ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

PROTOTIPO DE CENTRAL TELEFÓNICA PARA ENTORNOS DOMÉSTICOS

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

JUAN MANUEL BASTIDAS GUAYASAMÍN

juan08.man.bas@gmail.com

DIRECTOR: ING. ANA RODRIGUEZ MSc.

ana.rodriguez@epn.edu.ec

CO DIRECTOR: ING. CHRISTIAN TIPANTUÑA MSc.

christian.tipantuna@epn.edu.ec

Quito, julio 2016

DECLARACIÓN

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

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

_____

Juan Manuel Bastidas Guayasamín

CERTIFICACIÓN

Certificamos que el presente trabajo fue desarrollado por Juan Manuel Bastidas Guayasamín, bajo nuestra supervisión.

_____

Ing. Ana Rodríguez MSc.
DIRECTOR DEL PROYECTO

_____

Ing. Christian Tipantuña MSc.
CO DIRECTOR DEL PROYECTO

AGRADECIMIENTO

En primer lugar agradezco a Dios por todas las bendiciones que me ha concedido y ha sabido guiar mi vida. Agradezco a mis padres Galo y Anita por todo su amor, comprensión, cariño y apoyo en cualquier decisión. También por haberme dado la oportunidad de educarme.

Agradezco a mi hermano Galo Andrés por su compañía y sus palabras de apoyo brindadas en los buenos y malos momentos, de igual manera a mi tía Amparito por su cariño y tenerme siempre en sus oraciones a mí y a mi familia.

Un agradecimiento muy especial a la Ing. Anita Rodríguez, directora del proyecto, al Ing. Christian Tipantuña codirector del mismo y al Ing. José Antonio Estrada, por la guía que me han dado y por el tiempo de dedicación que muy gentilmente me han concedido para la realización del presente proyecto.

Además me gustaría agradecer al Ing. Diego Guacho por sus consejos y aportes de ayuda durante las etapas del proyecto, de igual manera a los funcionarios de la Dirección Nacional de Investigación Especial (DIE) de la Agencia de Regulación y Control de las Telecomunicaciones (ARCOTEL), de manera especial a María Fernanda Negrete, Anita Jácome, Roberto Pérez y Diego Noboa.

Finalmente quiero agradecer a mis maestros de la Escuela Politécnica Nacional por haberme transmitido sus conocimientos y a todos los amigos que conocí en la universidad con quienes compartí muchas experiencias inolvidables durante esta etapa de la vida, y con los cuales aprendí que el trabajo en equipo nos lleva a conseguir grandes metas.

DEDICATORIA

Dedico la realización de este proyecto a mi padre Galo por su ejemplo de padre trabajador, cariñoso y sumamente responsable; y mi madre Anita por su dedicación, amor y preocupación para conmigo.

CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE CÓDIGOS

RESUMEN

El presente proyecto de titulación está conformado de cuatro capítulos. En el primer capítulo se realiza una descripción de los conceptos básicos relacionados con la Telefonía IP. Dicha descripción abarca los componentes típicos que existen en un entorno que utiliza Telefonía IP, los servicios que se pueden ofrecer y las herramientas Open Source más conocidas empleadas para su implementación.

Además se incluye una revisión de las características de Raspberry Pi, dispositivo que ha sido utilizado para desempeñar el papel de servidor de Telefonía IP del prototipo, para concluir se describen algunos conceptos sobre el modelo incremental, la metodología de desarrollo XP y el patrón de diseño Modelo Vista Controlador *(MVC)*, utilizados como referencia para el desarrollo de la aplicación de Gestión de Telefonía IP.

El segundo capítulo se encuentra dividido en dos partes, la primera parte comprende el análisis de requerimientos de los servicios de telefonía IP para un entorno doméstico, así como también los requisitos de la aplicación que se necesitaron para desarrollar el prototipo. Finalmente, la segunda parte de este capítulo describe el diseño del prototipo.

El tercer capítulo corresponde a la implementación del prototipo desarrollado, para lo cual se ha divido la implementación a nivel de hardware y la implementación de la aplicación web divida en los niveles de Presentación, Lógica de Negocios y Datos.

A continuación, se presentan las pruebas realizadas en el prototipo entre las cuales se tiene: pruebas de funcionamiento del mismo, pruebas de uso de memoria y CPU en Raspberry Pi, pruebas unitarias y de aceptación de la aplicación, y finalmente las pruebas de usabilidad de la aplicación web.

El capítulo final corresponde a las conclusiones y recomendaciones que se han obtenido al culminar el proyecto.

PRESENTACIÓN

A pesar de la exorbitante utilización del servicio de Internet en los hogares y de la existencia de nuevos servicios de comunicación basados en mensajería, la comunicación telefónica convencional continúa siendo muy utilizada, especialmente desde el hogar, ya que su costo suele ser más reducido que el de la telefonía móvil.

El servicio telefónico implica un egreso económico para una familia y por ello es conveniente poder gestionarlo para que se use adecuadamente. Del mismo modo, al ser una puerta hacia el exterior, el contacto que puedan tener miembros vulnerables de la familia mediante el servicio telefónico, es siempre una preocupación. Esta gestión se implementa comúnmente en entornos corporativos, pero requiere de personal técnico que la administre y un egreso económico importante que quizás una familia no lo pueda afrontar.

En base a estos argumentos, se propone el diseño y desarrollo de un prototipo de Central Telefónica que permita gestionar la comunicación telefónica doméstica. Para ello, se plantea el uso del computador de placa reducida Raspberry Pi, que tiene un costo muy reducido para las capacidades de procesamiento y memoria que ofrece. Además, se utilizó herramientas de software libre como base del prototipo, lo que evita que se tenga que pagar licencias por su utilización.

Con la finalidad de ofrecer una interfaz usable a personas no técnicas, el prototipo consta de una aplicación web para la gestión del servicio de telefonía. Dicha aplicación podría permitir gestionar de manera relativamente sencilla el tráfico de voz en entornos domésticos, con el fin de ser una herramienta de control a la que tienen acceso los padres de familia en el hogar.

CAPÍTULO 1

1 MARCO TEÓRICO

1.1 CONCEPTOS BÁSICOS DE LA TELEFONÍA IP

1.1.1 TELEFONÍA IP

La telefonía IP es un concepto que se relaciona con la VoIP (*Voice Over IP*), la cual es una tecnología que transmite audio por la red IP, el término telefonía IP es utilizado para explicar la convergencia de VoIP con servicios tales como fax y multimedia sobre IP [1], además relaciona otros aspectos como protocolos de transporte y diseño del plan de marcación.

1.1.2 COMPONENTES DE UN SISTEMA DE TELEFONÍA IP

Los componentes de telefonía IP pueden ser variados, dependiendo del escenario donde se quiera levantar este servicio. A continuación se especificarán los elementos más comunes.

1.1.2.1 Servidor o Central Telefónica [2], [3]

Es el integrante más importante, puesto que se encarga de la inteligencia del sistema de telefonía IP. La Central IP debe almacenar las direcciones IP asociadas a los equipos terminales y hacer el mapeo respectivo.

Una Central Telefónica tradicional cumple con las siguientes funciones:

- Establecer la conexión extremo-extremo entre dos usuarios, ya sean internos (usuarios de la misma red) o un usuario interno y un externo; el usuario externo puede ser otra Central PBX.

- Supervisar el circuito para detectar las peticiones de las llamadas, responderlas, realizar las conexiones, tareas de señalización y por último el colgado.

- Deshacer la ruta o el enlace entre los dos usuarios, es decir liberar el recurso asignado anteriormente.

- En la actualidad, las centrales IP realizan las mismas tareas, para lo cual interactúan con otros servicios que les permitan encaminar la llamada.

1.1.2.2 Equipos terminales

Los equipos terminales son los elementos que interactúan con los usuarios, como es el caso de un teléfono IP, y otros que son automáticos como las contestadoras; en ambos casos el dispositivo es el punto final que termina la llamada y los media streams[1].

1.1.2.3 Gateway

Es un dispositivo que se encarga de la conversión de un protocolo de señalización a otro, como por ejemplo, la conversión de SIP a ISDN. Lo más común es utilizar el gateway para permitir la comunicación con la PSTN (*Public Switched Telephone Network*) [2].

1.1.2.4 Bridge de Conferencia

Permite que varias personas participen en una llamada telefónica; el más claro ejemplo es cuando se permite a los usuarios marcar desde su propio teléfono en una reunión virtual [4].

1.1.2.5 Direccionamiento

En la telefonía IP el direccionamiento es un mecanismo para identificar a los terminales o usuarios utilizando los URIs (*Uniform Resource Identifiers*) o los números.

---

[1] Media Stream: Flujos de información como videos o imágenes

Los URIs definidos en RFC 2396 permiten identificar un espacio de nombres registrado independiente de la ubicación; se caracterizan por ser fáciles de recordar para un usuario, pero puede resultar complicada la marcación en algunos dispositivos.

Otra forma de identificación es mediante la enumeración de E.164, referido al plan de marcación internacional de telecomunicación, el cual incluye hasta 15 dígitos numéricos y un signo más (+), dicho símbolo es sustituido por el código para llamadas internacionales 00 [2].

1.1.3 PRINCIPALES SERVICIOS DE LA TELEFONÍA IP [5]

1.1.3.1 Llamada en Espera

El servicio de Llamada en Espera permite a un usuario aguardar unos minutos antes de que pueda conversar con el destino que marcó, se caracteriza porque el servicio corta la transmisión de audio entre los dos elementos de la comunicación, sin embargo mantiene la conexión entre ellos.

1.1.3.2 Transferencia de Llamada

La transferencia de llamada permite redirigir la llamada, es decir, si originalmente se tenía una conversación entre dos UA (*User Agent*) A y B, se realiza un cambio de sesión para hacer una nueva conexión entre B y C.

1.1.3.3 Mensajes en Espera

El mensaje en espera es un servicio que se puede usar durante una llamada telefónica para atender otra, permitiendo generar un mensaje audible a quien lo escucha, el servicio puede indicarse de varias maneras, como por ejemplo generando tonos intermitentes en el teléfono o con luces parpadeantes de algunos teléfonos IP.

1.1.3.4 Grabación de Llamadas

La grabación de llamadas es un servicio muy útil que puede ser usado para el control de calidad de un servicio o con fines de entrenamiento en ambientes

empresariales, el servicio puede ser de gran utilidad permitiendo tener datos de la grabación como la fecha y la hora de cuando fue realizada la llamada, así como también el número del empleado en la misma.

1.1.4 HERRAMIENTAS DISPONIBLES PARA CENTRALES IP

1.1.4.1 Asterisk

Asterisk es un software de tipo *Open Source*[2], flexible en lo que respecta al manejo de la telefonía IP, se encuentra compuesto por varios módulos los cuales pueden ser usados o removidos según sea la necesidad del sistema telefónico que se desee implementar. Desde otro punto de vista, Asterisk es una Central Telefónica privada, la cual permite conectar una o más líneas telefónicas con otras [6].

La manera de funcionamiento de Asterisk permite dar solución a dos problemas como son la incompatibilidad y la funcionalidad limitada, los cuales pueden ser dos problemas presentes en sistemas de carácter propietario [7].

Asterisk presenta algunas características como:

- Llamada Extensión-Extensión
- Línea Troncal
- Registros de Llamadas
- Grabación de Llamadas
- Interrupción de Llamadas
- Correo de Voz
- IVR (*Interactive Voice Response*)

[2] Open Source: Software de código abierto, desarrollado y distribuido libremente, puede ser alterado o modificado con total libertad.

1.1.4.1.1 Llamadas Extensión-Extensión [6]

Asterisk permite hacer una llamada extensión-extensión, lo que significa que un usuario puede marcar de un teléfono a otro mediante un número denominado extensión.

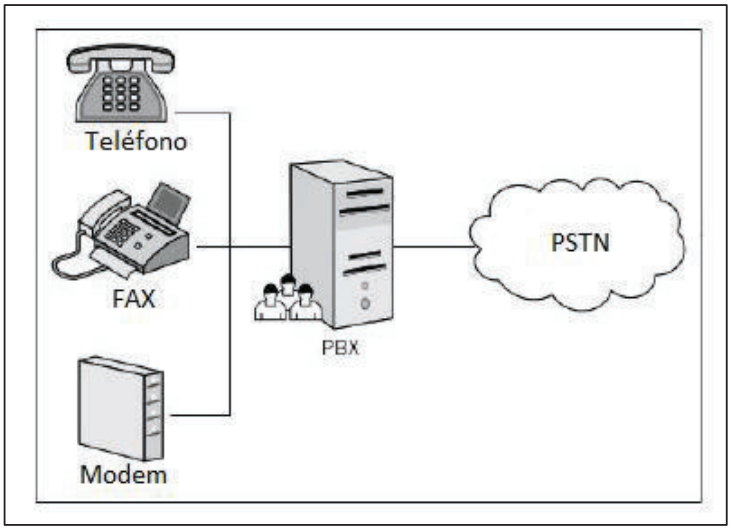

Figura 1.1 Central Telefónica [6]

En la **Figura 1.1** se muestra tres dispositivos conectados a la Central Telefónica, los cuales pueden llamarse unos a otros identificados por una extensión, para lo cual la central crea el enlace directo entre ambos dispositivos.

1.1.4.1.2 Línea Troncal [6]

Una línea troncal comparte el acceso para múltiples líneas telefónicas, por lo general se emplea para conectar a la PSTN (*Public Switched Telephone Network*). Dichas conexiones pueden ser: una troncal analógica, varias troncales analógicas o un enlace digital de alta capacidad.

1.1.4.1.3 Registros de Llamadas [6]

Asterisk mantiene un conjunto de registros sobre las llamadas realizadas en la red telefónica denominados CDRs[3] (*Call Detail Records*). Los CDRs permiten tener un control sobre la red telefónica puesto que muestran información de gran utilidad

[3] CDR: Call Detail Records, son los registros de llamadas que maneja Asterisk.

como la fecha, hora, duración de la llamada, extensiones involucradas en la llamada, entre otras.

1.1.4.1.4 Grabación de Llamadas [6]

La grabación de las llamadas es una de las características de Asterisk, que permite grabar las llamadas que se producen en la red telefónica. Un ejemplo de aplicación de este servicio es una llamada en un ambiente cliente-vendedor.

1.1.4.1.5 Interrupción de llamadas [6]

Esta característica permite interrumpir la comunicación entre dos usuarios para que un tercero pueda comunicarse con uno de los participantes sin perder la comunicación entre ellos. Por ejemplo, este servicio podría utilizarse cuando un empleado y un cliente mantienen una conversación y otra persona, como un supervisor, podría interrumpir la comunicación para asesorar a su empleado sin la necesidad de que el cliente lo escuche o se entere de la situación.

1.1.4.1.6 IVR (Interactive Voice Response) [6]

Un IVR es un mensaje o una grabación que permite reproducir la Central Telefónica, para interactuar con el usuario que se comunica con la red telefónica. El IVR puede ser personalizado y es de gran utilidad cuando los clientes llaman a pedir información o hacen consultas y no es necesario una atención personalizada.

1.1.4.1.7 Correo de voz [6]

El correo de voz, que proporciona Asterisk, es un servicio en el cual cada usuario que pertenece a la Central Telefónica puede tener un buzón donde le lleguen los mensajes de voz. Asterisk puede informar o notificar a los usuarios las llegadas de dichos mensajes.

Los correos de voz pueden ser configurados en el archivo **voicemail.conf** de Asterisk, donde se configuran parámetros como la cantidad máxima de mensajes a guardar, el sitio donde guardarlos, entre otros parámetros.

La configuración del archivo **voicemail.conf** comprende una parte de la configuración total del servicio de correo de voz, para que el servicio este completamente funcionando, se debe vincular el servicio con los canales y el plan de marcado configurados en los archivos **sip.conf** o **iax.conf** y el archivo **extensions.conf.**

1.1.4.1.8 Plan de Marcado

El plan de Marcado es la parte fundamental de Asterisk debido a que define cómo fluyen las llamadas en la red telefónica, por medio de este plan se puede atender a todas las llamadas que llegan a la Central Telefónica y dirigirlas al destino adecuado conforme lo indique el plan. El plan de marcado pude escribirse de tres maneras que son:

- Manera tradicional, utilizando el archivo **extensions.conf**
- Utilizando AEL (*Asterisk Extension Logic*), configurando el archivo ***extensions.ael***
- Utilizando LUA, configurando el archivo **extensions.lua** [7]

El plan de marcado básico que se encuentra dentro del archivo **extensions.conf** consiste en un script[4] que se basa en cuatro conceptos fundamentales: Contextos, Extensiones, Prioridades y Aplicaciones.

1.1.4.1.8.1 Contextos

Los contextos son las partes en las cuales se divide el plan de marcado. Dentro de cada contexto existen extensiones las cuales son aisladas de las extensiones de otro contexto y pueden interactuar o comunicarse mediante algunas aplicaciones como **Goto**[5] o **Gotolf**.[6]

[4] Script: Programa pequeño escrito en un archivo comúnmente escrito en texto claro.
[5] GoTo: Aplicación de Asterisk que permite realizar saltos dentro del plan de marcado.
[6] Gotolf: Aplicación de Asterisk usada para saltar de un sitio del plan de marcado a otro si cumple una condición.

La sintaxis de un contexto consiste en un nombre encerrado entre corchetes como indica la **Figura 1.2**, donde el nombre de dicho contexto puede incluir letras de A-Z, mayúsculas o minúsculas e incluso números.

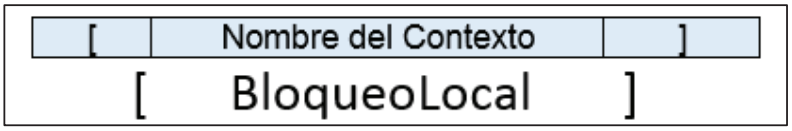

Figura 1.2 Sintaxis de un Contexto

1.1.4.1.8.2 Extensiones

Una extensión puede representar un número marcado dentro de una llamada telefónica, pero debido a la naturaleza del plan de marcado, una extensión puede representar una secuencia de pasos dentro de un contexto que se ejecutan al marcar dicho número.

La sintaxis de una extensión consta de cuatro partes, la primera es la palabra **exten** seguida de los símbolos igual y mayor (**=>**), a continuación se coloca el número de la extensión, la prioridad y la aplicación separadas por comas, como se observa en la **Figura 1.3**.

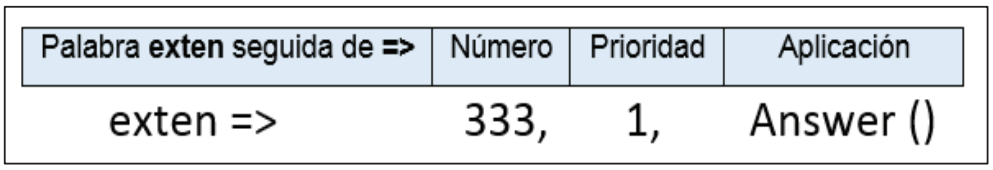

Figura 1.3 Sintaxis de una Extensión

1.1.4.1.8.3 Prioridades

La prioridad indica el orden en el cual se van desarrollando las extensiones, se encuentran enumeradas desde el número 1; en muchas ocasiones se pueden tener grandes cantidades de pasos y añadir uno puede ocasionar conflictos con la numeración, para lo cual es recomendable utilizar la letra n desde el segundo paso como lo muestra la **Figura 1.4**.

```
exten =>333,1,Answer()
exten =>333,n,Hacer algo!
exten =>333,n,Hacer algo!
exten =>333,n,Hangup()
```

Figura 1.4 Ejemplo de uso de Prioridades

1.1.4.1.8.4 Aplicaciones

Las aplicaciones permiten realizar tareas específicas dentro de cada instrucción en el plan de marcado como por ejemplo reproducir un sonido; la mayoría de las aplicaciones requieren una información adicional llamada argumentos, los cuales van entre paréntesis. Las aplicaciones más comunes son **Answer()**[7], **Hangup()**[8] y **Playback(nombre del archivo de sonido)**[9].

1.1.4.1.9 Canales de comunicación

Asterisk permite la comunicación entre dispositivos mediante muchos protocolos; sin embargo, los más comunes son SIP e IAX2, para los cuales se deben configurar los archivos **sip.conf** o **iax.conf**. Dentro de cada archivo existe la configuración para cada dispositivo terminal que va a pertenecer a la red telefónica. El archivo se divide en secciones donde existe información del tipo de controlador de cada canal.

Cada sección de estos archivos inicia con el nombre de la sección entre corchetes y todo lo que se encuentra a continuación representa las características del canal, tales como el contexto del plan de marcado al que pertenece o el identificador del dispositivo telefónico. En la **Figura 1.5** se muestra un ejemplo de configuración de un archivo **sip.conf**.

---

[7] Answer(): Aplicación de Asterisk que permite contestar un canal que está timbrando.
[8] Hangup(): Aplicación de Asterisk que termina o cuelga el canal activo
[9] Playback(): Aplicación de Asterisk utilizada para reproducir un sonido, dentro de los paréntesis se coloca el nombre del archivo sonido.

```
[general]
context=unauthenticated        ; contexto por defecto para las llamadas entrantes
allowguest=no                  ; evitar llamadas no autenticadas
srvlookup=yes                  ; habilitar busqueda del registro DNS SRV en llamadas salientes
udpbindaddr=0.0.0.0            ; escuchar peticiones UDP en todas las interfaces
tcpenable=no                   ; deshabilitar soporte TCP

[Juan]          ;seccion del terminal Juan
type=friend     ;indica el controlador del canal que asocia el nombre del Terminal y la IP
context= Juan   ;contexto al cual se dirigen las llamadas hechas por este terminal
host=dynamic    ;el dispositivo registrara el telefono e Asterisk
nat=yes         ;asume que el terminal esta detras de un NAT
secret=juan123  ;un password de seguridad para que el telefono se asocie
dtmfmode=auto   ;indica que acepta tonos de marcado de otros dispositivos
disallow=all    ;deshabilita todos los códecs de voz
allow=ulaw      ;primer códec que acepta
allow=alaw      ;segundo códec que acepta
username=Juan   ;nombre con el cual se registra el dispositivo
```

Figura 1.5 Ejemplo de Archivo sip.conf

La **Figura 1.5** muestra una sección `[general]`, esta es una sección estándar que se encuentra en la parte superior del archivo la cual indica parámetros por defecto y la manera cómo el protocolo se relaciona con el sistema.

1.1.4.2 Elastix

Elastix es un sistema de comunicaciones unificado basado en Asterisk que permite incluir varios servicios como: fax, mensajería instantánea, correo, video conferencia y voz sobre IP. Es un sistema de código abierto y cuenta con la licencia GPLv2, es decir quien lo adquiera, deberá estar sujeto a condiciones de dicha licencia. La versión más reciente que se tiene es la versión 2.5.0 [8].

Además de los servicios ya mencionados que ofrece Elastix, se puede nombrar algunos adicionales, tales como [8]:

- Interconexión entre PBXs
- Identificación de origen de la llamada
- Grabación de llamadas
- Soporte para smartphones
- Reportes Avanzados

1.1.4.3 FreePBX

FreePBX es un sistema de administración gráfica de Asterisk de código abierto y licenciado bajo la licencia GNU GPL (*General Public License*). El sistema incluye el sistema operativo basado en Linux, la herramienta Asterisk y la interfaz gráfica de administración. El esquema de comunicación se ilustra en la **Figura 1.6**.

Los usuarios interactúan con la interfaz gráfica de FreePBX vía web, la cual a su vez se comunica con los módulos de FreePBX, cada uno tiene funciones distintas y almacenan información en una base de datos que interactúa con el framework de FreePBX, que a su vez trabaja con Asterisk para escribir los archivos necesarios.

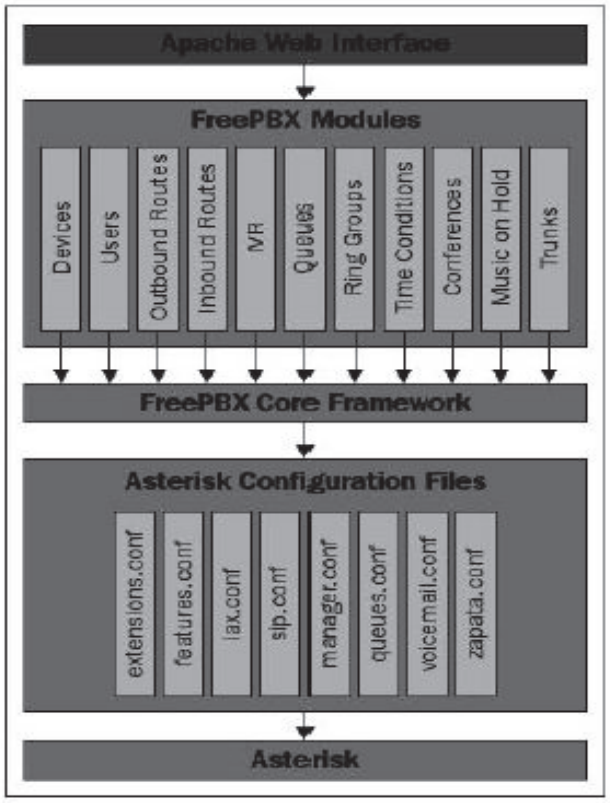

Figura 1.6 Comunicación de Módulos de FreePBX [9]

1.2 PLATAFORMA DE CÓMPUTO RASPBERRY PI

Raspberry Pi es un computador de placa reducida del tamaño de una tarjeta de crédito y de bajo costo. A pesar de tener una capacidad menor a la de un computador normal, es capaz de realizar muchas de las tareas que un computador

de escritorio lo hace, tales como navegar por Internet, reproducir videos de buena calidad, utilizar hojas de cálculo, procesadores de texto, entre otras. Además, es utilizado para motivar a personas de cualquier edad al aprendizaje de lenguajes de programación como Scratch y Python [10].

Raspberry Pi cuenta con dos modelos: el tipo A y el tipo B. Una de las diferencias marcadas entre ambos es la capacidad de memoria que en el caso del tipo B es de 512 MB, mientras que en el tipo A es de 256 MB. La segunda diferencia entre ambos a nivel físico, es que el modelo tipo B ofrece mayor cantidad de puertos, considerado el puerto Ethernet que le da una gran ventaja frente al tipo A.[11]

1.2.1 ESTRUCTURA GENERAL DE UN RASPBERRY PI [11]

La placa Raspberry Pi modelo B+ que va a ser utilizado en el presente proyecto cuenta con un procesador Broadcom BCM2835 basado en la arquitectura ARM, el cual es el encargado del procesamiento del Raspberry Pi, así como también del proceso asociado a gráficos y tareas de entrada/salida.

La placa también dispone de una memoria RAM (*Random Access Memory*) para el almacenamiento temporal de datos, dichos datos serán borrados cuando el Raspberry Pi se encuentre apagado.

Raspberry Pi cuenta con puertos USB que le permiten conectarse a un teclado y mouse, de igual manera posee un puerto RJ-45 Ethernet, en el caso de un modelo B para conectarse a una red, además de un puerto HDMI que utiliza para conectarse a un monitor o a un televisor. Finalmente, posee una entrada de audio y un puerto micro USB para la alimentación energética [11]. La **Figura 1.7** muestra la estructura de la placa Raspberry Pi tipo B+.

Raspberry Pi también tiene 20 pines GPIO en el caso de la versión B+ para propósitos de entrada/salida y para facilitar la conexión a otro tipo de hardware. En la parte derecha del conector HDMI se encuentra el puerto CSI (*Camera Serial Interface*), que le permite tener una conexión de alta velocidad a un módulo con cámara.

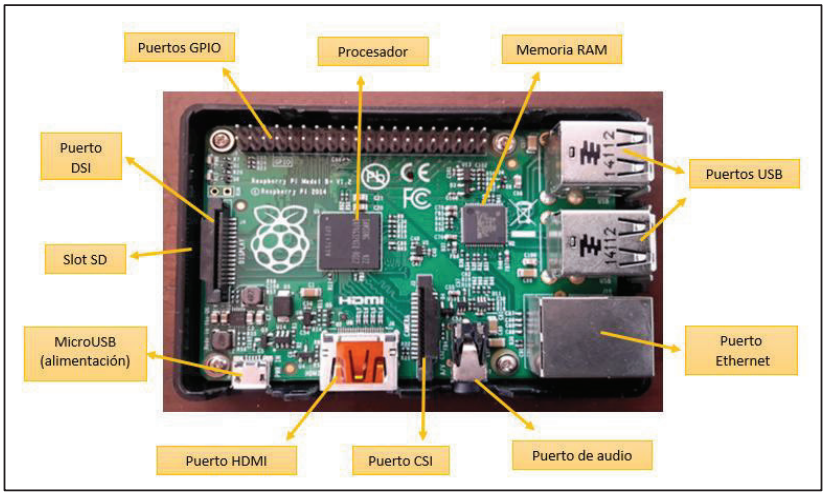

Figura 1.7 Estructura de Raspberry Pi

Del lado opuesto a los puertos USB y Ethernet como muestra la **Figura 1.7** se encuentra una ranura donde se coloca la tarjeta SD (*Secure Digital*) por la parte inferior de la placa la cual es utilizada para el almacenamiento del Sistema Operativo, datos, programas, entre otros. Por otro lado, el Display o puerto DSI (*Display Serial Interface*) permite la conexión a un panel digital, el cual es utilizado principalmente por desarrolladores de sistemas embebidos [11].

1.2.2 CARACTERÍSTICAS DE SOFTWARE

Raspberry Pi utiliza un procesador BCM2835 con arquitectura ARM, de 32 bits. Esta arquitectura fue creada en la década de los ochenta por Arcor Computers y combina la arquitectura RISC (*Reduced Instruction Set Computing*) con la capacidad de bajo consumo de energía, para ejecutar tareas del procesador. Esta arquitectura es utilizada por dispositivos móviles y permite que el Raspberry Pi funcione con un voltaje de 5V y consuma una corriente de 1A [11].

El sistema operativo más común en este tipo de placas es Raspbian, derivado de la distribución Debian, que cuenta con los paquetes estrictamente necesarios para funcionar en una plataforma de prestaciones reducidas.

Raspbian incluye los paquetes necesarios para poder utilizar un navegador web, realizar programación en Python, de igual manera incorpora un ambiente de escritorio denominado Lightweight X11 Desktop Enviroment (LXDE) basado en ventanas y de fácil uso.

1.3 DESARROLLO DE SOFTWARE

1.3.1 ANÁLISIS DE REQUERIMIENTOS

El análisis de requerimientos es la primera etapa para el desarrollo de software, consiste en tomar todos los requisitos necesarios para elaborar el software.

Los requerimientos pueden ser clasificados en dos tipos: requerimientos funcionales y requerimientos no funcionales.

1.3.1.1 Requerimientos Funcionales

Los requerimientos funcionales son aquellos que indican qué tarea debe realizar el producto (software) para satisfacer la necesidad del cliente [11]. Un requerimiento funcional, por ejemplo, puede ser el obtener el promedio de las calificaciones de un alumno.

1.3.1.2 Requerimientos No Funcionales

Los requerimientos no funcionales son características, propiedades o cualidades que debe tener el software para cumplir con la tarea para lo cual fue hecho [12]. Un requerimiento no funcional es representado por un adjetivo o cualidad. Por ejemplo, el software debe ser de fácil uso y rápido para ejecutar sus tareas.

1.3.1.3 Casos de Uso[13]

Los casos de uso son técnicas que se emplea para especificar los requerimientos en el diseño de software, que consiste en describir el comportamiento del usuario final frente al sistema. Un caso de uso puede estar elaborado en una tabla que puede tener los siguientes elementos:

- Autor: Indica el nombre de la persona que toma el requerimiento.
- Actor: Usuario que interviene en el sistema que se va a desarrollar.
- Fecha: Fecha en la que se elaboró el caso de uso.
- Código: Identificador del caso de uso.

- Título: Nombre que se asignó al caso de uso.
- Procedimiento Exitoso: Conjunto de pasos que expresa un proceso con éxito del caso de uso.
- Procedimiento Alternativo: Procedimiento a tomar frente a un fallo del procedimiento principal.

La **Tabla 1.1** detalla un ejemplo del formato de un caso de uso, en el que se muestra cada una de las partes indicadas.

Autor: Andrés Castillo	**Actor:** Mesero
Código: CU-01	**Fecha:** 24 de marzo del 2014
Título: Crear pedidos	
Procedimiento exitoso: 1.- El mesero se logeo en el sistema, y oprime el botón escoger mesa 2.-Se muestra una lista de acciones como pedido nuevo, pedidoanterior, etc 3.-Escojo la opción pedido nuevo. 4.-Se muestra una lista de todos los platos disponibles para el cliente. 5.-El mesero va escogiendo cada plato con un checkbox y escribe en el campo número la cantidad de los mismos pedidos por el cliente. 6.- El mesero llena un identificativo del pedido. 7.-El mesero presiona el botón guardar y enviar . 8.- Se actualizan los datos en la base 9.-Se muestra el formulariode los pedidos	
Procedimiento alternativo: 4a.- Falla el envío 5a.- Muestra mensaje de error 6a.- Retorno al formulario de los platos	

Tabla 1.1 Ejemplo de un Caso de Uso

Los casos de uso pueden ser representados mediante un diagrama, en el que intervienen los siguientes componentes: el actor, casos de usos, relaciones [14].

