ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

DESARROLLO DE UN PROTOTIPO DE APLICATIVO WEB BASADO EN WEBRTC Y VOIP

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN TECNOLOGÍAS DE LA INFORMACIÓN

JIMMY ANDRES PARDO VILCACUNDO

jimmy.pardo@epn.edu.ec

DIRECTOR: PhD. LUIS FELIPE URQUIZA AGUIAR

luis.urquiza@epn.edu.ec

DMQ, Enero 2022

CERTIFICACIONES

Yo, Jimmy Andres Pardo Vilcacundo declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

Jimmy Andrés Pardo Vilcacundo

Certifico que el presente trabajo de integración curricular fue desarrollado por Jimmy Andrés Pardo Vilcacundo, bajo mi supervisión.

PhD. Luis Felipe Urquiza Aguiar
DIRECTOR DEL TRABAJO DE TITULACIÓN

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el (los) producto(s) resultante(s) del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

Jimmy Andres Pardo Vilcacundo

PhD. Luis Urquiza

DEDICATORIA

A mis padres por brindarme su amor y apoyo incondicional, en cada etapa de mi vida, todos mis triunfos serán para ustedes. Gracias por inculcarme todos los valores morales y éticos que poseo y aconsejarme sobre los que debo mejorar.

A mi hermana que siempre me ha apoyado y con quien he compartido muchas vivencias y experiencias, gracias por siempre estar ahí cuando más te he necesitado.

AGRADECIMIENTO

Primero, el agradecimiento a Dios por darme desde el principio y cada día muchas fuerzas para continuar y no decaer en el camino.

Deseo manifestar un agradecimiento para los ingenieros e ingenieras que ayudaron con ideas y soluciones para poder lograr este trabajo de titulación, en especial a Luis Urquiza que supo guiar hasta el final para conseguir lo que nos habíamos propuesto.

Agradezco también de manera especial a mis amigos con los que sin duda he compartido mil y una experiencias a lo largo de esta carrera, muchas gracias a todos por la confianza y la amistad brindada. Para: Erick, Josué, Giss, Stalyn y Sofy.

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE CÓDIGOS

RESUMEN

El presente proyecto describe la implementación de un prototipo de software publicado en Internet para la comunicación de audio y video entre navegadores web, así como, de comunicación de audio con la PSTN mediante una línea analógica. Los usuarios utilizan una aplicación web, que implementa un cliente sipML5, mediante una librería escrita en JavaScript. Esta librería incorpora la tecnología WebRTC, también se encarga de generar la señalización SIP que se necesita para el establecimiento de una sesión. El transporte de los datos se realiza mediante el protocolo Websocket, que permite tener comunicación bidireccional entre el cliente web y el servidor de comunicaciones. El servidor de comunicaciones implementa los siguientes servicios: Web, Websockets, SIP, IDS.

Para la fase de diseñó del prototipo fue necesario estudiar conceptos y tecnologías tales como: WebRTC, sipML5, Asterisk, Fail2ban, HTML5, Webscoket y EC2 de AWS. Además de analizar los requerimientos de software obtenidos mediante encuestas.

Se describió la instalación de las herramientas necesarias y la implementación de cada capa establecida en la etapa de diseño para el desarrollo del prototipo. Finalmente se presentaron las pruebas de funcionamiento y validación, de acuerdo con los casos de uso definidos en la etapa de diseño, siendo estas satisfactorias.

PALABRAS CLAVE: WebRTC, VoIP, sipML5, EC2 de AWS.

ABSTRACT

This document describes a project that proposes the implementation of web prototype published on the Internet for audio and video communication between web browsers, as well as audio communication with the PSTN through an analog line. Users use a web application, which implements a sipML5 client, through a library written in JavaScript. This library incorporates WebRTC technology, it is also responsible for generating the SIP signaling that is needed to establish a session. The transport of data is carried out through the WebSocket protocol, which allows bidirectional communication between the web client and the communications server. The communication server implements the services: Web, WebSocket, SIP, IDS.

The prototype design phase considers to study concepts and technologies such as: WebRTC, sipML5, Asterisk, Fail2ban, HTML5, Websocket and AWS EC2. Also, in this phase the software requirements obtained through surveys, were analyzed.

The installation of the necessary tools and the implementation of each layer established in the design stage for the development of the prototype were described. Finally, the functional and validation tests were presented, according to the use cases defined in the design phase, these were satisfactory.

KEYWORDS: WebRTC, VoIP, sipML5, AWS Elastic Computing.

1 INTRODUCCIÓN

La necesidad de comunicarse en tiempo real mediante aplicaciones web que permitan transmitir audio y video se ha vuelto más importante gracias al auge del trabajo remoto.

Este proyecto fue inspirado por la librería sipML5 ya que permite crear aplicaciones web basadas en WebRTC, Websockets y señalización SIP que es la más empleada en VoIP. Utilizando Asterisk como servidor de comunicaciones, se puede lograr que los clientes web, que implementan el API sipML5, se comuniquen con cualquier dispositivo SIP, ofreciendo convergencia, ya que empleando un Gateway se podría digitalizar la voz proveniente, por ejemplo, de una línea analógica convencional y de esta manera comunicar un navegador web con la PSTN.

Además, Asterisk ofrece varios conectores con distinto motores de bases de datos, lo que facilita la creación de aplicaciones que permitan la administración de usuarios que en palabras simples serian clientes web que utilizan el API sipML5 para comunicarse.

Ya que la idea del proyecto comenzó como una solución simple para poder comunicar a dos usuarios que no se encontrarán compartiendo una misma red, el prototipo está publicado en Internet, utilizando una instancia EC2 de AWS para este propósito. Esto facilita mucho el uso del aplicativo ya que no se necesita ningún plugin adicional, lo único necesario es siempre tener acceso a Internet y un navegador web.

1.1 Objetivo general

Desarrollar un prototipo de aplicación web publicada en Internet basado en WEBRTC y VOIP.

1.2 Objetivos específicos

- Analizar los conceptos fundamentales para el desarrollo del prototipo.
- Diseñar los módulos del prototipo de aplicación web.
- Implementar los módulos de la aplicación web y su funcionalidad.
- Verificar el funcionamiento del prototipo.

1.3 Alcance

El presente proyecto tiene como finalidad desarrollar un prototipo de aplicación web para la transmisión de audio y video mediante WebRTC (Web Real Time Communication) y cualquier dispositivo que soporte el protocolo SIP. Para el alojamiento de la aplicación web se utilizará una instancia EC2 de AWS (Amazon Web Services). Para el apartado de seguridad se implementará un IDS (Intrusion Detection System) por software para evitar ataques de denegación de servicio, adicional se utilizará protocolos seguros en capa aplicación.

1.4 Marco teórico

En este capítulo se abordarán conceptos que servirán para el desarrollo del prototipo. Se detallará sobre VoIP, WebRTC y Asterisk como servidor de comunicaciones en tiempo real, así como AWS como plataforma de virtualización en la nube y sobre la definición de IDS.

1.4.1 VoIP

VoIP o voz sobre IP se refiere a la posibilidad de transmitir señales de voz sobre redes basadas en el protocolo IP, a diferencia de las redes conmutadas tradicionales, VoIP aprovecha el proceso de digitalización de la voz para inyectar estos bits en la red como cualquier otro servicio que utiliza IP como protocolo de enrutamiento [1]. Los elementos necesarios para una comunicación VoIP son:

- **Cliente. -** Es el que consume el servicio de VoIP mediante un equipo que posea un micrófono, puede ser dedicado como un teléfono IP o emulado a través de un software como un softphone.
- **Servidor. -** Es el encargado de gestionar todas las operaciones necesarias para el inicio, mantenimiento y enrutamiento de las llamadas establecidas por los clientes, así como del registro de estos. Generalmente en los servidores se instala software dedicado siendo estos propietarios como en el caso de Cisco o Avaya o pueden ser de código abierto como Asterisk. Se los denominan como IP-PBX [1].

- **Gateway. -** Su misión se centra en intercomunicar sistemas tradicionales de telefonía para ofrecer convergencia entre usuarios VoIP y usuarios externos.

1.4.2 Protocolo SIP

El protocolo SIP o Protocolo de Inicio de Sesión fue desarrollado por la IETF como mecanismo para iniciación, modificación y finalización de sesiones de usuario donde intervienen elementos multimedia como el video, voz, mensajería instantánea, etc. Se complementa con el protocolo SDP para describir la sesión; parámetros como IP, puertos, códecs que serán utilizados en la sesión se negocian mediante este protocolo [2].

Las funciones básicas del protocolo incluyen, determinar la ubicación de los usuarios y el control de las sesiones entre múltiples usuarios.

En la siguiente tabla se puede observar los mensajes principales que utiliza SIP para el establecimiento de sesiones.

Tabla 1.1. Mensajes SIP

Mensaje	**Descripción**
Invite	Se lo utiliza para iniciar una sesión. Generalmente una llamada telefónica.
ACK	Utilizado para confirmación de respuestas.
Bye	Se lo utiliza en la terminación de una sesión establecida.
Cancel	Utilizado para cancelar una petición que aún no ha sido atendida.
Options	Utilizado para consultar las capacidades de otro cliente SIP. Códecs, métodos, etc
Register	Utilizado para registrar un cliente SIP en un servidor SIP.

1.4.3 Protocolo RTP

El protocolo RTP o protocolo de transporte en tiempo real es un protocolo de capa aplicación que se utiliza para la transmisión de audio y video en aplicaciones en tiempo real como videoconferencias. Al igual que el protocolo SIP fue desarrollado por la IETF y junto con este representan la base de la industria VoIP.

Generalmente usa UDP en la capa transporte dentro del modelo TCP/IP, además de RTCP (RTP Control Protocol) también a nivel aplicación para el control de la información en una sesión RTP. Es posible proporcionar una capa de seguridad mediante SRTP

(Secure Real-time Transport Protocol). Utiliza Estándares de Encriptación Avanzada (AES) como cifrado de encriptación por defecto.

1.4.4 SDP

SDP o protocolo de inicio de sesión establece un estándar para definir los parámetros que se usaran en el intercambio de media entre dos dispositivos o endpoints. El SDP típicamente se encapsula en otro protocolo siendo el protocolo SIP el más usado.

El protocolo SDP permite declarar las capacidades y especificaciones que posee un endpoint para establecer el flujo de media. De manera general se declara:

- La dirección IP y puerto habilitado para recepción y envió de media.
- El protocolo que soporta el endpoint para la transmisión de media.
- El tipo de media que espera recibir generalmente audio o video y los códecs que soporta el endpoint.

SDP solo se limita a la negociación de un set de parámetros compatibles para el intercambio de media, por lo que por sí solo no puede establecer una sesión.

En un proceso típico de configuración de sesión entre dos endpoints, veremos como interactúan el protocolo SIP y SDP para iniciar y establecer una sesión. Una vez la sesión sea establecida la transmisión de media será realizada por el protocolo RTP.

Con estos conceptos hemos definido de manera general, como se establecen la mayor parte de las comunicaciones en el mundo VoIP.

1.4.5 WebRTC

WebRTC o Web Real-Time Communication es una tecnología que permite establecer comunicaciones en tiempo real sobre los navegadores Web de forma nativa [3]. El mayor uso de la tecnología WebRTC son audio y / o videollamadas en tiempo real, conferencias web y transferencia de datos. En la Figura 1.1 se puede observar la arquitectura

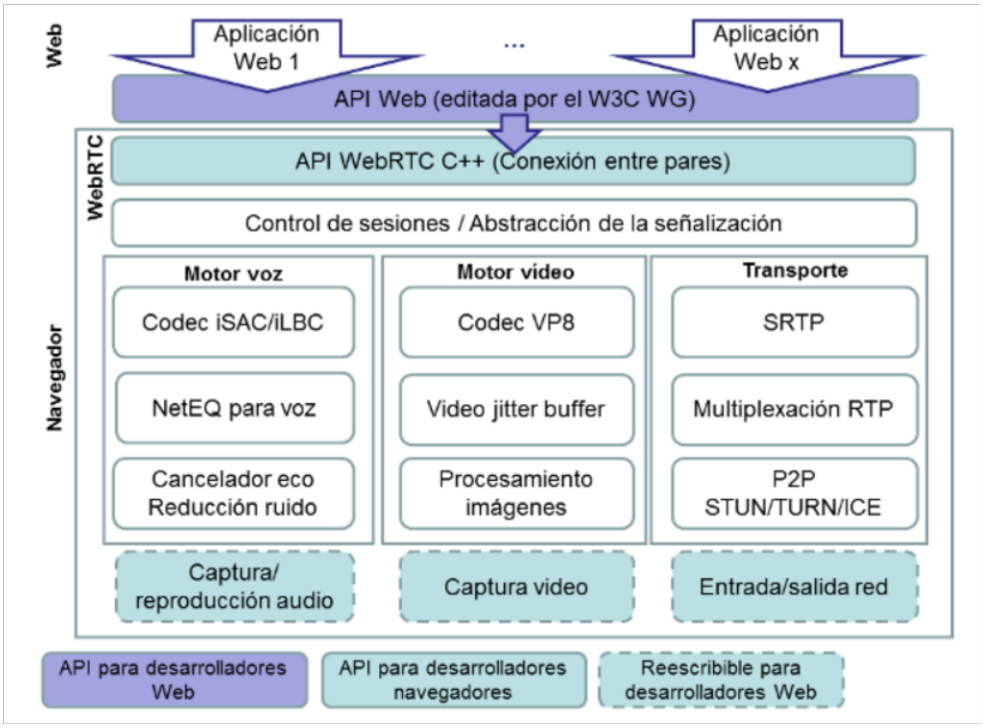


Figura 1.1 Arquitectura WebRTC

Como se visualiza en la figura la arquitectura WebRTC ofrece distintos API enfocados para desarrolladores, así también define protocolos para el transporte en la red y códecs para el audio/video. Todos estos parámetros son configurados mediante las API, usadas principalmente por los desarrolladores web.

La arquitectura define características y parámetros para poder hacer uso de sus API. Define los códecs a utilizar para la transmisión de audio/video, así como los protocolos para el transporte. De esta manera se utilizará SRTP para transmitir los paquetes de audio y video y se requiere del protocolo STUN para prevenir inconvenientes ocasionados por las traducciones NAT.

Para el control de sesiones WebRTC no define ningún protocolo y lo deja a libertad del desarrollador, esto presenta una ventaja ya que flexibiliza la integración con sistemas de comunicaciones en tiempo real basados en SIP.

A continuación, se presenta una tabla comparativa entre los protocolos que se usan en VoIP y en la arquitectura WebRTC.

Tabla 1.2. Comparación entre VoIP y WebRTC

Característica	**VoIP**	**WebRTC**
Señalización	SIP (principalmente), H.323	No se define
Medios	RTP	SRTP
Códecs de voz	Serie G.7xxx (principalmente)	Opus y G711
Códecs de video	H.264	VP8

1.4.6 SipML5

SipML5 es un cliente SIP programado en JavaScript de código abierto [4]. Para la transferencia del flujo de audio o video depende de la arquitectura WebRTC. Para la señalización utiliza el protocolo Websocket embebiendo a SIP como subprotocolo. De esta forma se puede conectar al cliente con cualquier infraestructura RTC basada en SIP.

SipML5 nos ofrece un API para poder implementar un cliente SIP desde un navegador web, proporcionando todas las funciones para el control y descripción de las sesiones.

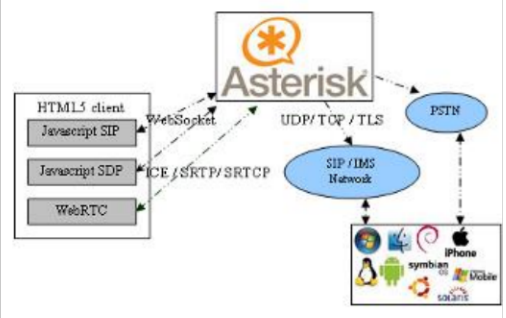


Figura 1.2 Comunicación SipML5 y Asterisk como servidor VoIP.

En la Figura 1.2 se puede observar la comunicación de un cliente SIPML5 con Asterisk. El cliente escrito en JavaScript implementa la señalización (SIP) y descripción de sesión (SDP), mediante el protocolo Websocket se comunica con Asterisk. La transmisión del audio y video se lo realiza utilizando WebRTC. Debido a la flexibilidad de Asterisk es posible comunicar a un cliente SipML5 con cualquier otro sistema VoIP. De esta forma por ejemplo podemos realizar una llamada desde un navegador web hacia cualquier número disponible en la PSTN.

1.4.7 Websocket

El protocolo WebSocket permite la comunicación bidireccional y full dúplex utilizando únicamente un socket TCP. El objetivo de esta tecnología es proporcionar un mecanismo para aplicaciones basadas en navegador que necesitan comunicación bidireccional con servidores que no depende de la apertura de múltiples conexiones HTTP. WebSocket está estandarizado por la IETF como el RFC 6455 [5].

WebSocket es un protocolo independiente basado en TCP. La única relación con HTTP es que su protocolo de enlace es interpretado por Servidores HTTP como una solicitud de actualización.

1.4.8 STUN

STUN o Session Traversal Utilities for NAT por sus siglas en inglés, es un protocolo del tipo cliente servidor que permite que las aplicaciones descubran la presencia y los tipos de NAT y cortafuegos entre ellos y la Internet pública.

También proporciona la capacidad para que las aplicaciones determinen las direcciones IP públicas que el NAT les asigna. STUN funciona con muchos NAT existentes y no requiere ningún comportamiento especial de ellos, está estandarizado bajo el RFC 5389 [6]. La conexión con un servidor STUN se realiza generalmente a través del puerto 3478 UDP.

1.4.9 PHP

PHP o Hypertext Preprocessor es un lenguaje de programación que se utiliza habitualmente en el desarrollo web como lenguaje de programación del lado del servidor. PHP es un lenguaje interpretado por lo que para ser procesado necesita un intérprete, generalmente este interprete es implementado como un módulo o un daemon.

La sintaxis de PHP se fundamenta en los principios de programación de C. El intérprete de PHP solo ejecuta código que se encuentra entre sus delimitadores. Estos delimitadores son <?php para abrir una sección de código y ?> para cerrarla. De esta forma se pude separar el código PHP del resto de código, generalmente HTML.

1.4.10 HTML

HTML o lenguaje de Etiquetas de Hipertexto es el empleado para la creación y estructura de una página web. Utiliza etiquetas para describir los elementos que conforman una página web. Dentro de estos elementos se puede identificar etiquetas especializadas para algún tipo de dato en específico como las etiquetas <img>.

1.4.11 CSS

CSS (Cascading Style Sheets) es un lenguaje de hojas de estilo, es decir, te permite aplicar estilos de manera selectiva a un documento estructurado que por lo general es un HTML. Gracias a CSS se puede crear páginas webs visualmente atractivas y que facilita la presentación de la información en los documentos HTML.

1.4.12 JavaScript

JavaScript es un lenguaje de programación interpretado que se utiliza mayormente del lado del cliente. Los navegadores web actuales cuentan con un intérprete JavaScript que permite implementar mayor dinamismo y funcionalidad a una página web.

JavaScript se define como un lenguaje orientado a objetos, débilmente tipado y dinámico. Su sintaxis se diseñó tomando en cuenta el lenguaje C y adoptando nombres y convenciones del lenguaje Java. Sin embargo, solo comparten el nombre ya que cada uno tiene propósitos diferentes.

1.4.13 Asterisk

Asterisk es un proyecto de código abierto que permite implementar un sistema potente de comunicaciones en tiempo real. El Core de Asterisk esta desarrollado en lenguaje C y puede ser desplegado en cualquier sistema Operativo Linux o basado en Unix.

1.4.13.1 Arquitectura de Asterisk

Asterisk es un programa con muchos componentes que se relacionan entre sí para brindar funcionalidad [7]. Los componentes de Asterisk son representados por módulos. Todos estos módulos son abstraídos por medio de archivos de configuración con extensión *. conf. El componente de Asterisk que permite gestionar y orquestar el

funcionamiento de estos módulos es conocido como Core. El Core de Asterisk es el núcleo del sistema y se encarga entre muchas otras funciones de leer los archivos de configuración, cargar los módulos e interactuar con el plan de marcado.

1.4.13.2 Plan de marcado

El plan de marcado o diaplan es el método para dirigir el comportamiento de Asterisk. Se lo puede definir como un conjunto de reglas que indicarán a Asterisk que hacer con los números que se marquen. La característica principal del diaplan es que es una serie de instrucciones escritas en un lenguaje tipo scripting. Para la programación del plan de marcado se debe editar el archivo extension.conf. De forma general se debe tener en cuenta tres conceptos claves para poder programar el dialplan, Estos conceptos son: contexto, extensión, prioridad y aplicación.

Un contexto es un bloque de código que sirve como entrada hacia el dialplan. Dentro de un contexto se puede declarar extensiones. Para cada extensión se puede realizar una o más acciones, estas acciones serán determinadas por la aplicación. Para determinar el orden de ejecución de estas acciones se utiliza la prioridad.

1.4.13.3 ARA

ARA (Asterisk real time architecture) es un conjunto de drivers y funciones que permite la comunicación con otras fuentes de datos, esto para mantener independiente el almacenamiento de diferentes objetos [8]. El principal beneficio de ARA es el soporte de bases de datos. Para la comunicación con nuestra base de datos utilizaremos ODBC además de ARA que nos permite cargar y actualizar objetos. Los objetos que necesitamos cargar son llamados Realtime SIP Peers que corresponden a los peers de cada usuario. El archivo que necesitamos modificar para trabajar con ARA es llamado extconfig.conf.

1.4.13.4 AMI

Asterisk Manager Interface es una interfaz de administración del tipo cliente/servidor sobre TCP [9]. Con la interfaz de administrador, se podrá escuchar y generar eventos relacionados con: originar llamadas, monitorear canales, así como ejecutar comandos de Asterisk. Para configurar el módulo AMI debemos modificar el archivo manager.conf.

1.4.13.5 Servidor web Asterisk

El Core de Asterisk provee un pequeño servicio web. Ciertos módulos de Asterisk pueden hacer uso de este servicio, como Asterisk Manager Interface sobre HTTP y módulos que soporten Websockets [10]. Para poder configurar este servicio debemos modificar el archivo http.conf.

1.4.14 AWS

Amazon Web Services o AWS es un conjunto de servicios de computación en la nube publica ofrecida a través de Internet por Amazon.com. Es considerada como una de las más populares opciones dentro del Cloud Computing.

1.4.14.1 Arquitectura

AWS está desplegada a través de 25 Regiones geográficas y con el fin de ofrecer disponibilidad proporciona zonas dentro de dichas regiones. Una zona de disponibilidad son centros de datos redundantes y aislados dentro de una región de AWS [11]. Las zonas de disponibilidad permiten que los clientes operen bases de datos y aplicaciones de producción con un nivel de disponibilidad, tolerancia a errores y escalabilidad mayor que el que ofrecería un centro de datos único.

AWS ofrece una variedad muy grande de servicios tanto de almacenamiento como para la puesta en marcha de varias aplicaciones. Nos centraremos en el servicio EC2 que AWS ofrece para la creación de máquinas virtuales dentro de una nube privada virtual.

