
;----------------------------------------- NAT SUPPORT ------------------------
nat=auto_force_rport,auto_comedia
externip=66.35.42.243
localnet=172.17.0.0/255.255.255.0
media_address = 66.35.42.243
;---------------------------------- MEDIA HANDLING ---------------------------------
directmedia=no
;---------------------------------------- REALTIME SUPPORT ------------------------
rtcachefriends=yes
rtupdate=yes
;----------------------------------------- Domain ------------------------
domain=172.18.1.135              ; Add IP address as local domain
```

rtp.conf

```
[general]

rtpstart=10000
rtpend=10100
```

res_odbc.conf

```
[asterisk]
enabled => yes
dsn => asterisk-connector
username => asterisk
password => contraseña
pooling => no
limit => 3
pre-connect => yes
```

modules.conf

```
[modules]
autoload=yes
```

extconfig.conf

```
[settings]
sipusers => odbc,asterisk,sipusers
sippeers => odbc,asterisk,sipusers
sipregs => odbc,asterisk,sipregs
```