El actor es el usuario que interviene en el sistema, mientras que los casos de uso son acciones que se van a producir en el sistema a desarrollarse, y finalmente las

relaciones son los nexos entre actores y casos de uso. El diagrama puede ser utilizado para definir el comportamiento y la comunicación que existe entre los usuarios y el sistema que se va a desarrollar.

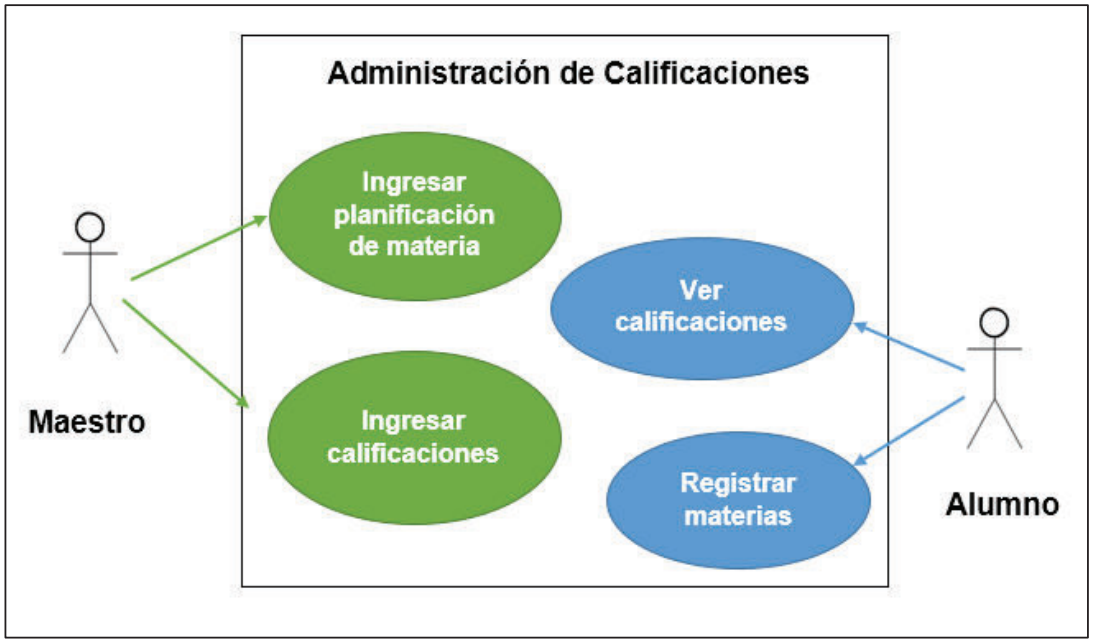

Figura 1.8 Ejemplo de Diagrama de Casos de Uso

La **Figura 1.8** muestra un ejemplo de un diagrama en el que los casos de uso están representados por un óvalo, el sistema se representa con un rectángulo y los actores se los representa fuera del sistema.

1.3.1.4 Historias de Usuario

Es una herramienta para representar los requerimientos de diseño de modo más informal y sencillo que los casos de uso. El cliente describe su requerimiento en sus propias palabras en una nota. El formato de una historia de usuario puede ser orientado a identificar tres aspectos que son: el actor, la razón y el beneficio.

Un ejemplo de la manera en que podría el cliente escribir la nota sería: “Como [**cliente habitual**], quiero [**ver productos relacionados con mis compras anteriores**] para [**ver si hay otros productos que me puedan interesar**]” [15].

Las historias de usuario pueden tener las siguientes características:

- Independientes unas de otras, combinan aquellas que dependan entre sí.

- Negociables, no son muy formales, como un contrato, pero se debe aclarar el alcance mediante una discusión con los clientes.
- Valoradas, dan valor a lo que el cliente quiere, no a lo que el desarrollador desee.
- Estimables, indican el tiempo que tomará en completar cada historia, lo que ayudará con el tiempo de estimación del proyecto.
- Pequeñas, deben ser lo más concretas posibles.
- Verificables, generalmente cubren un requerimiento funcional por lo que pueden verificarse.

1.3.2 ETAPA DE DISEÑO

1.3.2.1 Modelos del Proceso de Software

Un modelo del proceso de software representa una visión general de un proceso desde una perspectiva en particular. En un modelo se pueden incluir actividades relacionadas al proceso de software, el papel que cumplen las personas que participan en el proceso de ingeniería de software y el producto final que es el software a desarrollar [16].

Existen tres ejemplos de modelos, entre ellos: el modelo en cascada, el modelo iterativo o incremental y el modelo basado en componentes. Sin embargo el modelo incremental fue escogido en base a las características que se presentan en la **sección 1.3.2.1.1.**

1.3.2.1.1 Modelo Incremental

El modelo incremental se fundamenta en elaborar una primera versión de un sistema y entregárselo al cliente para una primera opinión, después de ello se analizarán los cambios especificados por el cliente y nuevamente se hará una entrega de una segunda versión, repitiendo el procedimiento hasta alcanzar versión final.

Cada versión entregada es denominada incremento o entregable, y se va definiendo con los clientes y programando según la urgencia del cliente. El modelo incremental es recomendable en casos en los cuales el cliente tenga una necesidad muy urgente de tener cierta funcionalidad del prototipo y conforme se avance, se puede ir dando mayor complejidad al sistema [13].

Una ventaja de este modelo es que el producto final estará más alineado a los requerimientos del cliente debido a los incrementos. Como aspecto negativo, el modelo incremental presenta una estructura muy deficiente debido a los cambios frecuentes, incrementado el costo del sistema y haciéndolo más complejo.

Se recomienda utilizar este modelo en sistemas en donde existan hasta 500000 líneas de código [16], y también cuando se dispone de poco personal para realizar la implementación en el tiempo en el cual se marca la entrega [13].

1.3.2.2 Metodologías de Desarrollo de Software

Una metodología de desarrollo de software es una perspectiva estructurada del desarrollo del software que tiene como finalidad facilitar la creación del software de alta calidad de manera factible.

1.3.2.2.1 Metodologías de Desarrollo Ágiles

1.3.2.2.1.1 Principios de Desarrollo de las Metodologías Ágiles [17]

Los principios por medio de los cuales las metodologías ágiles se llevan o fundamentan para el desarrollo de software son los siguientes:

1. “Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software con valor.”

2. “Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.”

3. “Entregamos software funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible.”

4. “Los responsables de negocio y los desarrolladores trabajamos juntos de forma cotidiana durante todo el proyecto.”

5. “Los proyectos se desarrollan en torno a individuos motivados. Hay que darles el entorno y el apoyo que necesitan, y confiarles la ejecución del trabajo.”

6. “El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara.”

7. “El software funcionando es la medida principal de progreso.”

8. “Los procesos ágiles promueven el desarrollo sostenible. Los promotores, desarrolladores y usuarios debemos ser capaces de mantener un ritmo constante de forma indefinida.”

9. “La atención continua a la excelencia técnica y al buen diseño mejora la agilidad.”

10. “La simplicidad, o el arte de maximizar la cantidad de trabajo no realizado, es esencial.”

11. “Las mejores arquitecturas, requisitos y diseños emergen de equipos auto-organizados.”

12. “A intervalos regulares el equipo reflexiona sobre cómo ser más efectivo para a continuación ajustar y perfeccionar su comportamiento en consecuencia.”

1.3.2.2.2 Metodología XP

XP (*Extreme Programming*) es una metodología de desarrollo de software que se fundamenta en cuatro valores expuestos en los doce principios de las Metodologías Ágiles que son: comunicación, simplicidad, coraje o valor y realimentación [18].

La comunicación entre los desarrolladores y los clientes, permite un mejor entendimiento de las cosas, la simplicidad que garantice una manera fácil de entender para los programadores y para los clientes; el coraje trata de explicar el optimismo de tener todo bien estructurado, de manera que si un cliente pide un cambio en algún requerimiento, este pueda ser implementado. Finalmente, la realimentación que permite mostrar de qué manera puede mejorar la funcionalidad, presentación o estructura de cierto módulo en particular.

XP se caracteriza porque permite una interacción entre el cliente y el grupo de desarrollo más profunda, también permite o se basa en la propia experiencia del programador en proyectos pasados. En lo que respecta a la rapidez, permite una entrega de resultados lo más rápido posible al cliente.

XP utiliza las historias de usuario para la obtención de los requerimientos, las mismas que son implementadas como un conjunto de tareas [16]. Una de las características más relevantes es la programación en pares, uno de los miembros del grupo se encarga de escribir el código, mientras que el otro realiza una supervisión, reduciendo el riesgo de fallos [18].

1.3.2.3 Patrón de Diseño Modelo Vista Controlador

"MVC (*Model-View-Controller*) es un patrón de diseño que considera dividir una aplicación en tres módulos claramente identificables y con funcionalidad bien definida" [19]. Los cuales son: el Modelo, la Vista y el Controlador.

MVC nació con la finalidad de reducir el esfuerzo de la programación durante el desarrollo de las aplicaciones. La característica fundamental se encuentra en proporcionar facilidad al momento de efectuar cambios debido a la separación de los tres componentes [20].

MVC presenta las siguientes ventajas:

- "Separación clara entre componentes de un programa; lo cual permite su implementación por separado."[19]
- "API (*Application Programming Interface*) muy bien definida, cualquiera que use el API podrá reemplazar el Modelo, la Vista o el Controlador, sin aparente dificultad."[20]
- La conexión entre el Modelo y las vistas es dinámica puesto que se produce en tiempo de ejecución y no en tiempo de compilación [20].

La relación de los tres componentes de MVC puede verse en la **Figura 1.9.**

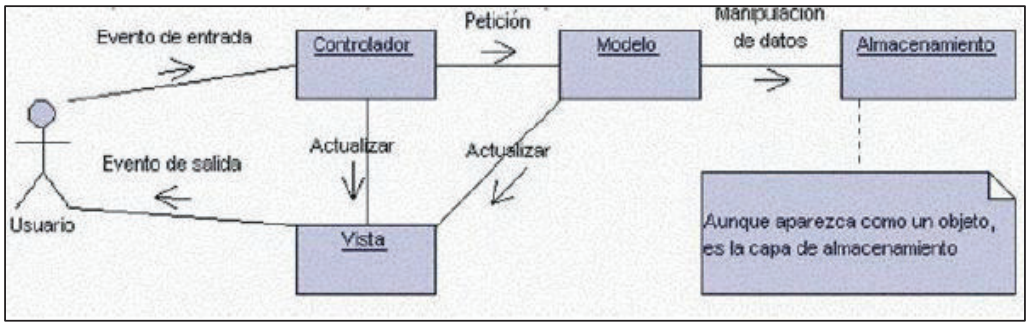

Figura 1.9 Interacción componentes de MVC [20]

1.3.2.3.1 Modelo

El modelo es el componente que contiene la información del mundo real que va a ser procesada por el sistema, puede estar representado por un conjunto de clases [10] las cuales tienen la función de acceder a la capa de almacenamiento de datos.

1.3.2.3.2 Vista

La vista es el componente que se encarga de mostrar los datos necesarios del modelo al usuario que manipula el sistema, también puede efectuar la actualización

[10] Clase: Abstracción de cualquier cosa del mundo real, utilizado en el paradigma de la Programación Orientada a Objetos

de los datos del modelo en un sistema activo[11]. Las vistas pueden estar en una relación de muchos a uno con el modelo, cada una con su respectivo controlador.

1.3.2.3.3 Controlador

El controlador es el módulo que actúa como un nexo entre la vista y el modelo, puede recibir eventos generados por el usuario (tales como un click) lo que le permite hacer modificaciones en el modelo. El controlador se basa en reglas de gestión de eventos como “Si Evento Z, entonces Acción W” [20].

[11] Sistema Activo: Sistema en el cual la información se va actualizando.

CAPÍTULO 2

2 ANÁLISIS DE REQUERIMIENTOS Y DISEÑO DEL PROTOTIPO

2.1 REQUERIMIENTOS DEL SISTEMA

Para determinar los requisitos de los servicios de telefonía que debe cumplir el prototipo se realizó una encuesta a padres de familia, quienes serán los usuarios encargados de configurar el sistema. Además se efectuaron historias de usuario como se detalla en la **sección 1.3.1.4** para recolectar los requisitos que debe cumplir la aplicación, misma que se encargará de la gestión del sistema telefónico, tomando como guía la metodología de desarrollo XP.

2.1.1 ENCUESTA PARA REQUISITOS DE LOS SERVICIOS DE TELEFONÍA

Se decidió aplicar la encuesta a una población de 2000 personas, referentes al número de padres de familia de los estudiantes de la facultad de Ingeniería Eléctrica y Electrónica matriculados en el semestre 2014B [21]; para el cálculo de la muestra se asumió el nivel de confianza del 95%, un error máximo de aceptación del 8.3% y una probabilidad de éxito del 50%, a fin de obtener una muestra aceptable para realizar el estudio. El nivel de confianza del 95% indica que 19 de 20 muestras de la misma población analizada van a generar intervalos de confianza que contendrán el parámetro de población [22]. Se consideró una probabilidad de éxito del 50% para reflejar el peor caso y el error fue fijado en 8.3% con la finalidad de no generar una muestra muy grande debido a que la mayoría de estudiantes no responden con seriedad las encuestas.

De acuerdo a la fórmula de la **Ec.(1)** se obtuvo una muestra de 140 personas a quienes se les aplicó la encuesta.

$$\text{tamaño de la muestra} = \frac{N \times Z^2 \times p \times q}{e^2 \times (N-1) + (Z^2 \times p \times q)} \qquad \textbf{Ec.(1)} \text{ [23]}$$

Donde:

N = tamaño de la población

Z=Nivel de confianza o seguridad

p=Probabilidad de éxito

q=1-p

e=Error máximo aceptado

Una vez procesadas las respuestas obtenidas de la encuesta, se llegó a conseguir los resultados necesarios para establecer los requisitos que debería cumplir el sistema en lo que concierne a los servicios de telefonía, y también respecto a algunos parámetros que se deben incluir en la interfaz gráfica del sistema.

La **Tabla 2.1** indica los resultados correspondientes a las respuestas más representativas de las encuestas realizadas, las cuales serán tomadas como referencia para la futura implementación del prototipo.

PREGUNTA	RESPUESTA	NÚMERO DE RESPUESTAS	% EQUIVALENTE
Destinos más frecuentes del padre de familia	Llamadas Locales	130	91,50%
	Llamadas Celulares	45	31,70%
Inconvenientes(molestias) más comunes	Recepción de llamadas no deseadas	74	52,10%
	Falta de detalle de llamadas	58	40,80%
	Llamadas de duración excesiva	37	26,10%
Número de hijos que viven con los padres	2 hijos	71	50%
	3 hijos	37	26,00%
Horario "adecuado" para realizar una llamada (hijos edad<15 años)	13:00-20:00	28	45,90%
	8:00-20:00	13	21,30%
Horario "adecuado" para realizar una llamada (hijos 15años<edad<20años)	Todo el día	23	35,90%
	7:00-22:00	21	32,80%
Tiempo de duración de una llamada de un hijo	10 minutos	53	37,30%
	3minutos	42	29,60%
	15 minutos	29	20,40%
Destinos más frecuentes para bloqueo de llamadas salientes	Llamadas Celulares	99	69,70%
	Llamadas Internacionales	70	49,30%
	Números 1800	44	31%
Servicios adicionales de telefonía	Contestadora Automática (IVR)	76	53,50%
	Buzón de voz	64	45,10%

Tabla 2.1 Resumen de resultados de la encuesta

El Anexo A corresponde al cuestionario que se utilizó en la realización de las encuestas, en el anexo además se incluye un detalle de los resultados que se obtuvieron en la encuesta por cada pregunta planteada.

2.1.2 HISTORIAS DE USUARIO

Las historias de usuario que se realizaron se presentan a continuación en forma de tabla. Las mismas que incluyen un código para identificar la historia de usuario, el autor, el usuario, la fecha en que fue realizada, el título de la misma, la descripción de las preferencias del usuario y un campo de observaciones.

Para aplicar la entrevista, se escogió al padre de una familia modelo en Ecuador, teniendo en cuenta como único parámetro el conocimiento del uso un computador.

2.1.2.1 Primera iteración

En la primera iteración se realizó una entrevista a un padre de familia para recolectar información sobre los requisitos que debería tener el sistema. En base a esta entrevista se elaboró las historias de usuario que se muestran desde la **Tabla 2.2** hasta la **Tabla 2.7**.

Autor: Juan Manuel Bastidas	**Usuario:** Padre de familia
Código: HU1-01	**Fecha:** 26 de mayo del 2015
Título: Crear Usuario	
Descripción: El sistema me debe dejar ingresar los datos de los miembros de mi casa, datos como el nombre y su teléfono.	
Observaciones: Ingreso de solo letras en el nombre de la persona y en el caso de los teléfonos solo números.	

Tabla 2.2 Historia de Usuario HU1-01, Crear Usuario

Autor: Juan Manuel Bastidas	**Usuario:** Padre de familia
Código: HU1-02	**Fecha:** 26 de mayo del 2015
Título: Restricción de las llamadas	
Descripción: El sistema me debe dejar escoger a dónde le permito llamar a mi hijo: como a celulares, números locales o provinciales, internacionales, números de emergencia. También quiero que pueda escoger el horario en que pueda hacer las llamadas y ponerle un tiempo límite para que no se demoren mucho.	
Observaciones: Asegurarse de que las opciones de horarios correspondan a los números permitidos para hora y minutos.	

Tabla 2.3 Historia de Usuario HU1-02, Restricción de Llamadas

Autor: Juan Manuel Bastidas	**Usuario:**Padre de familia
Código: HU1-03	**Fecha:** 26 de mayo del 2015
Título: Visualización de Detalle de Llamadas	
Descripción: Quiero poder ver un detalle de las llamadas que se hacen en la casa, cuánto duran en promedio las llamadas hechas. También quisiera ver las llamadas que llegan de afuera.	
Observaciones: Presentar un número de registros adecuado en la tabla para una correcta visualización .	

Tabla 2.4 Historia de Usuario HU1-03, Visualización de Destalle de Llamadas

Autor: Juan Manuel Bastidas	**Usuario:** Padre de familia
Código: HU1-04	**Fecha:** 26 de mayo del 2015
Título: Modificar los Datos de los Usuarios	
Descripción: El sistema me debe dejar cambiar la información de los usuarios como los números de teléfono, habilitar las llamadas que antes le bloquee.	
Observaciones: Validar a usuarios con el mismo nombre.	

Tabla 2.5 Historia de Usuario HU1-04, Modificar los Datos de los Usuarios

Autor: Juan Manuel Bastidas	**Usuario:** Padre de familia
Código: HU1-05	**Fecha:** 26 de mayo del 2015
Título: Borrar Usuarios	
Descripción: Debería poder borrar a los usuarios por ejemplo a los que ya no vayan a vivir en mi casa como un familiar que vino de visita.	
Observaciones: Asegurarse que el usuario sea borrado adecuadamente.	

Tabla 2.6 Historia de Usuario HU1-05, Borrar Usuarios

Autor: Juan Manuel Bastidas	**Usuario:** Padre de familia
Código: HU1-06	**Fecha:** 26 de mayo del 2015
Título: Buzón de Mensajes y Contestadora	
Descripción: Sería bueno que cada uno tenga un buzón de voz donde dejen un mensaje si no contestamos y una contestadora que nos conteste los mensajes .	
Observaciones: Implementar un buzón de voz para cada usuario y una contestadora automática, para la recepción de llamadas.	

Tabla 2.7 Historia de Usuario HU1-06, Buzón de Mensajes y Contestadora

2.1.2.2 Segunda Iteración

En la segunda iteración, se presentó una primera versión del sistema y se realizó una prueba a cinco usuarios para determinar qué tan fácil resulta el poder manipular la interfaz gráfica, la elección de los cinco padres de familia se basó únicamente en el conocimiento sobre el uso de un computador. A fin de cumplir con las tareas que se determinaron en las historias de usuario de la primera iteración.

En base a las respuestas obtenidas de cinco padres de familia se generaron las historias de usuario que se muestran en **Tabla 2.8**,**Tabla 2.9**,**Tabla 2.10**,**Tabla 2.11** y **Tabla 2.12**.

Autor: Juan Manuel Bastidas	**Usuario:** Padre de familia
Código: HU2-01	**Fecha:** 29 de julio del 2015
Título: Iconos más claros y con título	
Descripción: Los íconos de ayuda deberían ser más visibles y ponerles un título, también cambiar las flechas de regreso a la parte superior izquierda.	
Observaciones: Mejorar la presentación de los íconos.	

Tabla 2.8 Historia de Usuario HU2-01, Iconos más claros y con Título

Autor: Juan Manuel Bastidas	**Usuario:** Padre de familia
Código: HU2-02	**Fecha:** 29 de julio del 2015
Título: Bloqueo Simple y Avanzado	
Descripción: El bloqueo simple y el bloqueo avanzado deben ser más claros y específicos, no se entiende en que se diferencian.	
Observaciones: Presentar de forma más clara (notoria) la presentación de los tipos de bloqueo.	

Tabla 2.9 Historia de Usuario HU2-02, Bloqueo Simple y Avanzado

Autor: Juan Manuel Bastidas	**Usuario:** Padre de familia
Código: HU2-03	**Fecha:** 29 de julio del 2015
Título: Creación del Usuario	
Descripción: Sería bueno que la parte de la creación del usuario se especifique más claro lo del teléfono, porque se confunde con el celular de cada persona.	
Observaciones: Cambiar la palabra teléfono por extensión, cambiar usuario por persona o miembro del hogar.	

Tabla 2.10 Historia de Usuario HU2-03, Creación del Usuario

Autor: Juan Manuel Bastidas	**Usuario:** Padre de familia
Código: HU2-04	**Fecha:** 29 de julio del 2015
Título: Instructivo	
Descripción: El sistema debe tener un instructivo en el menú principal, poner las tareas que hace el sistema para no perderse.	
Observaciones: Identificar los pasos que se requieren para cumplir ciertas tareas.	

Tabla 2.11 Historia de Usuario HU2-04, Instructivo

Autor: Juan Manuel Bastidas	**Usuario:** Padre de familia
Código: HU2-05	**Fecha:** 29 de julio del 2015
Título: Manejo de Perfil	
Descripción: El perfil se confunde con usuario, al momento de crear un usuario da la impresión que es de crear el perfil de un usuario.	
Observaciones: Sustituir la palabra **Perfil** en la presentación por la palabra **Grupo**, y dejar 3 grupos predefinidos: adulto, niño y adolescente.	

Tabla 2.12 Historia de Usuario HU2-05, Manejo de Perfil

2.1.2.3 Tercera iteración

La tercera iteración consistió en presentar una segunda versión del sistema y aplicar una prueba similar a la realizada en la segunda iteración al mismo grupo de cinco usuarios, de este modo se obtuvieron nuevos requisitos que se detallan en **Tabla 2.13, Tabla 2.14** y **Tabla 2.15.**

Considerando todas las Historias de Usuario que se obtuvieron por parte de los usuarios, se realizó un resumen de los requisitos que la aplicación necesitaba cumplir, dicho resumen puede verse en la **Tabla 2.16**.

Autor: Juan Manuel Bastidas	**Usuario:** Padre de familia
Código: HU3-01	**Fecha:** 11 de noviembre del 2015
Título: Excepciones en Bloqueo Simple	
Descripción: Permitir agregar unas dos excepciones en los bloqueos de destinos, por ejemplo en los celulares sería util dejar sin bloquear mi número y el de mi esposa, por cualquier emergencia.	
Observaciones: Presentar dos excepciones en los bloqueos de destinos, según lo manifestado por el usuario.	

Tabla 2.13 Historia de Usuario HU3-01, Excepciones en Bloqueo Simple

Autor: Juan Manuel Bastidas	**Usuario:** Padre de familia
Código: HU3-02	**Fecha:** 11 de noviembre del 2015
Título: Agregar Bloqueo de Llamadas Etrantes	
Descripción: Quiero poder bloquear llamadas de números no deseados.	
Observaciones: Agregar la funcionalidad dicha por el usuario y bloquear estos números para todos los usuarios.	

Tabla 2.14 Historia de Usuario HU3-02, Agregar Bloqueo de Llamadas Entrantes

Autor: Juan Manuel Bastidas	**Usuario:**Padre de familia
Código: HU3-03	**Fecha:** 11 de noviembre del 2015
Título: Mejora Interfaz de Creación de Usuarios	
Descripción: Quiero que la plantilla donde pongo los datos sea más grande y espaciada, y que sean más claros los ejemplos de como poner los datos.	
Observaciones: Cambiar la interfaz de agregar miembros del hogar según las especificaciones descritas en descripción.	

Tabla 2.15 Historia de Usuario HU3-03, Mejora Interfaz de Creación de Usuarios

Requerimiento	Tipo de Requerimiento	Historia de Usuario	Iteración
Creación de Usuarios(Miembros del Hogar)	Funcional	HU1-01	Primera
Restricciones de Llamadas	Funcional	HU1-02	
Visualización de los Registros de Llamadas	Funcional	HU1-03	
Modificación de la Información de los Usuarios	Funcional	HU1-04	
Eliminación de Usuarios del Sistema	Funcional	HU1-05	
Servicios de Telefonía de Buzón de Mensajes y Contestadora Automática	Funcional	HU1-06	
Interfaz gráfica con íconos claros y con título	No Funcional	HU2-01	Segunda
Presentación más clara sobre los tipos de bloqueo Simple y Avanzado.	No Funcional	HU2-02	
Claridad en los términos utilizados para los datos y títulos	No Funcional	HU2-03/HU2-05	
Instructivo(Ayuda) en el sistema	Funcional	HU2-04	
Implementación de excepciones en el bloqueo de destinos (Bloqueo Simple)	Funcional	HU3-01	Tercera
Implementación de un Bloqueo de Llamadas Entrantes	Funcional	HU3-02	
Formulario de Agregación de Usuarios más claro y organizado	No Funcional	HU3-03	

Tabla 2.16 Resumen de Requerimientos de la Aplicación

2.2 DISEÑO DEL SISTEMA

2.2.1 NIVEL DE DATOS

Para explicar el nivel de datos de la aplicación se utilizó un diagrama del modelo Entidad-Relación y un modelo de tablas.

2.2.1.1 Modelo Entidad-Relación

El modelo Entidad-Relación que se realizó para la base de datos consta de cuatro entidades: Persona, Perfil, Destino e Información de Llamadas; cada una se muestra en la **Figura 2.1** con sus respectivos atributos, así como también con las relaciones que las unen.

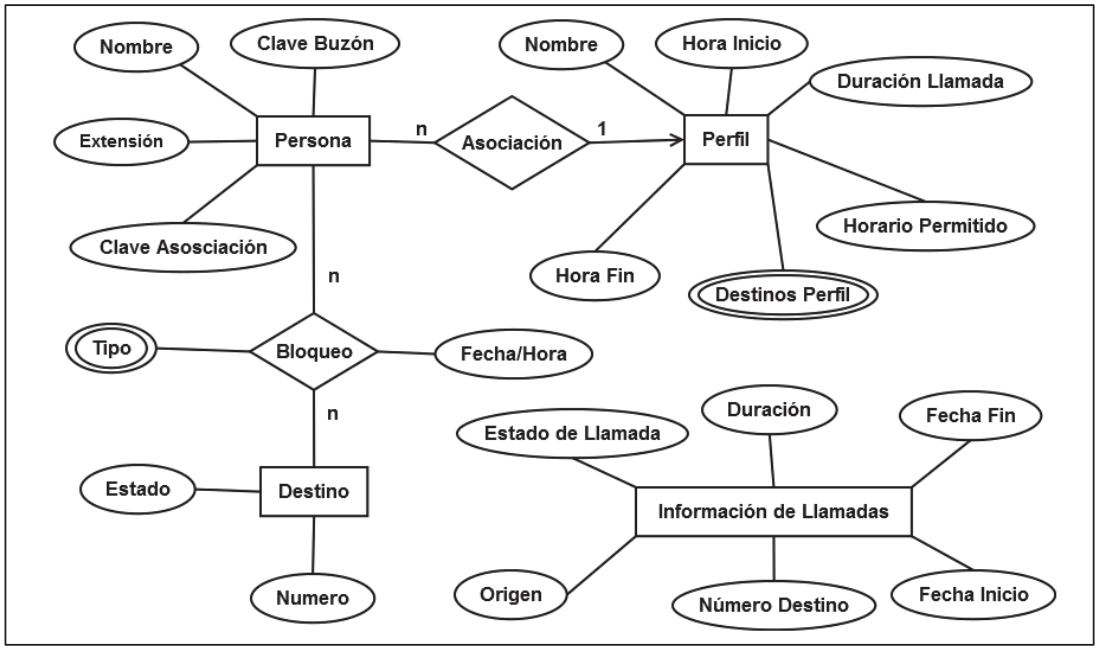

Figura 2.1 Diagrama Entidad-Relación de la Aplicación

2.2.1.2 Diagrama Relacional

En base al modelo Entidad-Relación se procedió a realizar el diagrama de tablas que será el modelo que se utilizará en el prototipo.

El diagrama consta de nueve tablas que permiten almacenar todos los datos necesarios para el funcionamiento del prototipo, dichas tablas son: Persona, Perfil, Bloqueo Simple, Bloqueo Avanzado, DestinosPerfil, PerfilUsuario, Destino y LlamadaEntrante.

Las tablas y las relaciones entre cada una de ellas se puede observar en el diagrama relacional de la **Figura 2.2**, donde se muestran las claves primarias[12] con color rojo, mientras que las claves foráneas[13] con letra verde. Para el diseño se consideró como clave primaria a un entero incremental en cada una de las tablas.

---

[12] Clave Primaria: Un campo, o combinación de varios campos que identifica de manera única un registro de una tabla de base de datos

[13] Clave Foránea: Una clave primaria de una tabla que representa una columna de otra tabla, indica la relación entre dos tablas.

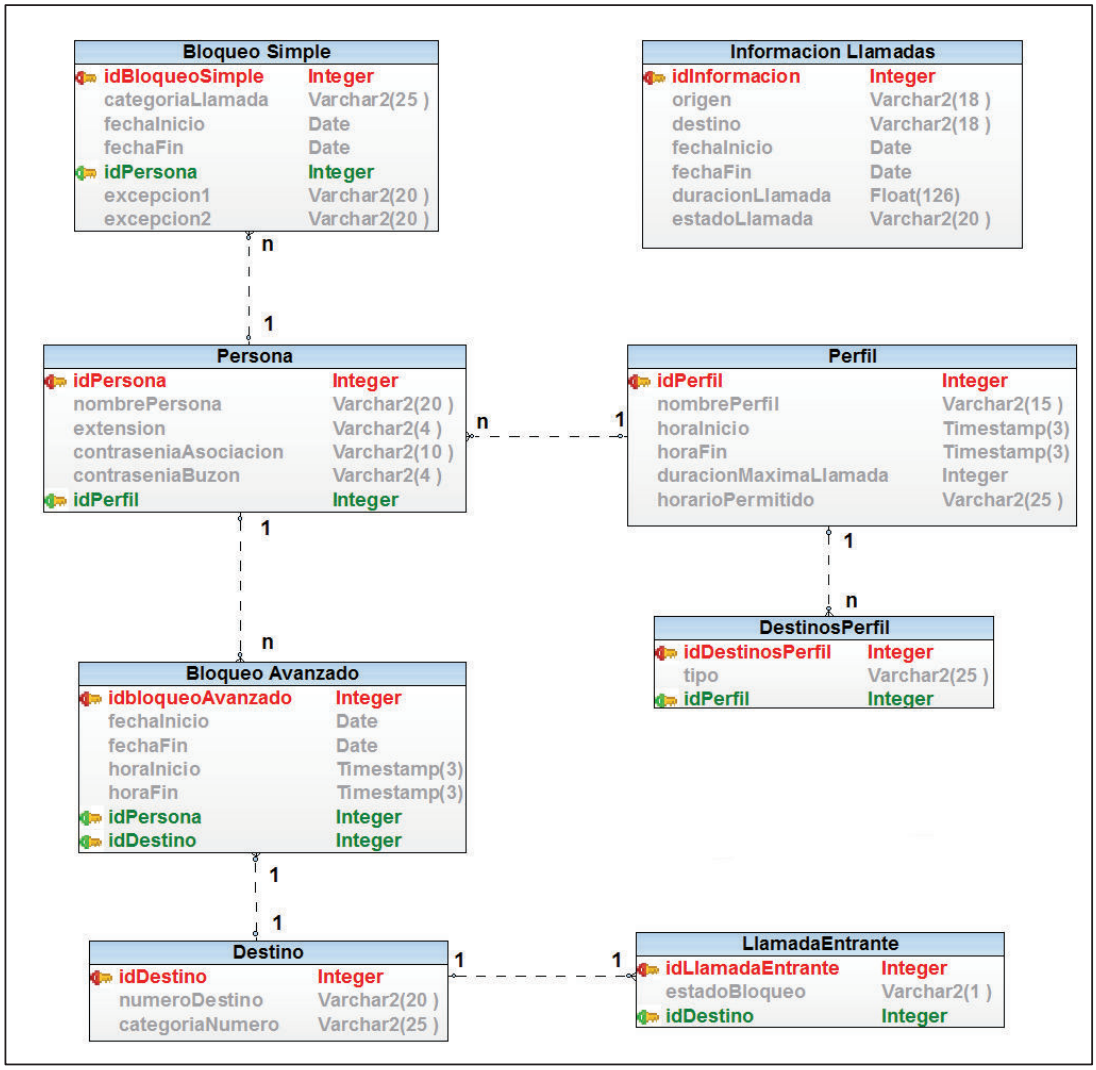

Figura 2.2 Diagrama Relacional

2.2.2 NIVEL DE LÓGICA DE NEGOCIOS

2.2.2.1 Diagrama de la Aplicación

La aplicación que va a gestionar el servicio de telefonía es una aplicación web, por esta razón se decidió trabajar con el Patrón de Diseño Modelo Vista Controlador, el cual es muy utilizado para desarrollar este tipo de aplicación como se muestra en **Figura 2.3**.