1.4.14.2 EC2

Amazon Elastic Compute Cloud o Amazon EC2 permite a los usuarios alquilar máquinas virtuales en las cuales se puede correr sus aplicaciones o servicios propios. Este servicio proporciona capacidad de cómputo y almacenamiento, pagando solo por la capacidad utilizada. La capacidad en EC2 se alquila por horas.

Las máquinas virtuales desplegadas en EC2 son llamadas instancias y el usuario puede crear, lanzar o terminar instancias tanto como necesite pagando por hora por servidor activo. Dichas instancias pueden ser desplegadas en distintas zonas de disponibilidad ofreciendo un nivel alto a tolerancia a errores.

1.4.15 IDS

Un sistema de detección de intrusos o IDS (Intrusion Detection System) es una herramienta que se utiliza dentro de la seguridad de la información que tiene como finalidad detectar accesos no autorizados a servicios o recursos en una red.

1.4.15.1 Funcionamiento

El funcionamiento se basa en el análisis del tráfico de una red o de un nodo en particular. El tráfico entrante es comparado con patrones o firmas de ataques conocidos. Estas firmas suelen disponerse desde una base de datos y representan los ataques conocidos y más comunes. Normalmente un IDS se integra con un firewall ya que de esta forma se puede crear reglas para bloquear a las amenazas atacantes.

1.4.15.2 Sistemas pasivos y sistemas reactivos

Una manera de clasificar a los IDS se basa en la capacidad de reacción que el sistema tendrá frente a un ataque.

En un sistema pasivo, al detectarse un posible ataque el IDS documenta el suceso y envía una alerta. Por otro lado, un sistema reactivo es capaz de responder a la actividad sospechosa reprogramando las reglas del firewall para que de esta forma se bloquee el tráfico proveniente de la red del atacante [12]. Estos tipos de IDS son muy utilizados para prevenir ataques de DDOS en servicios que por su naturaleza están expuestos a redes no seguras.

2 METODOLOGÍA

2.1 Definición de Requerimientos

En este apartado, se define la situación actual del sistema de comunicación que maneja el emprendimiento Biosphere. Posteriormente se detallas los requerimientos funcionales y no funcionales en base a las entrevistas realizadas y procesadas.

Biosphere nace en 2020 con el objetivo de poner a disposición de las empresas un servicio de asesoría, consultoría y auditoría ambiental [13].

2.1.1 Entrevistas

Se aplicó una entrevista dirigida a tres miembros de Biosphere. La entrevista formulada consta de seis preguntas, el modelo se presenta en el ANEXO A. A continuación, en la tabla se presenta el resumen de las encuestas realizadas.

Tabla 2.1 Resumen de entrevistas

N.º	Pregunta	Respuesta	
		SI	NO
1	¿Dispone de un sistema especializado para realizar llamadas y videollamadas entre los miembros de la organización?	0%	100%
2	¿Dispone de algún sistema que le permita tener registro de las llamadas que se realizan por motivos no ajenos a las actividades de la organización?	0%	100%
3	¿Dispone de algún sistema que le permita grabar el audio de las llamadas y videollamadas?	0%	100%
4	¿Dispone de algún sistema que le permite escuchar y descargar las grabaciones de llamadas?	0%	100%
5	¿Dispone de algún sistema que le permita visualizar las llamadas que se están realizando?	0%	100%
6	¿Dispone de un sistema que le permita administrar a los usuarios?	0%	100%

La primera pregunta permite conocer si en la organización se cuenta con algún sistema para realizar la comunicación de audio y video entre sus miembros. La segunda pregunta busca conocer si en la organización se cuenta con algún sistema donde se registren las llamadas realizadas entre sus miembros. La tercera pregunta dispone conocer si la organización cuenta con algún sistema que le permita grabar el audio de las comunicaciones.

La cuarta pregunta busca conocer si la organización cuenta con algún sistema donde se pueda escuchar y descargar las grabaciones de llamadas. La quinta pregunta busca conocer si la organización cuenta con algún sistema que permita visualizar las llamadas que se estén realizando. La sexta pregunta busca conocer si en la organización se cuenta con algún sistema donde se puedan administrar los usuarios.

2.1.2 Análisis de requerimientos

Con la información obtenida en las entrevistas realizadas podemos establecer de forma general los requerimientos. Se necesita contar con un aplicativo que:

- Permita realizar al menos una llamada o videollamada simultánea
- Permita realizar al menos una llamada simultánea a un número accesible desde la PSTN.
- Permita tener un registro de las llamadas.
- Permita visualizar las llamadas en tiempo real.
- Permita tener administración de los usuarios.

2.1.2.1 Requerimientos funcionales

Para el planteamiento de los requerimientos funcionales se ha tomado en cuenta los requisitos mínimos a cumplir.

- Se tendrán dos tipos de usuarios: administrador, agente.
- El administrador se encargará de la creación, modificación, eliminación de agentes y la entrega de credenciales de acceso a los agentes.
- El administrador podrá visualizar las llamadas que estén realizando los agentes.
- El administrador podrá ver y descargar los registros de las llamadas de los agentes.
- El administrador podrá descargar las grabaciones de las llamadas.
- El administrador podrá ejercer también como agente.
- El agente podrá realizar llamadas y videollamadas con otros agentes.
- El agente podrá realizar llamadas hacia la PSTN.

El requerimiento que menciona las llamadas y videollamadas es imprescindible y es uno de los más importantes, ya que la resolución a todos los demás requerimientos gira en torno a la resolución de éste.

2.1.2.2 Requerimientos no funcionales

- Se usa un modelo cliente servidor para el prototipo de aplicación web.
- Los datos se almacenarán en una base de datos PostgreSQL.
- Las contraseñas serán cifradas con md5 antes de ser almacenadas en la base de datos.
- Es necesario que los usuarios cuenten con acceso a Internet.

Es imprescindible que se cumpla el requisito no funcional que menciona el acceso a Internet ya que la solución se plantea publicada en el Internet.

2.2 Diseño

En esta sección, se detalla el diseño del prototipo de aplicación web basado en WebRTC y VOIP.

La Capa de Datos se describe mediante un diagrama relacional. La Capa de Lógica de Negocios incluye el diagrama de casos de uso y diagrama de clases. En la Capa de Presentación se muestran los bosquejos de las pantallas que tendrá el prototipo.

2.2.1 Diseño de la capa de datos

En esta sección, al usar PostgreSQL, se realiza el diseño del diagrama entidad relación de la base de datos que se va a utilizar teniendo en cuenta los requerimientos obtenidos previamente.

2.2.2 Diagrama entidad relación

Un diagrama entidad relación permite ilustrar como las entidades, que pueden ser personas, objetos u conceptos se relacionan entre si dentro de un sistema. En la siguiente figura se aprecia el diagrama entidad relación para la base SQL que permitirá el almacenamiento de los datos para el prototipo de aplicación web.

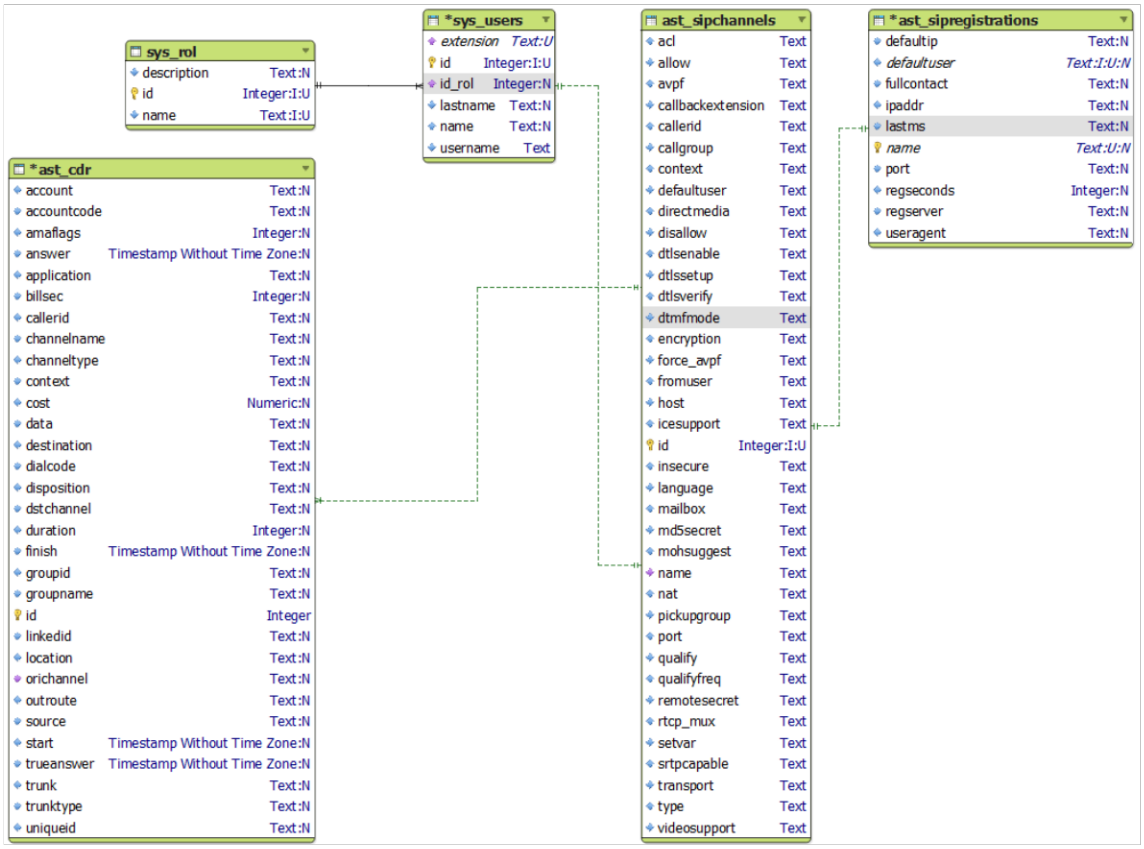


Figura 2.1 Diagrama entidad relación del prototipo

En el diagrama se puede apreciar todas las tablas que definen al prototipo, así como la cardinalidad que existen entre ellas. Las tablas *sys_user* y *sys_rol* representan a los usuarios y su extensión, así como el rol que tienen. La cardinalidad entre estas dos tablas es de uno a uno ya que un usuario solo podrá tener un rol a la vez.

Las tablas *ast_sipchannels* y *ast_sipregistration* relacionan a los usuarios con los endpoints que se crearan para el funcionamiento tanto de llamadas y videollamadas. La cardinalidad que existe entre la tabla *ast_sipchannels* y *sys_users* es de uno a muchos ya que un usuario tendrá dos endpoints. La tabla *ast_cdr* servirá para almacenar los registros de llamadas realizados por los usuarios.

2.2.3 Diseño de la capa lógica de negocios

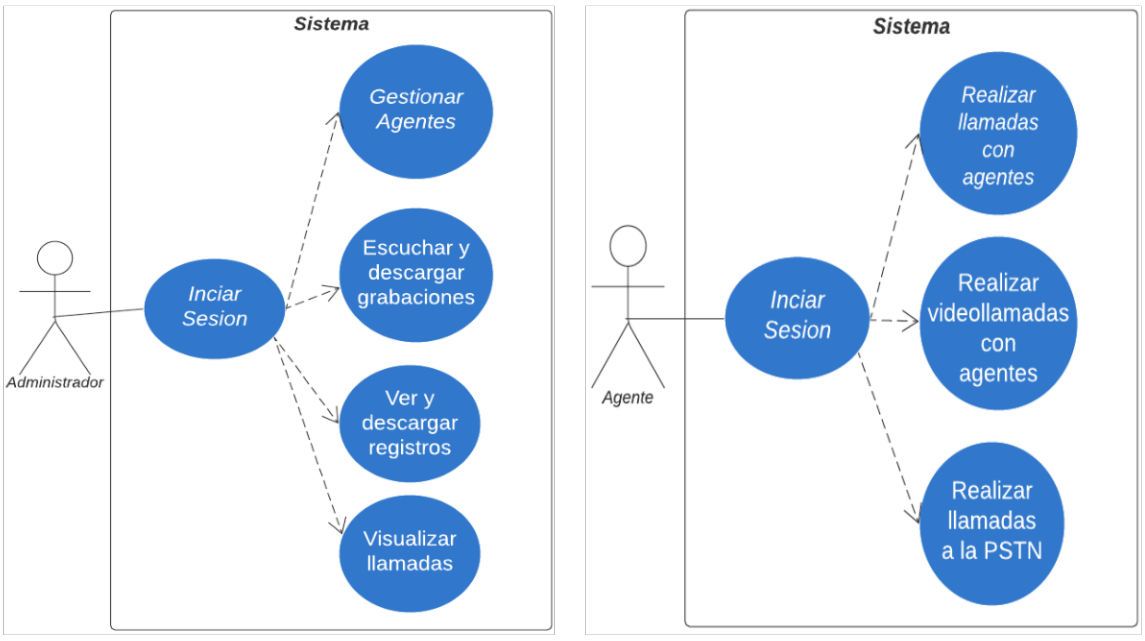
En esta capa, se diseña toda la lógica necesaria para el desarrollo del sistema, para que el usuario a través de una interfaz gráfica pueda acceder a las funcionalidades que presenta la aplicación web. Para esto se utilizará los diagramas de Casos de Uso y el diagrama de clases.

2.2.4 Diagramas de caso de uso

En un diagrama de casos de uso se representa las funcionalidades del sistema desde la perspectiva de un usuario (actor en UML). En el prototipo propuesto se definen dos tipos de actores: Agente y Administrador. A continuación, se detalla el rol de cada uno de los actores:

Administrador: este actor hace referencia a un usuario al que se le ha otorgado el rol de Administrador. Un administrador podrá crear, modificar o eliminar agentes. Podrá visualizar los registros de llamadas, así como escuchar y descargar las grabaciones de estas. Podrá visualizar las llamadas que se estén realizando en ese momento. El administrador también podrá ejercer el rol de usuario

Agente: este actor hace referencia a un usuario al que se le ha otorgado el rol de Agente. Un Agente podrá realizar llamadas o videollamadas con otros agentes. También podrá realizar llamadas externas hacia la PSTN.



a) Diagrama caso de uso administrador b) Diagrama caso de uso agente

Figura 2.2 Diagramas de caso de uso

2.2.5 Diagrama de clases

Un diagrama de clases permite representar de manera clara los elementos y componentes de un sistema. Los diagramas de clase son muy utilizados en paradigmas de programación orientado a objetos y definen las clases como un conjunto de atributos y funciones que las describe.

En la Figura 2.3 se observa la clase Database que es la más importante dentro del diseño del prototipo, ya que nos permite realizar la conexión con la base de datos e implementa todas las funciones necesarias para la persistencia de información. En esta clase se definen todas las funciones para crear, editar y eliminar usuarios, así como para la consulta de registros de llamadas, grabaciones, usuarios y el registro de ingreso al sistema.

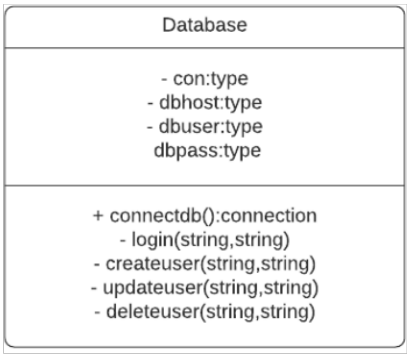


Figura 2.3 Diagrama de clase Database

2.2.6 Diseño de la capa presentación

La capa presentación también conocida como interfaz gráfica es la encargada de interactuar con el usuario final recopilando información y eventos para ser procesados por las capas inferiores y obtener un resultado. Para el diseño de esta capa se propuso la realización de bosquejos que representa las principales pantallas o ventanas que se le mostraran al usuario final para la interacción con el aplicativo. Se utilizará NinjaMock [14], una herramienta en línea, para este propósito. En la Figura 2.4 se observa el bosquejo de la pantalla que permite a un usuario realizar una llamada o videollamada con otro usuario del sistema.

Figura 2.4 Bosquejo de pantalla control de llamadas.

Se propone que cada usuario pueda visualizar el directorio con el nombre y extensión de los usuarios con los que puede comunicarse. En la parte derecha se tendrán botones que permitan contestar, colgar o llamar, así como una ventana para el manejo de las videollamadas.

Los bosquejos restantes se pueden observar en el ANEXO B.

3 IMPLEMENTACION

En esta fase del proyecto, se prepara el ambiente de desarrollo necesario, es decir, instalando las herramientas que se ajusten a la necesidad del proyecto. En el proceso de codificación se inicia con la Capa de Datos, seguido de la Capa de Negocio y finalmente la Capa Presentación, correspondiente al diseño realizado.

3.1 Instalación de las herramientas necesarias

En este apartado se describe la instalación de las herramientas y complementos necesarios para el desarrollo del prototipo de aplicación web.

3.1.1 Creación de instancia EC2 en AWS

Para el hosting del prototipo web se utilizará una instancia de EC2 en AWS. Para poder acceder a la consola de Amazon debemos crear primeramente una cuenta e ingresar con las credenciales. Una vez dentro podemos lanzar una instancia en EC2 utilizando el asistente de creación de instancias. La guía de instalación se adjunta en el Anexo C.

En la Figura 3.1 se observa el dashboard principal de administración de instancias.

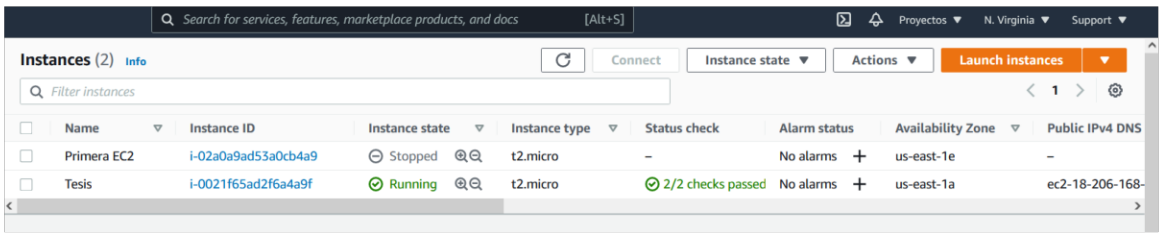


Figura 3.1 Dashboard de instancias EC2 de AWS.

En la parte superior derecha se visualiza la zona de disponibilidad en donde se encuentra desplegadas las instancias, en este caso Norte de Virginia. Entre la información más relevante se tiene el id, estado y tipo de instancia, así como la IP pública que AWS asigna automáticamente.

3.1.2 Instalación del servidor web

Para la instalación del servicio web utilizamos el gestor de paquetes de CentOS yum. En el Código 3.1 se observar la instalación tanto del servicio como de la dependencia *mod_ssl* para el manejo de certificados digitales.

```
1 yum -y install httpd mod_ssl
```

Código 3.1 Instalación apache y dependencias.

Una vez tengamos el servicio web, se necesita instalar el intérprete de php y todos los módulos necesarios para poder utilizarlo como lenguaje de programación del lado del servidor. En el código se detalla la instalación, primeramente, se instala el repositorio necesario (línea 1) y se habilita (línea 2). El repositorio habilitado corresponde a php74. (PHP versión 7.4)

Finalmente instalamos el intérprete y los módulos o también llamadas extensiones (línea 3). Los módulos necesarios son *php-pgsql* que nos brinda una librería con funciones para la gestión de conexión hacia la base de datos, *php-mbstring* que proporciona una librería para manejar la codificación de string y *php-xml*.

```
1 yum install -y http://rpms.remirepo.net/enterprise/remi-release-7.rpm
2 yum-config-manager --enable remi-php74
3 yum -y install php php-mbstring php-pgsql php-xml
```

Código 3.2 instalación de PHP y extensiones.

3.1.3 Instalación de PostgreSQL-11

Continuaremos utilizando yum para le gestión de paquetes. Instalaremos PostgreSQL 11 como DBMS además de los módulos necesarios para la integración de la base de datos.

```
1 yum -y install https://download.postgresql.org/pub/repos/yum/reporpms/
2 EL-7-x86_64/pgdg-redhat-repo-latest.noarch.rpm
3 yum -y install postgresql11-server postgresql11-odbc
4 yum -y install postgresql11-contrib
5 yum -y install postgresql11-devel postgresql11-plperl
```

Código 3.3 instalación de PostgreSQL y extensiones.

Una vez instalado todos los paquetes se procede a inicializar la base de datos (postgres) ejecutando el comando mostrado en la figura

```
1 /usr/pgsql-11/bin/postgresql-11setup initdb
```

Código 3.4 Inicialización de la BD de PostgreSQL

3.1.4 Instalación de Asterisk

La instalación la podemos realizar descargando el código fuente desde la página oficial de Asterisk. El código fuente se ha descargado en el directorio */usr/local/src* tomando la recomendación de la página oficial de Asterisk.

```
1 wget https://downloads.asterisk.org/pub/telephony/asterisk/asterisk-16-current.tar.gz
```

Código 3.5 Descarga de código fuente de Asterisk.

Antes de iniciar con la compilación del código fuente es necesario instalar las librerías y dependencias necesarias. Para esto podemos hacer uso de un script de verificación que está incluido junto al código fuente y otros addons. Una vez tengamos las dependencias instaladas debemos ejecutar el comando de la figura.

```
1 ./configure --with-jansson-bundled
```

Código 3.6 Verificación de dependencias.

Una vez terminada la ejecución del comando debemos ingresar al menú de selección de módulos, aquí podemos elegir los múdalos, códecs, aplicaciones, etc que serán instaladas dentro del Core de Asterisk. Para empezar a compilar ejecutamos los comandos que se muestran en el Código 3.7, las líneas 1 y 2 compilan el código fuente y las líneas 3 y 4 preparan el servicio de Asterisk dentro del gestor de servicios de CentOS

```
1 make
2 make install
3 make config
4 ldconfig
```

Código 3.7 instalación de Asterisk.

Una vez compilado e instalado podemos ejecutar el comando *make samples* para crear archivos de configuración de ejemplo que nos servirán de guía para la configuración de los diferentes módulos de Asterisk. Los archivos de configuración se crean en el directorio */etc/asterisk*. Para iniciar Asterisk ejecutamos el comando descrito en el Código 3.8.

```
1 service asterisk start
```

Código 3.8 Arranque de Asterisk.

3.1.5 Instalación de Fail2ban

Utilizaremos Fail2ban, una herramienta de código abierto escrita en Python [15], para configurarla como IDS. Para instalar fail2ban utilizaremos el gestor yum ejecutando el comando que se muestre en el Código 3.9.

```
1 yum -y install fail2ban
```

Código 3.9 Instalación de fail2ban.

3.2 Implementación de la capa de datos

En esta sección, se presenta la parte más importante de la codificación de la Capa de Datos. Tanto en esta, como en todas las capas de programación que componen el prototipo, se utiliza en su mayoría el idioma inglés. Esto con el fin de cumplir buenas prácticas de programación que invitan a evitar el uso de acentos, caracteres especiales y palabras reservadas.

En el Código 3.10 se muestra la creación de la base de datos llamada *develop* donde se almacenará la información del sistema.

```
1 create database develop
```

Código 3.10 Creación de BD.

El Código 3.11 corresponde a la creación de la tabla "*sys_users*". En la línea 1 se establece el nombre de la tabla y el esquema. Entre la línea 2 y 7 se crean las columnas y los tipos de datos de cada una, así como las propiedades para la clave primaria y Not null para columnas que exigen el ingreso de datos para la creación de un registro.

```
1 CREATE TABLE public.sys_users (
2     id integer NOT NULL,
3     name text,
4     lastname text,
5     username text NOT NULL,
6     extension text NOT NULL,
7     id_rol integer
8 );
```

Código 3.11 Creación de la tabla sys_users.

El Código 3.12 corresponde a la creación de la función "*delete_account*". En la línea 1 se establece el nombre de la función el tipo de dato que retornará la función al ejecutarse con éxito y el tipo de dato que aceptará como parámetro de entrada.

En la línea 2 se declara que el lenguaje a utilizarse será SQL Procedural Language (plpgsql). Entre la línea 4 y 6 se declara las variables a utilizarse. La línea 5 es una variable que guarda el valor del parámetro de entrada de la función.

Las líneas 8 a 30 corresponden al cuerpo de la función. Aquí se ejecuta todas las acciones necesarias para eliminar un usuario. En la línea 31 se retorna el tipo de valor especificado en el prototipo de la función.

```
1  CREATE FUNCTION public.delete_account(text) RETURNS integer
2      LANGUAGE plpgsql
3      AS $_$
4  DECLARE
5          id_exten ALIAS FOR $1;
6          integer_var integer;
7
8  BEGIN
9          -- Delete user
10 delete from sys_users where extension=id_exten;
11 GET DIAGNOSTICS integer_var = ROW_COUNT;
12
13         --Delete Websocket client
14 delete from ast_sipchannels where name=concat('+',id_exten);
15 GET DIAGNOSTICS integer_var = ROW_COUNT;
16
17         --Delete SIP client
18 delete from ast_sipchannels where name= id_exten;
19 GET DIAGNOSTICS integer_var = ROW_COUNT;
20
21         --Delete registration WS
22 delete from ast_sipregistrations where name=concat('+',id_exten);
23 GET DIAGNOSTICS integer_var = ROW_COUNT;
24
25         --Delete registration SIP
26 delete from ast_sipregistrations where name= id_exten;
27 GET DIAGNOSTICS integer_var = ROW_COUNT;
28
29         return integer_var;
30 END;
31 $_$;
```

Código 3.12 Creación de la función delete_account.

El código restante que abarca la creación de todas las tablas, vistas y funciones que componen la capa de datos se encuentran en el ANEXO D.

3.3 Implementación de la capa de lógica de negocios

En este apartado, se presenta la implementación de las clases, el servicio web, el servicio SIP y el servicio IDS.

3.3.1 Codificación de clases

Para la implementación de las clases que se comunicaran por un lado con la capa de datos y por otro lado con la capa de presentación se utilizará php.

El Código 3.13 muestra parte del archivo *database.php*. En la línea 1 se declara la clase Database que se encargara de conectarse con la base de datos y realizar todas las acciones necesarias que se requieran desde la Capa Presentación.

Entre las líneas 3 y 7 se declaran los atributos de la clase. Estos atributos permiten la conexión con la base de datos. Las líneas de la 9 a la 11 corresponde a la declaración del constructor de la clase. Cada vez que se instancie un objeto de la clase Database se ejecutara la función *connect_db()*.

```
1 class Database{
2
3     private $con;
4     private $dbhost="localhost";
5     private $dbuser="postgres";
6     private $dbpass="paradise";
7     private $dbname="develop";
8
9     function __construct(){
10         $this->connect_db();
11     }
12
13     public function connect_db(){
14         $dns= "host=" . $this->dbhost . " dbname=" . $this->dbname ."
15 user=" . $this->dbuser . " password=" .$this->dbpass;
16         $this->con = pg_connect($dns) or die ("Connection Error:".
17 pg_last_error());
18     }
19 }
```

Código 3.13 Creación de la clase Database.

El Código 3.14 muestra una función que permite crear un nuevo usuario en la base de datos, en la línea 1 se declara el nombre de la función y los parámetros de entrada que serán comandos de la Capa Presentación. En la línea 11 se crea un string con la consulta que se realizara a la base de datos. En esta consulta se hace uso de la función *create_account* descrita en la Capa de Datos.

```
1  public function createnew($name,$lname,$username,$pass,$exten,$codec){
2       $sql = "Select username,extension from sys_accounts_v where
3          username = '$username' or extension = '$exten'";
4       $res = pg_query($this->con, $sql);
5       $flag = pg_num_rows($res);
6          if(0==$flag){
7            $callerid=$name . ' ' . $lname . ' <'. $exten . '>';
8            $caracteresNoPermitido = array("<","/",">","ò","Ç","Ø","ø",
9              "Å","å","Æ","æ","ß","¤","¡","ä","ö","ü","à","Ü","Ö","Ä");
10           $namenew=str_replace($caracteresNoPermitido,'',$name);
11           sql = "select create_account('$namenew','$lname','$username',
12                 '$pass','$exten','$callerid','$codec')";
13           $res = pg_query($this->con, $sql);
14           AMI::connect_ami();
15           return true;
16      }
17      else{
18           return false;
19      }
20 }
```

Código 3.14 Declaración de función createnew.

El código restante que abarca todas las funciones necesarias para la comunicación con la capa de datos se encuentra en el ANEXO E.

3.3.2 Codificación del servicio web

3.3.2.1 Obtención del dominio

Para la obtención del dominio se utilizará Freenom. Freenom es un proveedor de dominios gratuitos y de pago. Su misión es facilitar que la gente pueda acceder a internet y ayudar a que los países desarrollen su economía digital [16]. Estos dominios gratuitos funcionan exactamente igual que cualquier otro nombre de dominio. Para el desarrollo del prototipo hemos registrado el dominio realtel.ml.

Para poder asociar el nombre del dominio con la IP pública de la instancia EC2 se debe ingresar en Manage Domain. Una vez dentro creamos los registros necesarios.

3.3.2.2 Instalación de certificado SSL

Para la obtención del certificado digital se utilizará Lets Encript. Lets Encript es una CA gratuita, automatizada y abierta [17]. Para la instalación del certificado se usará el cliente ACME Certbot. Se utilizará el gestor de paquetes yum para instalarlo. En el Código 3.15 se muestra el comando.

```
1 yum -y install cerbot
```

Código 3.15 instalación del cliente Cerbot.

Para descargar el certificado debemos asegurarnos de que el servicio web se encuentre detenido (Código 3.16, línea 1). Para instalar los archivos correspondientes al certificado digital se ejecuta el comando descrito en el Código 3.16, línea 2. Todos los archivos se descargarán en el directorio */etc/letsencrypt/realtel.ml/*

```
1 service httpd stop
2 cerbot certonly --standalone -d realtel.ml
```

Código 3.16 instalación del certificado digital.

3.3.2.3 Configuración del servicio Web

Para poder configurar el servicio web procedemos a modificar el archivo *httpd.conf* el cual se encuentra alojado en el directorio */etc/httpd/conf/*. En el Código 3.17 se muestra parte del contenido del archivo *httpd.conf*.

En la línea 1 se declara un Virtual Host que escucha en todas las interfaces del servidor en el puerto 443. Entre las líneas 2 a la 6 se declara el directorio hacia donde se dirigirán todas las peticiones HTTPS. También se otorgan los permisos de acceso. En la línea 7 se declara el Server Name que en este caso es el dominio realtel.ml. En la línea 8 se declara el *DocumentRoot* que es el directorio raíz desde donde se buscaran todos los recursos solicitados por las peticiones. Generalmente coincide con el declarado en las etiquetas *Directory*. En la línea 9 activamos el módulo para trabajar con certificados digitales y finalmente en las líneas 10 y 11 se indica los directorios de los archivos correspondiente a los certificados.

Para forzar que el servidor web trabaje de manera segura creamos otro Virtual Host que escuche en el puerto 80 y realice una redirección hacia nuestro servicio seguro. Este código se describe entre las líneas 14 a 17.

```
1  <VirtualHost *:443>
2    <Directory "/var/www/html/develop">
3      Options Indexes FollowSymLinks
4      AllowOverride All
5      Require all granted
6    </Directory>
7      ServerName realtel.ml
8      DocumentRoot "/var/www/html/develop"
9      SSLEngine on
10     SSLCertificateFile /etc/letsencrypt/live/realtel.ml/fullchain.pem
11     SSLCertificateKeyFile /etc/letsencrypt/live/realtel.ml/privkey.pem
12 </VirtualHost>
13
14 <VirtualHost *:80>
15     ServerName realtel.ml
16     Redirect / https://realtel.ml
17 </VirtualHost>
```

Código 3.17 Configuración del servicio web.

Una vez modificado el archivo debemos reiniciar el servicio web.

3.3.3 Configuración de Asterisk

Asterisk maneja varios archivos de configuración para controlar el funcionamiento de cada uno de sus módulos. Estos archivos se encuentran en el directorio /etc/asterisk/.

El archivo *sip.conf* nos permite configurar todo lo referente con el protocolo SIP además que nos facilita crear endpoints SIP que pueden ser asociados a un usuario y además nos permite conectarnos con proveedores SIP para la comunicación con la PSTN.

En el Código 3.18 se observa una parte de la sección general que contiene la configuración por defecto para todos los usuarios. Se puede sobrescribir los valores por defecto en las configuraciones de cada usuario o peer. En las líneas de 2 a 6 estamos habilitando el servicio para los protocolos UDP, TCP y TLS cada uno en su puerto por defecto y para todas las interfaces del servidor. Los puertos de escucha por defecto para el servicio SIP es el 5060 UDP/TCP y 5061 TCP para TLS.

Las líneas 13 y 14 donde se configura tanto la IP externa o IP pública del servidor como la red local son muy importantes, ya que el servicio estará expuesto al Internet. El no

configurar estos parámetros ocasiona inconvenientes en la transmisión de los flujos de audio y video.

```
[general]
udpbindaddr=0.0.0.0
tcpenable=yes
tcpbindaddr=0.0.0.0
tlsenable=yes
tlsbindaddr=0.0.0.0
realm=develop
callerid=Develop <Develop>
websocket_enabled=true
disallow=all
allow=alaw,ulaw,opus
directmedia=no
externaddr=18.206.168.146
localnet=172.31.31.188/20

```

Código 3.18 Configuración del módulo SIP.

En el Código 3.19 podemos observar la declaración de un peer del tipo *friend*. Este tipo de peer permite tanto realizar como recibir llamadas. El peer home se utilizará para realizar llamadas externas hacia la PSTN. Para este registro son necesarios principalmente los parámetros *secret*, el nombre del usuario y el parámetro *dynamic* en host.

```
[home]
type=friend
context=internal
host=dynamic
secret=paradise
nat=force_rport,comedia
disallow=all
allow=alaw,ulaw,g729,opus
qualify=4000
```

Código 3.19 Creación de un endpoint SIP.

Para la creación de los peers tipo *friend*, que corresponde a los usuarios no se utilizara este archivo directamente. Esto debido a que codificarlos en el archivo *sip.conf* le quitaría flexibilidad al diseño. Para llevar a cabo esta tarea haremos uso de la Arquitectura en Tiempo Real de Asterisk más conocida como ARA por sus siglas en inglés.

En el Código 3.20 se observa los parámetros necesarios para configurar un peer que soporte WebRTC. En la línea 1 se activa la encriptación del streaming de audio y video (SRTP) así como el uso de DTLS (línea 2).

```
1 encryption=yes
2 avpf=yes
3 force_avp=yes
4 icesupport=yes
5 dtlsenable=yes
6 dtlsverify=fingerprint
7 dtlssetup=actpass
8 rtcp_mux=yes
```

Código 3.20 Parámetros para soporte de WebRTC.

Estos parámetros serán guardados para cada peer en la base de datos y leídos a través de ARA.

3.3.4 ARA

Como se mencionó en el marco teórico ARA es la arquitectura en tiempo real de Asterisk y permite cargar ciertos objetos de manera dinámica, a través de un conector hacia una base de datos.

El archivo que necesitamos modificar para trabajar con ARA es llamado *extconfig.conf*. La Figura 3.2 muestra un esquema ejemplo de la utilización de ARA.

```
<family> => <realtime driver>,<res_<driver>.conf class name>[,<table>]
```

Figura 3.2 Esquema de ARA.

En el Código 3.21 podemos observar la configuración para trabajar con Realtime SIP peers. Mediante ODBC se realizarán consultas SQL a las tablas *ast_sipchannels* y *ast_sipregistrations* pertenecientes a la base de datos *develop*.

```
1 [settings]
2 sippeers => odbc,develop,ast_sipchannels
3 sipregs => odbc,develop,ast_sipregistrations
```

Código 3.21 Configuración de ARA.

Ahora debemos configurar ODBC ya que lo utilizaremos como driver dentro de ARA. Para esto modificaremos el archivo *res_odbc.conf* Como se puede visualizar en el Código 3.22 de manera general para configurar este archivo se necesita configurar las credenciales

de acceso a la base de datos (líneas 3 a 5). Es importante recordar el DSN ya que se necesitará para la configuración del driver para PostgreSQL.

```
1 [develop]
2 enabled => yes
3 dsn => develop
4 username => postgres
5 password => paradise
6 pre-connect => yes
7 sanitysql => select 1
8 max_connections => 20
```

Código 3.22 Configuración del módulo ODBC.

El siguiente Código 3.23 describe la configuración del archivo odbc.ini (driver odbc) para poder utilizarlo con nuestra base de datos en PostgreSQL. La línea 6 Database debe coincidir con el nombre del DSN configurado en el Código 3.22.

```
1  [develop]
2  Description             = Database Connection
3  Driver                  = PostgreSQL
4  Trace                   = Yes
5  TraceFile               = /var/log/asterisk/odbc
6  Database                = develop
7  Servername              = localhost
8  Port                    = 5432
9  Protocol                = 7.4
10 ReadOnly                = No
```

Código 3.23 Configuración del controlador ODBC.

3.3.5 CDR

CDR o Call Detail Records es usado como un histórico de todas las llamadas realizadas en un sistema, generalmente se guardan en bases de datos, pero también pueden ser almacenados en archivos de texto plano.

Asterisk maneja el almacenamiento de los registros de CDR en su propia base de datos. Sin embargo, podemos aprovechar la configuración del ODBC previamente realizada y por medio de este, guardar los registros en nuestra base de datos personalizada. Utilizaremos el archivo *cdr_adaptative_odbc.conf* para lograr esto. En el Código 3.24 podemos observar la configuración del CDR través de ODBC.

En la línea 1 se declara la conexión ODBC que se utilizará (Es la misma creada en el archivo *res_odbc.conf*), en la línea 3 se hace referencia a la tabla creada en nuestra base

de datos. Entre la línea 5 a la 12 se describen las columnas que debe contener la tabla *ast_cdr* para que los registros se guarden correctamente. La línea 10 describe la columna channel que hace referencia al canal que originó la llamada.

```
1  [cdr]
2  connection=develop
3  table=ast_cdr
4  usegmtime=no
5  alias clid => callerid
6  alias src => source
7  alias dst => destination
8  alias end => finish
9  alias dcontext => context
10 alias channel => orichannel
11 alias lastapp => application
12 alias lastdata => data
```

Código 3.24 Configuración CDR.

3.3.6 Dialplan

El Dialplan o plan de marcado permite programar el enrutamiento de las llamadas dentro del sistema (core de Asterisk) mediante un lenguaje de programación del tipo scripting

En el Código 3.25 se puede visualizar los contextos creados. Entre la línea 3 a 22 se define el contexto *dial-internal* el cual se encarga de realizar la grabación, llamadas y videollamadas entre las extensiones. Para realizar las llamadas hacia los endpoints que soportan WebRTC se utiliza la aplicación Dial (línea 14).

Para la codificación del contexto *internal* además de la extensión creada en la línea 20 que permite marcar hacia números de la PSTN se utiliza el script declarado en la línea 1.

```
#exec '/etc/asterisk/internos.pl'

[dial-internal]
exten => _X.,1,Noop(${CHANNEL})
same => n,ChanIsAvail(SIP/+${EXTEN},a)
same => n,Set(WSAVAILORIGCHAN=${AVAILORIGCHAN})
same => n,NoOp(Free ws channels are: ${WSAVAILORIGCHAN})
same => n,ChanIsAvail(SIP/${EXTEN},a)
same => n,Set(SIPAVAILORIGCHAN=${AVAILORIGCHAN})
same => n,NoOp(Free sip channels are: ${SIPAVAILORIGCHAN})
same => n,MixMonitor(telephony/${EXTEN}/${CDR(uniqueid)}.WAV,b,
        sh /opt/scripts/eje.sh ${CDR(uniqueid)} ${EXTEN})
same => n,GotoIf($["${WSAVAILORIGCHAN}"!=""]?ws:sip)
same => n(ws),Dial(SIP/+${EXTEN},15)
same => n,Hangup
same => n(sip),ExecIf($["${SIPAVAILORIGCHAN}"!=""]?
        Dial(SIP/${EXTEN},15):Hangup(20))

[internal]
exten => _X.,1,Noop(${EXTEN})
same => n,ChanIsAvail(SIP/home,a)
same => n,NoOp(FXO Status: ${AVAILORIGCHAN})
same => n,MixMonitor(telephony/${EXTEN}/${CDR(uniqueid)}.WAV,b,
        sh /opt/scripts/eje.sh ${CDR(uniqueid)} ${EXTEN})
same => n,ExecIf($["${AVAILORIGCHAN}"!=""]?
       Dial(SIP/home/${EXTEN},25):Hangup(20))
```

Código 3.25 Codificación del plan de marcado.

El script internos.pl está escrito en lenguaje perl y permite realizar una consulta a la base de datos hacia la tabla *sys_users*. Nos interesa extraer las extensiones de todos los usuarios para agregarlas al contexto *internal*. El Código 3.26 muestra la codificación del script internos.pl. Como se visualiza en la línea 14 se crea las extensiones y cada una ejecutará la aplicación *Goto* que permite cambiar al contexto *dial-internal*.

```
1  #!/usr/bin/perl
2  use strict;
3  use DBI;
4  my $dbh = DBI->connect("DBI:Pg:dbname=develop;host=localhost;port=5432",
5          "postgres", "paradise", {'RaiseError' => 1});
6  my $ext;
7  my $sth_h = $dbh->prepare("SELECT DISTINCT extension
8            FROM public.sys_users ORDER BY extension");
9  $sth_h->execute();
10 my $ext .= "\n[internal]\n";
11 while (my $ref = $sth_h->fetchrow_hashref()) {
12        my $exten = $ref->{'extension'};
13                  $ext .= "exten =>
14                  $exten,1,Goto(dial-internal,\${EXTEN},1)\n";
15       }
```

Código 3.26 Codificación script internos.pl.

3.3.7 AMI

AMI es una interfaz de administración que permite gestionar canales, módulos y algunas características del core de Asterisk, es del tipo cliente/servidor.

Para configurar AMI debemos modificar el archivo *manager.conf*. En el Código 3.27 entre las líneas 1 a 5 habilitamos la interfaz en el puerto 5038. La línea tres habilita el acceso a través del servidor web de Asterisk, de esta forma podremos escuchar y generar eventos mediante peticiones HTTP. El código de creación de un usuario para conexión se describe entre las líneas 7 a 11. Este usuario tendrá todos los permisos para lectura y escritura.

```
1  [general]
2  enabled = yes
3  webenabled = yes
4  port = 5038
5  bindaddr = 0.0.0.0
6
7  [admin]
8  secret=paradise
9  permit=127.0.0.1
10 read=all
11 write=all
```

Código 3.27 Configuración de AMI.

3.3.8 Servidor web Asterisk

Para poder configurar este servicio debemos modificar el archivo *http.conf* En la línea 2 y 3 del Código 3.28 se habilita el servicio y el soporte de páginas web (HTML, CSS, JavaScript). Entre las líneas 5 a 8 habilitamos el servicio en el puerto 8089 y declaramos el certificado digital a usarse que es el mismo empleado en nuestro servicio web apache. Con esta configuración garantizamos el uso de Websockets seguro y https.

```
1 [general]
2 enabled=yes
3 bindaddr=0.0.0.0
4 bindport=8088
5 enablestatic=yes
6 tlsenable=yes
7 tlsbindaddr=0.0.0.0:8089
8 tlscertfile=/etc/letsencrypt/live/realtel.ml/fullchain.pem
9 tlsprivatekey=/etc/letsencrypt/live/realtel.ml/privkey.pem
```

Código 3.28 Configuración servidor web Asterisk.

El soporte de páginas web es necesario ya que utilizaremos AMI sobre HTTP para realizar el panel de monitoreo de llamadas. Para esto nos basaremos en un ejemplo descrito en la documentación de Asterisk llamado ajamdemo.html. Esto será descrito más a detalle en la Fase de Implementación de la Capa Presentación. El directorio por defecto del servidor web de Asterisk es */var/lib/asterisk/static-http/*.

3.3.9 Configuración del IDS

Como sistema de detección de intrusos se aplicará un sistema activo, de esta manera no solo se analizarán los patrones de ataque, si no, se reaccionará ante éstos de manera que se bloquee el tráfico y no se vea comprometido el servicio.

Para la configuración de nuestro IDS utilizaremos fail2ban. Primeramente, copiaremos el archivo *jail.conf* dentro del directorio /etc/fail2ban/jail.d/ y lo renombraremos como *jail.local*.

En el Código 3.29 se observa parte del archivo *jail.local* que define la configuración por defecto que se aplicará. Dentro de la configuración de los parámetros por defecto es necesario definir el tiempo de bloqueo de las IP, así como el *findtime* que define el temporizador de espera de intentos de ataque. Si el número de ataques definido en

maxretry se ha cumplido dentro del tiempo *findtime* entonces fail2ban actuara de acuerdo con lo descrito en las jaulas configuradas.

```
[DEFAULT]
ignoreip = 127.0.0.1/8 ::1
bantime = 240m
findtime = 10m
maxretry = 5
```

Código 3.29 Configuración de parámetros por defecto.

El Código 3.30 descrito presenta la configuración de una jaula que es como se conoce a un bloque de código que representa a un servicio. Esta parte de código también se encuentra en el archivo *jail.local*

La jaula que se configurará protegerá el servicio SIP, en la línea 3 configuramos los puertos que serán analizados. Las líneas entre la 4 y 6 describen la acción a realizarse una vez se haya detectado el ataque. En este caso bloquemos la IP por medio de *iptables*. La línea 7 especifica el archivo de log que se estará analizando en busca de patrones de ataques ya definidos. Finalmente, en la línea 8 se configura el número máximo de intentos de ataque, una vez se cumpla este número se procederá según lo indicado en las acciones. (action). La línea 9 define la ubicación del archivo de log del servicio que estará analizando fail2ban en busca de patrones de ataque.

```
[asterisk]
enabled = true
port    = 5060,5061,5038
action  = %(banaction)s[name=%(__name__)s-tcp, port="%(port)s",
          protocol="tcp", chain="%(chain)s", actname=%(banaction)s-tcp]
          %(banaction)s[name=%(__name__)s-udp, port="%(port)s",
          protocol="udp", chain="%(chain)s", actname=%(banaction)s-udp]
          %(mta)s-whois[name=%(__name__)s, dest="%(destemail)s"]
logpath = /var/log/asterisk/messages
maxretry= 3
```

Código 3.30 Configuración de la jaula Asterisk.