El bloque Vista es aquel que interactúa con el cliente y con el controlador, este bloque es representado como una página web es decir un archivo con extensión **.html**.

El bloque del Controlador además de interactuar con el bloque Vista, interactúa con el nivel de datos y también con la Central Telefónica Asterisk, más concretamente con los archivos **sip.conf**, **extensions.conf**, **voicemail.conf** y **Master.csv.**

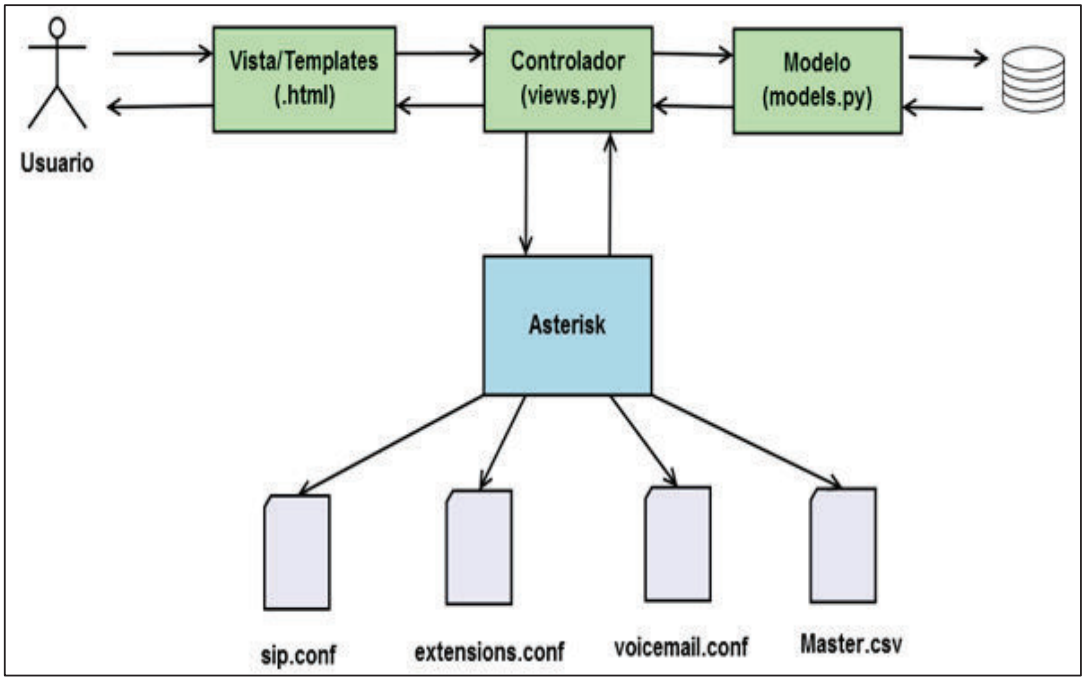

Figura 2.3 Diagrama de la Aplicación

Finalmente, el bloque del Modelo mostrado en la **Figura 2.3** se encarga de gestionar el acceso a la base de datos de acuerdo a lo que el Controlador le solicite o le envíe.

2.2.2.2 Diagrama de Clases

En vista de que el patrón MVC es un modelo orientado a objetos [19], es decir que trabaja con lenguajes de programación orientados a objetos como java, c#, Python, entre otros, se justifica el diagrama de clases que se muestra en la **Figura 2.4**.

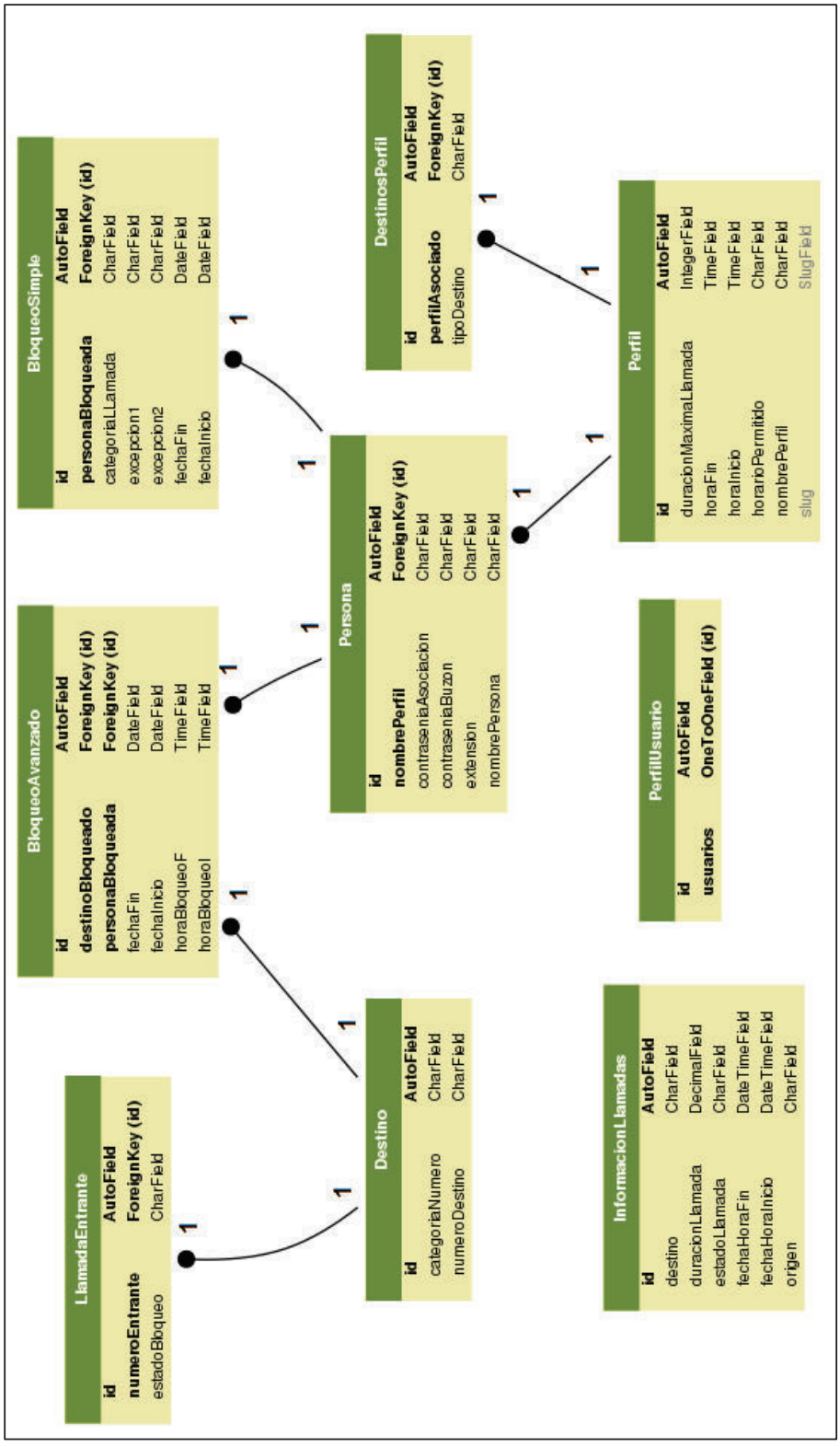

Figura 2.4 Diagrama de Clases

Donde el punto (**.**) representa la navegabilidad de la relación entre ambas clases.

2.2.2.3 Diagrama de Actividades

Para representar la parte dinámica de la aplicación se realizaron diagramas de actividades, los cuales permiten mostrar de manera gráfica el flujo de las interacciones en un escenario específico.

El diagrama de actividades utiliza las mismas figuras que un diagrama de flujo. El rombo para simbolizar una decisión, rectángulos redondeados para representar una función específica y flechas para conectar los anteriores elementos [13].

Los diagramas de actividades que se realizaron se pueden observar desde la **Figura 2.5** hasta la **Figura 2.15.**

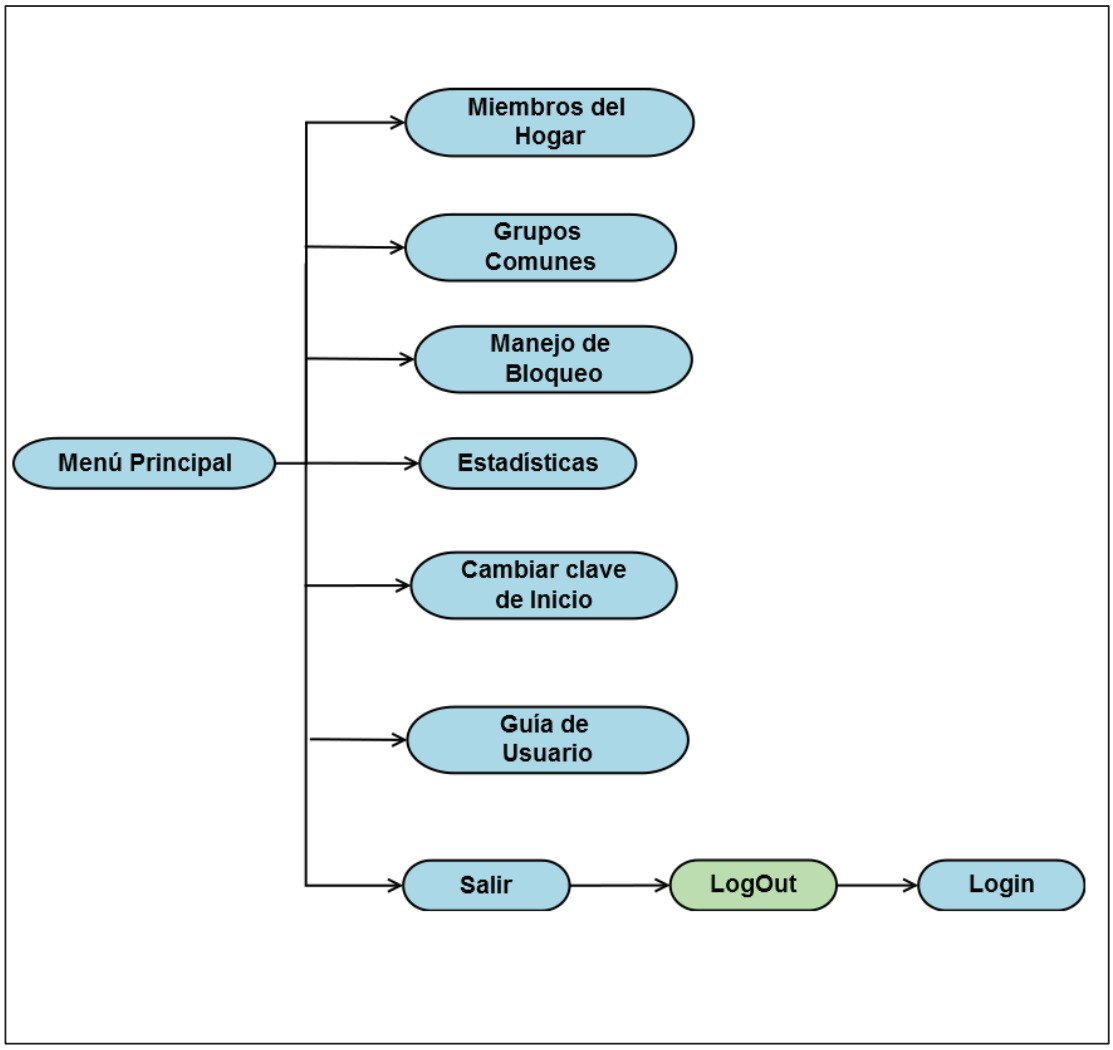

Figura 2.5 Diagrama de Actividades, Menú Principal

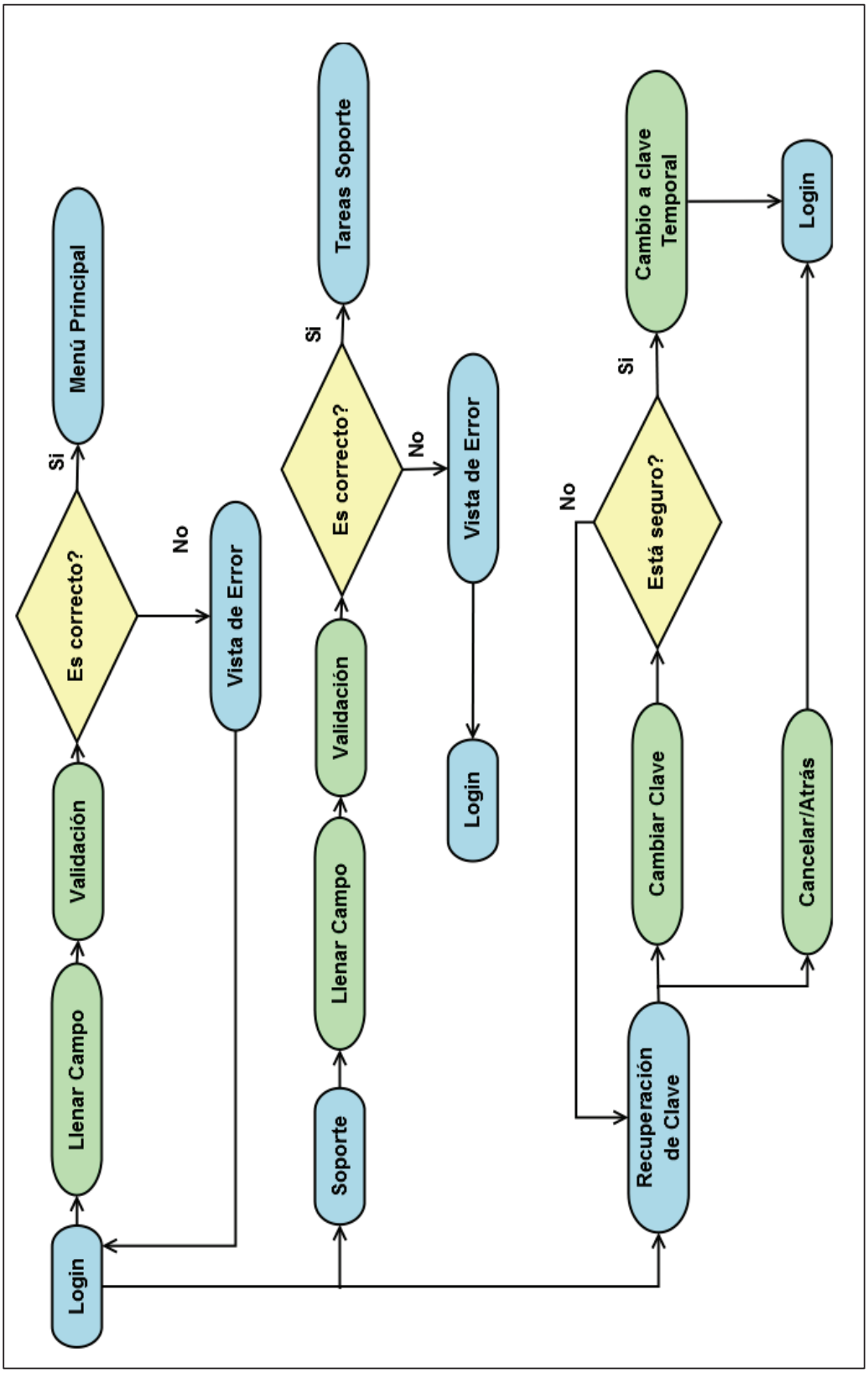

Figura 2.6 Diagrama de Actividades, Login

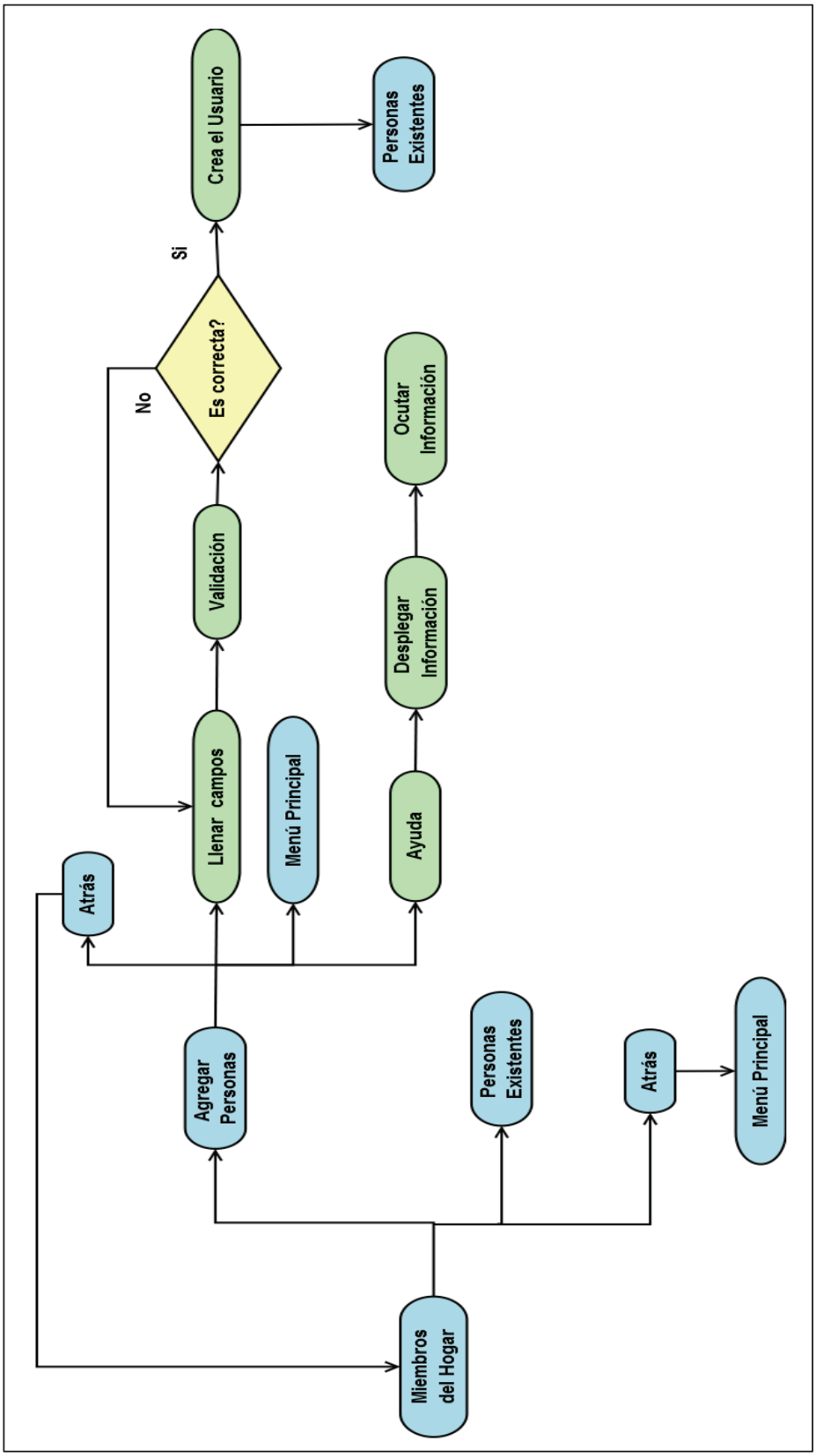

Figura 2.7 Diagrama de Actividades, Agregar Persona

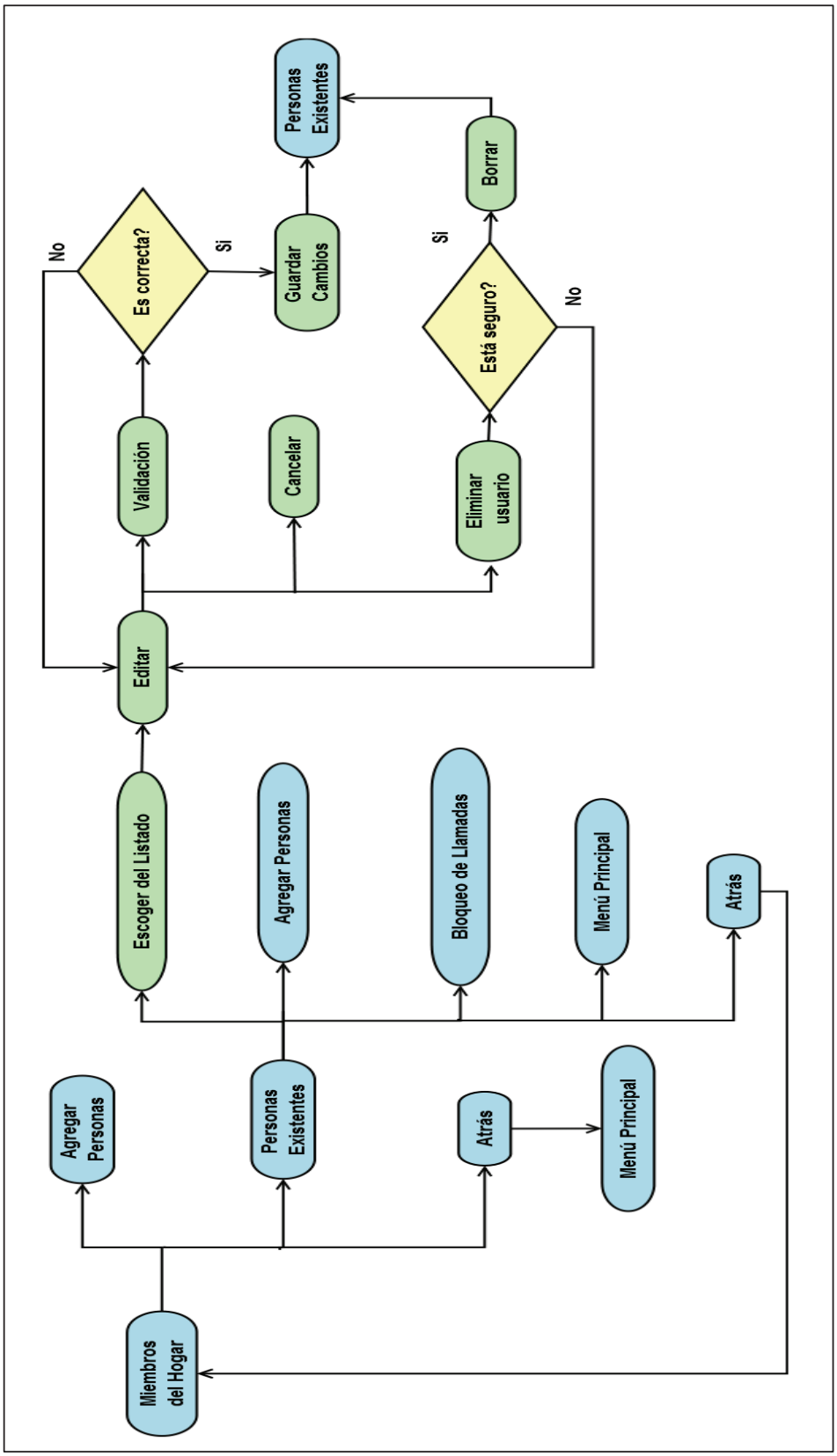

Figura 2.8 Diagrama de Actividades, Personas Existentes

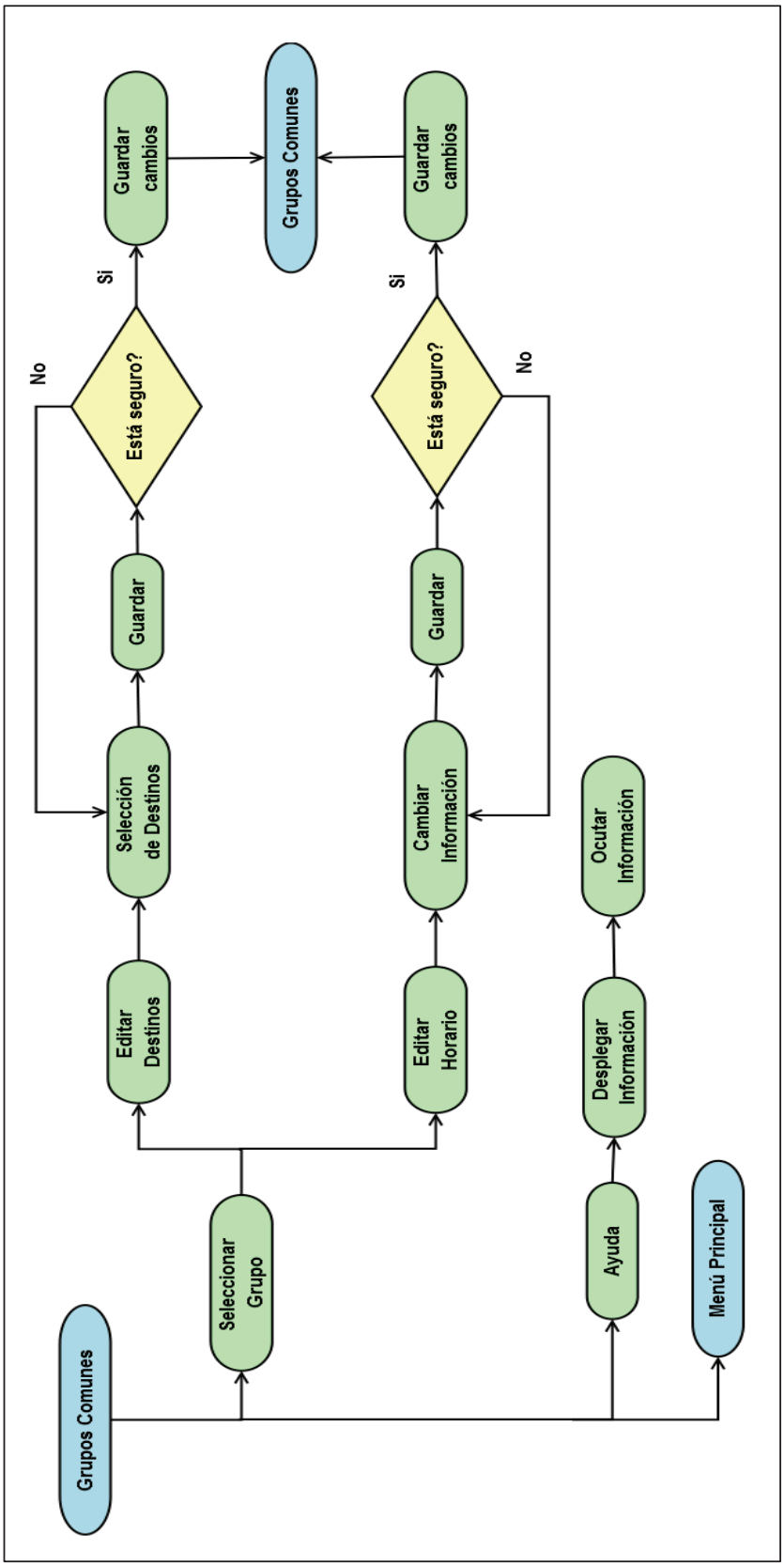

Figura 2.9 Diagrama de Actividades, Grupos Comunes

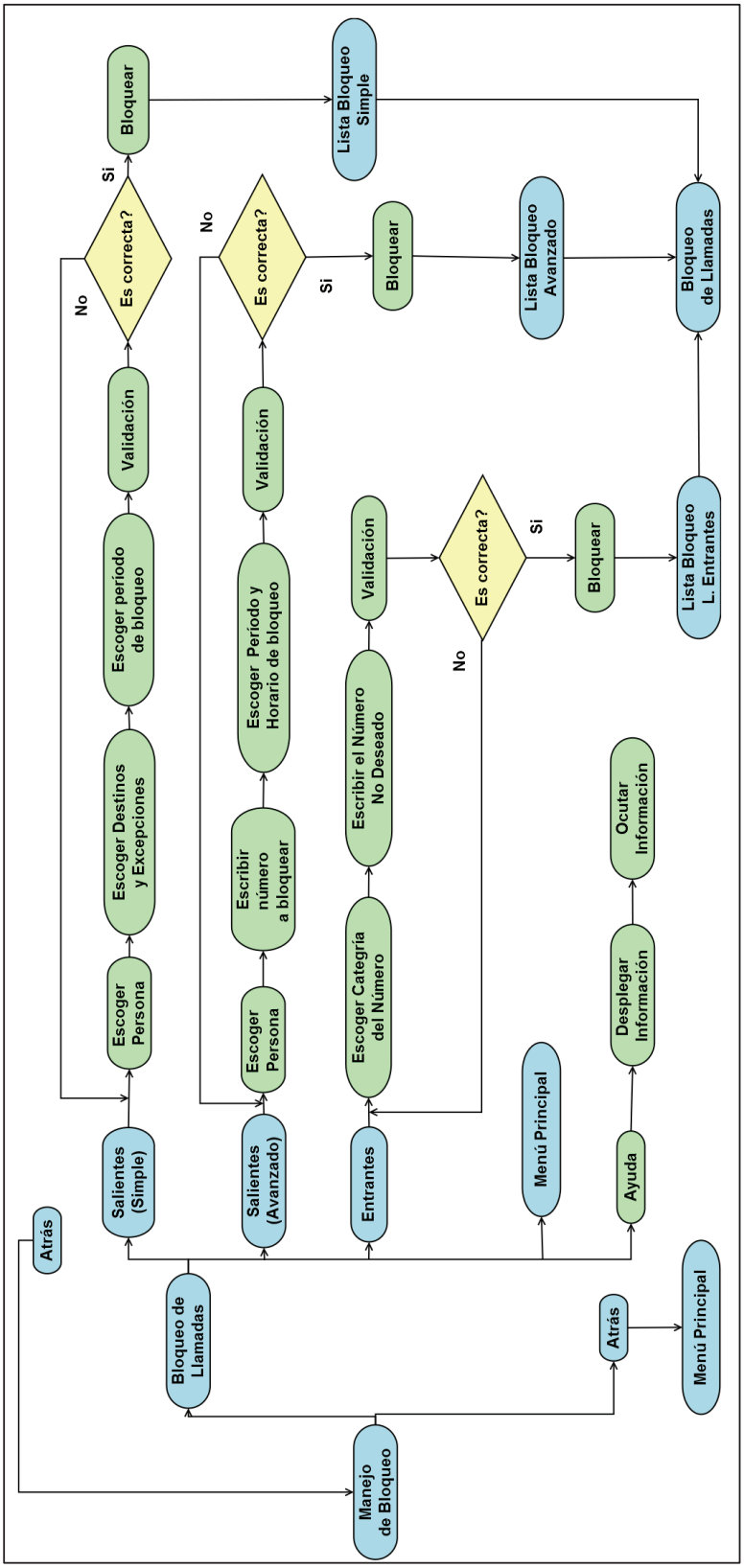

Figura 2.10 Diagrama de Actividades, Bloqueo de Llamadas

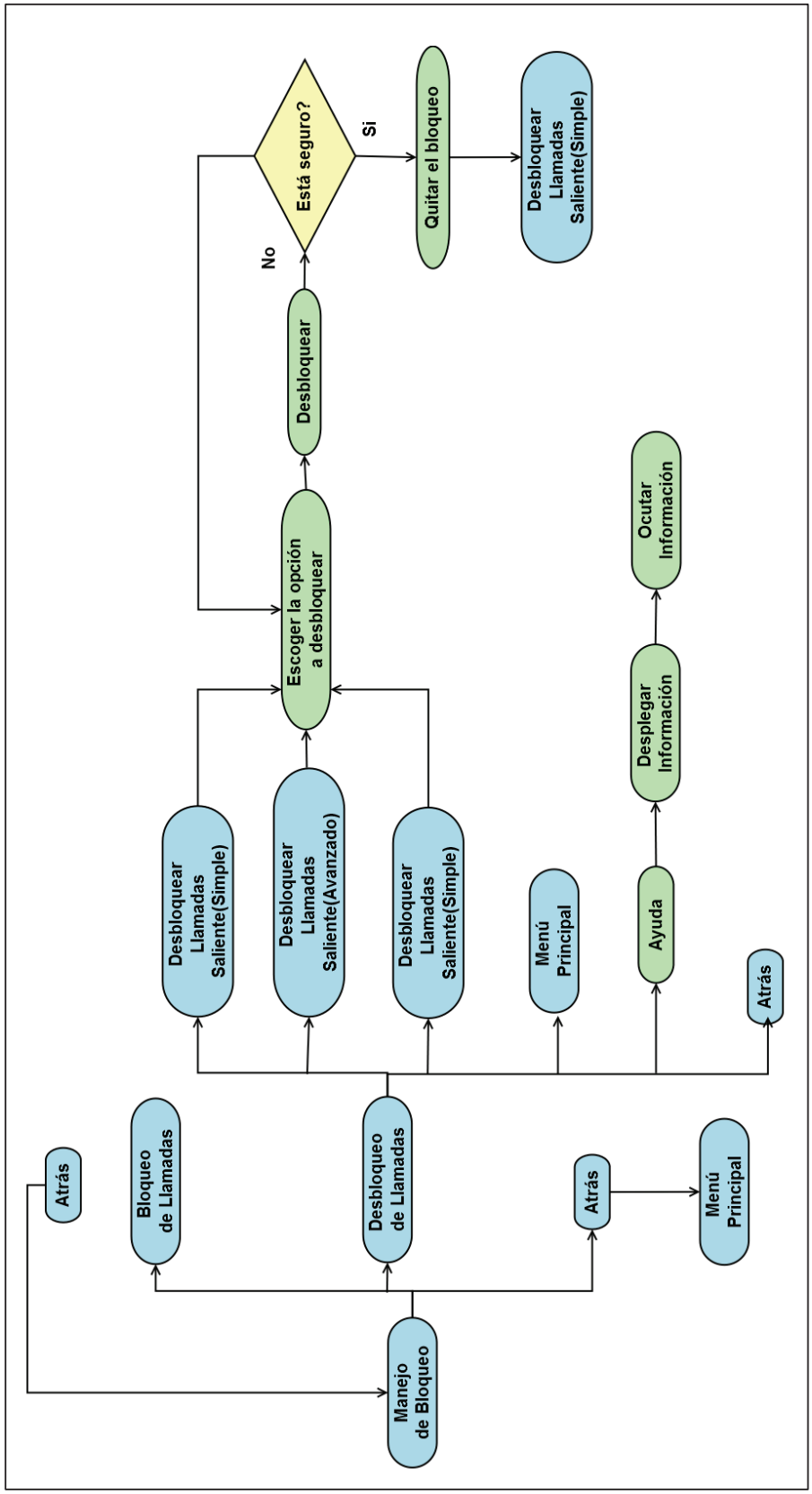

Figura 2.11 Diagrama de Actividades, Desbloqueo de Llamadas

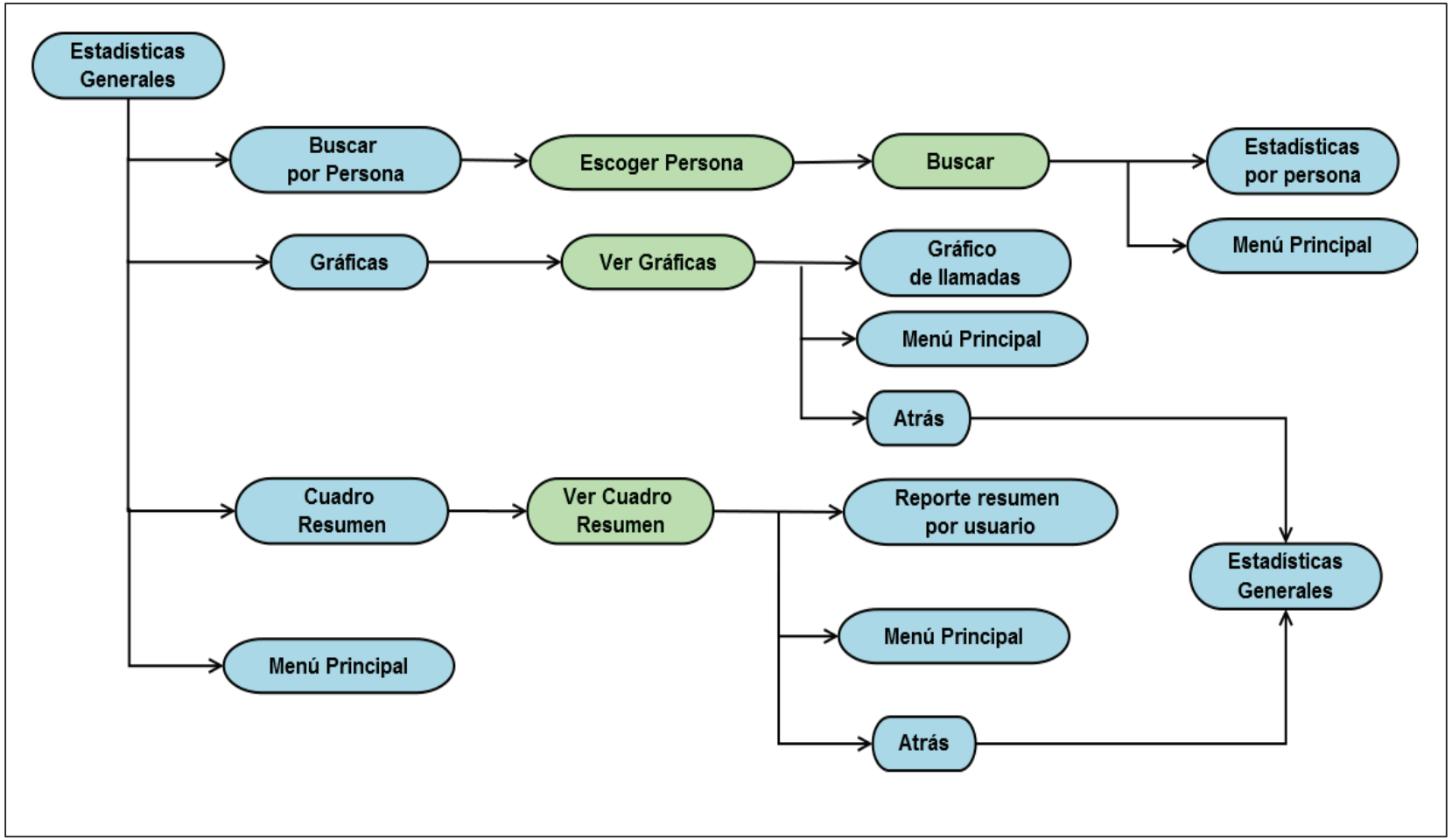

Figura 2.12 Diagrama de Actividades, Estadísticas Generales

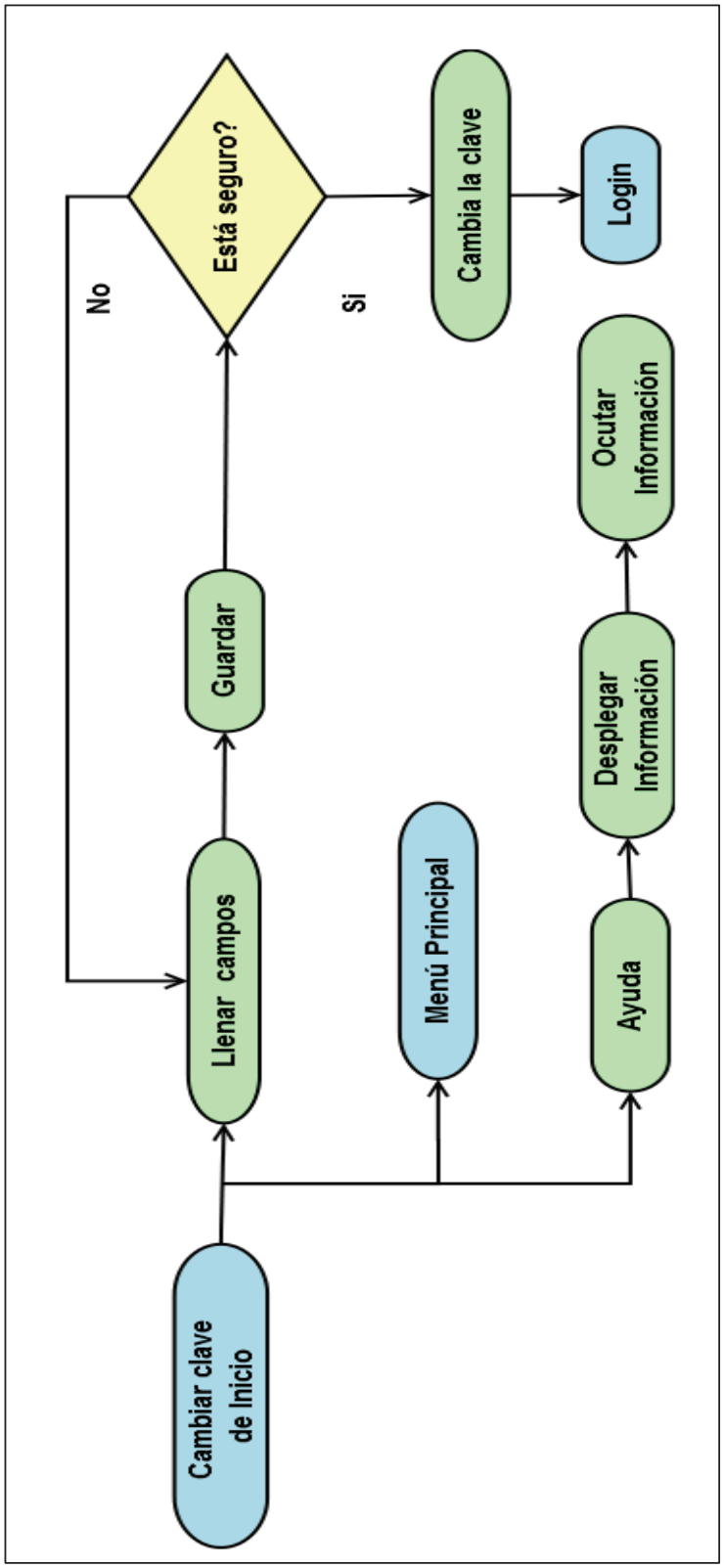

Figura 2.13 Diagrama de Actividades, Cambiar Clave de Inicio

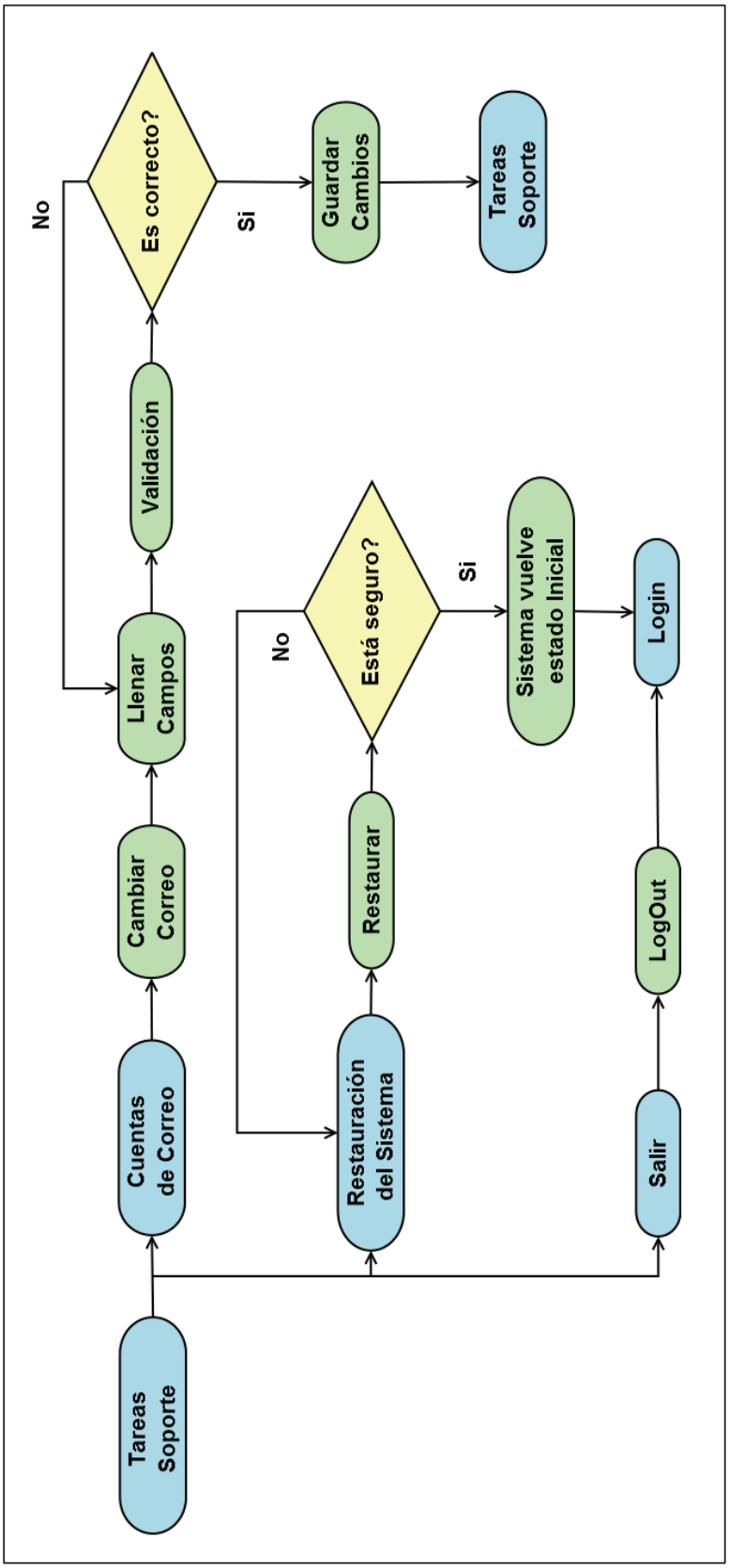

Figura 2.14 Diagrama de Actividades, Tareas Soporte

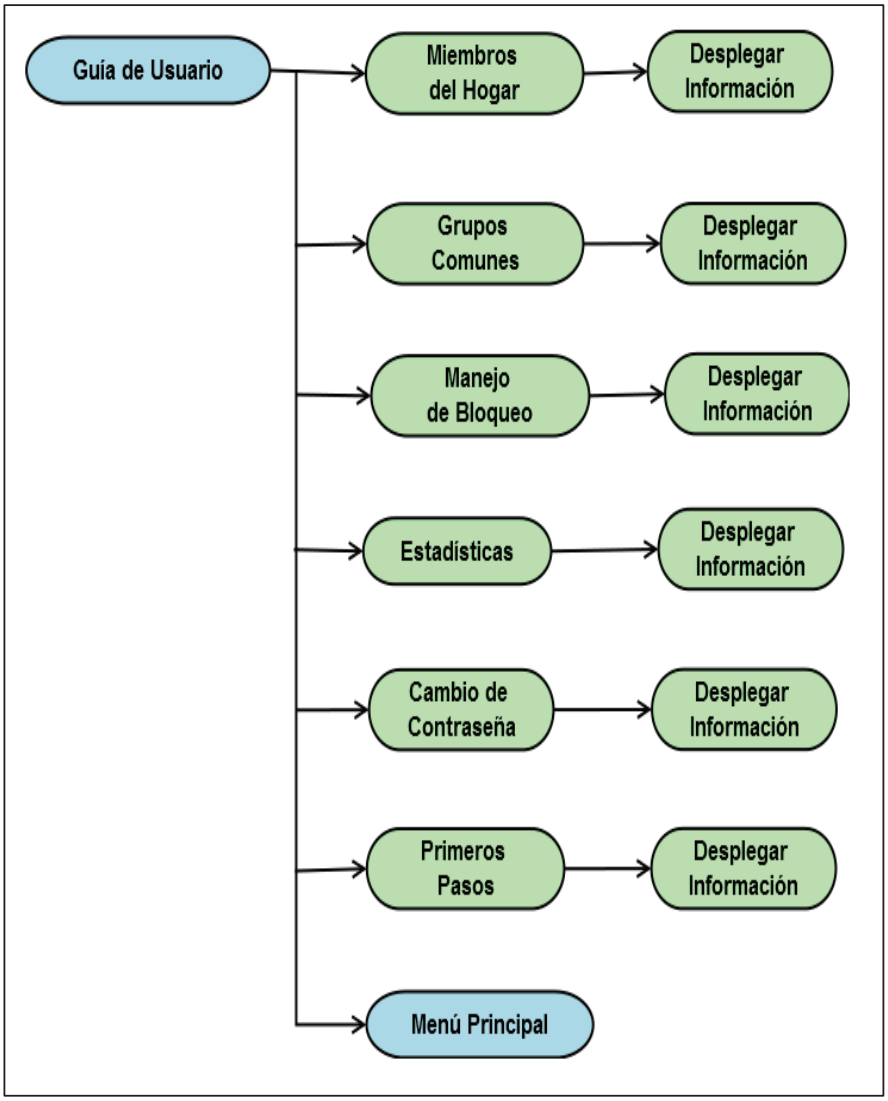

Figura 2.15 Diagrama de Actividades, Guía de Usuario

2.2.3 NIVEL PRESENTACIÓN

Las interfaces gráficas del prototipo tuvieron tres iteraciones relacionadas a las historias de usuario que se presentaron en la **sección 2.1.2**. En primer lugar se realizó un bosquejo de las interfaces gráficas iniciales las cuales pueden ser vistas en el Anexo B, al igual que las diferentes interfaces en las iteraciones previas.

En base a los requisitos manifestados por los usuarios después de realizar cada iteración, se obtuvieron las siguientes interfaces gráficas mostradas desde la **Figura 2.16** hasta la **Figura 2.35.**

Figura 2.16 Interfaz Final, Inicio de Sesión

Figura 2.17 Interfaz Final, Administración de Usuarios

Figura 2.18 Interfaz Final, Menú Principal

Figura 2.19 Interfaz Final, Agregar Miembro del Hogar

Figura 2.20 Interfaz Final, Personas Existentes

Figura 2.21 Interfaz Final, Manejo de Bloqueo

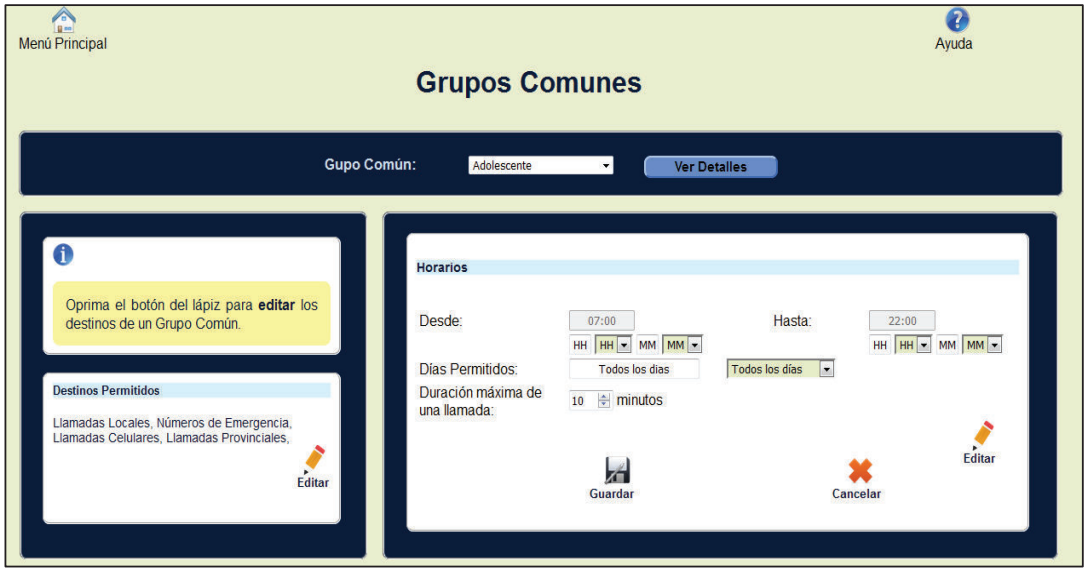

Figura 2.22 Interfaz Final, Grupos Comunes

Figura 2.23 Interfaz Final, Bloqueo de Llamadas Salientes (Avanzado)

Figura 2.24 Interfaz Final, Bloqueo de Llamadas Salientes (Simple)

Figura 2.25 Interfaz Final, Bloqueo de Llamadas Entrantes

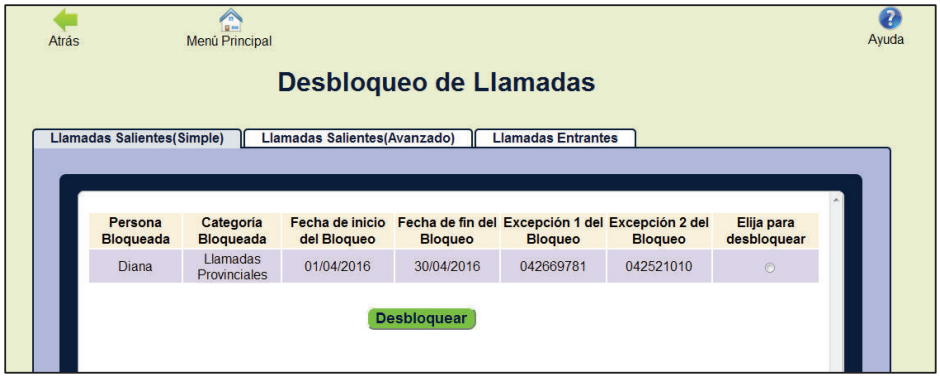

Figura 2.26 Interfaz Final, Desbloqueo de Llamadas

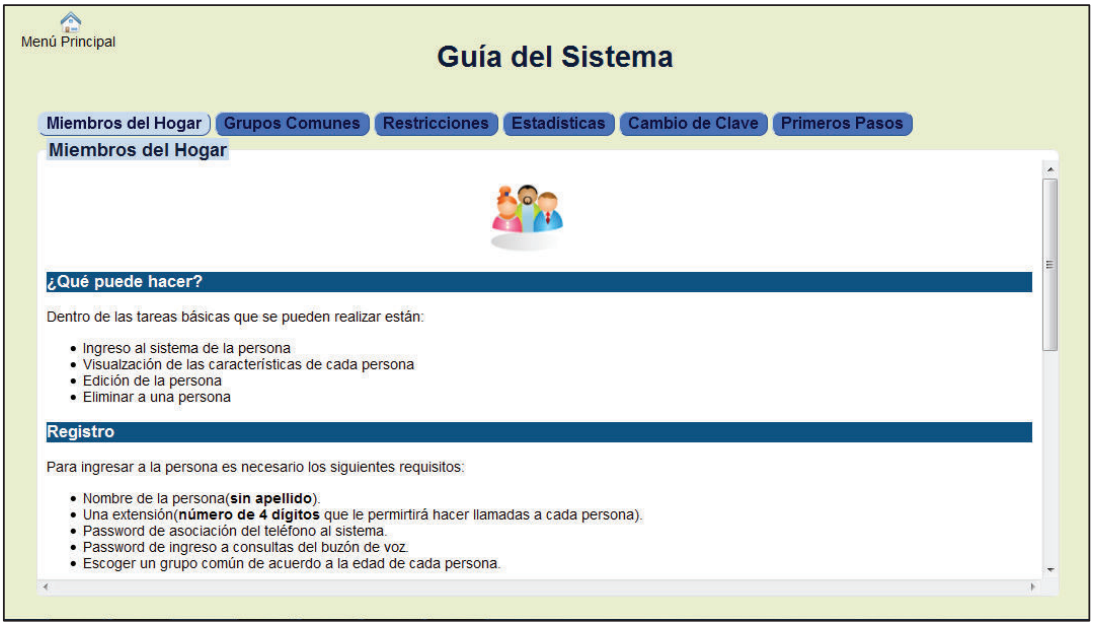

Figura 2.27 Interfaz Final, Guía del Sistema

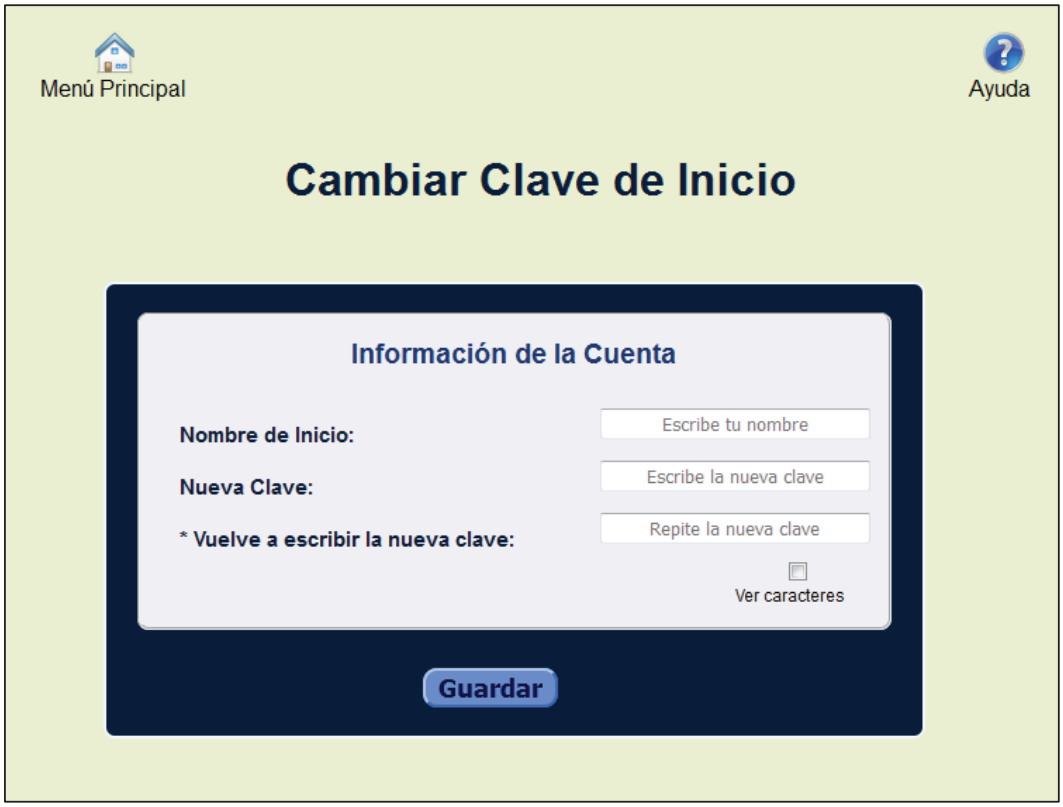

Figura 2.28 Interfaz Final, Cambiar Clave de Inicio

Figura 2.29 Interfaz Final, Estadísticas Generales Totales

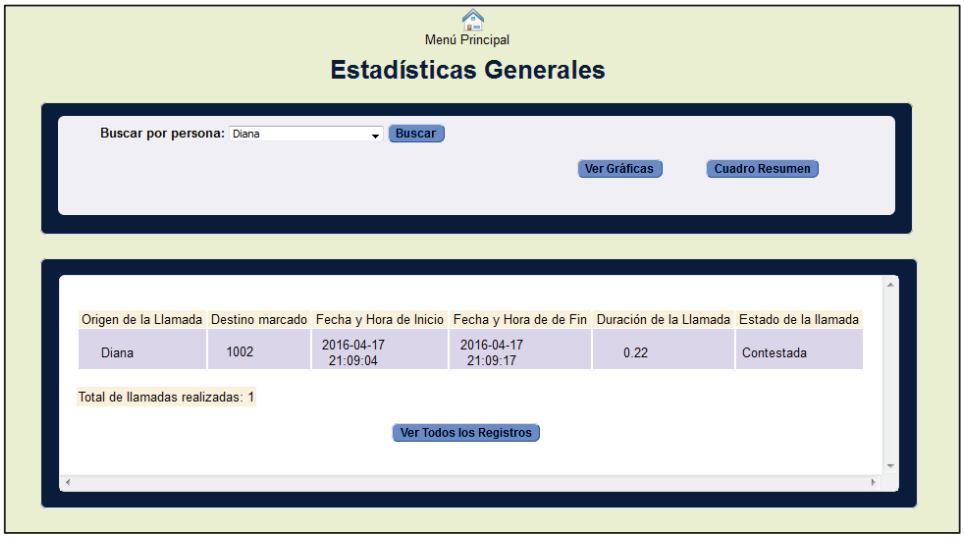

Figura 2.30 Interfaz Final, Estadísticas de Llamadas Específicas

Figura 2.31 Interfaz Final, Cuadro Resumen de Llamadas

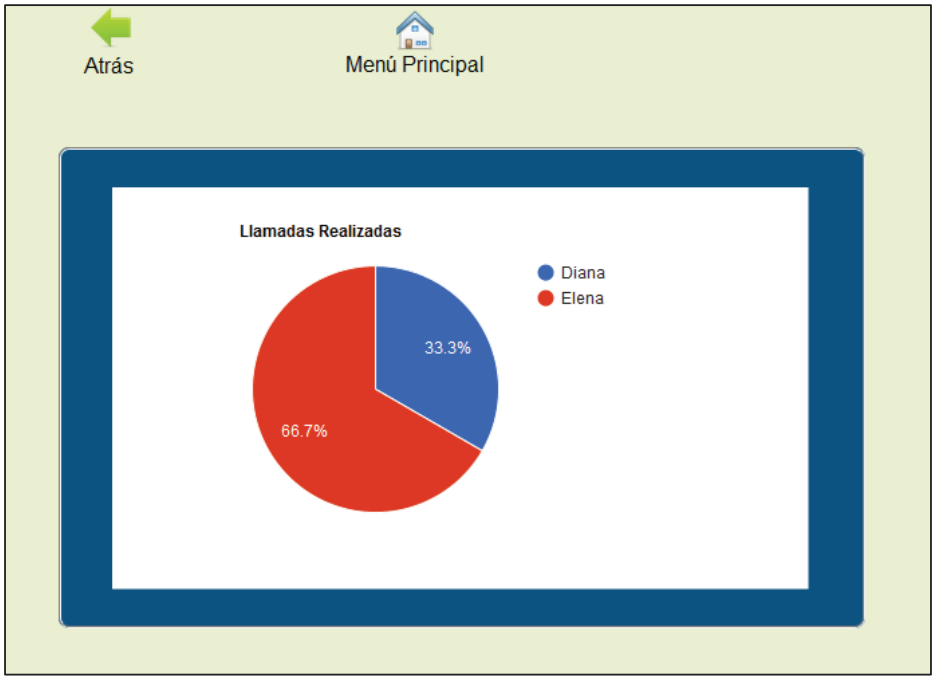

Figura 2.32 Interfaz Final, Gráficas

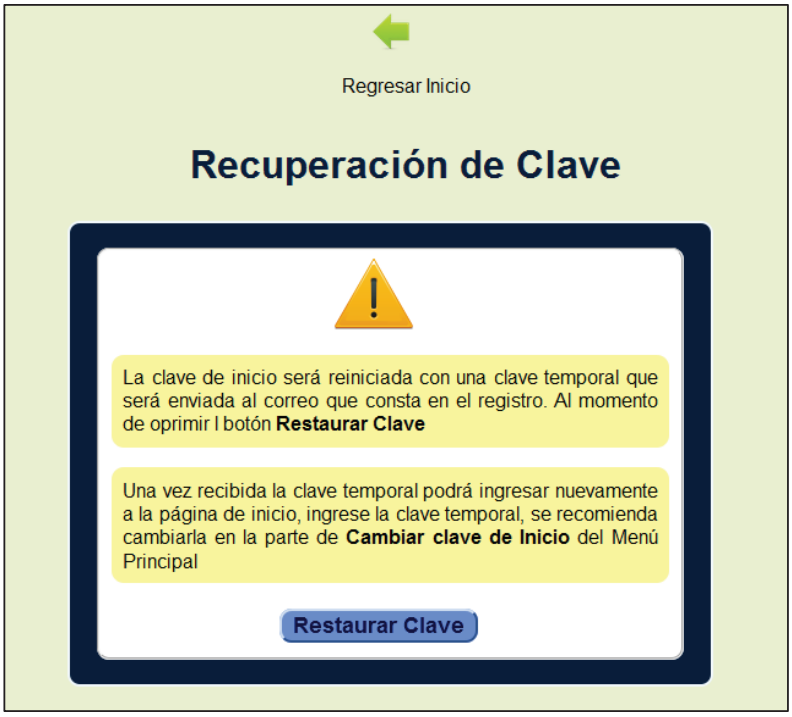

Figura 2.33 Interfaz Final, Recuperación de Clave de Inicio

En el prototipo además surgió la idea de implementar tareas de mantenimiento y soporte, para lo cual se realizó dos interfaces gráficas dedicadas exclusivamente a desempeñar esas tareas como se muestran en la **Figura 2.34** y la **Figura 2.35**.