Fail2ban al actuar como IDS activo analiza patrones de ataques y ejecuta una acción para evitar dichos ataques. Estos patrones de ataque se los puede configurar dentro del directorio */etc/fail2ban/filter.d/*. Existen archivos predeterminados para los servicios más comunes.

El código pertenece al archivo *asterisk.conf*, aquí se definen los patrones más comunes de ataque hacia el servicio SIP, como son, falla en la autenticación por contraseña fallida e intentos de llamadas no autorizadas. Todos estos patrones o regex (expresiones regulares) se crean a partir del análisis del archivo de log de asterisk */var/log/asterisk/messages*. En este caso se utilizará el archivo que viene por defecto con fail2ban ya que contiene una lista de patrones más comunes de ataques.

```
[Definition]
_daemon = asterisk
log_prefix= (?:NOTICE|SECURITY|WARNING)%(__pid_re)
            s:?(?:\[C-[\da-f]*\])?:? [^:]+:\d*
            (?:(?: in)? [^:]+:)?
prefregex = ^%(__prefix_line)s%(log_prefix)s
            <F-CONTENT>.+</F-CONTENT>$
failregex = ^Registration from '[^']*' failed for '
            <HOST>(:\d+)?' - (?:Wrong password
            |Username/auth name mismatch|
            No matching peer found|Not a local domain
            |Device does not match ACL
            |Peer is not supposed to register
            |ACL error \(permit/deny\)|Not a local domain)$
```

Código 3.31 Patrones de ataques.

En la Figura 3.3 se presenta la salida del comando *iptables -nL*. Se puede observar los Chain o cadenas creadas a partir de fail2ban. Dentro de los Chain f2b-asterisk se puede apreciar las IP atacantes bloqueadas con estatus Reject.

```
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
f2b-asterisk-udp  udp  --  0.0.0.0/0            0.0.0.0/0           multiport dports 5060,5061
f2b-asterisk-tcp  tcp  --  0.0.0.0/0            0.0.0.0/0           multiport dports 5060,5061

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination

Chain f2b-asterisk-tcp (1 references)
target     prot opt source               destination
REJECT     all  --  143.244.57.67        0.0.0.0/0           reject-with icmp-port-unreachable
REJECT     all  --  217.212.240.69       0.0.0.0/0           reject-with icmp-port-unreachable
REJECT     all  --  89.163.220.41        0.0.0.0/0           reject-with icmp-port-unreachable
REJECT     all  --  216.245.220.34       0.0.0.0/0           reject-with icmp-port-unreachable
REJECT     all  --  141.98.10.188        0.0.0.0/0           reject-with icmp-port-unreachable
RETURN     all  --  0.0.0.0/0            0.0.0.0/0

Chain f2b-asterisk-udp (1 references)
target     prot opt source               destination
REJECT     all  --  143.244.57.67        0.0.0.0/0           reject-with icmp-port-unreachable
REJECT     all  --  217.212.240.69       0.0.0.0/0           reject-with icmp-port-unreachable
REJECT     all  --  89.163.220.41        0.0.0.0/0           reject-with icmp-port-unreachable
REJECT     all  --  216.245.220.34       0.0.0.0/0           reject-with icmp-port-unreachable
REJECT     all  --  141.98.10.188        0.0.0.0/0           reject-with icmp-port-unreachable
RETURN     all  --  0.0.0.0/0            0.0.0.0/0
```

Figura 3.3 Visualización de IP bloqueadas por el IDS.

3.4 Implementación de la capa de presentación

3.4.1 Login

Al iniciar la aplicación web se muestra una página HTML correspondiente a la página de inicio de sesión. En el Código 3.32 entre las líneas 3 a la 8 se muestra la cabecera donde se importan las hojas de estilos y se define el nombre de la página y otros atributos correspondientes al esquema html.

En el código entre las líneas 9 a 25 se muestra el cuerpo de la página que consta de un form y los inputs (cajas de texto) que corresponden a las cajas de texto para el ingreso de credenciales. Entre las líneas 10 a 12 se ejecuta código PHP que permitirá presentar un mensaje cuando el intento de autenticación de un usuario sea incorrecto.

```
1  <html>
2    <head>
3      <meta charset="utf-8">
4      <title>Login</title>
5      <link href="https://fonts.googleapis.com/css?family=Roboto"
6            rel="stylesheet">
7      <link rel="stylesheet" href="assets/css/style.css">
8    </head>
9    <body>
10     <?php if(!empty($message)): ?>
11       <p> <?= $message ?></p>
12     <?php endif; ?>
13     <h1>Login</h1>
14     <form action="index.php" method="POST">
15       <input name="username" type="text"
16              placeholder="Ingresa tu usuario">
17       <input name="password" type="password"
18              placeholder="Ingresa tu contraseña">
19       <input type="submit" value="Ingresar">
20     </form>
21   </body>
22 </html>
```

Código 3.32 Codificación del archivo index.php.

En la Figura 3.4 se muestra la página principal que se presenta al usuario una vez se ingresa al url (https://realtel.ml) a través de un navegador web.

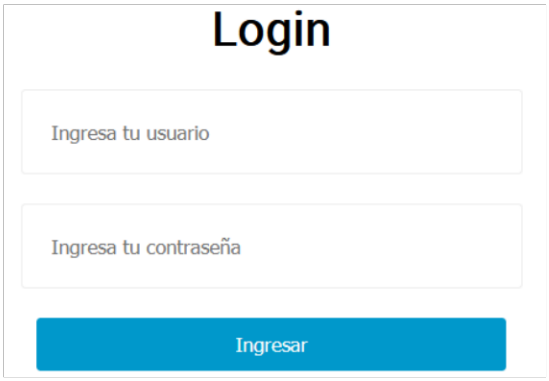


Figura 3.4 Pantalla de registro de ingreso.

3.4.2 Llamadas y videollamadas

En la Figura 3.5 se muestra la pantalla que se presenta a un usuario con rol Agente. La creación de esta pantalla se ha realizado en base al demo open source de Doubango Telecom. En la parte superior izquierda se visualiza el nombre del usuario junto con su número de extensión, en la parte inferior se presenta el directorio. El directorio es una tabla con tres columnas que brindan información tanto del nombre del usuario como el número de extensión y permite realizar directamente una llamada(audio) o videollamada(video). En la parte superior derecha se visualiza una caja de texto donde se puede ingresar un número para realizar una llamada hacia la PSTN.

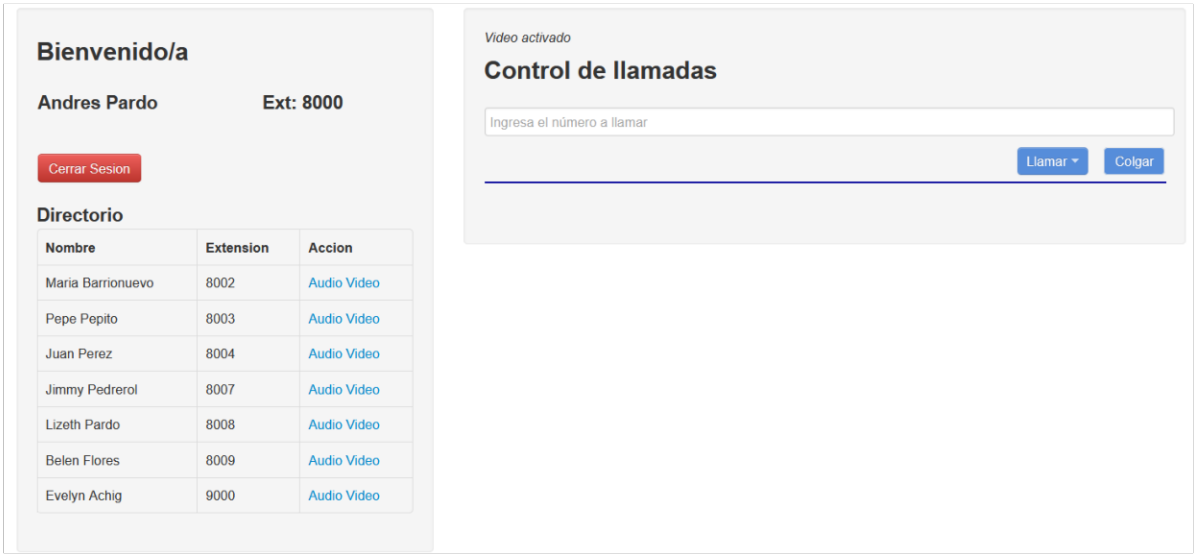


Figura 3.5 Pantalla de control de llamadas.

En el Código 3.33 se muestra parte de la cabecera del archivo *user.php*. En la línea 6 se inserta el API que implementa un cliente SIPml5, gracias a esta API escrita en JavaScript podremos establecer las sesiones necesarias para realizar las llamadas y videollamadas.

```
<title>Usuario</title>
<meta name="viewport" content="width=device-width,initial-scale=1.0"/>
<meta name="Description" content="HTML5 SIP client" />
<meta name="author" content="Develop" />
<script src="js/SIPml-api.js" type="text/javascript"> </script>
<link href="./assets/css/bootstrap.css" rel="stylesheet" />
<style type="text/css">
```

Código 3.33 Cabecera del archivo user.php.

El Código 3.34 muestra parte del cuerpo HTML del archivo *user.php* entre las líneas 4 a 29 se muestra la creación de la tabla que presentara el directorio de extensiones al usuario. Como se observa se utiliza PHP para que la creación de las filas de la tabla para que sea dinámica. Entre las líneas 20 a 25 se crea la columna que contendrá las acciones que se pueden ejecutar para cada miembro del directorio. Para cada fila se crea dos etiquetas <a> que a su vez tienen asociadas al evento onclick la función *sipCall()*. El parámetro de entrada de la función determina si se realizara una llamada (línea 21) o videollamada (línea 23).

```
1 <h3>
2     Directorio
3 </h3>
4 <table class="table table-bordered">
5     <thead>
6         <tr>
7           <th>Nombre</th>
8           <th>Extension</th>
9           <th>Accion</th>
10        </tr>
11            <?php
12              while ($row=pg_fetch_object($phonebook)){
13              $newname=$row->name;
14              $nlname=$row->lastname;
15              $newexten=$row->extension;
16            ?>
17        <tr>
18          <td><?php echo $newname.' '.$nlname;?></td>
19          <td><?php echo $newexten;?></td>
20          <td>
21            <a href="#" onclick='sipCall("call-audio","
22              <?php echo $newexten;?>","true");'>Audio</a>
23            <a href="#" onclick='sipCall("call-audiovideo","
24              <?php echo $newexten;?>","true");'>Video</a>
25          </td>
26        </tr>
```

```
27              <?php } ?>
28      </thead>
29 </table>
```

Código 3.34 Codificación del archivo user.php

En el Código 3.35 podemos observar la función *sipCall*, que es una sobreescritura de la misma función que nos ofrece el API y archivo base de Doubango. En la línea 1 se define el prototipo de la función que acepta tres parámetros de entrada, *s_type* que es el tipo de sesión que se iniciará, (solo audio o audio y video), *s_newvalue* que es la extensión con la que se establecerá la sesión y *s_flag* que es una bandera que nos permite identificar cuando una llamada se realiza desde el directorio telefónico o se realiza desde la caja de texto del control de llamadas.

```
1  function sipCall(s_type,s_newvalue,s_flag) {
2  if(s_flag=='true')
3   {
4    txtPhoneNumber.value = s_newvalue;
5   }
6  if (oSipStack && !oSipSessionCall
7      && !tsk_string_is_null_or_empty(txtPhoneNumber.value)) {
8    btnCall.disabled = true;
9    btnHangUp.disabled = false;
10   oSipSessionCall = oSipStack.newSession(s_type, oConfigCall);
11   var undef = s_newvalue || txtPhoneNumber.value;
12   txtPhoneNumber.value=undef;
13   if (oSipSessionCall.call(txtPhoneNumber.value) != 0) {
14        oSipSessionCall = null;
15        txtCallStatus.value = 'Failed to make call';
16        btnCall.disabled = false;
17        btnHangUp.disabled = true;
18        return;
19         }
20    }
21   else if (oSipSessionCall) {
22        txtCallStatus.innerHTML = '<i>Conectando...</i>';
23        oSipSessionCall.accept(oConfigCall);
24        }
25   else
26        { alert("Por favor ingrese un numero");}
27 }
```

Código 3.35 Codificación de la función sipCall.

Entre las líneas 2 y 5 se observa el condicional que evalúa el parámetro de entrada s_flag, cuando sea true se tratará de una sesión que se establece desde el directorio. En la línea 10 se crea la nueva sesión utilizando la función *newSession* que nos ofrece el API sipML5, como parámetros de entrada se envía la extensión y la variable *oConfigCall* que hace referencia a los id de las etiquetas video y audio del html (*user.php*). En la línea 13 se establece el inicio de sesión para una llamada, como parámetro de entrada se envía la extensión.

3.4.3 Panel de administración

En la Figura 3.6 se presenta la pantalla principal que se mostrará a un usuario con rol Administrador. En la parte central se puede visualizar una tabla con los usuarios creados, la última columna Acciones permite editar o eliminar al usuario. En la parte superior a la tabla se tiene tres botones que permiten agregar un usuario, ver el reporte de llamadas e ingresar a la pantalla *user.php* para poder realizar llamadas y videollamadas. En la parte superior derecha se cuenta con dos botones, uno que permite abrir el panel de llamadas que se estén realizando y el otro para cerrar la sesión. En la parte derecha se presenta una etiqueta con el nombre del usuario administrador.

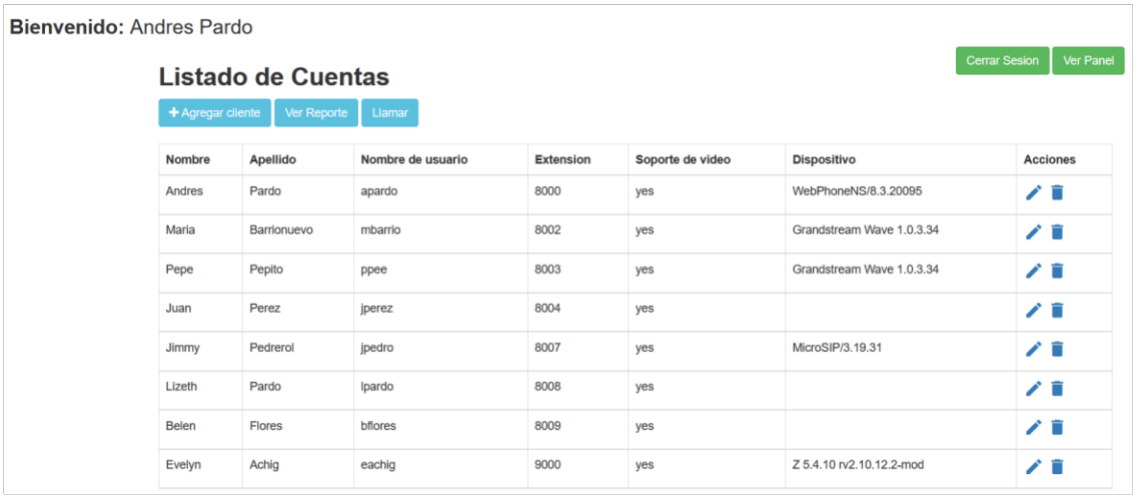


Figura 3.6 Pantalla principal de administración.

En el Código 3.36 entre las líneas 1 a la 36 se muestra la creación de la tabla que presenta a los usuarios existentes en el sistema, se utiliza la clase *table table-boardered* que es una clase de Bootstrap para darle estilo. Para la creación de las filas se utiliza PHP. En la línea 2 se incluye al archivo *database.php*, de esta manera se puede instanciar un objeto de la clase Database (línea 3) y utilizar la función *readnew()* que permite obtener todos los usuarios (línea 4).

Entre las líneas 5 y 30 se define el ciclo while que permite crear dinámicamente tantas filas como usuarios existan. Las líneas 21 a 28 permiten crear la columna “Acciones”, para realizar esto se utiliza dos etiquetas <a> que permiten redireccionar hacia los archivos *update.php* y *delete.php*, enviando el id del usuario.

```
1  <?php
2          include ('database.php');
3          $clientes = new Database();
4          $listado=$clientes->readnew();
5          while ($row=pg_fetch_object($listado)){
6          $id=$row->id_account;
7          $nombres=$row->name;
8          $apellido=$row->lastname;
9          $uname=$row->username;
10         $ext=$row->extension;
11         $video=$row->videosupport;
12         $uagent=$row->useragent;
13     ?>
14   <tr>
15     <td><?php echo $nombres;?></td>
16     <td><?php echo $apellido;?></td>
17     <td><?php echo $uname;?></td>
18     <td><?php echo $ext;?></td>
19     <td><?php echo $video;?></td>
20     <td><?php echo $uagent;?></td>
21     <td>
22       <a href="update.php?id=<?php echo $id;?>" class="edit"
23                       title="Editar" data-toggle="tooltip">
24                 <i class="material-icons">&#xE254;</i></a>
25       <a href="delete.php?ext=<?php echo $ext;?>" class="delete"
26                       title="Eliminar" data-toggle="tooltip">
27                 <i class="material-icons">&#xE872;</i></a>
28     </td>
29   </tr>
30 <?php } ?>
```

Código 3.36 Codificación de la tabla para administración.

3.4.3.1 Crear y editar usuarios

En la Figura 3.7 se muestra la pantalla para agregar un usuario. Los campos solicitados son nombre, apellido, extensión, nombre de usuario y password. Todos estos campos pueden ser alfanuméricos, pero se recomienda usar solo números para el campo extensión. Los campos nombre de usuario y password serán utilizados por el usuario para ingresar al aplicativo y poder realizar llamadas.

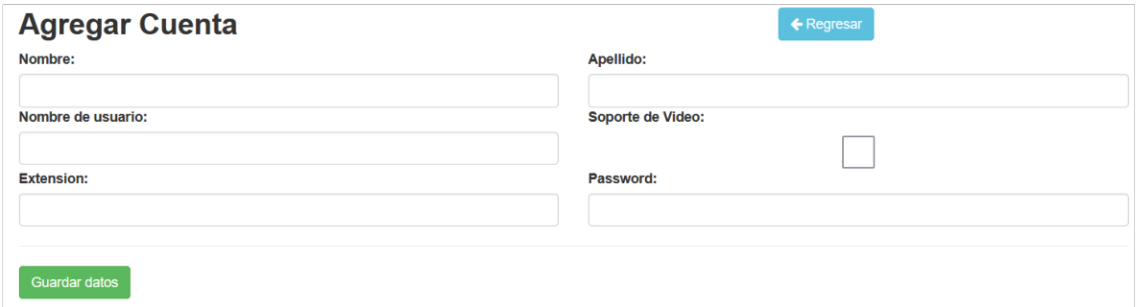


Figura 3.7 Formulario para creación de un usuario.

En el Código 3.37 se muestra la creación de un input (caja de texto) necesario para la toma de los datos del nuevo usuario. La palabra reservada *required* solicita de manera obligatoria se llenen los inputs antes de realizar un post en el formulario.

```
<div class="col-md-6">
    <label>Password:</label>
    <input type="password" name="passw" id="passw"
           class='form-control' maxlength="64" required>
</div>
```

Código 3.37 Creación de una caja de texto para toma de datos.

La pantalla para editar usuarios es muy similar a la pantalla de creación de usuario, se incluyen los mismos inputs y prácticamente el mismo código.

En el Código 3.38 se observa como mediante la etiqueta value y utilizando php los parámetros correspondientes a la extensión son llenados a partir de la información del usuario (línea 5). De manera similar ocurre con los campos adicionales. Al momento de realizar un submit todos estos valores serán enviados nuevamente a la base de datos y actualizados en caso de haberse realizado alguna modificación.

```
<div class="col-md-6">
    <label>Extension:</label>
    <input type="text" name="extension" id="extension"
        class='form-control' maxlength="64" required
        value="<?php echo $data_usuario->extension;?>">
</div>
```

Código 3.38 Creación de una caja de texto para edición de datos.

3.4.3.2 Reporte de llamadas

Figura 3.8 Pantalla para reporte de llamadas

En el Código 3.39 se muestra la parte más importante del archivo *report.php*. Entre las líneas 1 a 24 se crea la tabla que contendrá los registros de las llamadas. Se utiliza PHP para la creación dinámica de las filas, podemos ver este código entre las líneas 12 a 21. En la línea19 se utiliza la etiqueta audio que nos permitirá escuchar y descargar el audio de la grabación en formato ogg.

```
1  <?php
2  while ($row=pg_fetch_object($listado)){
3   $orig=$row->source;
4   $dest=$row->destination;
5   $state=$row->disposition;
6   $duration=$row->duration;
7   $talk=$row->billsec;
8   $start=$row->start;
9   $uniqueid=$row->uniqueid;
10  $path='audio/'.$uniqueid.'.ogg';
11 ?>
12 <tr>
13  <td><?php echo $orig;?></td>
14  <td><?php echo $dest;?></td>
15  <td><?php echo $state;?></td>
16  <td><?php echo $duration . ' s';?></td>
17  <td><?php echo $talk . ' s';?></td>
18  <td><?php echo $start;?></td>
19  <td><audio src="<?php echo $path;?>"
20          controls="controls"> </audio> </td>
21 </tr>
22 <?php
23 }
24 ?>
```

Código 3.39 Codificación del archivo report.php.

En la Figura 3.8 al momento de dar clic en Exportar obtendremos un archivo de excel con extensión xls. El nombre del archivo es *registro_llamadas.xls*. El archivo de excel contendrá todos los registros de las llamadas realizadas. Para exportar los registros de llamadas utilizaremos la cabecera Content-type. De este modo podemos transformar el código HTML que corresponde a la tabla de registros de llamadas.

En el Código 3.40 se muestra un fragmento del archivo *report-excel.php*. En la línea 3 se indica el tipo de archivo el tipo de archivo y la codificación del archivo que se exportará. En la línea 4 configuramos el nombre y extensión del archivo.

Entre la línea 14 a 20 podemos visualizar la primera fila correspondiente al encabezado de la tabla, el resto de las filas correspondientes a todos los registros serán creadas atreves de PHP.

```
1  <?php
2  include_once 'database.php';
3  header("Content-type: application/vnd.ms-excel; charset=UTF-8");
4  header("Content-Disposition: attachment;
5          filename=registros_llamadas.xls");
6  header("Pragma: no-cache");
7  header("Expires: 0");
8  $report= new Database();
9  $listado= $report->report_total();
10 ?>
```

Código 3.40 Cabecera del archivo report-excel.php.

3.4.3.3 Panel de llamadas

Para la realización de este panel se tomó como base el archivo ajamdemo.html que se crea por defecto dentro del directorio del servidor de Asterisk (*/var/lib/asterisk/static-http/*) y que permite conectarse a AMI por medio de peticiones web, utilizando archivos JavaScript (funciones) brindan la posibilidad de escuchar eventos (en este caso llamadas) y presentarlos en una página web. Todos estos archivos son de código libre y permiten a los usuarios desarrollar rápidamente páginas web que pueden interactuar con el core de Asterisk por medio de AMi y su servidor web. A esta tecnología Asterisk la define como AJAM [18].

En la Figura 3.9 se presenta la pantalla correspondiente al panel de monitoreo de llamadas. Dentro de esta pantalla se podrán observar los canales activos en tiempo real. Una llamada consta de dos canales activos, el emisor y el receptor. En la Figura 3.9

podemos visualizar una llamada en proceso, se identifica al llamante por medio del contexto *dial-internal*, en la figura el usuario Evelyn Achig realiza una llamada a Sofía Revelo. La columna contexto representa la extensión marcada y el punto de entrada en el dialplan.

Figura 3.9 Pantalla para visualización de llamadas.

El Código 3.41 pertenece al encabezado del archivo *panel.html* (archivo ajamdemo.html renombrado y modificado). Entre las líneas 6 a 9 cargamos las hojas de estilos desde las fuentes de Google, así como un script de Bootstrap. En las líneas 11 y 12 hacemos referencia al API que nos brinda el demo de Asterisk para poder acceder a los eventos del core mediante AMI, vía peticiones web. Esta API se compone de archivos javascript que implementan funciones que permiten traducir los eventos leídos desde AMI hacia una página web.

```
1  <meta charset="utf-8">
2  <meta http-equiv="X-UA-Compatible" content="IE=edge">
3  <meta name="viewport" content="width=device-width,initial-scale=1">
4  <link rel="stylesheet" href="https://fonts.googleapis.com/
5          css?family=Roboto|Varela+Round|Open+Sans">
6  <link rel="stylesheet" href="https://fonts.googleapis.com/
7        icon?family=Material+Icons">
8  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/
9        font-awesome/4.7.0/css/font-awesome.min.css">
10 <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/
11         bootstrap/3.3.7/css/bootstrap.min.css">
12 <script src="prototype.js"></script>
13 <script src="astman.js"></script>
14 <link href="astman.css" media="all" rel="Stylesheet" type="text/css"/>
```

Código 3.41 Cabecera del archivo panel.html.

Dentro del archive panel.html se definen funciones que hacen uso del archivo *astman.js*. En el Código 3.42 se puede observar la función *doLogin()* que permite autenticarse con AMI, las credenciales a usar se definieron en el Código 3.27. En la línea 3 se observa el uso de la función *sendRequest* definida en el API (*astman.js*) que permite enviar una petición de login con las credenciales definidas en las variables username (admin) y secret (paradise).

```
1 function doLogin() {
2     $('statusbar').innerHTML = "<i>Logging in...</i>";
3     astmanEngine.sendRequest('action=login&username='
4         + $('username').value + "&secret="
5         + $('secret').value, demo.logins);
6 }
```

Código 3.42 Función para registro de ingreso con AMI.

El Código 3.43 pertenece al archivo astman.js. Dentro de este archivo se declaran funciones que permiten manejar los eventos correspondientes a los canales y poder presentarlos dentro del archivo *panel.html*. En la línea 1 y 2 creamos la tabla e insertamos la fila de encabezado que nos permitirá visualizar los canales activos en tiempo real. Este comportamiento asíncrono se logra gracias a AJAX. En la línea 13 se declara el lazo for que permitirá seguir creando tantas filas en la tabla como canales existan.

```
This.channelTable = function(callback) {
var s, x;
var cclass, count=0;
var found = 0;
var foundactive = 0;
var fieldlist = new Array("channel", "callerid", "calleridname",
                "context", "extension", "priority");
me.chancallback = callback;
s = "<table class='chantable'>\n";
s = s + "\t<tr class='labels' id='labels'><td>Canal</td>
                <td>Usuario</td><td>Contexto</td></tr>";
count=0;
for (x in channels) {
 if (channels[x].channel) {
  if (count % 2)
      cclass = "chanlistodd";
  else
      cclass = "chanlisteven";
       if (me.selecttarget && (me.selecttarget == x)) {
               cclass = "chanlistselected";
               foundactive = 1;
       }
      count++;
      s = s+"\t<tr class='" + cclass + "' id='"+channels[x].channel
           + "' onClick='astmanEngine.clickChannel(event)'>";
      s = s + "<td>" + channels[x].channel + "</td>";
```

Código 3.43 Función de creación de tabla para visualización de llamadas.

4 RESULTADOS, CONCLUSIONES Y RECOMENDACIONES

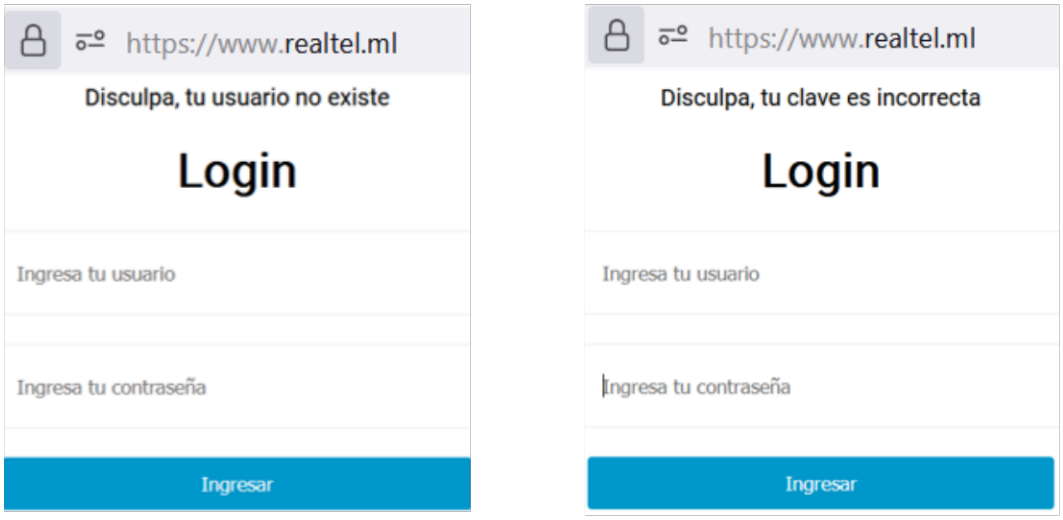
4.1 Pruebas de funcionamiento

En este apartado se realizarán pruebas de funcionamiento del prototipo web. De esta forma se validará que todos los componentes implementados cumplan los requisitos descritos en la fase de diseño.

4.1.1 Iniciar sesión en la aplicación

En la Figura 4.1 se presenta la interfaz gráfica para el inicio de sesión. En la sección del URL se observa que el protocolo para comunicación con el servicio web es https. En esta

prueba se valida que el usuario pueda ingresar al sistema. Se observa los dos casos que se pueden presentar cuando un usuario ingresa de manera incorrecta sus credenciales.



a) Usuario no existe b) Clave incorrecta

Figura 4.1 Interfaz de ingreso.

4.1.2 Agente

Si el usuario tiene el rol de agente una vez ingrese sus credenciales correctamente se presentará la interfaz mostrada en la Figura 4.2. En la parte superior izquierda se visualiza el nombre del agente y su extensión. En la parte inferior izquierda se presenta el directorio de todos los agentes que consten en el aplicativo. En la parte superior derecha se muestra el control de llamadas y videollamadas, desde aquí podemos contestar llamadas entrantes, colgar y realizar una llamada a la PSTN ingresando un número en la caja de texto.

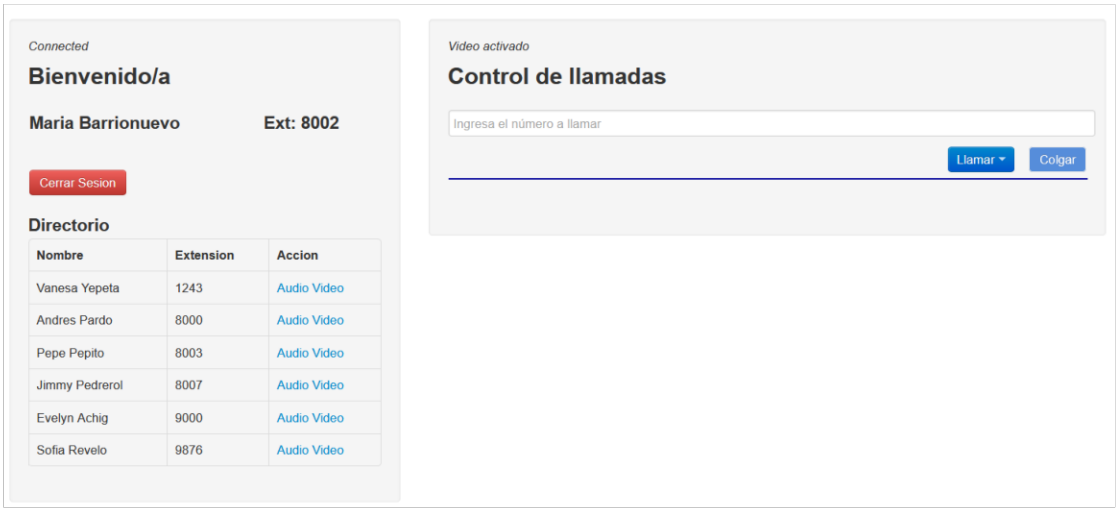


Figura 4.2 Interfaz para llamadas y videollamadas.

4.1.2.1 Videollamada

En este caso se realizará una videollamada a Sofia Revelo. Para esto usaremos el directorio. En la columna Acción damos clic en Video. El navegador nos solicitará permiso para usar el Micrófono y Cámara como en la Figura 4.3. En esta prueba se valida el funcionamiento de una videollamada con otro agente.

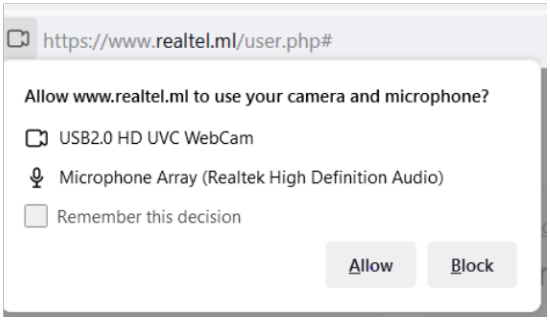


Figura 4.3 Permiso para utilización de micrófono y cámara.

Una vez la videollamada sea contestada se presentará dos recuadros en la parte inferior derecha como se visualiza en la Figura 4.4. El cuadro de mayor tamaño corresponde al video del otro extremo de la comunicación (video remoto), mientras el recuadro pequeño corresponde al video local.

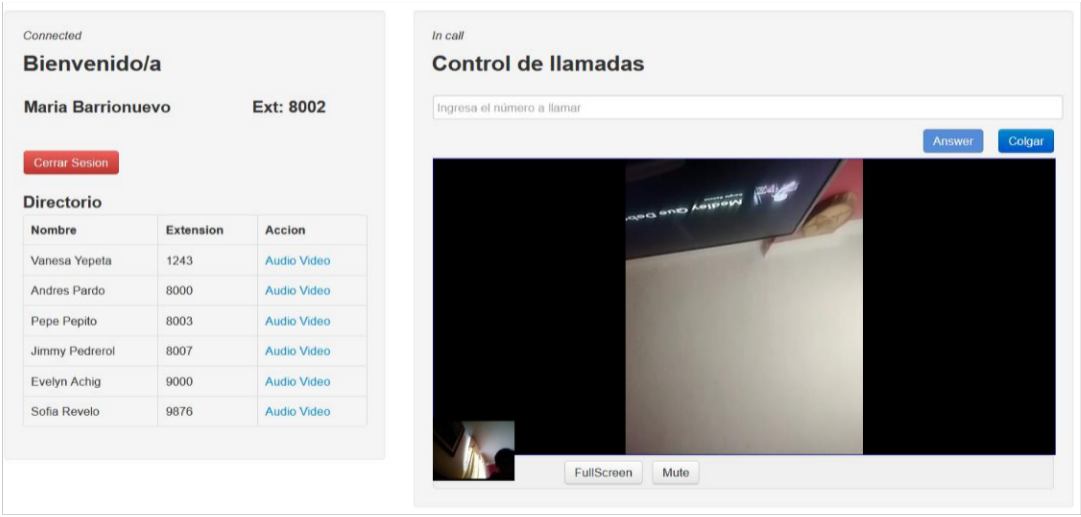


Figura 4.4 Videollamada.

4.1.2.2 Llamada interna

De igual manera para la llamada se realizará utilizando el directorio. En Acciones es necesario dar clic en Audio, la llamada se realizará a María Barrionuevo. Si se solicita acceso para el micrófono es necesario permitirlo, al igual que lo realizado en la

videollamada. En la Figura 4.5 se puede observar la interfaz para el agente María Barrionuevo en la parte superior derecha se visualiza la extensión desde la que se llama y se activa el botón para contestar la llamada. En esta prueba se valida el funcionamiento de una llamada con otro agente.

Figura 4.5 Llamada entrante.

4.1.2.3 Llamada a la PSTN

Para realizar una llamada hacia la PSTN se utilizará la caja de texto que se ubica en la parte superior derecha. Se llamará al número celular 0984078210. En la Figura 4.6 se observa la interfaz para iniciar la llamada. Es necesario dar clic en Llamar y Audio. En esta prueba se valida el funcionamiento de una llamada externa hacia la PSTN.

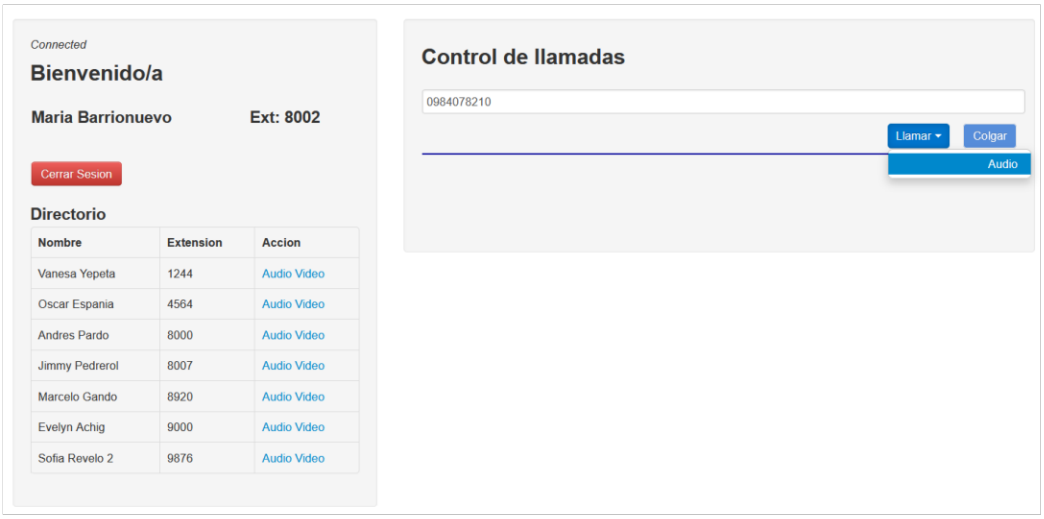


Figura 4.6 Llamada externa a un número celular.

En caso de que se intente realizar una llamada sin haber ingresado ningún número en la caja de texto se desplegara la alerta presentada en la Figura 4.7.

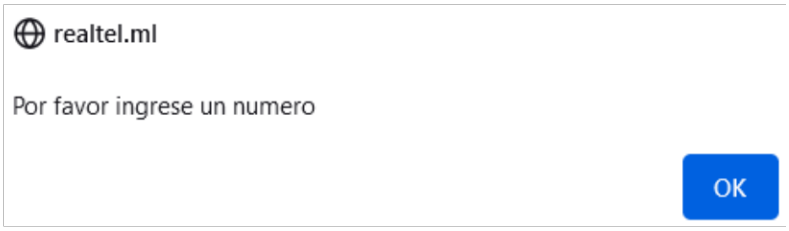


Figura 4.7 Alerta número no ingresado.

4.1.3 Administrador

Si el usuario tiene el rol de administrador una vez ingrese sus credenciales correctamente se presentará la interfaz mostrada en la Figura 4.8. En la parte superior izquierda se visualiza el nombre del administrador. En la parte central se muestra la tabla con los usuarios del sistema. Desde la columna acciones podemos editar o eliminar un usuario.

Figura 4.8 Interfaz de administración.

4.1.3.1 Crear Usuario

En la Figura 4.9 se observa la pantalla de creación de usuario, todos los campos son requeridos por lo que no se guardaran los datos sin llenar la información en todos los campos. Los campos nombre de usuario y extensión son únicos. En esta prueba se valida la creación de un agente.

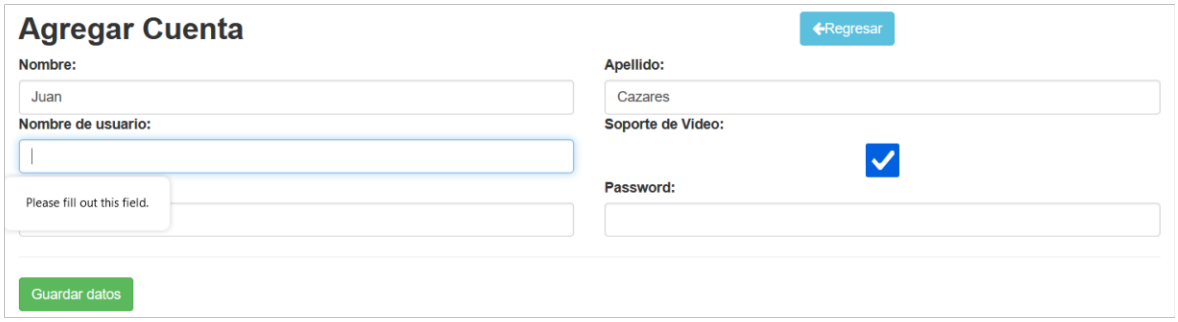


Figura 4.9 Ingreso de datos para creación de un usuario.

En la Figura 4.10 se observa una etiqueta informativa. Esta etiqueta se muestra cuando un usuario ha sido creado correctamente.

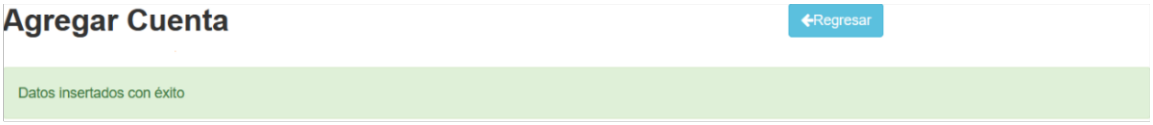


Figura 4.10 Etiqueta usuario agregado con éxito.

En la Figura 4.11 se observa otra etiqueta informativa, en este caso esta etiqueta se presenta cuando ha ocurrido un inconveniente, por ejemplo, se intenta crear otro usuario con la misma extensión, o cuando ya existe otra cuenta con el mismo nombre de usuario.

Figura 4.11 Etiqueta usuario no pudo ser agregado.

4.1.3.2 Editar Usuario

Para editar un usuario se debe dar clic en el icono remarcado en la Figura 4.12, este icono está ubicado en la columna Acciones que se puede visualizar para cada usuario creado en el aplicativo web. Esto se observa en el listado de usuarios que se presenta en la Figura 4.8. Una vez se dé clic en el icono se presenta la pantalla descrita en la Figura 4.13. En esta prueba se valida la edición de un agente.

Figura 4.12 Edición de un usuario.

Al igual que en la creación de usuario la pantalla presentada en la Figura 4.13 solicitada la misma información para la actualización de un usuario. Todos los campos son requeridos. Se realiza la misma validación tanto para el nombre de usuario y extensión ya que son parámetros únicos para cada agente.

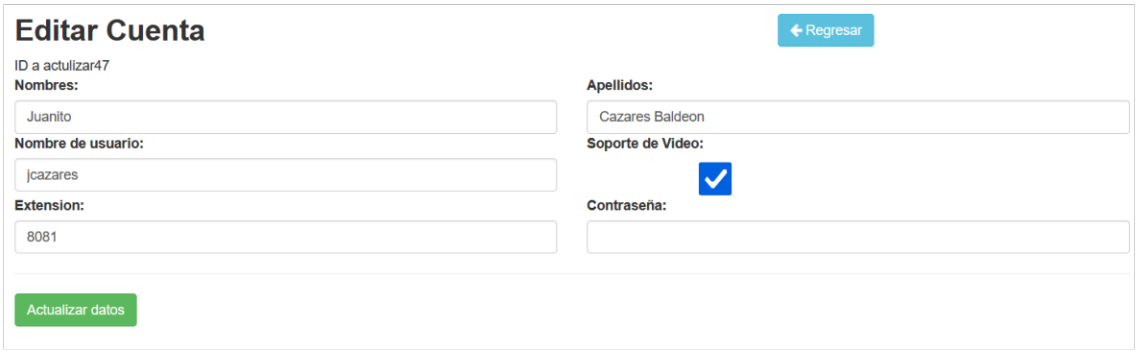


Figura 4.13 Ingreso de datos para edición de un usuario.

En la Figura 4.14 se muestra una etiqueta informativa para el caso de que la actualización no se haya realizado.

Figura 4.14 Etiqueta usuario no pudo ser actualizado.

La Figura 4.15 muestra una etiqueta informativa para una actualización correcta de un usuario.

Figura 4.15 Etiqueta usuario actualizado con éxito.

4.1.3.3 Eliminar Usuario

Para eliminar un usuario se debe dar clic en el icono remarcado en la Figura 4.16, este icono está ubicado en la columna Acciones que se puede visualizar para cada usuario creado en el aplicativo web. Esto se observa en el listado de usuarios que se presenta en la Figura 4.8. En esta prueba se valida la eliminación de un agente.

Figura 4.16 Eliminar un usuario.

4.1.3.4 Reporte de llamadas

En esta prueba se valida la visualización de registros de llamadas. Para visualizar los registros de llamadas se debe dar clic en el botón Ver Reportes en la pantalla que se presenta en la Figura 4.8. Una vez hecho esto se observa la pantalla descrita en la Figura 4.17. En esta pantalla se muestran los registros de todas las llamadas realizadas ordenadas desde la fecha de la última llamada, de forma descendente. Estos registros se presentan en una tabla como se visualiza en la Figura 4.17, que consta de siete columnas con datos como el origen, destino, estado, tiempo total de llamada, tempo hablado y grabación. El estado hace referencia a si la llamada ha sido contestada o no contestada, de esto dependerá si se tiene grabación o no de la llamada.