Dichas tareas ya no serían ejecutadas por el padre de familia, el principal usuario manipulador del sistema, si no por un usuario considerado de soporte.

Figura 2.34: Interfaz Final, Inicio de Sesión Usuario de Soporte

Figura 2.35 Interfaz, Tareas de Soporte

2.2.4 ELECCIÓN DE LA PLATAFORMA DE TELFONIA IP PARA RASPBERRY

Para implementar el servicio de telefonía se analizaron tres opciones *Open Source* que podrían utilizarse dentro del prototipo.

La primera opción que se consideró fue la versión de Elastix para Raspberry Pi denominada micro Elastix, la cual presenta todas las características de Elastix mencionadas en la **sección 1.1.4.2** exceptuando los módulos de *Call Center* y Mensajería Instantánea [24]. Micro Elastix provee una imagen ISO[14] donde se encuentran todos los paquetes necesarios para la instalación en la tarjeta SD de la placa Raspberry Pi.

Micro Elastix también posee una interfaz gráfica de administración y configuración, a la cual se accede vía web. La segunda opción que se consideró fue RasPBX, una distribución de Raspbian que contiene instalados tanto Asterisk 11 como FreePBX 13, además provee dos opciones adicionales y opcionales: HylaFAX[15] y Fail2Ban[16].

Para la elección de la herramienta que se va a emplear, se consideró las características previamente mencionadas y la comparación de la **Tabla 2.17** que contiene las principales diferencias entre cada opción analizada.

Parámetro	Distribución		
	Asterisk Puro	**MicroElastix**	**RasPBX**
Licencia	Open Source	Open Source	Open Source
Administración	Por consola	Vía web	Vía web
Sistema Operativo	Centos, Ubuntu, Raspbian	Distribución basada en Centos	Distribución basada en Raspbian
Herramientas	Telefonía	Telefonía,Mensajería instantánea, Video conferencia, entre otros	Telefonía, marcado por voz, Mensajes SMS

Tabla 2.17 Comparación de herramientas para Telefonía

Las dos opciones descritas anteriormente presentan una visión empresarial al incluir muchos servicios como los detallados en las secciones **1.1.4.2** y **1.1.4.3**, y los mostrados en la **Tabla 2.17**. Uno de los factores a considerar es la interfaz gráfica de administración que poseen las versiones de RasPBX y Micro Elastix, la cual es muy sobrecargada y resulta compleja de manipular para un usuario sin

[14]Imagen ISO: Archivo que es copia de un sistema ficheros basado en la norma ISO 9660, utilizados generalmente para distribuir Sistemas Operativos
[15] HylaFAX: Sistema tipo cliente-servidor utilizado para enviar y recibir fax
[16] Fail2BAN: Paquetes de Linux utilizados para escanear archivos logs para luego bloquear IPs encontradas en los archivos utilizando IpTables.

conocimientos técnicos. Esta es una desventaja el usuario padre de familia quien debe manipular la interfaz.

Por lo tanto, dichas herramientas no son viables para ser utilizadas en un ambiente doméstico, por este motivo se decidió utilizar a Asterisk puro y desarrollar una interfaz de administración mucho más sencilla y fácil de utilizar, dirigida a cualquier tipo de personas, y sobretodo que no requiera tener conocimientos técnicos. A la vez si se empleara RasPBX o Micro Elastix, se desperdiciaría recursos ya que el desarrollo del prototipo únicamente requiere herramientas de telefonía.

CAPÍTULO 3

3 IMPLEMENTACIÓN Y PRUEBAS DEL PROTOTIPO

3.1 ESQUEMA GENERAL DEL PROTOTIPO

El prototipo se encuentra formado por una Central Telefónica IP, una interfaz de administración web, un gateway de voz utilizado para conectar el prototipo a la PSTN, clientes SIP y un equipo de conectividad para comunicar todos los dispositivos en una misma red.

Para la elección de los dispositivos que conforma el prototipo se consideró un entorno doméstico en el cual el tráfico de llamadas no va a ser muy alto por el reducido número de personas, además la solución que se plantee debe ser de bajo costo para facilitar y motivar su adquisición a la población.

Tomando en cuenta los parámetros antes mencionados, se escogió Raspberry Pi, para implementar la Central Telefónica debido a su tamaño, bajo costo y características de funcionamiento detalladas en la sección **1.2**. El gateway de voz que se seleccionó fue Grandstream HT503 ya que presenta un puerto FXO necesario para la conexión a la PSTN y puertos Ethernet que permiten anexarlo en la red.

Los clientes SIP que se requieren deben ser implementados en un softphone que se pueda instalar en una laptop o un teléfono inteligente, los cuales son los dispositivos tecnológicos que más utilizan por las personas en la actualidad. Se eligió a **Zoiper**, un softphone que se adapta al escenario planteado, pues es fácil de utilizar, es multiplataforma y Open Source.

Para finalizar, el prototipo necesita un módulo que permita conectar todos los dispositivos mencionados previamente en red. Para la conexión se escogió un router inalámbrico, puesto que permite conectar dispositivos alámbricos e

inalámbricos. Los elementos mencionados anteriormente se encuentran conectados entre sí, como se muestra en la topología de la **Figura 3.1.**

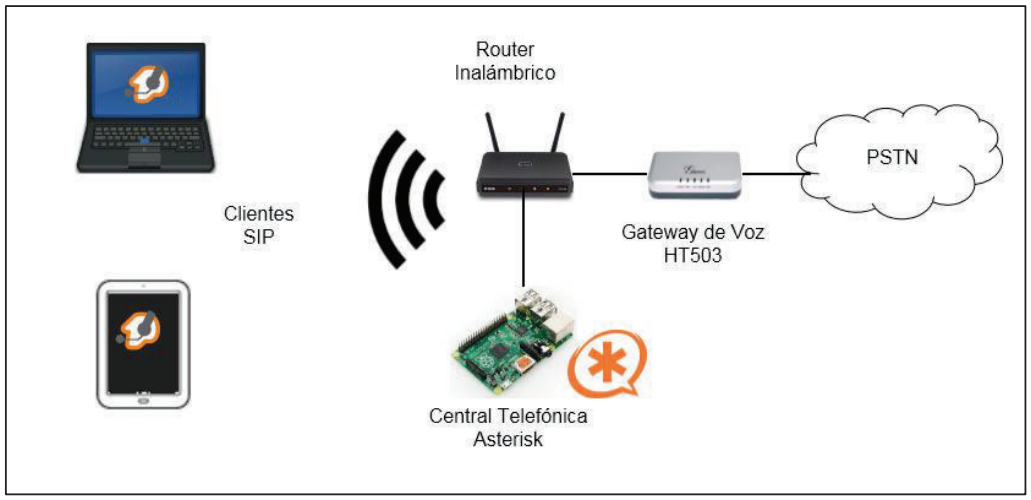

Figura 3.1 Topología Física Final del Prototipo

3.2 CONFIGURACIÓN DE RASPBERRY PI

Para configurar el Raspberry Pi debe considerar los siguientes pasos:

1. Descargar el Sistema Operativo Raspbian, del sitio oficial de Raspberry Pi: https://www.raspberrypi.org/downloads/raspbian/

2. Una vez descargado el Sistema Operativo, se procede a grabarlo en la tarjeta SD utilizando la herramienta Win32DiskImager [25], como muestra **Figura 3.2**.

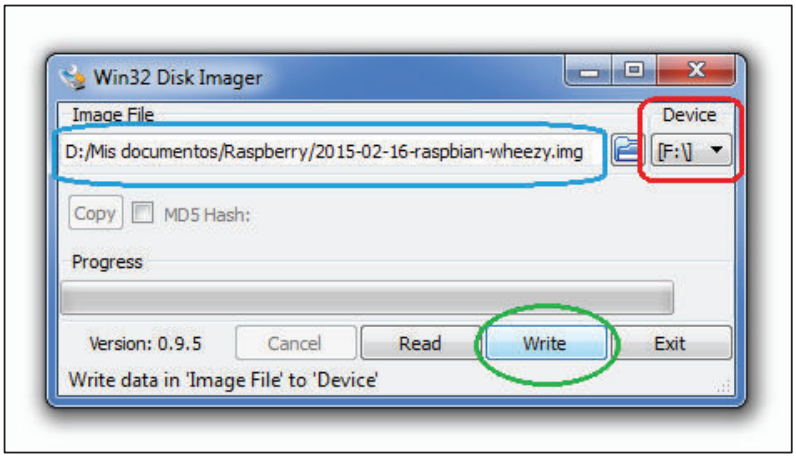

Figura 3.2 Escritura de la Imagen Raspbian en tarjeta SD

3. Una vez finalizada la escritura de la imagen del sistema operativo se inserta la tarjeta SD en la ranura respectiva dentro del Raspberry Pi y se conecta el Raspberry con el teclado, mouse y una pantalla para comenzar la configuración, ver **Figura 3.3**.

Figura 3.3 Conexión de Raspberry Pi

El Raspberry Pi arrancará sobre la imagen almacenada en la tarjeta SD e inmediatamente después aparecerá el menú mostrado en la **Figura 3.4** para la configuración correspondiente.

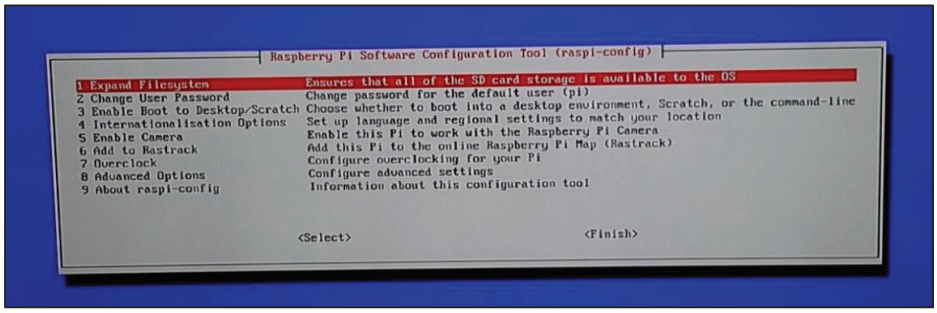

Figura 3.4 Pantalla de Configuración Inicial de Raspberry Pi

4. Existen 9 opciones de configuración, en la implementación únicamente se realizó la configuración para el teclado y la selección de la región.

5. A continuación se puede configurar la expansión del sistema operativo en toda la tarjeta SD para lo cual se escoge la opción 1 de la pantalla de Configuración Inicial mostrada en **Figura 3.4**, con el fin de ocupar todo el espacio de memoria disponible.

6. Una vez concluidos los pasos anteriores, se oprime el botón finalizar de la pantalla de Configuración Inicial.

7. Después de haber instalado Raspbian y realizado las configuraciones generales se inicia la sesión en la cuenta por defecto **Pi**, cuya clave por defecto es **raspberry**.

8. Configuración de una dirección IP estática, para poder acceder remotamente a Raspberry Pi vía SSH, esto se lo realiza editando el archivo **interfaces** ubicado en la ruta **/etc/network/interfaces**. En el Anexo C del manual de instalación se explica el proceso completo para configurar la dirección IP estática.

3.3 IMPLEMENTACIÓN Y CONFIGURACIÓN DEL MÓDULO DE COMUNICACIÓN

3.3.1 IMPLEMENTACIÓN DE LA CENTRAL TELEFÓNICA ASTERISK

Para la implementación de la Central Telefónica Asterisk, en primer lugar se debe instalar Asterisk, esto se lo puede hacer ejecutando el comando **`sudo apt-get install asterisk`**.

El proceso de instalación se realiza de manera automática exceptuando el paso en el que se debe escoger el código del país, por lo tanto para el presente proyecto de ingresó el código 593 perteneciente a Ecuador, ver **Figura 3.5**.

Antes de realizar la configuración de la Central Telefónica, se deben instalar los paquetes de audio en español para los distintos mensajes de voz que se requieren implementar, mediante los comandos mostrados en el **Código 3.1.**

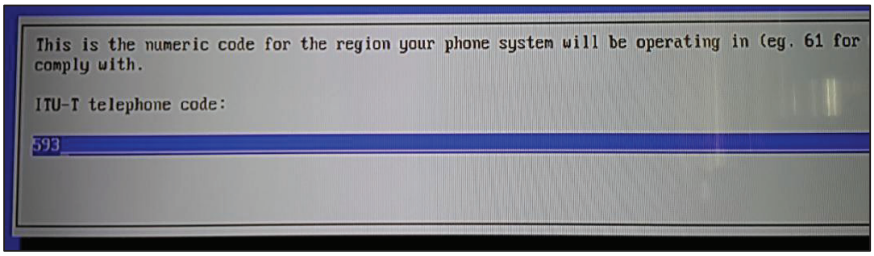

Figura 3.5 Pantalla de Especificación de Código Telefónico del País

```
wget -O core.zip http://www.asterisksounds.org/es-
ar/download/asterisk-sounds-core-es-AR-sln16.zip

wget -O extra.zip http://www.asterisksounds.org/es-
ar/download/asterisk-sounds-extra-es-AR-sln16.zip
```

Código 3.1 Comandos para descargar paquetes de audio en español para Asterisk

Una vez descargados los archivos y descomprimidos, se recomienda cambiar el formato al que utilice cada códec de acuerdo a la configuración del canal SIP o IAX2.

La configuración de Asterisk será gestionada por el padre de familia mediante la interfaz gráfica, dentro de Asterisk se deben configurar los archivos **sip.conf**, **extensions.conf** y **voicemail.conf**, los cuales vienen por defecto en la instalación. Como buena práctica, se renombró a estos archivos originales para trabajar con otros archivos nombrados como los archivos originales, de manera que si existe un error de edición o configuración, se pueda regresar a la condición inicial.

La configuración de los archivos nuevos debe contener cierta información ya establecida que el usuario no pueda cambiar o manipular. Para el archivo **sip.conf** se define la configuración mostrada en el **Código 3.2**.

La mayoría de los parámetros mostrados en el archivo **sip.conf** fueron explicados en la **sección 1.1.4.1.9**, a excepción de la sección **ht503fxo** la cual es la encargada de la configuración de la conexión con el Gateway de Voz.

Dentro de la configuración más importante en **ht503fxo** se encuentra la dirección IP del Gateway de Voz y el puerto 5062 que utiliza por defecto el Gateway HT503 empleado.

El archivo **extensions.conf** que contiene el plan de marcado, contiene los contextos por defecto especificados en **Tabla 3.1**. Dichos contextos son de carácter permanente, no pueden ser eliminados, pero si pueden ser editados.

sip.conf

```
[general]
context=unauthenticated ; contexto por defecto para las llamadas entrantes
allowguest=no ; evitar llamadas no autenticadas
srvlookup=yes ; habilitar busqueda del registro DNS SRV en llamadas salientes
udpbindaddr=0.0.0.0 ; escuchar peticiones UDP en todas las interfaces
tcpenable=no ; deshabilitar soporte TCP
language=es
;------------------------------------------------------------------------------
[ht503fxo]
type=peer
canreinvite=no
host=192.168.0.160
nat=no
port=5062
disallow=all
allow=alaw
allow=ulaw
allow=all
dtmfmode=inband
relaxdtmf=yes
qualify=yes
context=LlamadasInternas
username=MiCentral
;------------------------------------------------------------------------------
;AgregarCanales
```

Código 3.2 Configuración Inicial del Archivo sip.conf

Contexto	Propósito
Interno	Permite la comunicación de cada usuario dentro de la red interna.
Llamadas Internas	Analiza la llamada que ingresa de la PSTN y la dirige al usuario adecuado dependiendo de la extensión que reciba.
Salida Llamadas	Permite enviar la llamada hacia la PSTN.
nino	Permite especificar los destinos permitidos por el perfil niño.
adolescente	Permite especificar los destinos permitidos por el perfil adolescente.
adulto	Permite especificar los destinos permitidos por el perfil adulto.
tiempoNinio	Especifica el tiempo máximo de duración de una llamada de un usuario que pertenezca al perfil niño.
tiempoAdolescente	Especifica el tiempo máximo de duración de una llamada de un usuario que pertenezca al perfil adolescente.
tiempoAdulto	Especifica el tiempo máximo de duración de una llamada de un usuario que pertenezca al perfil adulto.

Tabla 3.1 Contextos por defecto del Prototipo

Al momento de gestionar el plan de marcado desde la interfaz gráfica, se crearán los contextos necesarios para el funcionamiento completo del prototipo.

Los requisitos de la **Tabla 2.1** se tomaron en cuenta para dejar algunos valores por defecto en los contextos por defecto del plan de marcado, los cuales se especifican en la **Tabla 3.2**.

Perfil	Horario Permitido		Duración Máx. Llamada
Niño	13:00-20:00	Lunes-Viernes	3 minutos
Adolescente	7:00-22:00	Todos los días	10 minutos
Adulto	00:00-23:55	Todos los días	30 minutos

Tabla 3.2 Valores por defecto de los Perfiles utilizados en el plan de marcado

El archivo **voicemail.conf** se configurará durante la operación del prototipo, es decir conforme el padre de familia vaya registrando a los miembros de su familia, se irán añadiendo las líneas correspondientes a la configuración del servicio de correo de voz.

3.3.2 CONFIGURACIÓN DEL GATEWAY DE VOZ

El Gateway de Voz que se utilizó para la conexión a la PSTN es un Gateway Grandstream HT503 que dispone de dos puertos RJ45 LAN y WAN, dos puertos RJ11 uno para FXS[17] y otro para FXO[18], un botón de reinicio y la entrada de alimentación energética, como se muestra en la **Figura 3.6.**

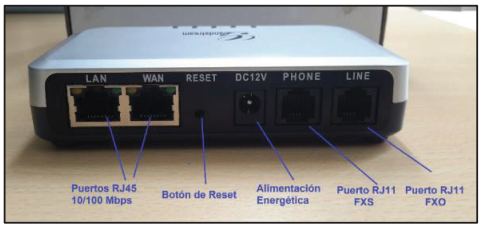

Figura 3.6 Estructura de Gateway de Voz Grandstream HT503

[17] FXS (Foreign Exchange Station): Puerto que actúa como una central puesto que genera timbrado, debe ser conectado a un dispositivo que detecta tono como un teléfono analógico.
[18] FXO (Foreign Exchange Oficce): Puerto que actúa como un teléfono analógico que detecta una señal de timbrado, debe ser conectado a un dispositivo que genere tono como por ejemplo una línea telefónica de la PSTN.

Para configurar el Gateway de Voz se ingresa al dispositivo vía web, utilizando la dirección IP por defecto que en este caso es 192.168.2.1, y que corresponde a la interfaz LAN. Una vez ingresado a dicha dirección IP se abrirá una pantalla de autenticación como la mostrada en la **Figura 3.7**.

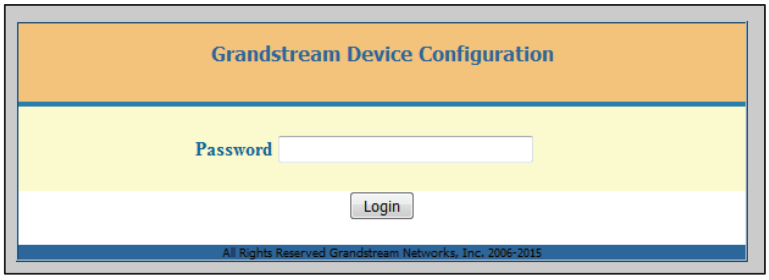

Figura 3.7 Pantalla de inicio del Gateway de Voz Grandstream HT503

Para ingresar al menú de configuración se utiliza la clave por defecto que es **admin**. Dentro del menú de configuración se realizaron las siguientes tareas:

1. Configuración de una dirección IP estática para el puerto WAN en la opción ***Basic Settings***, ver **Figura 3.8**.

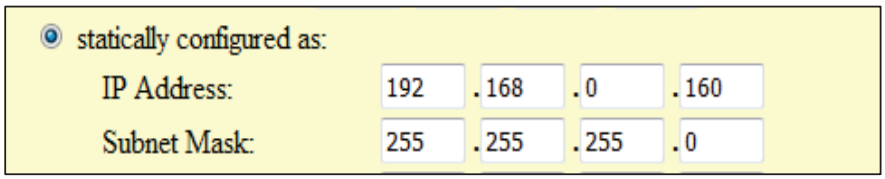

Figura 3.8 Configuración de una Dirección IP para el Gateway de Voz

2. Configuración de la dirección IP de la Central Telefónica del prototipo (dirección IP de Raspberry Pi) dentro de la opción ***Basic Settings***, ver **Figura 3.9**.

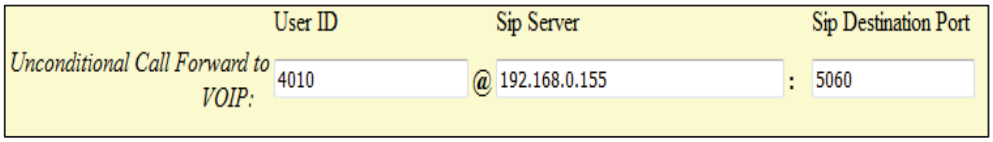

Figura 3.9 Vinculación con la Central Telefónica

3. Configuración de las frecuencias de operación del equipo para que funcione adecuadamente con la PSTN en Ecuador conforme se muestra en la **Figura 3.10**, este proceso se lo realiza en la opción ***Advanced Settings*** del menú de configuración [26].

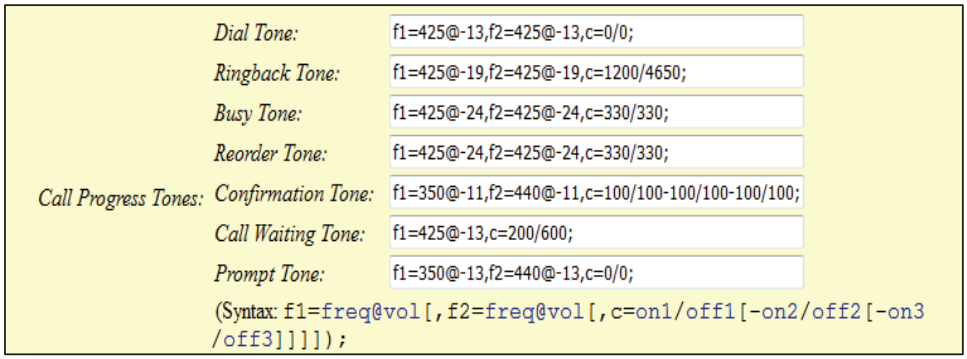

Figura 3.10 Configuración de las Frecuencias de Trabajo del Gateway de Voz

3.4 IMPLEMENTACIÓN DE LA APLICACIÓN DE GESTIÓN

3.4.1 LENGUAJE DE PROGRAMACIÓN UTILIZADO

El lenguaje de programación empleado para la aplicación web fue Python, el cual es un lenguaje interpretado, orientado a objetos que se caracteriza por ser amigable, fácil de aprender, flexible y portable [27], [28].

Python tiene una sintaxis simple, el código es mucho más fácil de entender y escribir comparado con otros lenguajes como PHP[19]. Además es utilizado por un gran número de desarrolladores y presenta herramientas de depuración del lenguaje que son muy fáciles de acceder [29]. Adicionalmente viene instalado por defecto en las distribuciones de Linux, incluyendo a Raspbian. Debido a estas características, se escogió a Python considerando la metodología de programación orientada a objetos.

3.4.2 FRAMEWORK DJANGO

Django es un Framework Web [20] del tipo *Open-Source* que permite crear y mantener una aplicación web de alta calidad con un mínimo esfuerzo. Aporta un alto nivel de abstracción en los patrones de desarrollo web comunes, atajos para las tareas más frecuentes de programación y convenciones claras sobre la forma de resolver los problemas [30].

---

[19] PHP (Hypertext Pre-processor): Es un lenguaje de programación orientado al desarrollo web de contenido dinámico.

[20] Framework Web: Es una infraestructura digital que permite crear aplicaciones web de manera mucho más sencillo.

3.4.2.1 Funcionamiento

Django se encuentra basado en MVC (*Model-View-Controller*), modelo que fue explicado en la **sección 1.3.2.3.** Django utiliza tres archivos basados en Python que son: **urls.py**, **views.py** y **models.py**, en conjunto con un template el cual es un archivo con extensión .html[21] y es utilizado para desarrollar la aplicación web.

El archivo **models.py** contiene una descripción de las tablas de una base de datos, cada tabla está representada por una clase de Python llamada modelo. Mediante este archivo se puede crear, actualizar y borrar registros de la base de datos utilizando código de Python en lugar de escribir las sentencias de SQL[22] [30].

En el **Código 3.3** se observa un ejemplo de un archivo **models.py** donde se encuentran dos modelos **Destino** y **LlamadaEntrante** con sus respectivos atributos.

Además cada uno contiene un método Unicode[23], el cual se utiliza para proporcionar una representación Unicode de un objeto[24] del modelo [31] .

models.py

```
from django.db import models
from django.db.models import Count
from django.template import defaultfilters
from django.db.models import Avg
from django.contrib.auth.models import User

class Destino(models.Model):


    numeroDestino = models.CharField(max_length=20,unique=True)
    categoriaNumero=models.CharField(max_length=25,null=True)

    def __unicode__(self):
        return self.numeroDestino

class LlamadaEntrante(models.Model):

    numeroEntrante = models.ForeignKey(Destino)
    estadoBloqueo= models.CharField(max_length=1)

    def __unicode__(self):
        return self.numeroEntrante
```

Código 3.3 Ejemplo de un Archivo models.py

[21] HTML (HyperText Markup Language): Lenguaje en el cual se encuentra escrito una página web.
[22] SQL(Structered Query Language): Lenguaje estándar utilizado para el acceso y manipulación de las bases de datos.
[23] Unicode: Sistema de codificación de caracteres utilizado para mostrar texto escrito en diferentes tipos de lenguajes.
[24] Objeto: Instancia de una clase en el paradigma Orientado a Objetos.

El archivo **views.py** es el encargado de realizar las tareas de lógica de negocios, es decir este archivo representa al controlador en MVC. Este archivo contiene funciones o métodos denominados controladores, como pueden observarse en el **Código 3.4**, se dispone de tres controladores **menuPrincipal**, **agregarPersona** y **manejoBloqueo**.

views.py

```
#Funcion para mostrar pagina de  menu principal
def menuPrincipal(request):
  if request.user.is_authenticated():
    return render(request,'menuPrincipal.html')
  else:
    return HttpResponseRedirect('/autRequerida/')

#Funcion para mostrar pagina de agregar nueva persona
def agregarPersona(request):
    context_listar={}
    listar=Perfil.objects.all()
    context_listar['tablas']=listar
    if request.user.is_authenticated():

      return render(request,'agragarPersona.html',context_listar)
    else:
      return HttpResponseRedirect('/autRequerida/')

#Funcion para mostrar pagina de manejo de bloqueo
def manejoBloqueo(request):
    if request.user.is_authenticated():
      return render(request,'restricciones.html')
    else:
      return HttpResponseRedirect('/autRequerida/')
```

Código 3.4 Ejemplo de Archivo views.py

Las tareas que desempeñan los controladores del ejemplo son mostrar las páginas web adecuadas cuando les llega una petición, previamente se realiza una comprobación de que el usuario ha iniciado sesión para poder mostrarle la página web correspondiente.

En el caso del controlador **agregarPersona**, para mostrar la página web se necesita tener información de la base de datos mediante una consulta, específicamente al modelo **Perfil**.

El siguiente archivo **urls.py** contiene un nexo entre la url llamada en una petición y el controlador asociado, tal como puede observarse en el **Código 3.5.**

```
from django.conf.urls import patterns, url
from telefonia import views

urlpatterns = patterns('',

        url(r'^iniciar/$',views.iniciar,name='iniciar'),
        url(r'^salir/$',views.salir,name='salir'),
        url(r'^soporte/$',views.soporte,name='soporte'),
```

Código 3.5 Ejemplo de archivo urls.py

En el ejemplo mostrado en el **Código 3.5** se observan las urls: **iniciar**, **salir** y **soporte** (color amarillo), las cuales son vinculadas con el controlador del mismo nombre respectivamente.

3.4.2.2 Creación de una Aplicación en Django

Antes de crear una aplicación en Django se debe crear un proyecto el cual es un conjunto de configuraciones de una instancia de Django, en el que se incluyen la configuración de la base de datos, configuraciones específicas de Django y de la aplicación [32].

A continuación se explicarán los pasos utilizados para la creación del proyecto.

1. Crear una carpeta en el sitio donde se desee crear el proyecto, ver **Código 3.6**

```
root@ubuntu:/home/juanm/Documentos# mkdir ProyectosDjango
```

Código 3.6 Creación de Carpeta para Proyectos de Django

2. Dirigirse a dicha carpeta y crear el proyecto mediante el comando mostrado en el **Código 3.7**.

```
django-admin.py startproject miProyecto
```

Código 3.7 Comando de Creación de un proyecto en Django

   El comando ejecutado previamente creará un archivo llamado **manage.py** y una carpeta con el mismo nombre del proyecto, en este caso **miProyecto** que contiene tres archivos **_init_.py**, **settings.py**, **urls.py** y **wsgi.py** como lo muestra **Figura 3.11**.

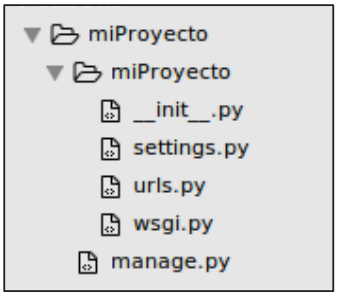

Figura 3.11 Estructura de Archivos de un Proyecto en Django

El archivo **manage.py**, considerado una utilidad para interactuar con el proyecto en sí, no debe ser modificado. La carpeta interna con el nombre **miProyecto** contiene los paquetes de Python necesarios para el proyecto.

El archivo **__init__.py** es un archivo que permite tratar al directorio **miProyecto** como un módulo de Python, el archivo **urls.py** contiene la información de las urls asociadas al proyecto y es considerado como la tabla de contenidos del proyecto de Django [32].

El archivo **settings.py** es un archivo de configuración del proyecto, contiene información de la ubicación de los archivos media como imágenes o videos que se van a utilizar en el proyecto, o incluso contiene la ubicación de los archivos html de las páginas web, entre otras cosas. Finalmente el archivo **wsgi.py** permite interactuar el proyecto con un servidor web WSGI[25] [32].

Una vez creado el proyecto, se procede a crear la aplicación mediante el comando mostrado en el **Código 3.8** donde **miAplicacion** es el nombre de la aplicación.

```
root@ubuntu:/home/juanm/Documentos/ProyectosDjango/miProyecto# python manage.py
startapp miAplicacion
```

Código 3.8 Creación de una Aplicación en Django

Al crear la aplicación se crean automáticamente los archivos necesarios de la aplicación entre los cuales se tiene **models.py**, **test.py**, **views.py**, **admin.py** e **__init__.py**; además crea una carpeta o directorio **migrations** como muestra la **Figura 3.12**

[25] WSGI: Es una interfaz simple y universal entre los servidores web y las aplicaciones web o frameworks.

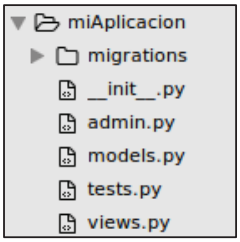

Figura 3.12 Estructura de archivos de una Aplicación en Django

Los archivos **models.py** y **views.py** cumplen las funciones ya detalladas en la **sección 3.4.2.1**, el archivo **admin.py** puede registrar los modelos creados y además crear una interfaz de administración, mientras que el archivo **test.py** permite guardar funciones para hacer pruebas de funcionamiento tales como pruebas unitarias.

El archivo **__init__.py** cumple con los mismos propósitos que el archivo del mismo nombre del proyecto y finalmente el directorio **migrations** guarda los cambios que se vayan haciendo a los modelos [33].

3.4.2.3 Desarrollo de una Aplicación

Una vez creada la aplicación se requiere hacer algunas configuraciones previas, las cuales se detallan a continuación.

1. Agregar la aplicación al proyecto, para lo cual se debe dirigir al archivo **settings.py** en el proyecto y añadir el nombre de la aplicación en **INSTALLED_APPS** como muestra el **Código 3.9**.

```
INSTALLED_APPS = (
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'miAplicacion',
)
```

Código 3.9 Agregar una Aplicación a un Proyecto

2. Tener una carpeta para almacenar todos los archivos media que se vayan a utilizar en la aplicación. Para este caso en concreto, se creó la carpeta **static**

para almacenar tanto las imágenes y los archivos con extensión css[26] y js[27], mismos que se utilizaron para la página web.