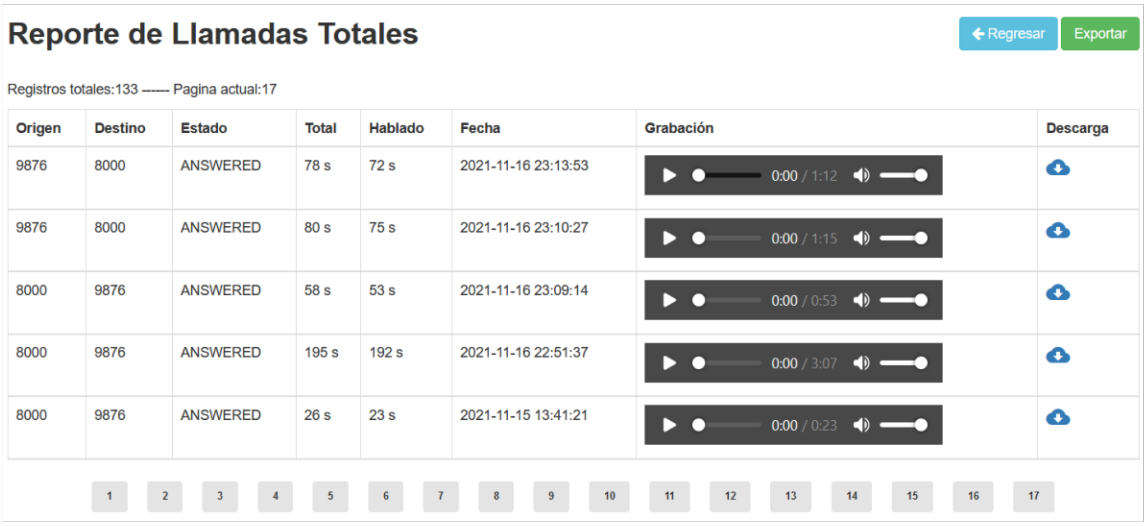
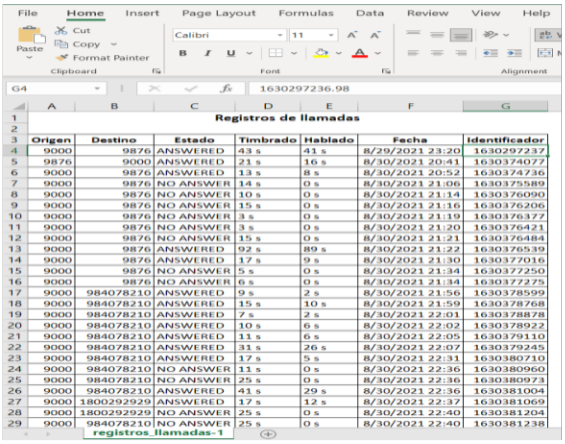


Figura 4.17 Interfaz de reporte y grabaciones de llamadas.

La tabla que contiene los registros de la llamada consta de ocho filas, en la parte inferior se observa un paginador que permite ir presentando todos los registros en la tabla. En la parte superior de la tabla se presenta una etiqueta que presenta la información del número total de registros y en que página se encuentra actualmente.

Para descargar estos registros en un archivo de Excel se debe dar clic en el botón Exportar ubicado en la parte superior derecha que se observa en la Figura 4.17. El archivo tendrá el siguiente nombre registros_llamadas.xls. Un ejemplo del reporte descargado se puede observar en la Figura 4.18, contiene las mismas columnas que la tabla presentada en la Figura 4.17, a excepción de la columna Grabación que ha sido reemplazada por el identificador único de la llamada. En esta prueba se valida la descargar del reporte con los registros de llamadas.



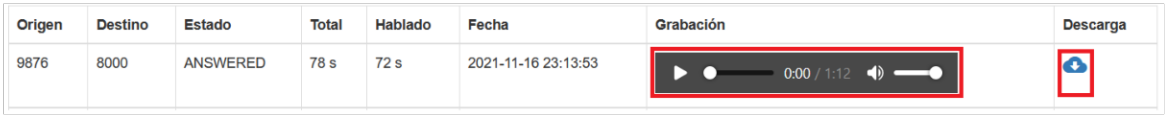
Registros de llamadas

Origen	Destino	Estado	Timbrado	Hablado	Fecha	Identificador
9000	9876	ANSWERED	43 s	41 s	8/29/2021 23:20	1630297237
9876	9000	ANSWERED	21 s	16 s	8/30/2021 20:41	1630374077
9000	9876	ANSWERED	13 s	8 s	8/30/2021 20:52	1630374736
9000	9876	NO ANSWER	14 s	0 s	8/30/2021 21:06	1630375589
9000	9876	NO ANSWER	10 s	0 s	8/30/2021 21:14	1630376090
9000	9876	NO ANSWER	15 s	0 s	8/30/2021 21:16	1630376206
9000	9876	NO ANSWER	3 s	0 s	8/30/2021 21:19	1630376377
9000	9876	NO ANSWER	3 s	0 s	8/30/2021 21:20	1630376421
9000	9876	NO ANSWER	15 s	0 s	8/30/2021 21:21	1630376484
9000	9876	ANSWERED	92 s	89 s	8/30/2021 21:22	1630376539
9000	9876	ANSWERED	17 s	9 s	8/30/2021 21:30	1630377016
9000	9876	NO ANSWER	5 s	0 s	8/30/2021 21:34	1630377250
9000	9876	NO ANSWER	6 s	0 s	8/30/2021 21:34	1630377275
9000	984078210	ANSWERED	9 s	2 s	8/30/2021 21:56	1630378599
9000	984078210	ANSWERED	15 s	10 s	8/30/2021 21:59	1630378768
9000	984078210	ANSWERED	7 s	2 s	8/30/2021 22:01	1630378878
9000	984078210	ANSWERED	10 s	6 s	8/30/2021 22:02	1630378922
9000	984078210	ANSWERED	11 s	6 s	8/30/2021 22:05	1630379110
9000	984078210	ANSWERED	31 s	26 s	8/30/2021 22:07	1630379245
9000	984078210	ANSWERED	17 s	5 s	8/30/2021 22:31	1630380710
9000	984078210	NO ANSWER	11 s	0 s	8/30/2021 22:36	1630380960
9000	984078210	NO ANSWER	25 s	0 s	8/30/2021 22:36	1630380973
9000	984078210	ANSWERED	41 s	29 s	8/30/2021 22:36	1630381004
9000	18002929229	ANSWERED	17 s	12 s	8/30/2021 22:37	1630381069
9000	18002929229	NO ANSWER	25 s	0 s	8/30/2021 22:40	1630381204
9000	984078210	NO ANSWER	25 s	0 s	8/30/2021 22:40	1630381238

Figura 4.18 Reporte exportado en un archivo Excel.

4.1.3.5 Grabaciones de llamadas

En esta prueba se valida la escucha de una grabación de llamada desde el aplicativo web. En la Figura 4.19 podemos observar una fila de la tabla presentada en la Figura 4.17. Esta fila corresponde a un registro de una llamada. En la penúltima columna podemos observar un control de audio que nos permite escuchar la grabación. Es importante que para que se pueda escuchar la grabación la llamada haya sido contestada (columna Estado) y tenga tiempo de hablado (columna Hablado).



Origen	Destino	Estado	Total	Hablado	Fecha	Grabación	Descarga
9876	8000	ANSWERED	78 s	72 s	2021-11-16 23:13:53	0:00 / 1:12	

Figura 4.19 Control de grabación de llamada.

Para descargar la grabación de llamada se debe dar clic en el icono en la columna Descarga. En la Figura 4.20 se puede observar un cuadro de dialogo que se genera al momento de descargar la grabación, el nombre de la grabación es el identificador único de llamada que se puede observar en el reporte. En esta prueba se valida la descarga de una grabación de llamada.

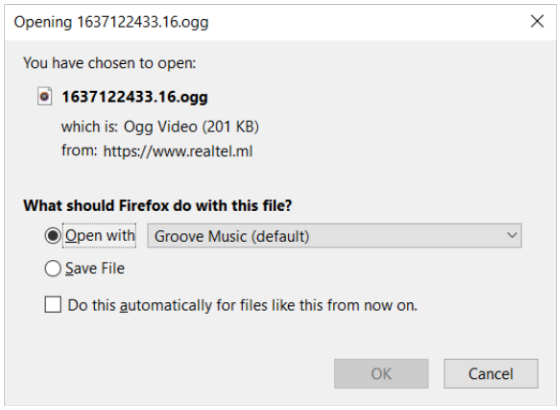


Figura 4.20 Cuadro de dialogo para descarga de grabación de llamada.

En la Figura 4.21 se puede observar un controlador de audio de un Navegador Chrome, este controlador de audio permite descargar fácilmente la grabación y ofrece otras características como variar la velocidad de reproducción.

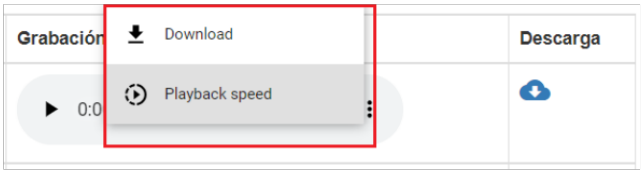


Figura 4.21 Control de audio de un navegador Chrome.

4.1.3.6 Panel de llamadas

En esta prueba se valida la visualización de llamadas en tiempo real. En la Figura 4.22 se observa el panel de llamadas, se considera una llamada como la comunicación de dos canales. El canal llamante, marcado en tono ocre y el canal receptor, marcado en tono violeta. Se presenta tres columnas que ofrecen información sobre cada canal, una de las más importantes es la columna Usuario ya que en esta columna se puede identificar las personas que están involucradas en la comunicación.

Figura 4.22 Visualización de una llamada interna

Las columnas Canal y Contexto nos proporcionan información acerca de la tecnología del canal (SIP) y el contexto (dialplan) en que los canales son utilizados para poder interactuar. Esta información es obtenida a través de AMI y los eventos que nos permite escuchar. En la Figura 4.23 se puede observar una llamada externa realizada por el usuario Andres Pardo hacia el número 02999999.

Figura 4.23 Visualización de una llamada externa.

Este tipo de llamadas se identifican ya que en la columna Usuario, en el segundo canal (segunda fila) se presenta la palabra PSTN, haciendo referencia a que se trata de una llamada hacia la red conmutada de telefonía. La columna contexto contiene el número marcado.

En la Figura 4.24 se puede observar un panel con dos llamadas simultaneas. La primera llamada se trata de una llamada interna entre los usuarios Andres Pardo y María Barrionuevo, la segunda llamada es realizada por Sofia Revelo y se trata de una llamada externa realizada al número celular 0984078210. En caso de existir una tercera llamada se visualizaría por debajo de la segunda, siguiendo el mismo esquema de dos canales.

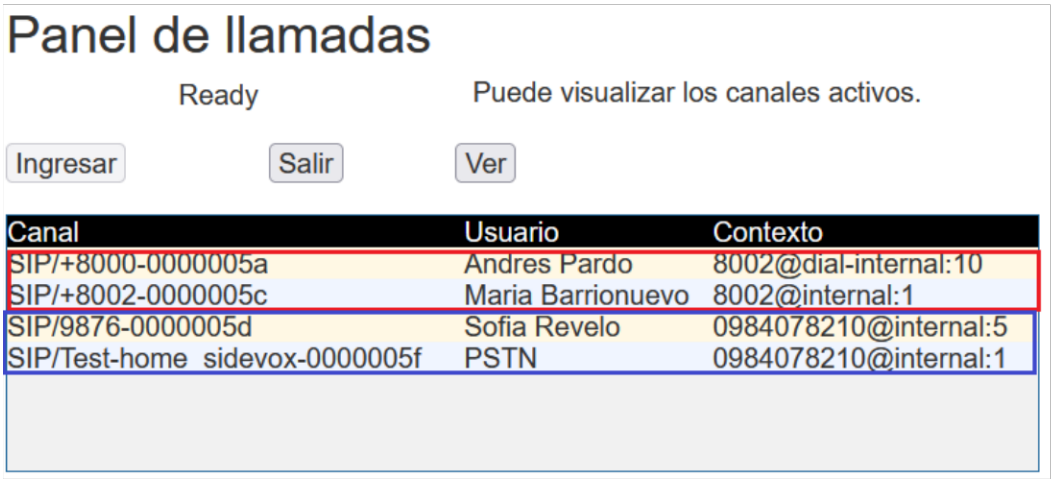


Figura 4.24 Visualización de dos llamadas simultaneas.

En el Anexo G se incluye un enlace a un video demostrativo donde se expone cada una de las pruebas de funcionamiento.

4.2 Pruebas de validación

Las pruebas de validación del prototipo de aplicación web se realizaron mediante cuestionarios cerrados a diferentes usuarios. Los cuestionarios fueron aplicados de la siguiente manera: a 3 personas que pertenecen a la organización Biosphere. Y tienen como finalidad garantizar que los casos de uso presentados se cumplen.

En la Tabla 4.1 se presentan los resultados del cuestionario para un usuario de tipo Administrador.

Tabla 4.1 Resultados de cuestionario realizado a un administrador.

N.º	Pregunta	Respuesta	
		SI	NO
1	¿Pudo ingresar a la aplicación web con el nombre de usuario admin y la contraseña paradise?	100%	0%

2	La aplicación web, ¿Le permitió crear un usuario?	100%	0%
3	La aplicación web, ¿Le permitió visualizar el/los usuarios?	100%	0%
4	La aplicación web, ¿Le permitió editar un usuario?	100%	0%
5	La aplicación web, ¿Le permitió eliminar un usuario?	100%	0%
6	La aplicación web, ¿Le permitió visualizar los registros de llamadas realizadas?	100%	0%
7	La aplicación web, ¿Le permitió descargar los registros de llamadas realizadas?	100%	0%
8	La aplicación web, ¿Le permitió escuchar online y descargar las grabaciones de las llamadas?	100%	0%
9	La aplicación web, ¿Le permitió visualizar las llamadas que se están realizando?	100%	0%

La pregunta uno, tiene como finalidad comprobar el control de acceso para un usuario con el rol administrador, arrojando como resultado que todos los entrevistados pudieron ingresar con las credenciales entregadas.

Las preguntas dos, tres, cuatro y cinco, tienen como objetivo comprobar que el administrador pueda realizar tareas tipo CRUD sobre los usuarios. El resultado que se obtuvo es que todos los entrevistados pudieron crear, visualizar, editar y eliminar un usuario.

Las preguntas seis y siete, tiene como finalidad comprobar que el administrador pueda visualizar los registros de las llamadas y descargar dichos registros. El resultado que se tuvo es que todos los entrevistados pudieron tanto visualizar como descargar los registros.

La pregunta 8, tiene como objetivo comprobar que el administrador pueda escuchar desde el navegador las grabaciones de las llamadas y además descargarlas. Todos los entrevistados pudieron realizar dichas acciones.

La pregunta 9, tiene como finalidad constatar que el administrador pueda visualizar las llamadas en tiempo real que se estén realizando. El resultado obtenido fue que todos los entrevistados pudieron visualizar las llamadas.

En la Tabla 4.2 se presentan los resultados del cuestionario para un usuario de tipo Agente.

Tabla 4.2 Resultados de cuestionario realizado a un agente.

N.º	Pregunta	Respuesta	
		SI	NO
1	¿Pudo ingresar a la aplicación web con el nombre de usuario y contraseña proporcionadas por el administrador?	100%	0%
2	La aplicación web, ¿Le permitió realizar una videollamada con otro agente registrado en el sistema?	100%	0%
3	La aplicación web, ¿Le permitió realizar una llamada con otro agente registrado en el sistema?	100%	0%
4	La aplicación web, ¿Le permitió realizar una llamada hacia un número externo?	100%	0%

La pregunta uno, busca comprobar que un usuario con el rol de agente pueda ingresar correctamente en el aplicativo web con las credenciales proporcionadas por el administrador. Se obtuvo como resultado que todos los entrevistados pudieron ingresar al aplicativo con las credenciales.

La pregunta dos, tres y cuatro tienen como objetivo comprobar el funcionamiento tanto para llamadas y videollamadas entre los agentes, así como, el funcionamiento para llamadas externas hacia la PSTN. Los resultados obtenidos fueron que todos los entrevistados pudieron realizar estas actividades.

4.3 Conclusiones

- Se analizaron los conceptos y las tecnologías necesarias para desarrollar el prototipo, haciendo énfasis en el API sipML5 que implementa la tecnología WebRTC que es el pilar fundamental para poder transmitir audio y video desde un navegador Web, así como de Asterisk que es el servidor de comunicaciones que permite gestionar las sesiones de los clientes y comunicarse con cualquier dispositivo VoIP.
- Se realizó el diseño del prototipo de aplicación web tomando en cuenta los requerimientos funcionales y no funcionales, se utilizó un modelo basado en capas para el desarrollo y se establecieron casos de uso para determinar así la funcionalidad que debe cumplir el proyecto.

- Se implementó la aplicación Web usando PHP como lenguaje del lado del servidor, para el desarrollo de cada una de las pantallas que se presentan al usuario se utilizaron HTML y CSS. Javascript fue totalmente necesario ya que el API sipML5 es una librería con un conjunto de funciones escritos en este lenguaje y gracias a esta API se pudo transmitir tanto audio y video desde el cliente web.

- Las pruebas de funcionamiento de la aplicación web se realizaron utilizando los navegadores Google Chrome, Microsoft Edge y Mozilla Firefox tomando en cuenta los casos de usos presentados en la etapa de diseño. Los resultados fueron satisfactorios.

- Para la persistencia de la información se usó una base de datos relacional y se empleó como motor de base de datos a PostgreSQL ya que presenta un tiempo de respuesta en consultas bastante bajo y la conexión mediante ODBC que se puede implementar con Asterisk es intuitiva y relativamente sencilla.

- Ya que uno de los requisitos funcionales era tener una aplicación web que fuera accesible desde cualquier red, se publicó el servicio en Internet usando una instancia EC2 de AWS. Para este prototipo se usó una instancia dentro de la capa gratuita, pero se podría potenciar los recursos de dicha instancia y aumentar la capacidad de usuarios máxima que esta podría soportar, así como, utilizar otras características que ofrece AWS como balanceadores de carga.

- Al tener expuesto el servicio a Internet es imprescindible proteger principalmente al servicio de señalización (SIP) contra ataques DDOS, se ha implementado un IDS mediante Fail2ban que protege al prototipo de los ataques principales, esto gracias a la base de datos de patrones que vienen incluidos en Fail2ban.

4.4 Recomendaciones

- El presente proyecto funciona perfectamente en un navegador web en PC de escritorio o portátil, sin embargo, el diseño no contempló ser responsive por lo que no está perfectamente adaptado a pantallas más pequeñas. Se recomienda a partir de este prototipo codificar los archivos CSS mediante media query, de esta manera se puede conseguir una buena experiencia en diferentes tamaños de pantallas.

- El API sipML5 ofrece muchas funciones que pueden permitir extender el funcionamiento del aplicativo, por ejemplo, podría permitir el envió de mensajes de texto e incluso la compartición de pantalla entre los clientes web.
- Se recomienda usar este aplicativo con el navegador Google Chrome o navegadores basados en Chromium ya que el soporte de WebRTC que poseen está en constante actualización y mantenimiento. Además, Google Chrome posee una herramienta de desarrollo (chrome://webrtc-internals/) que permite observar información acerca del uso de WebRTC.
- Se podría cambiar la arquitectura del presente proyecto y así aprovechar las características que ofrece AWS, por ejemplo, utilizando balanceadores de carga para el servicio HTTP o creando grupos de auto escalamiento que ofrecen crecimiento horizontal de acuerdo con métricas como carga y procesamiento.
- La implementación del IDS se puede extender incluyendo patrones de ataques propios y añadiendo seguridad a otros servicios, como el servicio web.

5 REFERENCIAS BIBLIOGRÁFICAS

[1] J. A. Carballar, *VOIP. La telefonia de Internet*. España, 2007.

[2] E. S. J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, "RFC 3261: SIP: Session Initiation Protocol." 2002.

[3] G. E. C. Holmberg, S. Hakansson, "RFC 7478: Web Real-Time Communication Use Cases and Requirements." 2015.

[4] Doubango Telecom, "sipML5 - The world's first open source HTML5 SIP client," 2012. https://www.doubango.org/sipml5/index.html?svn=236 (accessed Jul. 13, 2021).

[5] A. M. I. Fette, "RFC 6455: The WebSocket Protocol." 2011.

[6] R. M. J. Rosenberg, J. Weinberger, C. Huitema, "RFC 3489: STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)." 2003.

[7] "Asterisk Architecture, The Big Picture - Asterisk Project - Asterisk Project Wiki," 2014. https://wiki.asterisk.org/wiki/display/AST/Asterisk+Architecture%2C+The+Big+Picture (accessed Jul. 14, 2021).

[8] A. Project, "Realtime Database Configuration," 2013. https://wiki.asterisk.org/wiki/display/AST/Realtime+Database+Configuration (accessed Jan. 10, 2022).

[9] A. Project, "Asterisk Manager Interface," 2010. https://wiki.asterisk.org/wiki/display/AST/The+Asterisk+Manager+TCP+IP+API (accessed Dec. 20, 2021).

[10] A. Project, "Asterisk Builtin mini-HTTP Server," 2014. https://wiki.asterisk.org/wiki/display/AST/Asterisk+Builtin+mini-HTTP+Server (accessed Jan. 10, 2022).

[11] Amazon, "AWS." https://aws.amazon.com/es/what-is-cloud-computing/ (accessed Jan. 02, 2022).

[12] A. Corletti Estrada, *Seguridad en redes de Internet*. 2016.

[13] P. Mauricio, "Biosphere," 2020. https://www.flowcode.com/page/biosphere.

[14] “Ninjamock.” https://ninjamock.com/about-us (accessed Dec. 10, 2021).

[15] “Fail2ban.” https://www.fail2ban.org/wiki/index.php/Fail2Ban (accessed Dec. 12, 2021).

[16] “Freenom,” 2013. https://www.freenom.com/es/aboutfreenom.html (accessed Jan. 09, 2022).

[17] “Let’s Encrypt.” https://letsencrypt.org/es/ (accessed Jan. 03, 2022).

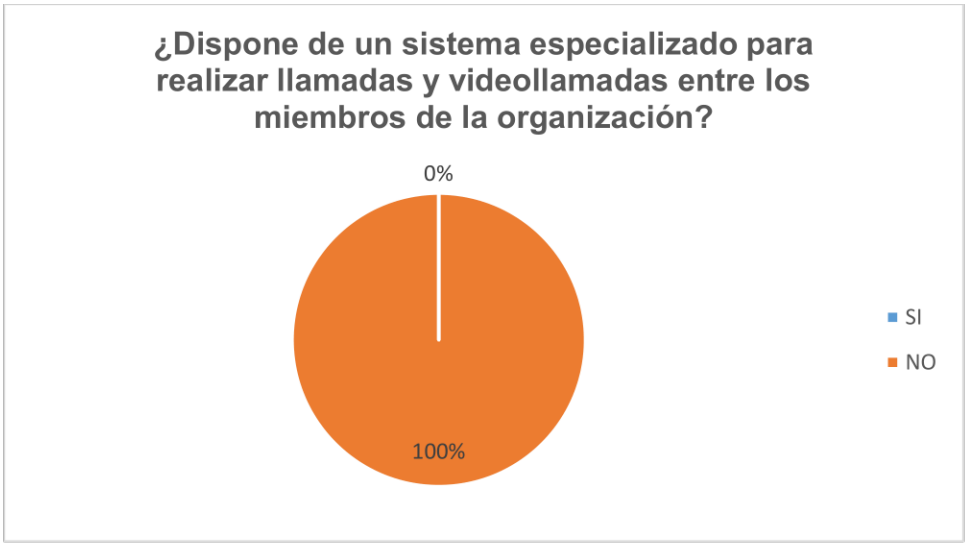
[18] A. Project, “Asynchronous Javascript Asterisk Manager (AJAM),” 2014. https://wiki.asterisk.org/wiki/pages/viewpage.action?pageId=4817256 (accessed Dec. 12, 2021).

ANEXOS

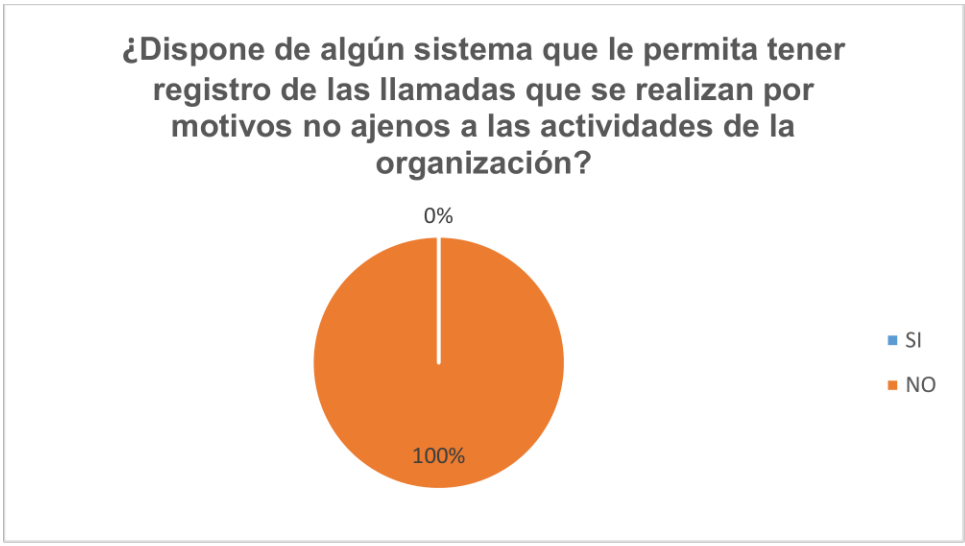
ANEXO A. Resumen de Entrevistas

A continuación, se presentan los resultados de las entrevistas realizadas para la toma de requerimientos resumidas en diagrama tipo pastel.

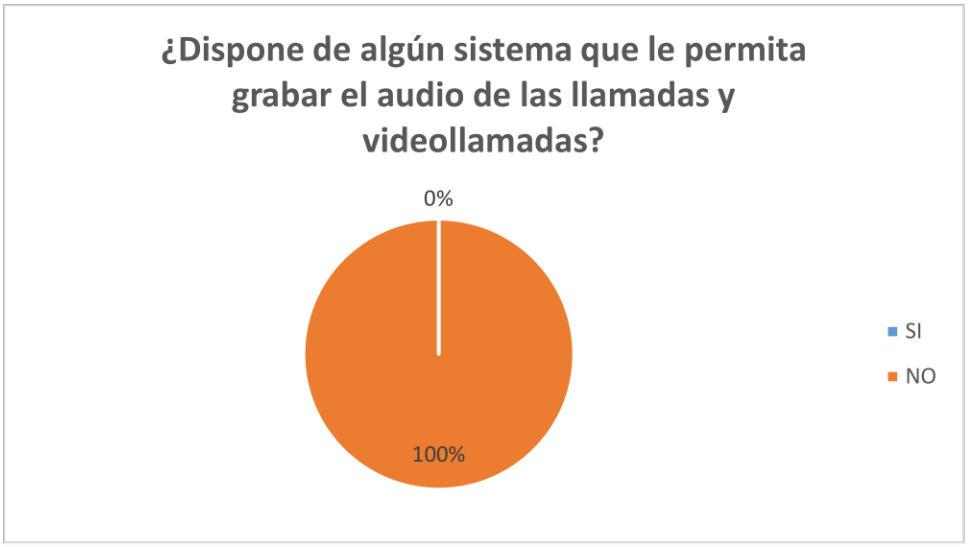
Pregunta 1. ¿Dispone de un sistema especializado para realizar llamadas y videollamadas entre los miembros de la organización?



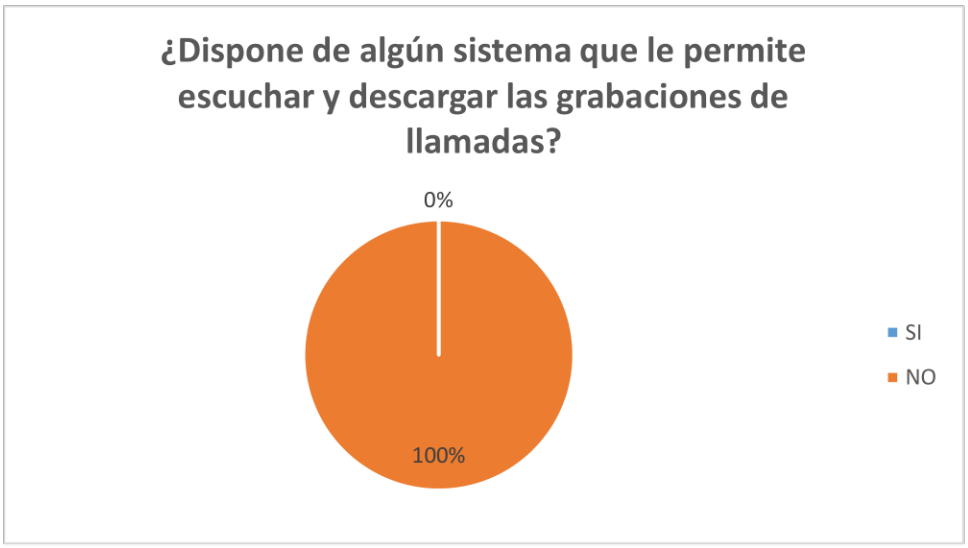
Pregunta 2. ¿Dispone de algún sistema que le permita tener registro de las llamadas que se realizan por motivos no ajenos a las actividades de la organización?



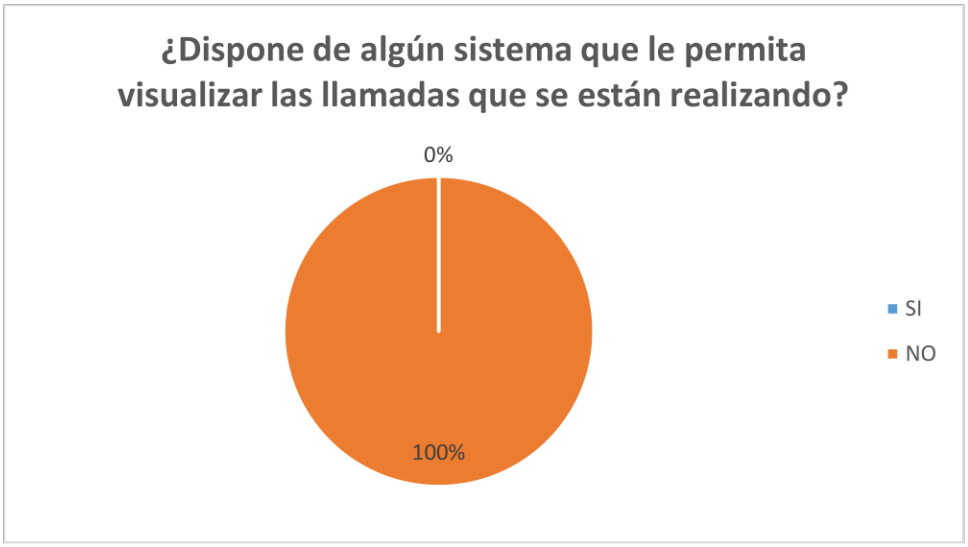
Pregunta 3. ¿Dispone de algún sistema que le permita grabar el audio de las llamadas y videollamadas?



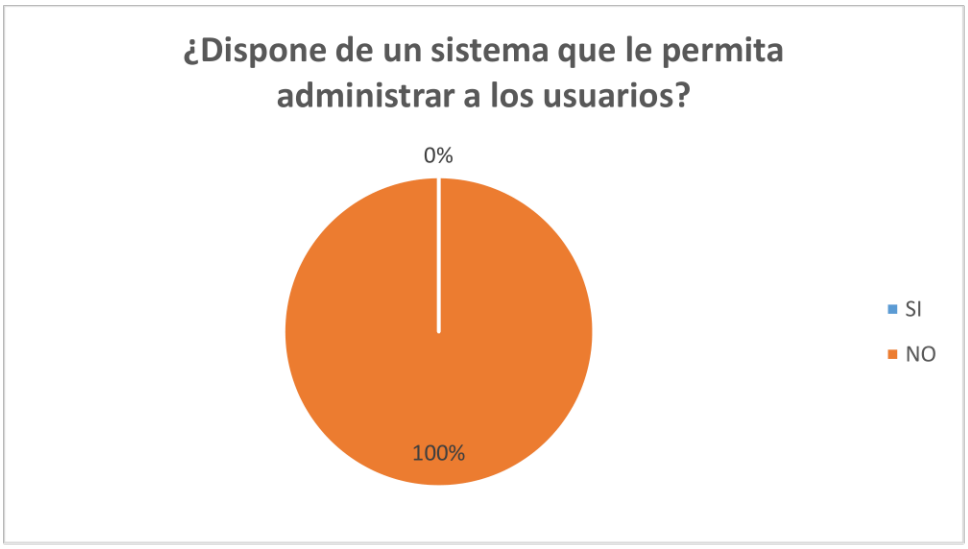
Pregunta 4. ¿Dispone de algún sistema que le permite escuchar y descargar las grabaciones de llamadas?



Pregunta 5. ¿Dispone de algún sistema que le permita visualizar las llamadas que se están realizando?



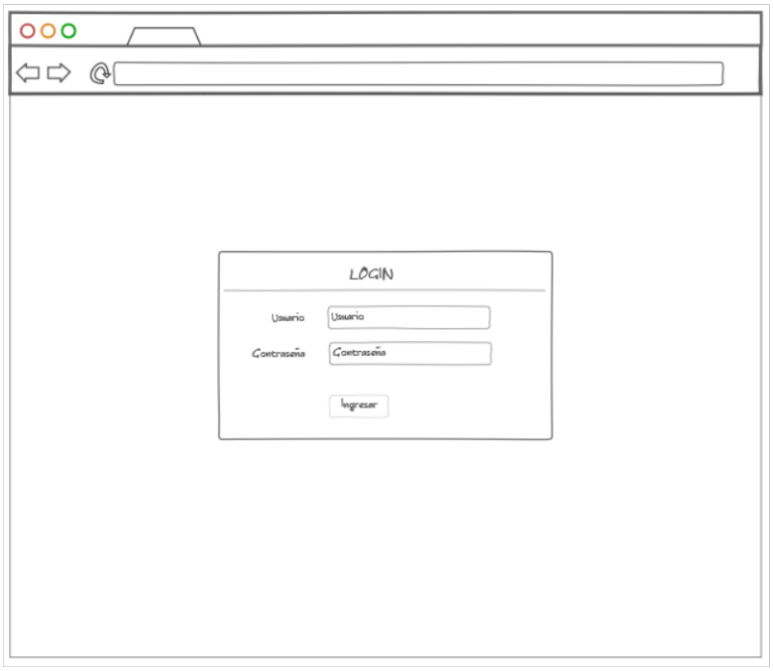
Pregunta 6. ¿Dispone de un sistema que le permita administrar a los usuarios?



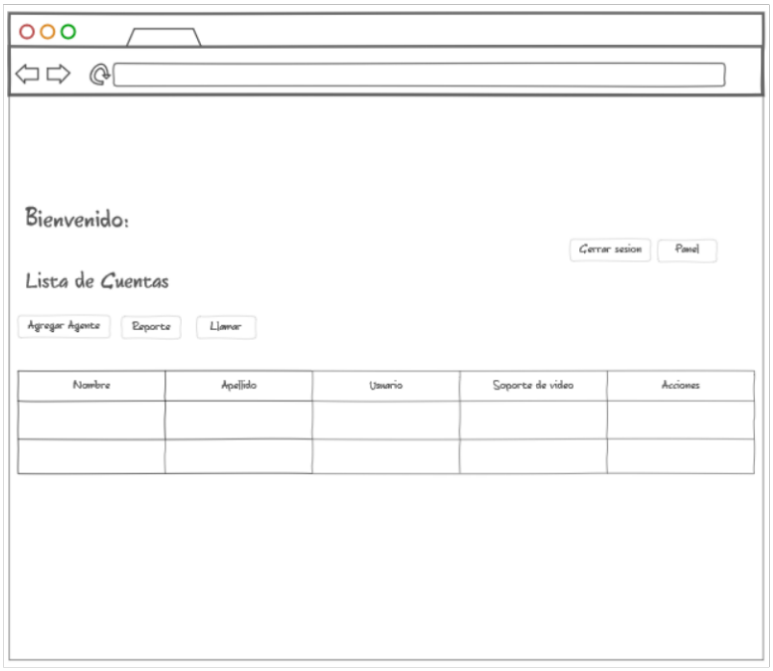
ANEXO B. Bosquejos de las Pantallas del Prototipo Web

Se presentan los bosquejos de las pantallas de usuario final que componen el prototipo web.

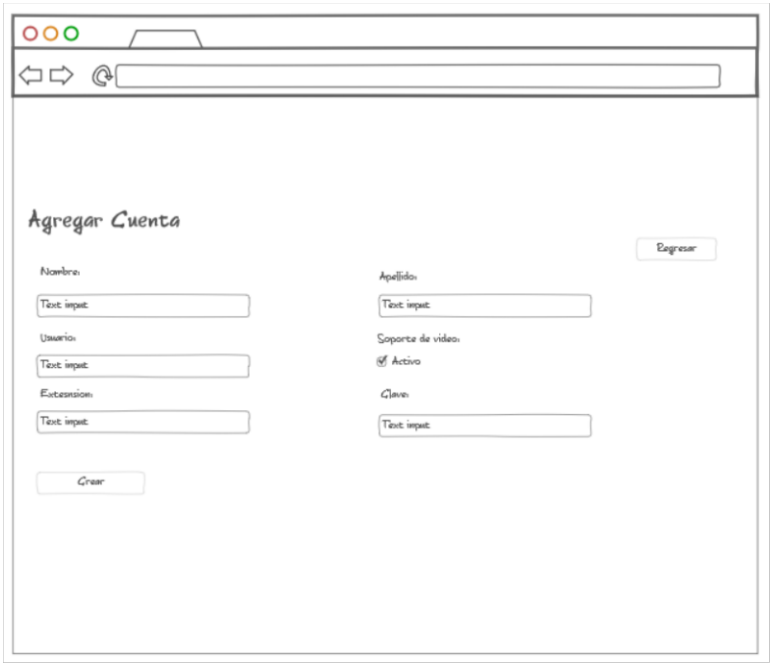
Bosquejo 1: Login. - Esta pantalla permite realizar el registro de ingreso al aplicativo web, valida las credenciales de usuario y permite ingresa tanto a la pantalla principal de administración (usuario administrador), como a la pantalla de usuario final (usuario agente).



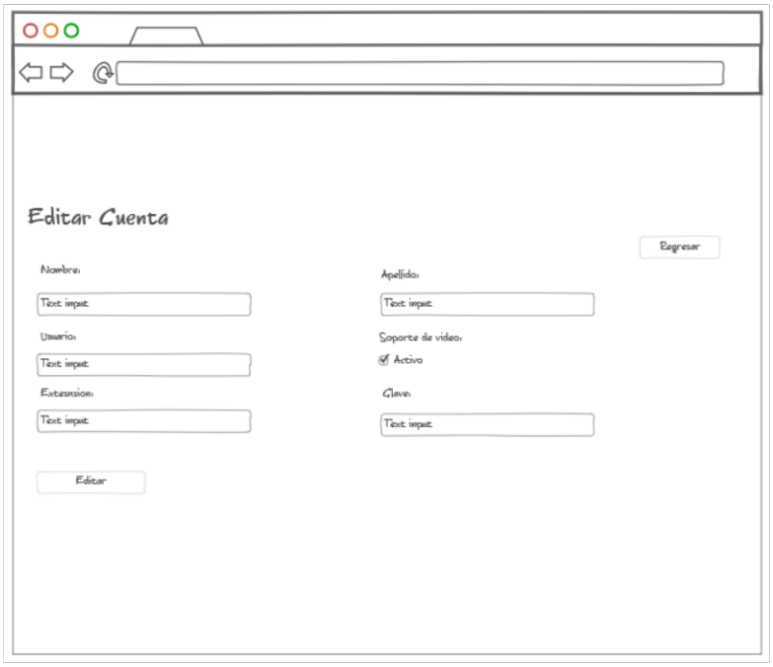
Bosquejo 2: Pantalla principal de administración. - Presenta la información de los usuarios creados en el sistema y permite ingresar al resto de pantallas de administración.



Bosquejo 3: Agregar Usuario. – Presenta la pantalla que permite ingresar la información para la creación de un nuevo usuario.

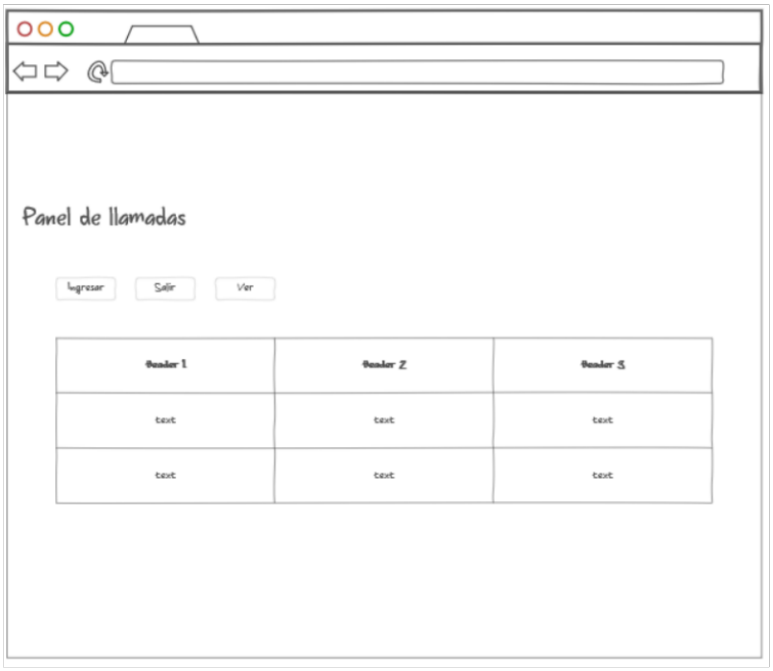


Bosquejo 3: Editar Usuario. – Presenta la pantalla que permite modificar la información de un usuario creado.



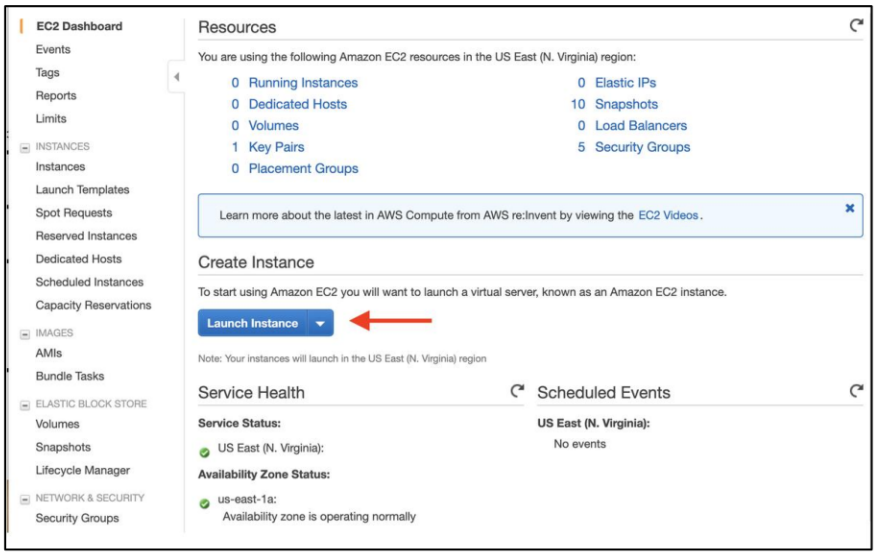
Bosquejo 4: Registros de llamadas. – Presenta la pantalla que permite visualizar las llamadas realizadas por los usuarios creados en el aplicativo. Además, descargar un reporte en Excel de estas llamadas y permite escuchar online y descargar las grabaciones de estas llamadas.

Bosquejo 5: Panel de llamadas. – Presenta la pantalla que permite visualizar las llamadas que se estén realizando en tiempo real.



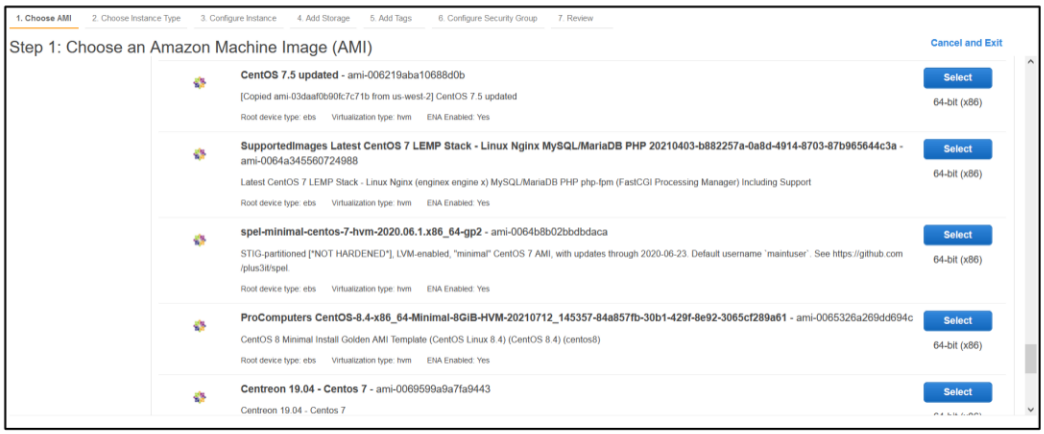
ANEXO C. Instalación de una Instancia EC2.

A continuación, se presenta un instructivo para la instalación de EC2 basado en el propuesto en la página oficial de AWS.



En la primera página, elegirá una imagen de Amazon Machine (“AMI”). La AMI elegida determinará el software de base que se instalará en la nueva instancia EC2. Esto incluye el sistema operativo y las aplicaciones instaladas en la máquina.