3. Una vez creada la carpeta mencionada en el paso anterior se debe especificar la ubicación de la carpeta en el proyecto de Django, para lo cual es necesario dirigirse a la parte final del archivo **settings.py** y añadir las líneas que muestra el **Código 3.10.**

```
STATIC_PATH = os.path.join(BASE_DIR,'static')
STATIC_URL = '/static/'
STATICFILES_DIRS = (
    STATIC_PATH,
)
```

Código 3.10 Especificaciones de la Ruta de los archivos Media

La variable **STATIC_PATH** permite crear una ruta a la carpeta donde se encuentran los archivos tipo media, **STATIC_URL** define una URL con la cual las aplicaciones de Django van a encontrar los archivos media cuando el servidor esté en funcionamiento y finalmente **STATICFILES_DIRS** especifica la ubicación de la carpeta **static** creado en el disco local [34].

4. Crear una carpeta donde ubicar los archivos con extensión html y vincular su ubicación en el archivo **settings.py,** este procedimiento es similar al que se realizó con la carpeta **static**, según muestra el **Código 3.11**.

```
TEMPLATE_PATH = os.path.join(BASE_DIR, 'templates')
TEMPLATE_DIRS = (TEMPLATE_PATH,)
```

Código 3.11 Especificaciones de Rutas de los Archivos html

5. Crear el archivo **urls.py** para la aplicación, mismo que se encargará de vincular la url que llega en una petición con el controlador adecuado. Esta tarea es opcional pero es considerada como buena práctica para manejar por separado al mapeo de URLs de distintas aplicaciones, lo único que se

[26] CSS (*Cascading Style Sheets*): Lenguaje utilizado para dar estilo a una página web.
[27] Js: Extensión de un archivo escrito en lenguaje JavaScript, el cual permite tener interfaces web más dinámicas.

debe hacer es vincular los archivos **urls.py** del proyecto y de la aplicación, para lo cual se configura el archivo **urls.py** del proyecto, ver **Código 3.12**

```
3
4   urlpatterns = patterns('',
5       # Examples:
6       # url(r'^$', 'miProyecto.views.home', name='home'),
7       # url(r'^blog/', include('blog.urls')),
8
9       url(r'^admin/', include(admin.site.urls)),
10      url(r'^',include('miAplicacion.urls')),
11  )
12
```

Código 3.12 Vinculación de los Archivos urls.py del Proyecto y la Aplicación

Una vez culminado el proceso anterior se puede probar la aplicación ejecutando el comando **`python manage.py runserver 192.168.0.155:8000`**, dentro de la carpeta de la aplicación. La ejecución de dicho comando iniciará el Servidor de Django con la dirección IP y puerto especificados en el comando.

3.4.3 IMPLEMENTACIÓN DEL NIVEL DE DATOS

Se desarrolló una aplicación de Django denominada Telefonía de acuerdo al procedimiento mostrado en los puntos anteriores.

3.4.3.1 Configuración de la Bases de Datos

Puesto que una clase definida en el archivo **models.py** en Django representa una tabla de una base de datos y una instancia de esa clase representa un registro en particular de dicha tabla. En base al diseño que muestra la **Figura 2.2**, se crearon los modelos necesarios que se van a emplear en el prototipo. Para crear los modelos se realizaron los siguientes pasos:

1. Definir los modeles en el archivo **models.py**, como muestra el **Código 3.13**, en este caso se ha definido el modelo **Persona**. Se puede observar claramente que hay un atributo llamado **nombrePerfil** el cual es una clave foránea[28] de la tabla **Perfil.**

[28] Clave foránea: Una clave primaria de una tabla que representa una columna de otra tabla, indica la relación entre dos tablas.

```
class Persona(models.Model):
    extension= models.CharField(max_length=4,unique=True)
    nombrePersona = models.CharField(max_length=20)
    contraseniaAsociacion= models.CharField(max_length=10, blank=True)
    contraseniaBuzon= models.CharField(max_length=4,blank=True)
    nombrePerfil= models.ForeignKey(Perfil)

    def __unicode__(self):

        return self.nombrePersona
```

Código 3.13 Modelo Persona

2. Dentro de la carpeta del proyecto, crear las tablas en la base de datos utilizando los comandos mostrados en el **Código 3.14.**

```
$ python manage.py makemigrations
$ python manage.py migration
```

Código 3.14 Comandos para Creación de un Modelo y Efectuar cambios en el mismo

Una vez aplicado los comandos, se mostrarán los resultados indicando que los cambios fueron realizados con éxito, ver **Figura 3.13**.

```
Migrations for 'telefonia':
  0002_persona.py:
    - Create model Persona

Operations to perform:
  Apply all migrations: admin, contenttypes, telefonia, auth, sessions
Running migrations:
  Applying miAplicacion.0002_persona... OK
```

Figura 3.13 Mensaje de confirmación de creación de la Tabla Persona

Las tablas restantes se crearon siguiendo el procedimiento explicado anteriormente. Cada tabla creada con sus respectivos atributos puede verse en la **Tabla 3.3**, además en el Anexo G se encuentra el diagrama de clases completo de la aplicación.

Tabla	Atributos	Tipo/Longitud
Persona	nombrePersona	CharField/20
	extensión	CharField/4
	contrseniaAsociacion	CharField/10
	contraseniaBuzon	CharField/4
	nombrePerfil	ForeingKey
Perfil	nombrePerfil	CharField/15
	horaInicio	TimeField
	horaFin	TimeField
	duracionMaximaLlamada	IntegerField
	horarioPermitido	CharField/25
DestinosPerfil	tipoDestino	CharField/25
	perfilAsociado	ForeingKey
BloqueoSimple	personaBloqueada	ForeingKey
	categoriaLlamada	CharField/25
	fechaInicio	DateField
	fechaFin	DateField
	excepcion1	CharField/20
	excepcion2	CharField/20
BloqueoAvanzado	personaBloqueada	ForeingKey
	destinoBloqueado	ForeingKey
	fechaInicio	DateField
	fechaFin	DateField
	horaBloqueoI	TimeField
	horaBloqueoF	TimeField
Destino	numeroDestino	CharField/20
	categoriaNumero	CharField/25
LlamadaEntrante	numeroEntrante	ForeingKey
	estadoBloqueo	CharField/25
InformacionLlamadas	origen	CharField/18
	destino	CharField/18
	fechaHoraInicio	CharField/20
	fechaHoraFin	CharField/20
	duracionLlamada	Decimal/5
	estadoLlamada	CharField/20
PerfilUsuario	nombreUsuario	CharField/20
	clave	CharField/20

Tabla 3.3 Resumen de los Modelos con sus respectivos atributos de la aplicación Telefonía

3.4.4 IMPLEMENTACIÓN DEL NIVEL LÓGICA DE NEGOCIOS

El nivel de Lógica de Negocios se encuentra representado por el archivo **views.py** del Framework de Django, como se había explicado anteriormente. Por ello se crearon los controladores que permiten realizar las tareas necesarias para el funcionamiento del prototipo.

Dichas tareas son: guardar usuarios, editar usuarios, eliminar usuarios, aplicación de los distintos tipos de bloqueo, realización de desbloqueo y visualización de un registro de llamadas.

A continuación se presentarán los ejemplos de los controladores más importantes que se implementaron en el prototipo, sin embargo el resto puede verse en el Anexo J referente al código fuente de la aplicación.

3.4.4.1 Creación de un Miembro del Hogar en el Sistema

La función **guardarUsuarioNuevo** que se observa en el **Código 3.15** se encarga de crear un usuario en el sistema. Esta función puede dividirse en tres partes: recolección de los datos necesarios para crear el usuario, verificación de un usuario repetido y creación del usuario.

La primera parte consiste en recolectar los datos de cada persona, provenientes del formulario presente en la página web respectiva, los cuales son: nombre, extensión, perfil, clave de asociación y clave de buzón.

```
def guardarUsuarioNuevo(request):
    nombreUsuario=request.POST.get('nombre')
    nombrepersona=darFormatoNombre(unicode(nombreUsuario))
    extension=request.POST.get('extension')
    passwordA=request.POST.get('contraseniaAso')
    passwordB=request.POST.get('contraseniaBuz')
    perfil=request.POST.get('perfilN')
    perfilEncontrado=Perfil.objects.get(nombrePerfil=perfil)
    existeP=True
    existeE=True
    try:
        personaExistente=Persona.objects.get(nombrePersona=nombreUsuario)
    except Exception, e:
        existeP=False
    try:
        personaExistente=Persona.objects.get(extension=extension)
    except Exception, e:
        existeE=False
    if existeP==True:
     print "ya existe la persona"
     return HttpResponseRedirect('/errorUnicaPersona/')
    else:
        if existeE==True:
          print "ya existe la persona"
          return HttpResponseRedirect('/errorUnicaExtension/')
        else:
          p=Persona(extension=extension,nombrePersona=nombreUsuario,
            contraseniaAsociacion=passwordA,contraseniaBuzon=passwordB,nombrePerfil=perfilEncontrado)
          p.save()
          persoG=Persona.objects.get(nombrePersona=nombreUsuario)
          pEnvio=Persona.objects.all()
          personaSip(passwordA,nombrepersona,extension)
          personaExtensions(extension,nombrepersona,perfilEncontrado.slug)
          personaVoiceM(extension,nombrepersona,passwordB)
          return redirect('/menuPrincipal/miembrosHogar/personasExistentes/')
```

Código 3.15 Función guardarUsuarioNuevo

Es necesario considerar que el usuario puede escribir algún nombre que incluya tildes, o incluso decida poner los dos nombres de uno de los miembros de su familia,

lo que puede traer inconvenientes en la configuración de los archivos de configuración de Asterisk.

Por esta razón se utilizó una función denominada **darFormatoNombre** para evitar este problema como se ve en el **Código 3.16**.

```
def darFormatoNombre(cadena):

    s = ''.join((c for c in unicodedata.normalize('NFD',unicode(cadena))
      if unicodedata.category(c) != 'Mn'))
    cadena=s.decode()
    cadenaF=cadena.replace(" ", "")
    return cadenaF
```

Código 3.16 Función darFormatoNombre

La función mencionada permite eliminar las tildes de una cadena que le llegue utilizando el módulo **unicodedata** de Python, y finalmente quitar el espacio en blanco que une los dos nombres. Por ejemplo la función cambiaría el nombre **María Fernanda** por **MariaFernanda**, en forma transparente para el usuario y de esta manera no se presentarán problemas en la escritura de los archivos de Asterisk.

La segunda parte del controlador verifica si los valores de las variables **nombreUsuario** y **extensión** ya existen en el sistema mediante el uso de excepciones. Para ese caso se le informará al padre de familia que dichos datos se encuentran repetidos y deberán ser cambiados, para esto el controlador devolverá la vista adecuada.

La última parte del controlador se encarga en concreto de la creación del usuario, primero se debe ingresar los valores a la tabla **Persona** con el **Código 3.17**, el cual muestra la manera de manipular los modelos en Django.

```
p=Persona(extension=extension,nombrePersona=nombreUsuario,
  contraseniaAsociacion=passwordA,contraseniaBuzon=passwordB,nombrePerfil=perfilEncontrado)
p.save()
```

Código 3.17 Código para guardar un Usuario en la Tabla Persona

Finalmente se deben escribir los archivos **sip.conf**, **extensions.conf** y **voicemail.conf**, con la ayuda de las funciones **personaSip**, **personaExtensions**, **personaVoiceM;** cada una de estas funciones puede observarse en el **Código 3.18**,**Código 3.19** y **Código 3.20** respectivamente.

```
def personaSip(clave,nombre,extension):
  cadenaComparativa=';todo\n'
  archivo=open("/etc/asterisk/sip.conf",'r')
  lineas=archivo.readlines()
  archivo.close()

  archivo=open("/etc/asterisk/sip.conf",'w')
  for li in lineas:
    archivo.write(li)
    if li==cadenaComparativa:

      archivo.write('['+nombre+'] ;'+nombre+'\n')
      archivo.write('type=friend ;'+nombre+'\n')
      archivo.write('context= '+nombre+' ;'+nombre+'\n')
      archivo.write('host=dynamic ;'+nombre+'\n')
      archivo.write('nat=yes ;'+nombre+'\n')
      archivo.write("secret=" +clave +';'+nombre+'\n')
      archivo.write('dtmfmode=auto ;'+nombre+'\n')
      archivo.write('disallow=all ;'+nombre+'\n')
      archivo.write('allow=ulaw  ;'+nombre+'\n')
      archivo.write('allow=alaw ;'+nombre+'\n')
      archivo.write("username="+nombre+';'+nombre+'\n')
      archivo.write("mailbox="+extension+'@default;'+nombre+'\n')

  archivo.close()
  subprocess.call(['service','asterisk','restart'])
```

Código 3.18 Función personaSip

```
def personaExtensions(extension,nombre,perfil):

  cadenaComparativa=';AgrearPersona\n'
  cadenaComparativa2='[Interno]\n'
  archivo=open("/etc/asterisk/extensions.conf",'r')
  lineas=archivo.readlines()
  archivo.close()
  archivo=open("/etc/asterisk/extensions.conf",'w')
  for li in lineas:
    archivo.write(li)
    if li==cadenaComparativa2:
      archivo.write("exten =>"+extension+",1,Answer() ;"+nombre+"\n")
      archivo.write("\t same =>n,Dial(Sip/"+nombre+",20) ;"+nombre+"\n")
      archivo.write("\t same =>n,VoiceMail("+extension+"@default) ;"+nombre+"\n")
      archivo.write("\t same =>n,Hangup() ;"+nombre+"\n")
    if li==cadenaComparativa:
      archivo.write("["+nombre+"] ;"+nombre+"\n")
      archivo.write("include =>"+perfil+";"+nombre+"\n")
      archivo.write("exten =>"+extension+",1,Answer() ;"+nombre+"\n")
      archivo.write("\t same =>n,Dial(Sip/"+nombre+") ;"+nombre+"\n")
      archivo.write("\t same =>n,VoiceMail("+extension+"@default) ;"+nombre+"\n")
      archivo.write("\t same =>n,Hangup() ;"+nombre+"\n")
      archivo.write("exten => xxxx,1,Goto(Interno,${EXTEN},1) ;"+nombre+"\n")
      archivo.write("exten =>55,1,Goto(consultarBuzon,${EXTEN},1) ;"+nombre+"\n")

  archivo.close()
  subprocess.call(['service','asterisk','restart'])
```

Código 3.19 Función personaExtensions

```
def personaVoiceM(extension,nombre,password):
  cadenaComparativa='[default]\n'
  archivo=open("/etc/asterisk/voicemail.conf",'r')
  lineas=archivo.readlines()
  archivo.close()
  archivo=open("/etc/asterisk/voicemail.conf",'w')
  for li in lineas:
    archivo.write(li)
    if li==cadenaComparativa:
      archivo.write(extension+" => "+password+","+nombre+"\n"

  archivo.close()
  subprocess.call(['service','asterisk','restart'])
```

Código 3.20 Función personaVoiceM

Las tres funciones mostradas anteriormente abren el archivo de su nombre respectivo, escriben las líneas adecuadas según el formato de Asterisk, cierran el archivo y para guardar los cambios efectúan un reinicio de Asterisk mediante el uso del módulo **subprocess** de Python.

3.4.4.2 Implementación de Bloqueo Simple

El bloqueo simple es una funcionalidad del prototipo que permite bloquear llamadas salientes a destinos comunes como llamadas a celulares, llamadas locales, entre otros; durante un período de tiempo. La implementación de dicha funcionalidad fue aumentando en dificultad acorde a los requerimientos manifestados por el usuario (padre de familia) en la parte de gráfica y en la parte de lógica.

El controlador **bloqueoSimple** debe verificar los tipos de destinos que el padre de familia bloqueó a uno de los miembros del hogar, y de acuerdo a esa verificación escribir el bloqueo adecuado. A continuación se mostrará un ejemplo en el caso de que el padre de familia haya bloqueado las llamadas celulares a un miembro de su hogar.

- Primero se verifica si el padre de familia excluyó dos números del bloqueo, a continuación se extrae el valor de los números que se excluyen del bloqueo como se ve en el **Código 3.21**.

```
try:
    celulares=request.POST.get('celularB')
except Exception, e:
    celulares="none"
exp1Celular=str(request.POST.get('txtExcepCel1'))
exp2Celular=str(request.POST.get('txtExcepCel2'))
```

Código 3.21 Primera parte del Controlador BloqueoSimple

- Segundo se obtiene la persona a quien ha bloqueado el padre de familia, la fecha de inicio y de fin que durará el bloqueo. Por último se preparan los datos para poder escribir en los archivos de configuración, tal como muestra el **Código 3.22**.

```
idpersona=request.POST.get('pEscogida')
personaConsultada=Persona.objects.get(id=idpersona)
nombrePersonaC=darFormatoNombre(unicode(personaConsultada))
fechaBloquearI=request.POST.get('cargarRango1')
fI=fechaBloquearI.split("-")
diaInicio=fI[2]
anioInicio=fI[0]
envMes=fI[1]
mesInicio=obtenerMes(envMes)

fechaBloquearF=request.POST.get('cargarRango2')
fF=fechaBloquearF.split("-")
diaFin=fF[2]
envMes2=fF[1]
anioFin=fF[0]
mesFin=obtenerMes(envMes2)
```

Código 3.22 Segunda parte del Controlador BloqueoSimple

- A continuación se debe verificar, si los números que el padre de familia ingresa como excepciones se encuentran guardados en la tabla de Bloqueo Avanzado lo cual puede traer inconvenientes al momento de efectuar una llamada.
- Después de ello se verifica si el bloqueo simple que se desea aplicar ya ha sido previamente ingresado pero con una fecha anterior, por lo que se procede a actualizar la fecha de duración del bloqueo.

El diagrama mostrado en la **Figura 3.14**, muestra claramente los pasos que se sigue antes de aplicar el bloqueo.

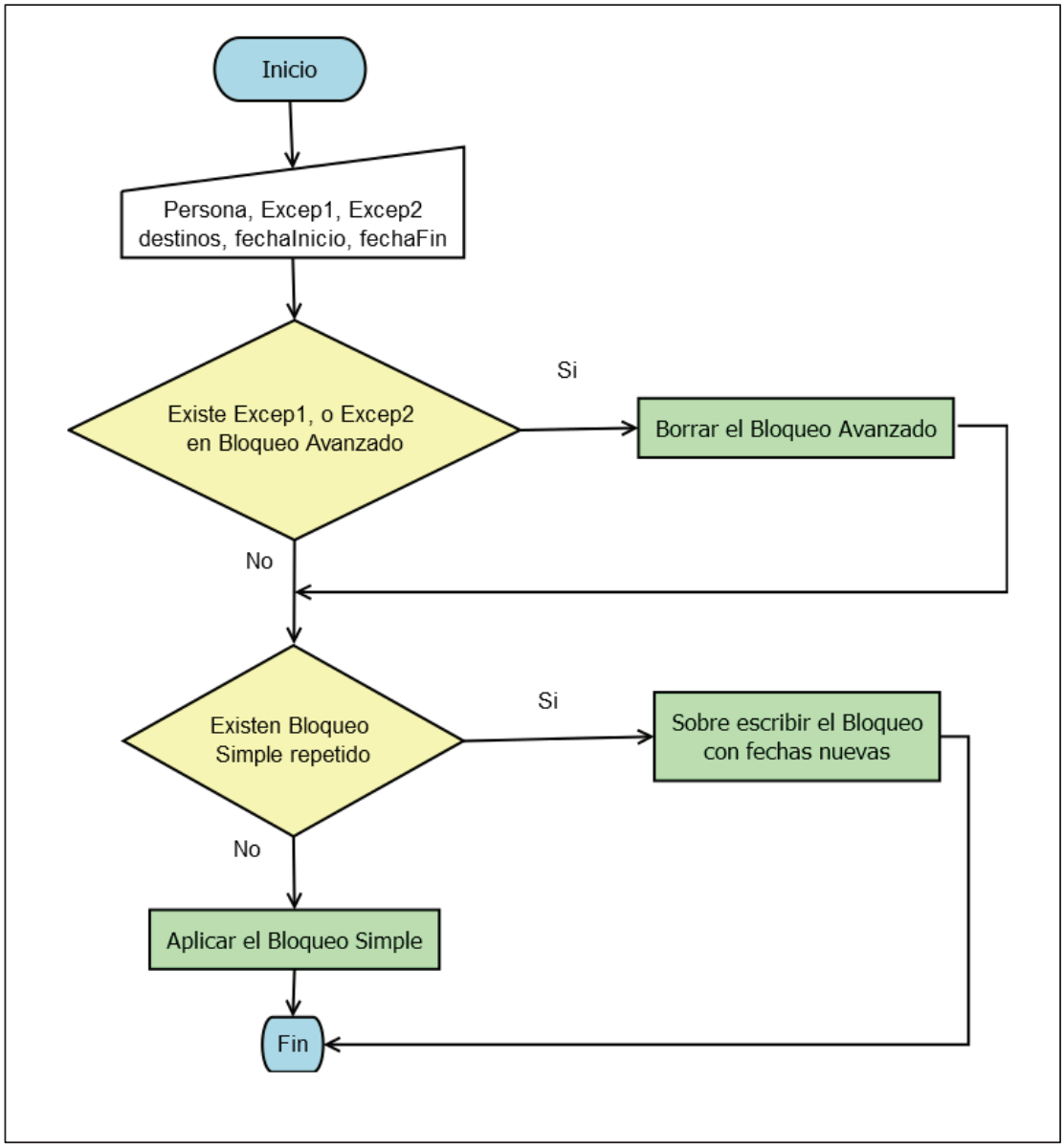

Figura 3.14 Diagrama de Flujo aplicado en Bloqueo Simple

- Finalmente, si se trata de un nuevo caso de bloqueo se guarda el bloqueo configurado en el modelo **BloqueoSimple** con los datos necesarios, como muestra el **Código 3.23**, para luego aplicar el bloqueo en el plan de marcado de Asterisk.

En el **Código 3.23** también puede observarse a las funciones **bloquearPatrones**, **bloquearExtension** y **agregarListasBlancas**, las cuales se encargan de adecuar el plan de marcado en el archivo **extensions.conf** para efectuar el bloqueo en Asterisk.

```
if celulares=='on':
    bc=BloqueoSimple(personaBloqueada=personaConsultada,
        categoriaLLamada="Llamadas Celulares",fechaInicio=fechaBloquearI,
        fechaFin=fechaBloquearF,excepcion1=exp1Celular,excepcion2=exp2Celular)
    bc.save()
    bloquearPatrones(nombrePersonaC,diaInicio,mesInicio,anioInicio
        ,diaFin,mesFin,anioFin,'BloqueoCelular')
    bloquearExtension(nombrePersonaC,'BloqueoCelular')
agregarListasBlancas(lstBCelulares,nombrePersonaC,'BloqueoCelular')
```

Código 3.23 Parte Final del Controlador BloqueoSimple

La tarea de la función **bloquearExtension** mostrada en el **Código 3.24** es escribir en el contexto de la persona bloqueada, las líneas necesarias para que se efectúe el bloqueo, es decir, reestructura el plan de marcado para indicar que antes de que la persona bloqueada realice una llamada telefónica se verifique si existe un bloqueo.

```
def bloquearExtension(persona,categoria):
  cadena='['+str(persona)+'] ;'+str(persona)+"\n"
  cadena2=categoria+str(persona)+' ;'+str(persona)+categoria+"\n"
  archivo=open("/etc/asterisk/extensions.conf",'r')
  lineas=archivo.readlines()
  archivo.close()

  archivo=open("/etc/asterisk/extensions.conf",'w')
  for li in lineas:

    archivo.write(li)
    if li==cadena:
      archivo.write("include =>"+cadena2+"\n")
  archivo.close()
```

Código 3.24 Función bloquearExtensión

La tarea de la función **agregarListasBlancas** que se encuentra en el **Código 3.25**, es enviar la llamada directamente al contexto de salida de llamadas, en caso de que la persona bloqueada digite uno de los números que se excluyeron del bloqueo.

```
def agregarListasBlancas(lista,persona,tipo):
  tipoLista=tipo
  cadena='['+tipoLista+str(persona)+'] ;'+str(persona)+ tipoLista+"\n"

  if len(lista)!=0:
    archivo=open("/etc/asterisk/extensions.conf",'r')
    lineas=archivo.readlines()
    archivo.close()
    archivo=open("/etc/asterisk/extensions.conf",'w')
    for li in lineas:
      archivo.write(li)
      if li==cadena:
        for i in range(0,len(lista)):
          archivo.write('exten =>'+str(lista[i])+',1,Goto(SalidaLlamadas,${EXTEN},1) ;'
            +str(persona)+tipoLista+"\n")

    archivo.close()
    subprocess.call(['service','asterisk','restart'])
```

Código 3.25 Función agregarListasBlancas

Por último, la función **bloquearPatrones**, se encarga de crear contextos temporales de bloqueo simple para cada persona. Esta función agregará las líneas necesarias para hacer que el sistema bloquee las llamadas, de acuerdo a los bloqueos configurados por el padre de familia.

En el ejemplo del bloqueo de llamadas a celulares, la función **bloquearPatrones** presente en el **Código 3.26** escribirá en el plan de marcado la condición de bloqueo según el período de duración del bloqueo especificado por el padre de familia, y le notificará a la persona bloqueada a través de un mensaje que no puede realizar llamadas.

```
def bloquearPatrones(persona,diaI,mesI,anioI,diaF,mesF,anioF,categoria):
  cadena1=';Agregar BloqueosSimples\n'
  cadena2='['+categoria+str(persona)+'] ;'+str(persona)+ categoria+"\n"
  archivo=open("/etc/asterisk/extensions.conf",'r')
  lineas=archivo.readlines()
  archivo.close()
  archivo=open("/etc/asterisk/extensions.conf",'w')
  for li in lineas:

    archivo.write(li)
    if li==cadena1:
      archivo.write(cadena2)
      if categoria== 'BloqueoLocal':
        archivo.write('exten => _[2-7]XXXXXX,1,Answer() ;'+str(persona)+categoria+"\n")
        if mesI==mesF:
          archivo.write('\t same =>n,GotoIfTime(*,*,'+diaI+'-'+diaF+','+mesF+'?bloqueo) ;'+str(persona)+categori
          archivo.write('\t same =>n(bloqueo),Playback(bloqueoSaliente);'+str(persona)+categoria+'\n')
          archivo.write('\t same =>n,Hangup();'+str(persona)+categoria+'\n')
        else:
          archivo.write('\t same =>n,GotoIfTime(*,*,'+diaI+'-'+diaF+','+mesI+'-'+mesF+'?bloqueo) ;'+str(persona)
          archivo.write('\t same =>n(bloqueo),Playback(bloqueoSaliente) ;'+str(persona)+categoria+'\n')
          archivo.write('\t same =>n,Hangup() ;'+str(persona)+categoria+'\n')

      elif categoria == 'BloqueoCelular':
        archivo.write('exten => _09XXXXXXXX,1,Answer();'+str(persona)+categoria+"\n")
        if mesI==mesF:
          archivo.write('\t same =>n,GotoIfTime(*,*,'+diaI+'-'+diaF+','+mesF+'?bloqueo);'+str(persona)+categoria
          archivo.write('\t same =>n(bloqueo),Playback(bloqueoSaliente);'+str(persona)+categoria+'\n')
          archivo.write('\t same =>n,Hangup();'+str(persona)+categoria+'\n')
        else:
          archivo.write('\t same =>n,GotoIfTime(*,*,'+diaI+'-'+diaF+','+mesI+'-'+mesF+'?bloqueo);'+str(persona)+
          archivo.write('\t same =>n(bloqueo),Playback(bloqueoSaliente);'+str(persona)+categoria+'\n')
          archivo.write('\t same =>n,Hangup();'+str(persona)+categoria+'\n')
```

Código 3.26 Parte de la Función bloquearPatrones

3.4.4.3 Implementación de Bloqueo Avanzado

El bloqueo avanzado de llamadas salientes tiene la intención de evitar una llamada de un miembro del hogar a un número en específico, en un período de tiempo especificado por el padre de familia, tanto en días como en horas. El controlador **bloqueoAvanzado** primero recolectará la información que llega de la interfaz gráfica como muestra el **Código 3.27**, donde se puede observar, el nombre de la persona a quien se aplicó el bloqueo, la fecha de inicio y fin del bloqueo, el horario de duración del bloqueo y el número a ser bloqueado.

```
def bloqueoAvanzado(request):

    personB=request.POST.get('pEscogidaA')
    numeroBloquear=request.POST.get('numeroEspecifico')
    personaConsultada=Persona.objects.get(nombrePersona=personB)
    nombrePersonaC=darFormatoNombre(unicode(personaConsultada))
    fechaBloquearI=request.POST.get('cargarRango1A')
    fechaBloquearF=request.POST.get('cargarRango2A')
    categoriaNumero=request.POST.get('tipoNumeroA')
    horaIV=request.POST.get('HoraInicioA'); horaFV=request.POST.get('HoraFinA')
    minIV=request.POST.get('MinutosInicioA');minFV=request.POST.get('MinutosFinA')
    guardarHoraI=horaIV+":"+minIV
    guardarHoraF=horaFV+":"+minFV
    existe=True
```

Código 3.27 Controlador bloqueoAvanzado, Primera Parte

A continuación el controlador debe guardar el bloqueo en el modelo respectivo. Previamente se debe hacer un análisis similar al bloqueo Simple, es decir comprobar si el número a bloquear es parte de una excepción de Bloqueo Simple, lo cual podría traer conflictos en el bloqueo. Además se verifica si el número fue bloqueado antes, lo que indicaría un cambio en el horario y la fecha de bloqueo. El diagrama de la **Figura 3.15** ilustra las acciones a tomar antes de realizar el bloqueo.

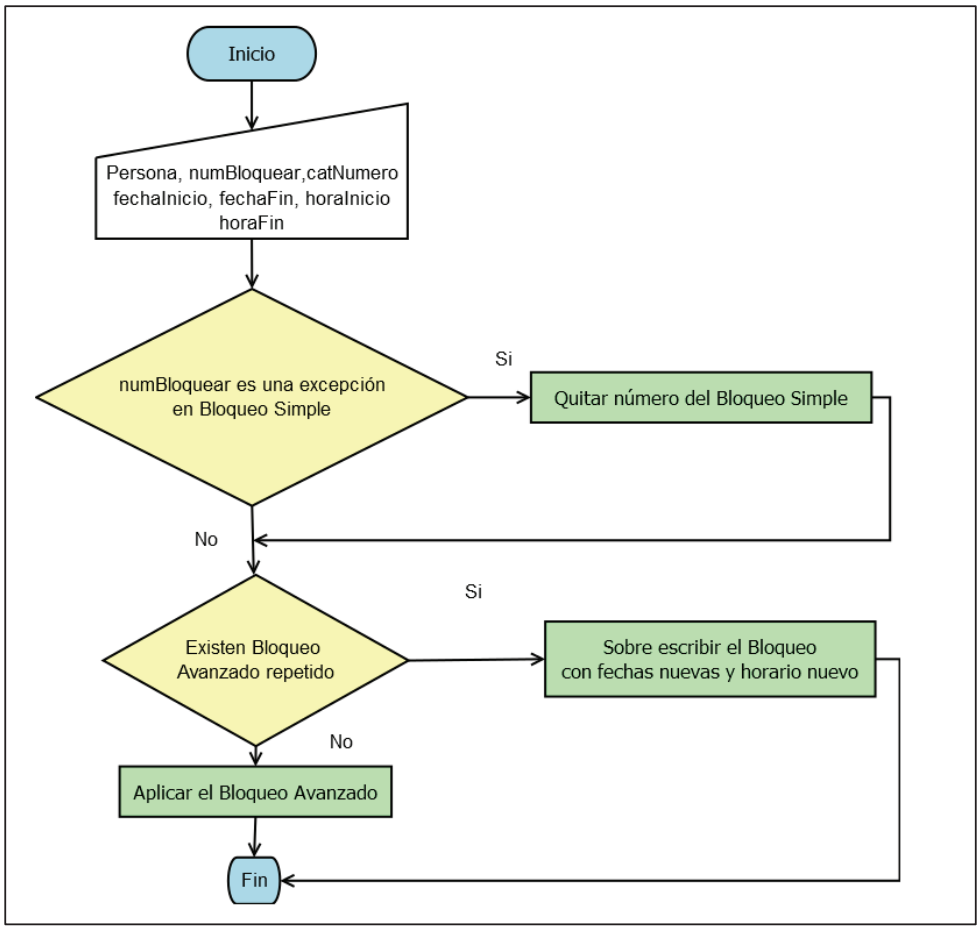

Figura 3.15 Diagrama de Flujo aplicado en Bloqueo Avanzado

Una vez terminado las verificaciones, se procede a aplicar el bloqueo con las sentencias mostradas en el **Código 3.28**.

```
d=Destino(numeroDestino=numeroBloquear, categoriaNumero=categoriaNumero)
d.save()
numero=Destino.objects.get(numeroDestino=numeroBloquear)
b=BloqueoAvanzado(personaBloqueada=personaConsultada,
  destinoBloqueado=numero,fechaInicio=fechaBloquearI,
  fechaFin=fechaBloquearF,horaBloqueoI=guardarHoraI,
  horaBloqueoF=guardarHoraF)
b.save()
bloqueoUnNumero(nombrePersonaC,numeroBloquear,diaInicio,enviarMesI,
  diaFin,enviarMesF,guardarHoraI,guardarHoraF)
```

Código 3.28 Controlador bloqueoAvanzado, Parte Final

Finalmente, el controlador debe escribir el bloqueo en el plan de marcado, previamente debe preparar los datos y enviarlos en un formato adecuado para escribirlos en el archivo de configuración y devolver la vista adecuada. Para ejecutar la tarea relacionada a la escritura se utiliza la función **bloqueoUnNumero,** la cual es mostrada en el **Código 3.29**.

```
#Funcion para hacer el bloqueo avanzado en el plan de marcado
def bloqueoUnNumero(persona,numeroB,dI,mI,dF,mF,hI,hF):
  cadenaPersona='['+str(persona)+'] ;'+str(persona)+"\n"
  cadenaHora= hI+"-"+hF
  cadenaDia=dI+"-"+dF
  cadenaMes=mI+"-"+mF
  print dI
  print dF
  archivo=open("/etc/asterisk/extensions.conf",'r')
  lineas=archivo.readlines()
  archivo=open("/etc/asterisk/extensions.conf",'w')
  for li in lineas:

    archivo.write(li)
    if li==cadenaPersona:
      if mI==mF:
        archivo.write("exten =>"+numeroB+",1,Answer();"+str(persona)+str(numeroB)+ "\n")
        archivo.write('\t same =>n,GotoIfTime('+cadenaHora+',*,'+cadenaDia+','+
          mF+'?bloqueo:llamar) ;'+str(persona)+str(numeroB)+'\n')
        archivo.write('\t same =>n(bloqueo),Playback(bloqueoSaliente);'+str(persona)+str(numeroB)+'\n')
        archivo.write('\t same =>n,Hangup();'+str(persona)+str(numeroB)+'\n')
        archivo.write('\t same =>n(llamar),Goto(SalidaLlamadas,${EXTEN},1);'
          +str(persona)+str(numeroB)+'\n')
        archivo.write('\t same =>n,Hangup();'+str(persona)+str(numeroB)+'\n')

      else:
        archivo.write('\t same =>n,GotoIfTime(*'+cadenaHora+',*,'+cadenaDia+','+
          cadenaMes+'?bloqueo:llamar) ;'+str(persona)+str(numeroB)+'\n')
        archivo.write('\t same =>n(bloqueo),Playback(bloqueoSaliente) ;'+str(persona)+str(numeroB)+'\n')
        archivo.write('\t same =>n,Hangup() ;'+str(persona)+str(numeroB)+'\n')
        archivo.write('\t same =>n(llamar),Goto(SalidaLlamadas,${EXTEN},1);'
          +str(persona)+str(numeroB)+'\n')
        archivo.write('\t same =>n,Hangup() ;'+str(persona)+str(numeroB)+'\n')

  archivo.close()
  subprocess.call(['service','asterisk','restart'])
```

Código 3.29 Función bloqueoUnNumero

En el **Código 3.29** se indica a la Central Telefónica que si el usuario marca el número bloqueado, se reproduzca una grabación indicando que está bloqueado, caso contrario se efectuará la llamada.

3.4.4.4 Implementación de Desbloqueo Avanzado

El prototipo incluye un módulo de desbloqueo, el controlador de desbloqueo es mucho más sencillo que los controladores mencionados anteriormente. El **Código 3.30** muestra un ejemplo de desbloqueo del caso avanzado.

```
def desbloqueoAvanzado(request):
    idDesbloquearA=request.POST.get('opcionBorradoA')
    elementoDesbloquear=BloqueoAvanzado.objects.get(pk=idDesbloquearA)
    persona=elementoDesbloquear.personaBloqueada
    personaEnviar=darFormatoNombre(unicode(persona))
    telefono=elementoDesbloquear.destinoBloqueado
    BloqueoAvanzado.objects.filter(pk=idDesbloquearA).delete()
    DesbloquearExtensionA(personaEnviar,telefono)
    context_listUsuario={}
    lista=BloqueoSimple.objects.all()
    context_listUsuario['bloqueSimple']=lista
    lista2=BloqueoAvanzado.objects.all()
    context_listUsuario['bloqueAvanzado']=lista2
    lista3=LlamadaEntrante.objects.all()
    context_listUsuario['destinos']=lista3
    return render(request,'desbloqueoGeneral.html',context_listUsuario)
```

Código 3.30 Controlador desbloqueoAvanzado

El controlador elimina el registro de la tabla **BloqueoAvanzado** y además utiliza una función **DesbloquearExtensionA** para realizar los cambios en el archivo de configuración del plan de marcado.

DesbloquearExtensionA requiere el nombre de la persona y el número que fue bloqueado, por esta razón, las cinco primeras líneas del **Código 3.30** realizan una consulta a la base de datos antes de borrar el registro.

La parte final del **Código 3.30** del controlador, crea unas listas de las tablas que almacenan los bloqueos para luego ser mostrados en la vista, de tal manera que el padre de familia pueda visualizar que se eliminó el registro.

3.4.5 IMPLEMENTACIÓN DEL NIVEL PRESENTACIÓN

En el nivel de presentación se crearon las páginas web adecuadas para el prototipo, cada vista del diseño final puede verse en la **sección 2.2.3**. Cada interfaz se compone de un archivo con extensión .html y los archivos .css y .js para dar estilo y dinamismo a la página web respectivamente.

A continuación se explicará parte del código de tres archivos que en conjunto muestran la vista para agregar un nuevo miembro del hogar.

Figura 3.16 Vista Agregar Miembro del Hogar

La **Figura 3.16,** presenta la interfaz que visualiza al padre de familia. En la parte superior se encuentran iconos de navegabilidad y de ayuda, a continuación se presenta el título de la interfaz seguido de un formulario con los principales datos que se requieren para configurar un miembro del hogar, finalizando se encuentra el botón de **Guardar.**

La parte fundamental de la interfaz se encuentra en el formulario, por este motivo se mostrará una porción del código que hace el formulario en los tres archivos.

Las líneas del **Código 3.31** presentan el código html que crea el formulario de manera básica, dentro del código se observan las etiquetas utilizadas para crear cada elemento de la página web.

```
<form action="/guardarUsuarioNuevo/" name="nombre" class="frmUsuarioNuevo" method="POST" id="
frmUsuarioNuevo">
{% csrf_token %}
<br>
        <fieldset class="general">
        <table class="tablaform">
            <tr>
                <td style="width: 28%;"><label >Nombre:</label></td>
                <td style="width: 32%;">
                    <input required type="text" placeholder="Escriba el nombre de la persona" name= "nombre
                    " id="nombreAgregar" onblur="limpia(1)" maxlength="20">
                </td>
                <td></td>
                <td style="width: 22%;"><p>Ejemplo:Jose </p></td>
            </tr>
            <tr>
                <td><label >Extensión:</label></td>
                <td>
                    <input required type="text"  placeholder="Escriba el número de extensión" name= "
                    extension" id="extensionAgregar" onblur="limpia(2)" maxlength="4">
                </td>
                <td></td>
                <td><p>Ejemplo:5320</p></td>
            </tr>
            <tr>
                <td><label>Clave de Asociación:</label></td>
                <td>
                    <input required  type="password" placeholder="Clave de 10 caracteres máximo." name= "
                    contraseniaAso" id="contraseniaAso" maxlength="10">
                </td>
                <td>
                    <input type="checkbox" name="verCA" id="verCA" ><div style="font-size:12px; font-
                    family: 'Arial';">Ver caracteres</div>
                </td>
                <td><p>Ejemplo:jose123</p></td>
            </tr>
```

Código 3.31 Parte del Archivo agregarPersona.html

Para organizar el formulario se creó una tabla de cuatro columnas, la primera columna llevará etiquetas **<label>,** las cuales son utilizadas para dar un texto, en el ejemplo mostrado puede observarse los textos **Nombre:, Extensión:** y **Clave de Asociación:**

La segunda columna contiene la etiqueta **<input>** del tipo texto, la cual es un control para que el padre de familia escriba el dato, en el ejemplo mostrado en el **Código 3.31** se observan los campos necesarios para que el padre de familia ingrese el nombre de la persona, la extensión y la clave de asociación.

La tercera columna contiene la etiqueta **<input>** del tipo **checkbox,** la cual permite que el padre de familia pueda ver las claves en texto claro. Esta columna únicamente contiene esta etiqueta en Clave de Asociación y en Clave de Buzón de Voz.

La última columna es utilizada para dar un ejemplo de la manera en que el padre de familia debe escribir cada uno de los datos utilizando la etiqueta **<p>**.

El archivo **agregarPersona.css** da el estilo a la página web, en css se puede dar el estilo de varias maneras:

1. Utilizando el nombre de la etiqueta a quien se le va a aplicar el estilo.
2. Mediante un atributo **class** de la etiqueta.
3. Utilizando un atributo **id** de una etiqueta.

En el **Código 3.32** se observa una parte del estilo de varios controles del formulario, utilizando el primer y tercer método. La etiqueta **<table>** presente en el **Código 3.31** contiene el atributo **class=tablaform**, por lo tanto **tablaform** presente en el **Código 3.32** da estilo a dicha tabla.

Dentro de **tablaform** se utilizan las propiedades **width, margin y background**, que se encargan de dar a la tabla ancho y margen respecto al control que contiene a la tabla, y color de fondo respectivamente.

```
label{
    display: block;
    color: #C2E2F6;

}
.tablaform{
    margin:10px;
    width: 96%;
    background: #F0EFF4;

}
.tablaform td{
    padding: 10px;

}


label, input[type="text"],input[type="password"]{
    font-family: "Arial";
    width: 100%;
    text-align: center;

}
```

Código 3.32 Parte del Archivo agregarPersona.css

En el **Código 3.32** se puede observar que a las etiquetas **label** e **input** del tipo texto y **password** se les asigna el tipo de fuente Arial, un tamaño del control del 100% y texto centrado.

Para complementar el estilo a la página web se utilizó el archivo **usuarios.js**. Este archivo contiene algunas líneas de código que se han utilizado para dar efectos más atractivos y dinámicos a la interfaz gráfica, ver **Código 3.33**.

```
//Evento para desplegar la ayuda de usuario nuevo
$("#AyudaUsrNuevo").click(function(){
    document.getElementById("frmUsuarioNuevo").style.opacity = "0.1";
    document.getElementById("formBarraSuperior").style.opacity = "0.1";

    $("#mostrarAyuda").slideDown(1500);

});
```

Código 3.33 Evento AyudaUsrNuevo

El **Código 3.33** tiene la función de opacar todo el formulario principal **frmUsuarioNuevo** y el fromulario **barraSuperior** de la barra de Ayuda, de manera que no se los pueda observar son claridad y que el mensaje de ayuda capte la atención del observador. El mensaje de ayuda llamado **mostrarAyuda** aparecerá y se podrá observar como lo muestra la **Figura 3.17.**

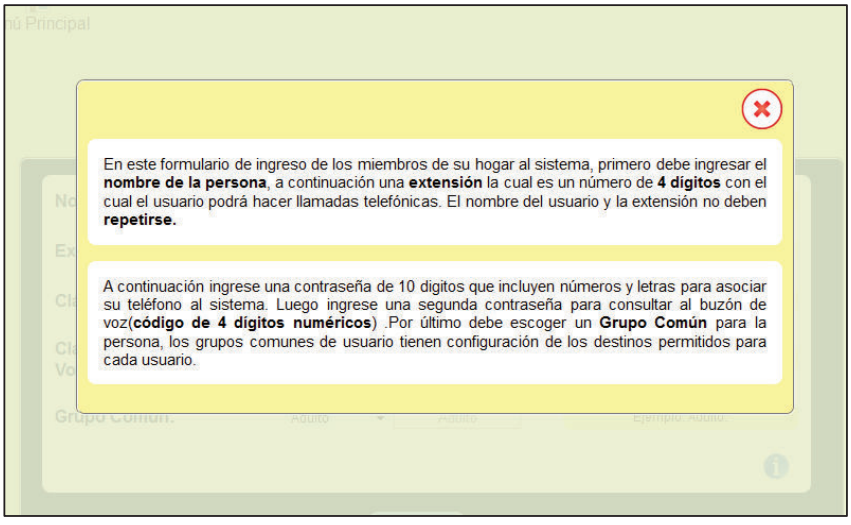

Figura 3.17 Mensaje de Ayuda en Vista Agregar Miembro del Hogar

La mayoría de los efectos realizados en los archivos con extensión .js se podrán apreciar de mejor manera en la experimentación práctica del prototipo.

3.5 PRUEBAS DE FUNCIONALIDAD

Las pruebas de funcionalidad que se realizaron para probar el funcionamiento del sistema telefónico fueron las siguientes:

- Realización de una llamada a la PSTN.
- Prueba de Bloqueo de Llamadas Salientes.
- Consulta de un Mensaje de Buzón de Voz.

Dichas pruebas pueden observarse en el Anexo H, el cual contiene los videos respectivos que demuestran la funcionalidad del servicio telefónico que se implementó en el prototipo.

3.6 PRUEBAS DE CARGA DEL SISTEMA

Las pruebas de carga que se realizaron en el sistema fueron orientadas al porcentaje de uso del CPU y a la cantidad de memoria RAM que Raspberry PI utilizó en el escenario más crítico.

Se consideró como escenario más crítico, a aquel en el que se encuentra trabajando la central telefónica y la aplicación web a la vez. Teniendo en cuenta dichos parámetros se preparó un escenario de prueba, considerando que una familia en promedio consta de 4 integrantes, según un estudio realizado por el INEC en el año 2012 [35]. Dicho escenario presenta las siguientes características:

- Puesta en marcha de la aplicación que gestiona el Servicio de Telefonía.
- Realización de 2 llamadas simultáneas, una llamada interna entre dos miembros del hogar y otra llamada dirigida hacia la PSTN por otro miembro de la familia.
- Manipulación de la aplicación que gestiona el Servicio de Telefonía (navegación por la aplicación, consulta de reportes, entre otros.)

El escenario mencionado anteriormente fue implementado según muestra la topología de la **Figura 3.18**.

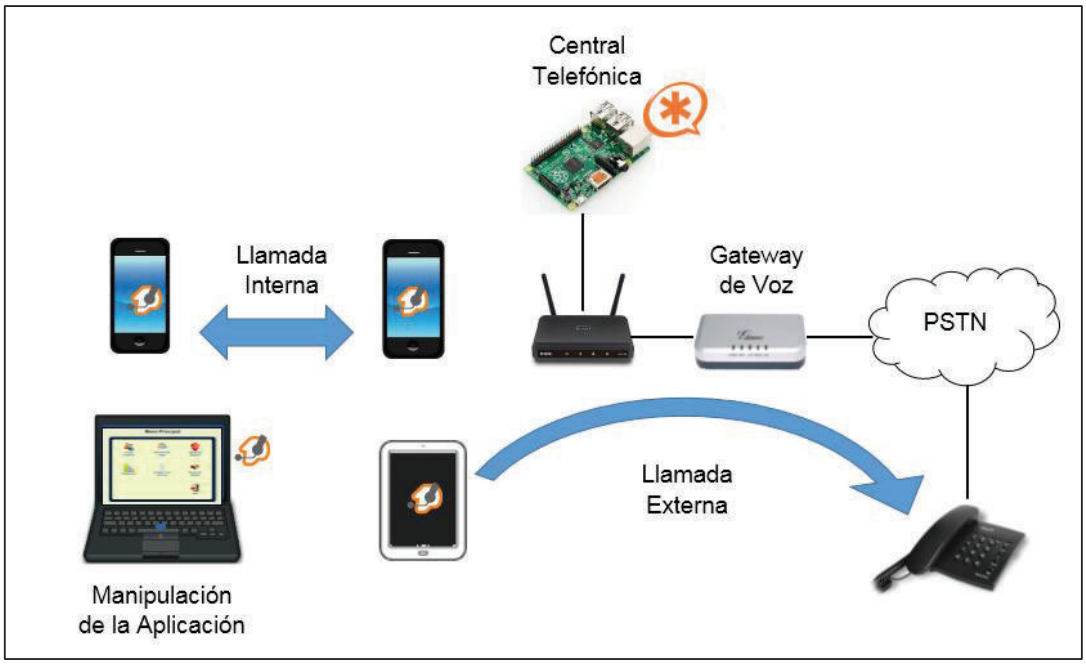

Figura 3.18 Topología del Prototipo utilizado para Pruebas de Carga

3.6.1 PRUEBAS DEL USO DE CPU

En base al escenario planteado en la **sección 3.6** se realizaron tres mediciones acerca del uso del CPU del Raspberry Pi, utilizando la herramienta de monitoreo de red PRTG[29] (*Paessler Router Traffic Grapher*). Dicha herramienta permite monitorear dispositivos de red así como también equipos de usuario, utilizando sensores para medir el comportamiento de memoria, de CPU, entre otros.

Mediante la Utilización de un sensor de medida de CPU propio de PRTG, se midió el porcentaje de utilización del CPU para cada una de las tareas que se planteó para la prueba, mencionadas en la sección **3.6**, obteniéndose las gráficas de Porcentaje de Uso de CPU vs Tiempo mostradas en la **Figura 3.19**, **Figura 3.20** y **Figura 3.21**.

[29] PRTG(*Paessler Router Traffic Grapher*): Es un programa gestor de monitoreo de la red, el cual permite monitorear los dispositivos presentes en una red local .

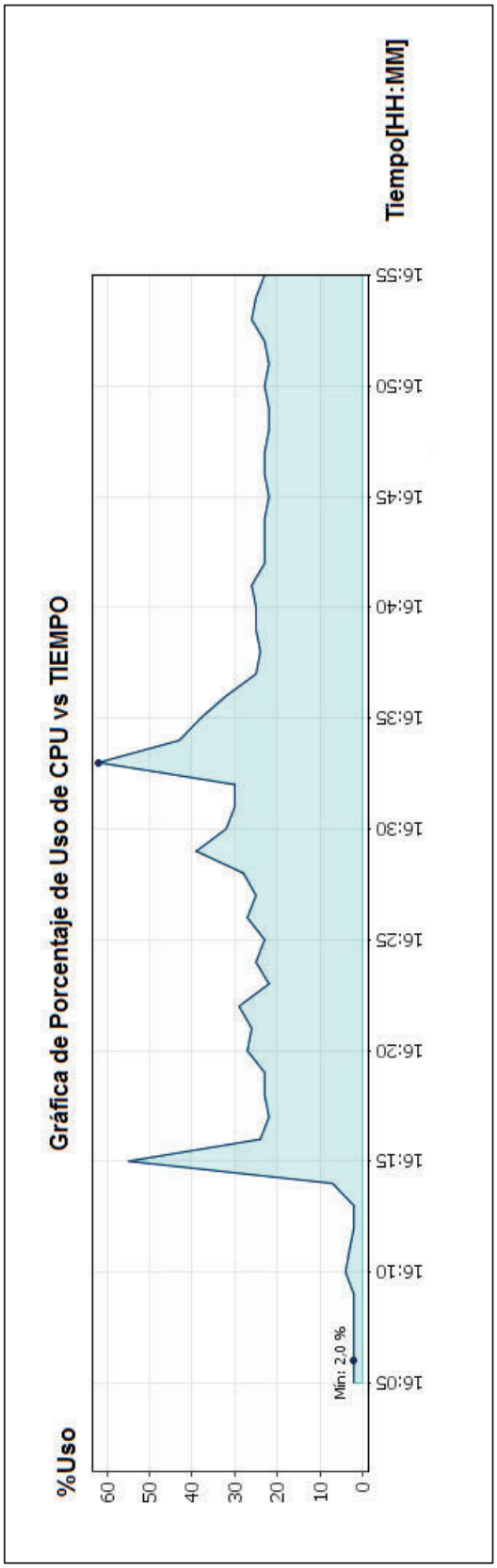

Figura 3.19 Gráfica de Porcentaje de Uso del CPU en Prueba 1

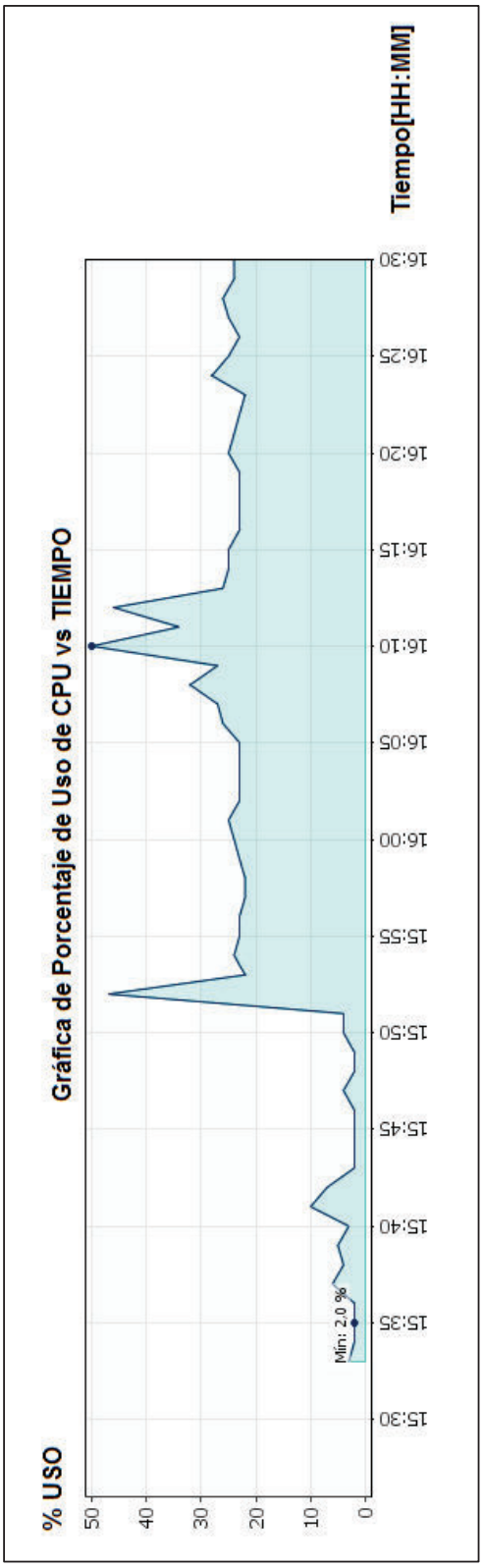

Figura 3.20 Gráfica de Porcentaje de Uso del CPU en Prueba 2

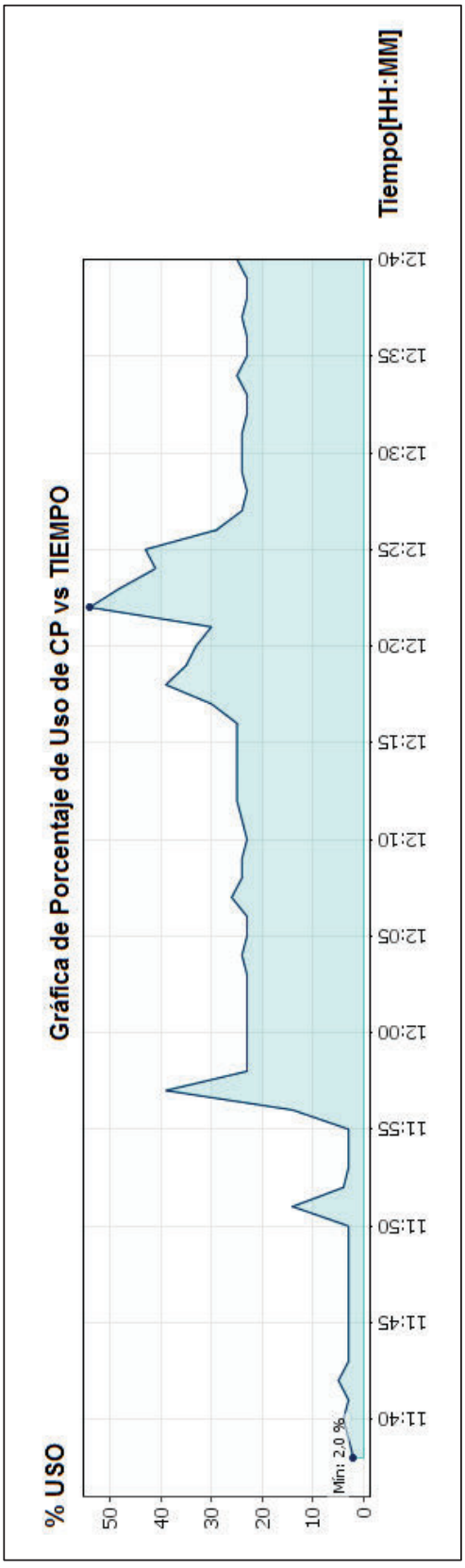

Figura 3.21 Gráfica de Porcentaje de Uso del CPU en Prueba 3

Las gráficas mostradas presentan la variación del porcentaje de uso del CPU del Raspberry Pi en distintos instantes de tiempo. En la **Tabla 3.4**, **Tabla 3.5** y **Tabla 3.6** se observan los parámetros que hicieron variar el uso del CPU, la hora aproximada del evento y la medición correspondiente obtenida de los datos proporcionados por PRTG.

Parámetro	Hora Aproximada	Porcentaje de Uso de CPU medido
Encendido de Raspberry Pi	16:05	2%
Encendido del Servidor(Aplicación)	16:14	55%
Conexión Clientes Sip	16:20	26%
Realización de la Primera Llamada	16:25	27%
Realización de la Segunda Llamada	16:30	32%
Navegación en la Aplicación (consulta de reportes, visualización de personas,etc)	16:32	62%

Tabla 3.4 Valores tomados del Porcentaje de Uso de CPU en Prueba1

Parámetro	Hora Aproximada	Porcentaje de Uso de CPU medido
Encendido de Raspberry Pi	15:33	3%
Encendido del Servidor(Aplicación)	15:51	47%
Conexión Clientes Sip	15:56	23%
Realización de la Primera Llamada	16:00	25%
Realización de la Segunda Llamada	16:07	32%
Navegación en la Aplicación (consulta de reportes, visualización de personas,etc)	16:09	50%

Tabla 3.5 Valores tomados del Porcentaje de Uso del CPU en Prueba 2

En el Anexo I se presentan los resultados completos obtenidos de las mediadas de Porcentaje de uso del CPU de Raspberry PI proporcionados por el monitor PRTG.

Mediante las tres mediciones que se realizaron se obtiene que en promedio, el porcentaje de uso del CPU del Raspberry Pi modelo B+, utilizado para las pruebas, es de 55% para el escenario más crítico. En este escenario se considera la

realización de 2 llamadas simultáneas, y navegación en la aplicación de administración del sistema telefónico utilizando 4 personas en el entorno doméstico. Por lo tanto se concluye que el prototipo soporta dichas condiciones puesto que las llamadas no se terminaron y el uso del CPU llega a utilizarse en un 55% en promedio de su capacidad.

Parámetro	Hora Aproximada	Porcentaje de Uso de CPU medido
Encendido de Raspberry Pi	11:35	2%
Encendido del Servidor(Aplicación)	11:57	39%
Conexión Clientes Sip	12:03	23%
Realización de la Primera Llamada	12:10	25%
Realización de laSegunda Llamada	12:18	39%
Navegación en la Aplicación (consulta de reportes, visualización de personas,etc)	12:21	54%

Tabla 3.6 Valores tomados del Porcentaje de Uso del CPU en Prueba 3

3.6.2 PRUEBAS DE MEMORIA FÍSICA

Con la finalidad de observar el comportamiento de la memoria física del Raspberry Pi según los parámetros planteados en la **sección 3.6**, se realizaron tres pruebas del uso de la memoria de Raspberry Pi utilizando la herramienta de monitoreo de red PRTG.

El sensor de memoria física que utiliza PRTG proporciona información de la memoria disponible y de la memoria total del dispositivo analizado, según muestran en las imágenes de la **Figura 3.22**,**Figura 3.23** y **Figura 3.24**.

En base a la información proporcionada por PRTG, se realizó el análisis respectivo de la memoria utilizada por Raspberry PI en los mismos instantes de tiempo en los cuales se analizó el porcentaje de uso del CPU como se observa en la **Tabla 3.7**, **Tabla 3.8** y **Tabla 3.9**.