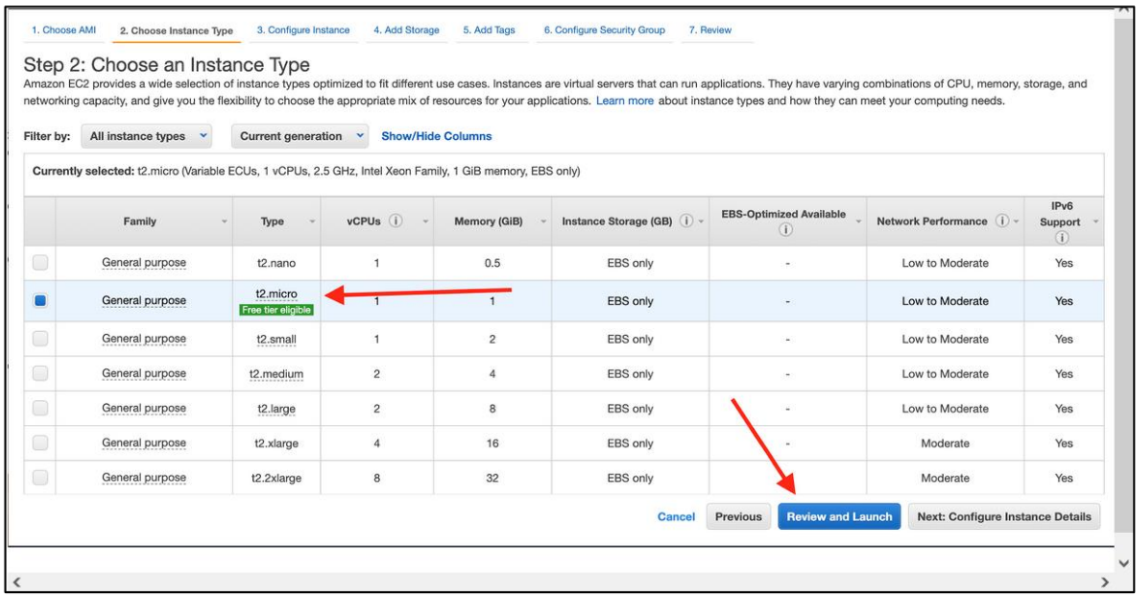
Muchas AMI son AMI de propósito general para ejecutar muchas aplicaciones diferentes, pero algunas están diseñadas específicamente para casos de uso específicos, como la AMI de aprendizaje profundo o numerosas AMI de AWS Marketplace.



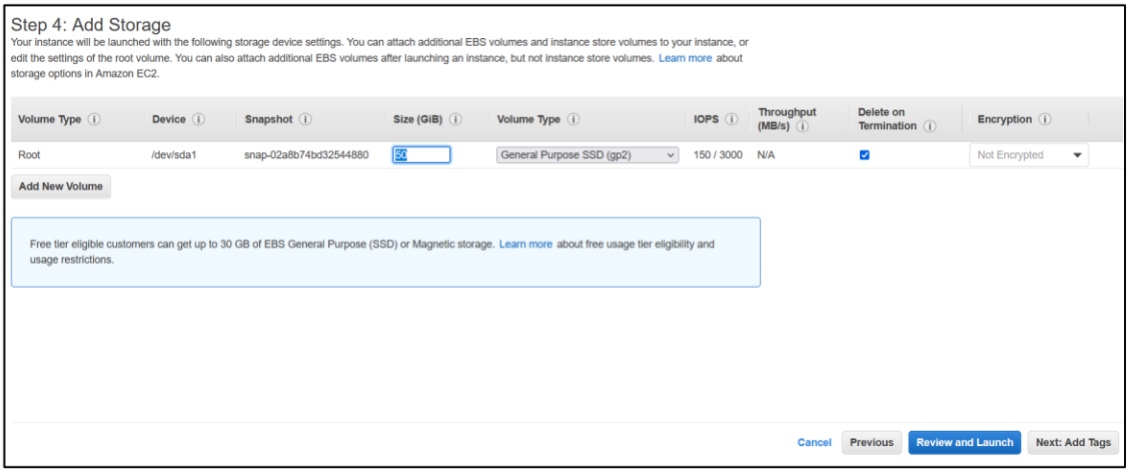
En la segunda pantalla del asistente de EC2, seleccionará un tipo de instancia EC2. Un tipo de instancia es una configuración particular de CPU, memoria (RAM), almacenamiento y capacidad de red.

AWS tiene una amplísima selección de tipos de instancias que abarca diferentes tipos de cargas de trabajo. Algunos están destinados a cargas de trabajo con un consumo de memoria intenso, como bases de datos y cachés, mientras que otros son ideales para cargas de trabajo que requieren una capacidad de cómputo elevada, como el procesamiento de imágenes o la codificación de video.

Amazon EC2 permite ejecutar 750 horas al mes de una instancia t2.micro dentro de la capa gratuita de AWS. Seleccione esta opción para este laboratorio de modo que no incurra en costos adicionales en la factura.

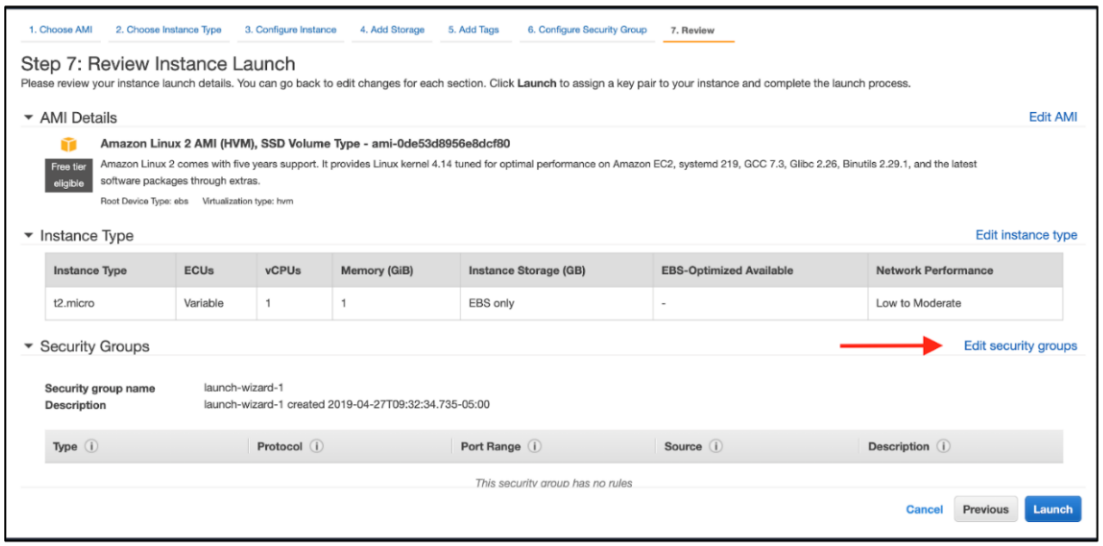


Añadir storage



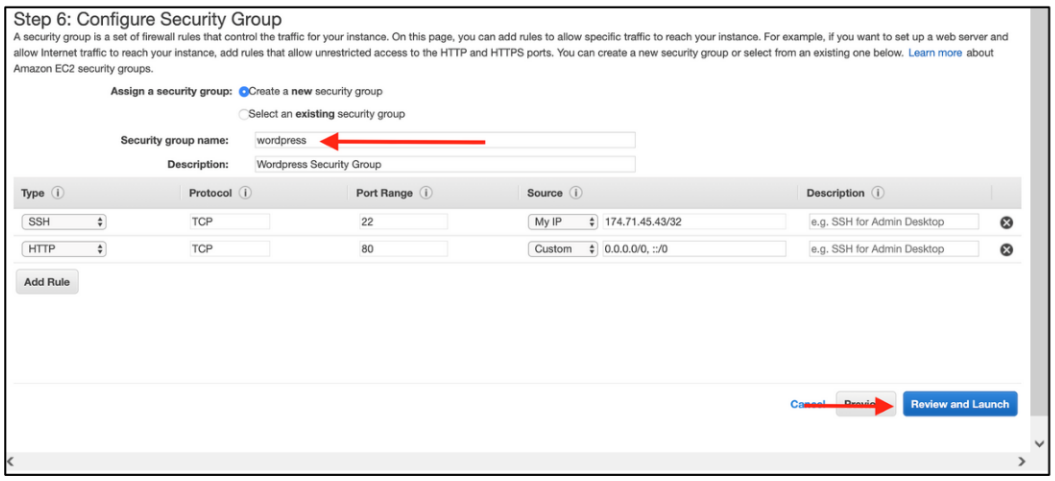
Después de hacer clic en el botón Revisar y lanzar, pasará a la pantalla Revisar lanzamiento de instancia. Es necesario configurar algo más antes de lanzar la instancia.

Los grupos de seguridad son reglas de red que describen el tipo de tráfico de red permitido en la instancia EC2. Para realizar esta configuración, haga clic en el enlace Editar grupos de seguridad en la página de revisión.



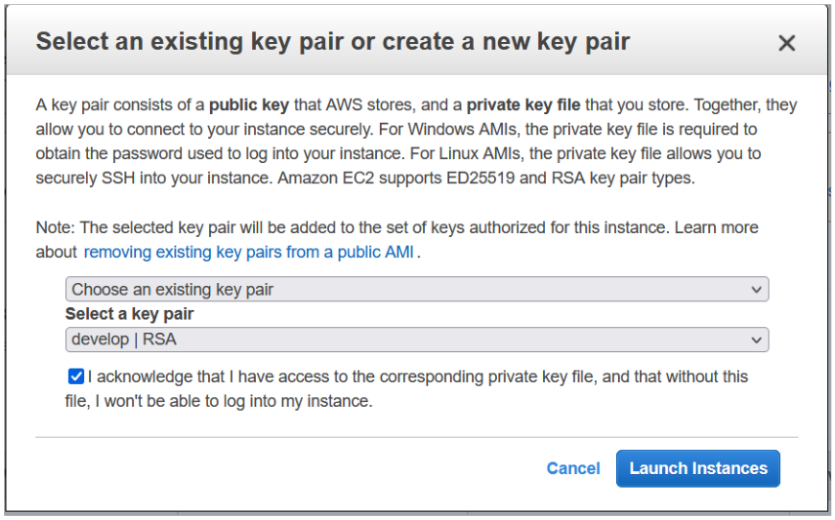
Aparecerán las reglas actuales del grupo de seguridad.

Hay una regla para SSH configurada, pero permite el acceso mediante SSH desde cualquier dirección IP. Haga clic en Origen para restringir el acceso a su dirección IP actual.



Aparecerán detalles que le indicarán cómo configurar un par de claves para la instancia. Usará el par de claves para conectar mediante SSH con la instancia, de forma que pueda ejecutar comandos en el servidor.

Cree un nuevo par de claves para la instancia y asígnele un nombre. Después, haga clic en el botón Descargar par de claves para descargar el archivo .pem en su equipo, ya que lo utilizará para conectarse remotamente a la instancia.



ANEXO D. Codificación de la Capa de Datos.

En este anexo se presenta parte de la codificación de la capa de datos que por su longitud no ha sido expuesta directamente en el capítulo correspondiente. Todo el código de la Capa de Datos estará adjunto en el archivo comprimido.

En el siguiente código se observa la creación de la vista sys_accounts_v, la cual nos sirve para presentar la información de los usuarios en la pantalla principal de administración. Entre la línea 3 y 17 vemos la creación de la vista como una consulta realizada a las tablas sys_users, ast_sipchannels y ast_sipregistrations y ordenada por el número de extensión de manera descendente.

```
1  CREATE VIEW public.sys_accounts_v AS
2   SELECT a.name,
3      a.lastname,
4      a.username,
5      a.id AS id_account,
6      a.extension,
7      b.allow,
8       CASE b.allow
9          WHEN 'alaw;ulaw;gsm;opus;g729;h264;vp8;mpeg4' THEN 'yes'
10         ELSE 'no'
11      END AS codec,
12     c.useragent,
13     b.md5secret
14    FROM ((public.sys_users a
15      JOIN public.ast_sipchannels b ON ((a.extension = b.name)))
16      JOIN public.ast_sipregistrations c ON ((b.name = c.name)))
17   ORDER BY a.extension;
```

A continuación, se presenta el código correspondiente a la creación de la función login que permite verificar si las credenciales ingresadas son correctas. Recibe tres parámetros de entrada declarados entre las líneas 6 a 8 y retorna una tabla con dos columnas (línea 2). En la línea 13 se prepara la contraseña ya que está compuesta de un hash entre la extensión la clave del usuario y la palabra develop (Realm de Asterisk). En la línea 16 se retorna el resultado de la consulta realizada a la vista sys_users_v donde se filtra por el nombre de usuario y su contraseña.

```
1  CREATE FUNCTION public.sys_login(text, text, text)
2  RETURNS TABLE(uname text, rolname text)
3      LANGUAGE plpgsql
4      AS $_$
5  DECLARE
6          uname ALIAS FOR $1;
7          pass ALIAS FOR $2;
8          ext ALIAS FOR $3;
9          md5Pass text;
10
11 BEGIN
12         --Prepare secret
13         md5Pass = concat(ext,':','develop',':',pass);
14
15         --Consultando
16         return query select username, rol from sys_users_v
17         where username=uname and md5secret = md5(md5Pass);
18
19 END;
20 $_$;
```

ANEXO E. Codificación de Clases.

En este anexo se presenta parte de la codificación de clases que por su longitud no ha sido expuesta directamente en el capítulo correspondiente. Todo el código correspondiente a las clases estará adjunto en el archivo comprimido.

En el siguiente código podemos observar tres funciones de la clase database.php. Entre la línea 2 a 6 se tiene a la función single_recordnew que permite obtener toda la información de un usuario por medio de su id. Entre la línea 9 a 14 se presenta la función current_user que permite obtener información como el nombre, apellido y extensión de un

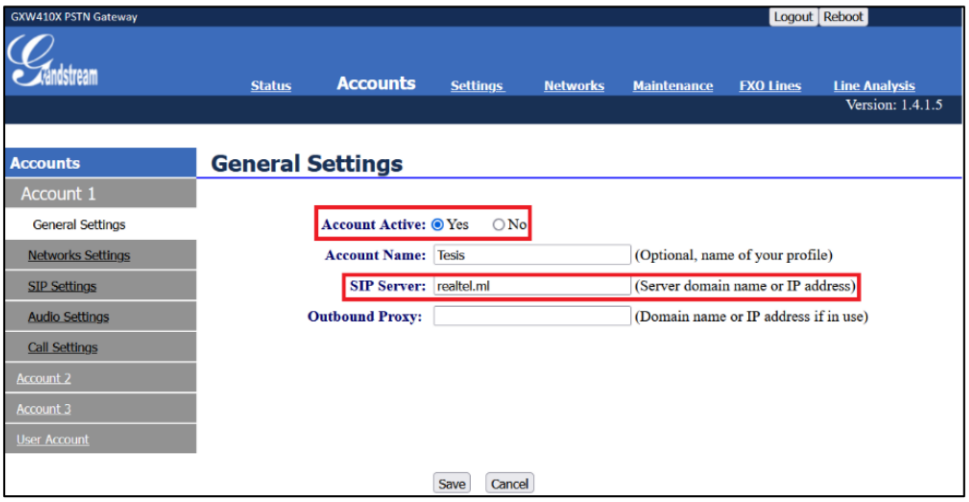
usuario por medio de su nombre de usuario. Las líneas entre la 17 a 22 muestran la función view_phonebook que trae toda la información necesaria para crear el directorio dentro de la pantalla de un usuario con rol de agente.

```
<?php
public function single_recordnew($id){
        $sql = "SELECT * FROM sys_accounts_v where id_account='$id'";
        $res = pg_query($this->con, $sql);
        $return = pg_fetch_object($res);
        return $return ;
}

public function current_user($uname){
        $sql = "SELECT name,lastname,username,extension
                FROM sys_accounts_v where username='$uname'";
        $res = pg_query($this->con, $sql);
        $return = pg_fetch_object($res);
        return $return;
}

public function view_phonebook($uname){
        $sql = "SELECT name,lastname,username,extension
                FROM sys_accounts_v where username != '$uname'";
        $res = pg_query($this->con, $sql);
        return $res ;
}
```

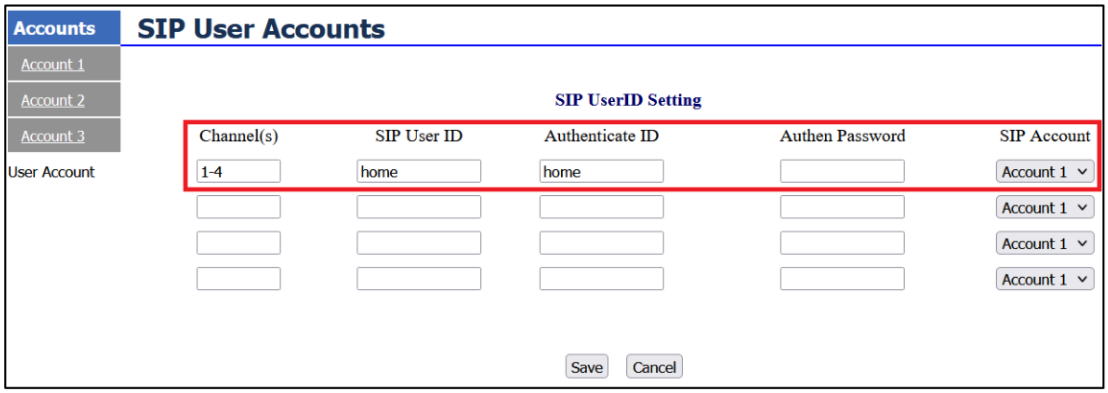
ANEXO F. Configuración Gateway FXO.

Se presenta la configuración de la cuenta SIP y de la línea analógica para la comunicación con la PSTN.

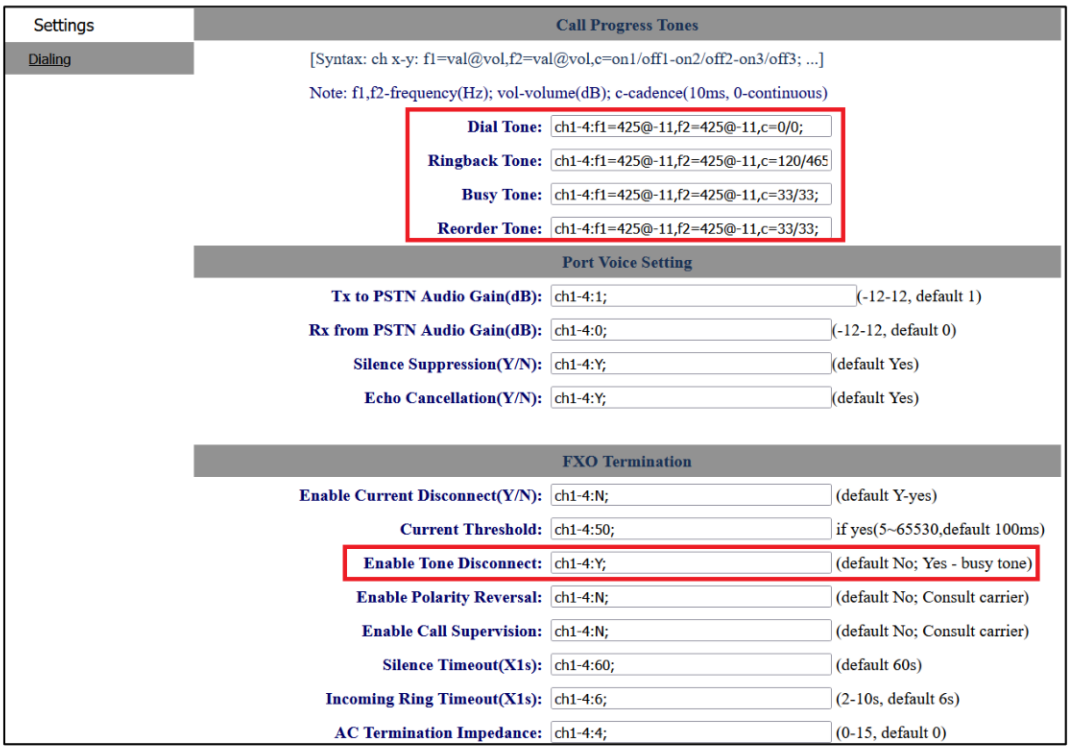
El equipo por utilizar es un Grandstream GXW410X PSTN Gateway. Lo primero que se debe configurar es la cuenta SIP, para esto se debe ingresar en Accounts > General Settings e ingresar la IP o dominio del servidor SIP. Una vez se haya ingresado se debe activar la cuenta y se guarda los cambios.



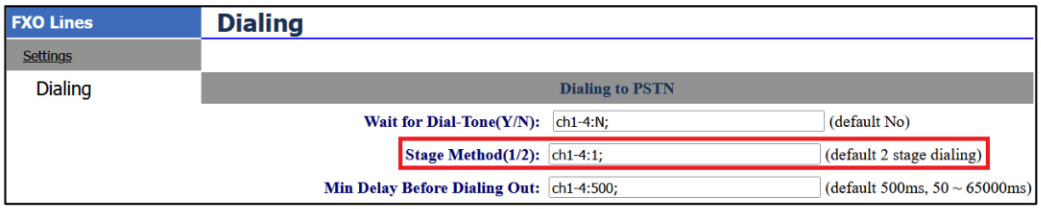
En este caso vamos a configurar al equipo como un cliente SIP que se registrará en nuestro servidor SIP, este tipo de configuración es útil ya que no requiere que se realice una redirección de puertos y no es necesario que el equipo este expuesto directamente al Internet. Se debe ingresar el nombre de la cuenta y la contraseña creadas en el servidor SIP (Asterisk, archivo sip.conf. Peer home).



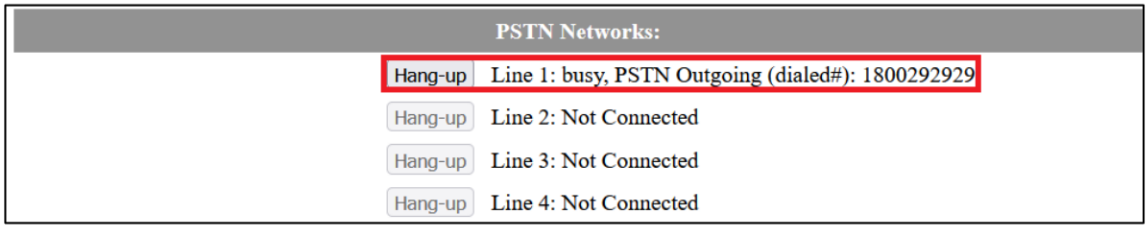
Una vez se haya creado la cuenta SIP y registrado se debe configurar las cadencias y frecuencias correspondientes a la línea analógica. Para esto nos podemos guiar en las recomendaciones expuestas por la IUT-T E.180/Q.35, sin embargo, éstas pueden cambiar de acuerdo con el proveedor de la línea. También es importante activar el tono de desconexión para evitar inconvenientes en el colgado.



Es importante cambiar el método de etapa a 1 en el apartado Dialing para poder realizar llamadas externas por la línea analógica. Una vez se hayan realizados estos ajustes es necesario guardar y reiniciar el equipo.



En la siguiente figura se observa una llamada saliente realizada por la línea 1, que en este caso es la única que se tiene conectada al equipo.



Anexo G. Video demostrativo

Enlace a video demostrativo donde se expone cada uno de los puntos descritos en las pruebas de funcionamiento.

https://1drv.ms/v/s!AqxLj10rWv3t6Epnq4bgFJrJJb_z?e=1Ij6RZ