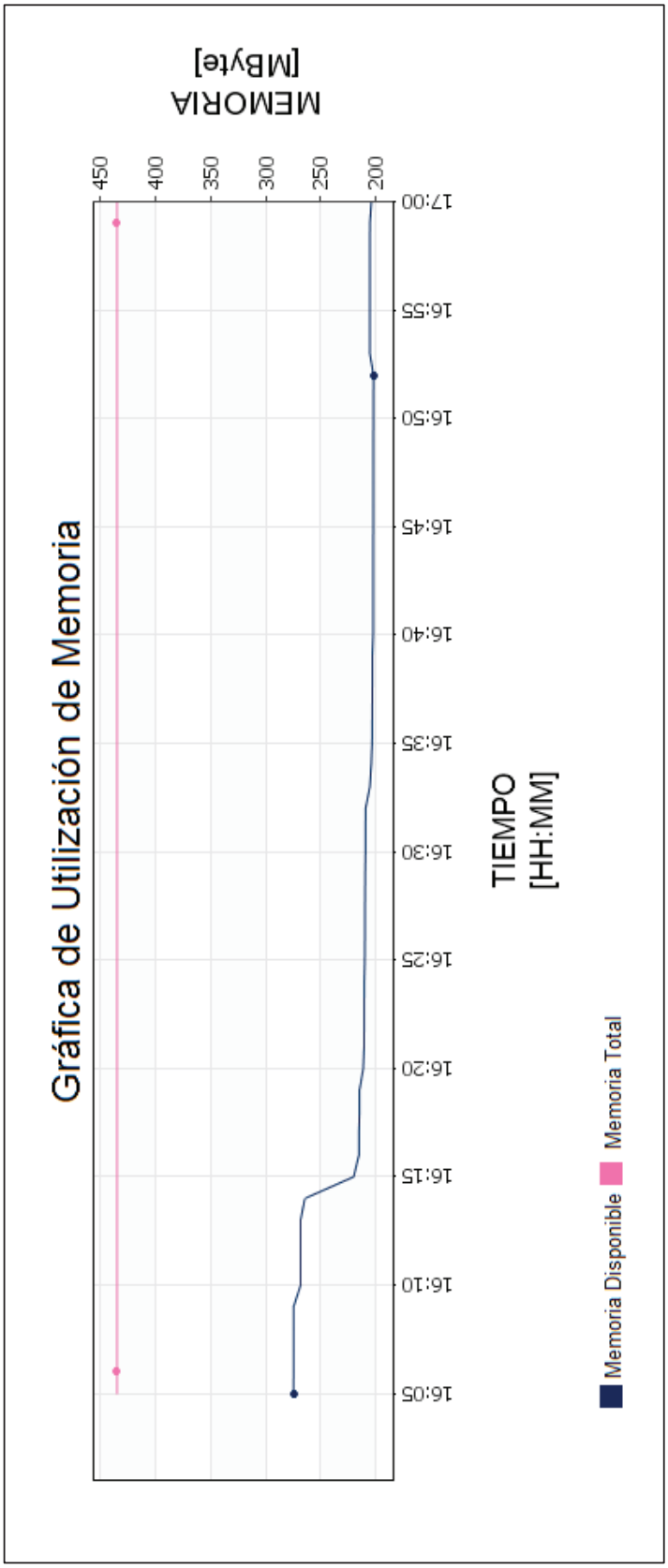

Figura 3.22 Gráfica de Utilización de Memoria en Prueba 1

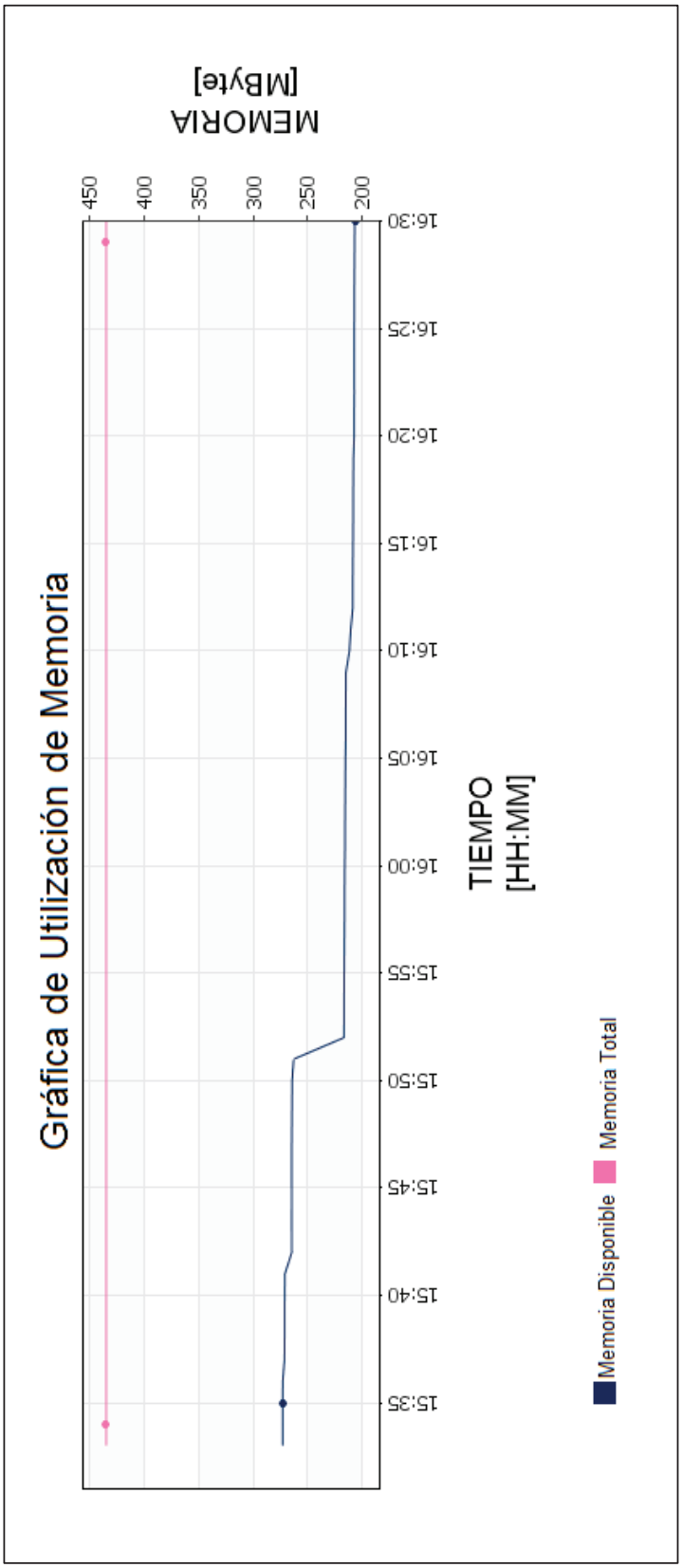

Figura 3.23 Gráfica de Utilización de Memoria en Prueba 2

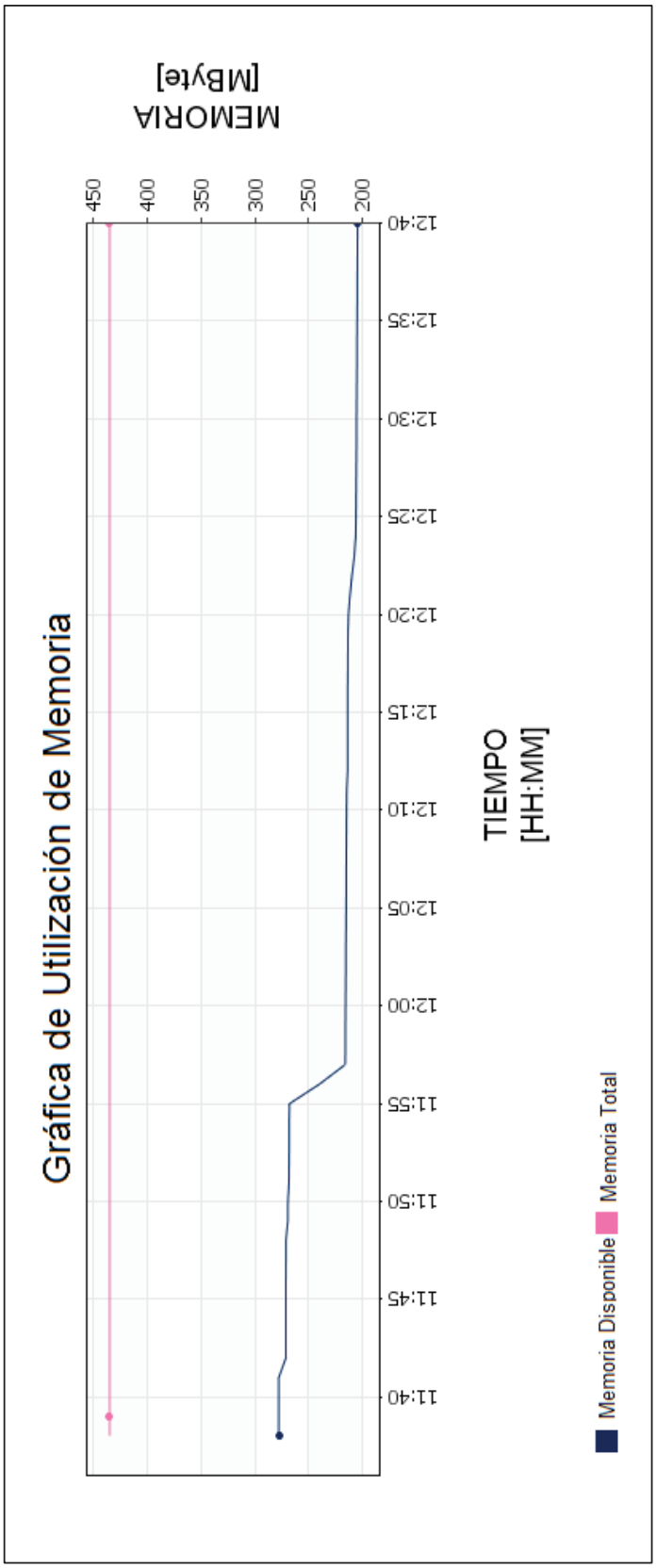

Figura 3.24 Gráfica de Utilización de Memoria en Prueba 3

Parámetro	Hora Aproximada	Memoria Total [MBytes]	Memoria Disponible [MBytes]	Memoria Utilizada [MBytes]
Encendido de Raspberry Pi	16:05	435	275	**160**
Encendido del Servidor(Aplicación)	16:14	435	220	**215**
Conexión Clientes Sip	16:20	435	211	**224**
Realización de la Primera Llamada	16:25	435	210	**225**
Realización de la Segunda Llamada	16:30	435	209	**226**
Navegación en la Aplicación (consulta de reportes, visualización de personas,etc)	16:32	435	205	**230**

Tabla 3.7 Valores tomados de Memoria en Prueba 1

Parámetro	Hora Aproximada	Memoria Total [MBytes]	Memoria Disponible [MBytes]	Memoria Utilizada [MBytes]
Encendido de Raspberry Pi	15:33	435	273	**162**
Encendido del Servidor(Aplicación)	15:51	435	217	**218**
Conexión Clientes Sip	15:56	435	216	**219**
Realización de la Primera Llamada	16:00	435	216	**219**
Realización de la Segunda Llamada	16:07	435	215	**220**
Navegación en la Aplicación (consulta de reportes, visualización de personas,etc)	16:09	435	212	**223**

Tabla 3.8 Valores tomados de Memoria en Prueba 2

Parámetro	Hora Aproximada	Memoria Total [MBytes]	Memoria Disponible [MBytes]	Memoria Utilizada [MBytes]
Encendido de Raspberry Pi	11:35	435	278	**157**
Encendido del Servidor(Aplicación)	11:57	435	216	**219**
Conexión Clientes Sip	12:03	435	216	**219**
Realización de la Primera Llamada	12:10	435	214	**221**
Realización de la Segunda Llamada	12:18	435	214	**221**
Navegación en la Aplicación (consulta de reportes, visualización de personas,etc)	12:21	435	210	**225**

Tabla 3.9 Valores tomados de Memoria en Prueba 3

En el Anexo I se presentan los resultados completos obtenidos de las mediadas de Memoria de Raspberry PI proporcionados por el monitor PRTG.

En resumen para el caso más crítico de análisis del prototipo se obtiene que en promedio, el uso de memoria física es de 226 Mbytes lo que equivale a un uso del 52% de la memoria total del Raspberry PI modelo B+ utilizado para el análisis.

3.7 PRUEBAS UNITARIAS DE LA APLIACIÓN

La metodología de desarrollo XP utiliza pruebas unitarias, las cuales permiten hacer una verificación del funcionamiento de una unidad estructural de código que pueden ser una clase o un método. Mediante el archivo **test.py,** se realizaron ocho pruebas unitarias correspondientes a las ocho clases presentes en el archivo **models.py**.

El archivo **test.py** presenta una clase denominada **Test,** dentro de dicha clase se definió un método **setUp** como muestra el **Código 3.34**. Este código, se encarga de crear objetos de las clases que van a ser evaluadas en las pruebas unitarias.

```
#Pruebas Unitarias
class Test(TestCase):

 # FUncion para ingresar la informacion
 def setUp(self):
    perfil=Perfil.objects.create(nombrePerfil="Adulto",horaInicio ="00:10",horaFin="23:59",
        duracionMaximaLlamada=30,horarioPermitido="Todos los dias")
    d1=DestinosPerfil.objects.create(tipoDestino="Llamadas Locales",perfilAsociado=perfil)
    d2=DestinosPerfil.objects.create(tipoDestino="Llamadas Celulares",perfilAsociado=perfil)
    d3=DestinosPerfil.objects.create(tipoDestino="Numeros de Emergencia",perfilAsociado=perfil)
    persona = Persona.objects.create(nombrePersona="Juan", extension="1010",
        contraseniaAsociacion="juan12345", contraseniaBuzon="2010",nombrePerfil=perfil)
    bloqueoS= BloqueoSimple.objects.create(personaBloqueada=persona,categoriaLLamada="Llamadas Celulares",
        fechaInicio="2015-07-05",fechaFin="2015-08-12",excepcion1="0983581978",excepcion2="0985212140")
    destino=Destino.objects.create(numeroDestino="2345010",categoriaNumero="Locales")
    destinoE=Destino.objects.create(numeroDestino="042362412",categoriaNumero="Provinciales")
    bloqueoA= BloqueoAvanzado.objects.create(personaBloqueada=persona,destinoBloqueado=destino,
        fechaInicio="2015-07-06",fechaFin="2015-07-23",horaBloqueoI="08:30",horaBloqueoF="13:00")
    bloqueoE=LlamadaEntrante.objects.create(numeroEntrante=destinoE,estadoBloqueo="B")
    info=InformacionLlamadas.objects.create(origen="Juan",destino="1020",
        fechaHoraInicio="2015-10-15 10:20:10",fechaHoraFin="2015-10-15 10:22:10",duracionLlamada=2,
        estadoLlamada="Contestada")
```

Código 3.34 Clase Test y Función setUp

A continuación, se proceden a crear ocho métodos que van a realizar las pruebas unitarias sobre las clases: **Persona, Perfil, DestinosPerfil, Destino, BloqueoSimple, BloqueoAvanzado, LlamadaEntrante** e **InfomaciónLlamadas.** Según se muestra desde el **Código 3.35** hasta el **Código 3.42.**

```
#Prueba Unitaria de Persona
def test_Persona(self):
    persona = Persona.objects.get(nombrePersona="Juan")
    self.assertEqual(persona.extension, "1010")
    self.assertEqual(persona.contraseniaAsociacion, "juan12345")
    self.assertEqual(persona.contraseniaBuzon, "2010")
    self.assertEqual(persona.nombrePerfil.nombrePerfil, "Adulto")
```

Código 3.35 Método test_Persona

```
#Prueba Unitaria de Perfil
def test_Perfil(self):
    perfil = Perfil.objects.get(nombrePerfil="Adulto")
    self.assertEqual(perfil.horaInicio,datetime.time(parser.parse("00:10")))
    self.assertEqual(perfil.horaFin,datetime.time(parser.parse("23:59")))
    self.assertEqual(perfil.duracionMaximaLlamada, 30)
    self.assertEqual(perfil.horarioPermitido, "Todos los dias")
```

Código 3.36 Método test_Perfil

```
#Prueba Unitaria de Perfil
def test_Destinos_Perfil(self):
    perfil = Perfil.objects.get(nombrePerfil="Adulto")
    d1=DestinosPerfil.objects.get(perfilAsociado=perfil,tipoDestino="Llamadas Locales")
    self.assertEqual(d1.tipoDestino,"Llamadas Locales")
```

Código 3.37 Método test_Destinos_Perfil

```
#Prueba Unitaria de Destino
 def test_Destino(self):
     destino = Destino.objects.get(numeroDestino="2345010")
     self.assertEqual(destino.categoriaNumero,"Locales")
```

Código 3.38 Método test_Destino

```
#Prueba Unitaria de Bloqueo SImple
def test_Bloqueo_Simple(self):

    personaB=Persona.objects.get(nombrePersona="Juan")
    bloqueado = BloqueoSimple.objects.get(personaBloqueada=personaB)
    self.assertEqual(bloqueado.personaBloqueada.nombrePersona, "Juan")
    self.assertEqual(bloqueado.categoriaLLamada, "Llamadas Celulares")
    self.assertEqual(bloqueado.fechaInicio,datetime.date(parser.parse("2015-07-05")))
    self.assertEqual(bloqueado.fechaFin,datetime.date(parser.parse("2015-08-12")))
    self.assertEqual(bloqueado.excepcion1, "0983581978")
    self.assertEqual(bloqueado.excepcion2, "0985212140")
```

Código 3.39 Método test_Bloqueo_Simple

```
#Prueba Unitaria de Bloqueo Avanzado
def test_Bloqueo_Avanzado(self):
    personaB=Persona.objects.get(nombrePersona="Juan")
    bloqueado = BloqueoAvanzado.objects.get(personaBloqueada=personaB)
    self.assertEqual(bloqueado.personaBloqueada.nombrePersona, "Juan")
    self.assertEqual(bloqueado.destinoBloqueado.numeroDestino, "2345010")
    self.assertEqual(bloqueado.fechaInicio,datetime.date(parser.parse("2015-07-06")))
    self.assertEqual(bloqueado.fechaFin,datetime.date(parser.parse("2015-07-23")))
    self.assertEqual(bloqueado.horaBloqueoI,datetime.time(parser.parse("08:30")))
    self.assertEqual(bloqueado.horaBloqueoF,datetime.time(parser.parse("13:00")))
```

Código 3.40 Método test_Bloqueo_Avanzado

```
#Prueba Unitaria de Bloqueo de Numeros Entrantes
def test_BloqueoLlamada_Entrante(self):
    numeroE=Destino.objects.get(numeroDestino='042362412')
    numBloqueado = LlamadaEntrante.objects.get(numeroEntrante=numeroE)
    self.assertEqual(numBloqueado.numeroEntrante.numeroDestino, "042362412")
    self.assertEqual(numBloqueado.estadoBloqueo, "B")
```

Código 3.41 Método test_BloqueoLlamada_Entrante

```
#Prueba Unitaria de Informacion de Llamadas
def test_Informacion_Llamadas(self):
    utc = pytz.utc
    fechaI = datetime.strptime("2015-10-15 10:20:10",'%Y-%m-%d %H:%M:%S')
    fechaF = datetime.strptime("2015-10-15 10:22:10",'%Y-%m-%d %H:%M:%S')

    personaB=Persona.objects.get(nombrePersona="Juan")
    info1=InformacionLlamadas.objects.get(origen=personaB)
    self.assertEqual(info1.origen, "Juan")
    self.assertEqual(info1.destino, "1020")
    self.assertEqual(info1.fechaHoraInicio,utc.localize(fechaI))
    self.assertEqual(info1.fechaHoraFin,utc.localize(fechaF))
    self.assertEqual(info1.duracionLlamada, 2)
    self.assertEqual(info1.estadoLlamada, "Contestada")
```

Código 3.42 Método test_Informacion_Llamadas

Cada uno de los métodos mostrados previamente trae el objeto creado o guardado por el método **setUp,** y lo compara con el valor que debería tener, para lo cual se utiliza la función ***asserEqual,*** la misma que compara dos parámetros separados por comas; el primer parámetro es el valor obtenido del objeto y el segundo parámetro es el valor esperado.

Para ejecutar las pruebas se utiliza el comando mostrado en el **Código 3.43,** dentro del directorio de la aplicación.

```
python manage.py test nombreApliacion
```

Código 3.43 Comando de ejecución de Pruebas Unitarias

Al finalizar la ejecución de las pruebas unitarias se muestra una pantalla en la que se indica que las ocho pruebas se completaron y funcionan adecuadamente, ver **Figura 3.25.**

```
root@ubuntu:/home/juanm/Documentos/ProyectosDjango/tesis2# python manage.py test telefon
ia
Creating test database for alias 'default'...
........
----------------------------------------------------------------------
Ran 8 tests in 0.052s

OK
Destroying test database for alias 'default'...
```

Figura 3.25 Resultado de Pruebas de Unitarias

De este modo se puede concluir que las clases analizadas en las pruebas unitarias funcionan de manera adecuada.

3.8 PRUEBAS DE ACEPTACIÓN

La metodología XP sugiere el uso de pruebas de aceptación, las mismas que permiten comprobar las características funcionales del producto final. Las pruebas de aceptación provienen de las historias de usuario obtenidas en la etapa de recolección de requerimientos, las mismas son visibles y revisables por el cliente [13].

En base a esta información se precedió a realizar la prueba de cada uno de los requerimientos funcionales determinados en la **sección 2.1.2** que se encuentran presentes en la **Tabla 3.10.** La prueba de aceptación, fue realizada por un padre de familia, el cual fue comprobando la funcionalidad de cada uno de los requerimientos, utilizando la aplicación.

Requerimiento	Tipo de Requerimiento	Historia de Usuario
Creación de Usuarios	Funcional	HU1-01
Edición de Usuarios	Funcional	HU1-04
Eliminación de Usuarios	Funcional	HU1-05
Visualización de los Registros de Llamadas	Funcional	HU1-03
Aplicación de Bloqueo Simple	Funcional	HU1-02
Aplicación de Bloqueo Avanzado	Funcional	HU1-02
Consulta de Buzón de Voz	Funcional	HU1-06
Comprobación de Contestadora Automática	Funcional	HU1-06
Visualización del Instructivo(Ayuda) en el sistema	Funcional	HU2-04
Implementación de excepciones en el bloqueo de destinos (Bloqueo Simple)	Funcional	HU3-01
Aplicación de Bloqueo de Llamadas Entrantes	Funcional	HU3-02

Tabla 3.10 Requerimientos Funcionales utilizados para las Pruebas de Aceptación

El padre de familia realizó las tareas detalladas en el Anexo D, y a continuación llenó una tabla indicando el porcentaje de cumplimiento del requisito e indicando el nivel de aceptación o satisfacción de cada uno de los requisitos.

La **Tabla 3.11** detalla las respuestas que el padre de familia llenó para constatar el cumplimiento de los requisitos. En el Anexo G se puede constatar la tabla original llenada a mano por el padre de familia.

Número de Tarea	Detalle	Porcentaje de Cumplimiento	Satisfacción
1	Creación de Usuarios	100%	Muy Satisfecho
2	Edición de Usuarios	100%	Muy Satisfecho
3	Eliminación de Usuarios	100%	Muy Satisfecho
4	Realización de Bloqueo Simple	100%	Muy Satisfecho
5	Aplicación de Bloqueo Avanzado	100%	Muy Satisfecho
6	Aplicación de Bloqueo de Llamadas Entrantes	100%	Muy Satisfecho
7	Consulta de Buzón de Voz	100%	Muy Satisfecho
8	Comprobación de Contestadora Automática	100%	Muy Satisfecho
9	Visualización de Registros de Llamadas	100%	Muy Satisfecho
10	Visualización de Instructivo(Ayuda) en el sistema	100%	Muy Satisfecho

Tabla 3.11 Detalle de Cumplimiento de Pruebas de Aceptación

3.9 PRUEBAS DE USABILIDAD DE LA APLICACIÓN

La usabilidad puede definirse como “la medida en que un producto puede ser usado por usuarios específicos para alcanzar metas específicas, con efectividad, eficiencia y satisfacción”, según define el estándar ISO 9241-11 1998 [36].

La presentación de las interfaces gráficas es de gran importancia en la usabilidad puesto que interactúan directamente con el usuario. Por lo tanto se debe diseñar las mismas, con el propósito de garantizar la satisfacción del cliente y la calidad del producto final (software). El diseño de una aplicación web usable permite al usuario interesarse en el cumplimiento de su trabajo mas no en la aplicación misma [36].

La usabilidad de la aplicación web se evaluó en base a tres parámetros que son la facilidad de aprendizaje, la eficiencia y la satisfacción.

3.9.1 MEDIDA DE LA FACILIDAD DE APRENDIZAJE

La facilidad de aprendizaje es una métrica que evalúa la rapidez y la facilidad con la que los usuarios pueden comenzar a realizar un trabajo productivo en el sistema. Puede ser medida como el tiempo que le toma al usuario llegar del nivel novato, al nivel experto [36].

La evaluación de este parámetro consistió en medir el número de intentos que realizaron 10 usuarios y la demora en cada intento para pasar del nivel novato al nivel experto. La prueba radicó en tomar el tiempo que cada persona se demoró en realizar las tareas descritas en el Anexo D.

En la **Tabla 3.12**, se muestran las mediciones de cada usuario, donde se puede apreciar el número de intentos en que los usuarios trataron de alcanzar un tiempo cercano al valor del usuario de nivel experto, el cual es de **2 minutos** con **50 segundos.**

Usuario	Tiempo 1	Tiempo 2	Tiempo 3	Tiempo 4	Tiempo 5
Diego Guacho	0:04:11	0:02:46			
René Arellano	0:07:04	0:05:32	0:03:44	0:03:23	0:02:58
María Fernanda Negrete	0:07:21	0:03:41	0:03:05	0:02:51	
Jorge Vallejo	0:04:39	0:03:21	0:02:53		
Joffre Bastidas	0:06:52	0:03:31	0:03:10	0:02:52	
Cristhina Encalada	0:05:23	0:03:27	0:02:36		
Ena Bastidas	0:07:05	0:04:33	0:03:57	0:03:28	0:02:57
Miguel Luna	0:05:13	0:03:40	0:03:08	0:02:45	
Roberto Guayasamín	0:05:34	0:03:37	0:02:50		
Patricia Guayasamín	0:05:08	0:03:56	0:02:40		

Tabla 3.12 Tiempos de Demora en Prueba de Facilidad de Aprendizaje

El parámetro de nivel experto que se consideró fue el tiempo que el desarrollador del prototipo se demoró en realizar las tareas indicadas en el Anexo D. Analizando a cada usuario se puede concluir que al realizar el segundo intento el tiempo de demora bajó casi a la mitad en la mayoría de los casos, y puede considerarse entonces que el sistema es más fácil de utilizar al segundo intento de conocerlo.

En los siguientes intentos el tiempo disminuirá según factores propios del usuario como la rapidez al digitar, la cual es propia de cada uno y depende del uso constante del computador.

Además, se preguntó a cada usuario la edad y el tiempo que utilizan un computador al día, estos resultados están estrechamente relacionados con la usabilidad del prototipo y se muestra en la **Tabla 3.13.**

De acuerdo a los datos de la **Tabla 3.12**, el 50% de los usuarios trato de acercarse al tiempo referencial al tercer intento. Al cuarto intento el 80% se acercó a la referencia, finalmente se realizó un quinto y último intento para los dos últimos usuarios, quienes lograron disminuir su tiempo y llegar a un valor muy cercano al tiempo referencial.

Usuario	Edad	Horas Promedio de Uso del Computador
Diego Guacho	27	10
René Arellano	55	1
María Fernanda Negrete	47	6
Jorge Vallejo	35	8
Joffre Bastidas	32	7
Cristhina Encalada	31	7
Ena Bastidas	33	2
Miguel Luna	39	6
Roberto Guayasamín	43	8
Patricia Guayasamín	54	8

Tabla 3.13 Edad y Horas Promedio de Uso del Computador de los Usuarios de Prueba

En la **Figura 3.26** se puede observar de mejor manera el número de intentos realizados y el porcentaje de usuarios por cada uno.

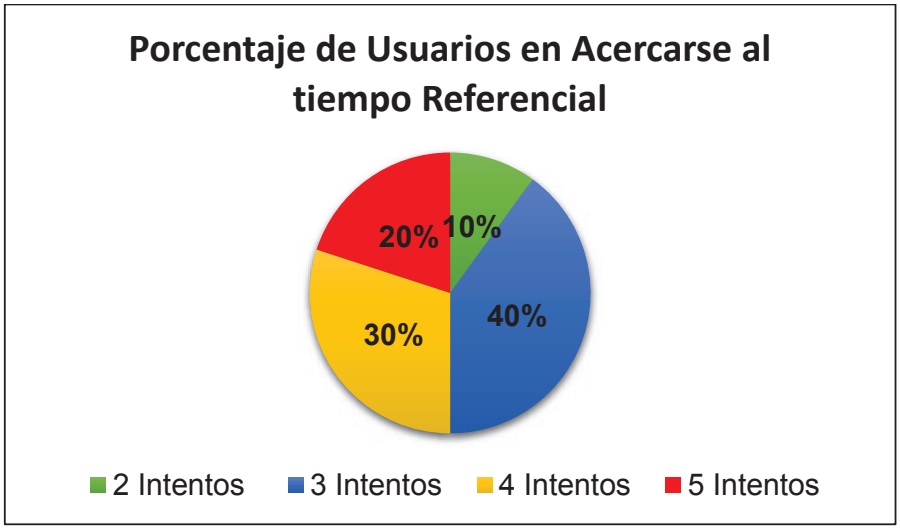

Figura 3.26 Porcentaje de Usuarios e Intentos en adquirir el Nivel Avanzado

Finalmente se realizó una gráfica de Intentos vs Tiempo de dos usuarios, se consideró al más rápido en aprender y el más lento, como muestra la **Figura 3.27**. Claramente se puede observar que el usuario más lento presenta una curva con una pendiente más pronunciada en color rojo en comparación a la curva en color azul perteneciente al usuario más rápido. El Anexo G contiene la gráfica de Intentos vs Tiempo de todos los usuarios que realizaron la prueba.

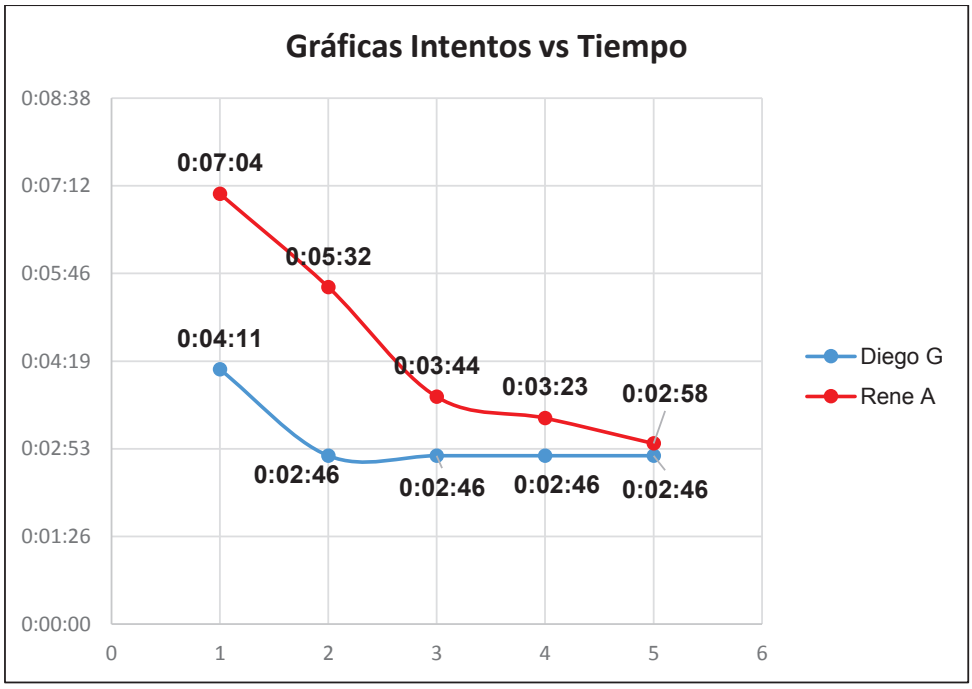

Figura 3.27 Gráfica comparativa de Facilidad de Aprendizaje

3.9.2 MEDIDA DE LA EFICIENCIA

La eficiencia es una métrica que evalúa el nivel de productividad del usuario cuando utiliza el sistema. Puede medirse como el número de tareas por unidad de tiempo que el usuario experto es capaz de llevar a acabo manipulando el sistema [36].

En base a este criterio se preparó un escenario presente en el Anexo D, para evaluar al mismo grupo de usuarios considerados en la **sección 3.9.1**. En primer lugar se midió el tiempo que se demora un usuario avanzado en completar las configuraciones del Anexo D, considerando al desarrollador del prototipo como usuario avanzado.

Posteriormente cada uno de los usuarios realizó las mismas configuraciones detalladas en el Anexo D, mientras realizaban la prueba se tomó el tiempo que se demoraron en cumplir las tareas que se señalan a continuación:

- Configuración de José (padre)
- Configuración de Elena (madre)
- Configuración de Miguel (hijo mayor)
- Configuración de Diana (hija menor)
- Aplicación del Bloqueo Simple
- Aplicación del Bloqueo Avanzado
- Aplicación del Bloqueo de una Llamada Entrante.

Los resultados obtenidos después de realizar la prueba se detallan en la **Tabla 3.14**.

Las tareas escogidas durante la prueba no tienen el mismo tiempo de demora para llevarse a cabo, ya que son distintas (distinto grado de complejidad), debido a esto se realizó un promedio de duración de las tareas para cada usuario con la finalidad de poder determinar el número de tareas por unidad de tiempo que se muestra en la **Tabla 3.15**.

USUARIO	Login y Creación del padre José	Creación de la madre Elena	Creación del hijo Miguel	Creación de la hija Diana	Aplicación del Bloqueo Simple	Aplicación del Bloqueo Avanzado	Bloquear el Número Entrante	Total
Usuario Avanzado	**0:00:45**	**0:00:34**	**0:00:35**	**0:00:33**	**0:01:09**	**0:00:58**	**0:00:33**	**0:05:07**
Diego Guacho	0:00:52	0:00:37	0:00:33	0:00:34	0:01:05	0:01:01	0:00:26	**0:05:08**
Jorge Vallejo	0:00:43	0:00:38	0:00:35	0:00:32	0:01:08	0:01:03	0:00:31	**0:05:10**
Roberto Guayasamín	0:00:46	0:00:36	0:00:33	0:00:34	0:01:06	0:01:02	0:00:28	**0:05:05**
Cristhina Encalada	0:00:47	0:00:37	0:00:38	0:00:36	0:01:08	0:01:10	0:00:27	**0:05:23**
Miguel Luna	0:00:47	0:00:37	0:00:37	0:00:34	0:01:07	0:01:10	0:00:32	**0:05:24**
René Arellano	0:01:01	0:00:39	0:00:36	0:00:37	0:01:05	0:01:11	0:00:28	**0:05:37**
María Fernanda Negrete	0:00:49	0:00:43	0:00:40	0:00:38	0:01:07	0:01:10	0:00:31	**0:05:38**
Patricia Guayasamín	0:00:49	0:00:40	0:00:39	0:00:37	0:01:22	0:00:57	0:00:35	**0:05:39**
Joffre Bastidas	0:01:02	0:00:49	0:00:45	0:00:40	0:01:04	0:01:08	0:00:25	**0:05:53**
Ena Bastidas	0:00:57	0:00:46	0:00:48	0:00:44	0:01:05	0:01:06	0:00:29	**0:05:55**

Tabla 3.14 Tiempos de Demora de los Usuarios en la Prueba de Eficiencia

En la segunda columna de la **Tabla 3.15** se presenta el tiempo medido en segundos, que se demora cada usuario en promedio en desarrollar una tarea. En la tercera columna se calcula el número de tareas por minuto considerando la información de la columna número 2.

USUARIO	Promedio de Duración de una tarea [S]	Número de Tareas por minuto
Usuario Avanzado	**44**	**1,36**
Diego Guacho	44	1,36
René Arellano	48	1,25
María Fernanda Negrete	48	1,25
Jorge Vallejo	44	1,36
Joffre Bastidas	50	1,2
Cristhina Encalada	46	1,30
Ena Bastidas	51	1,18
Miguel Luna	46	1,30
Roberto Guayasamín	44	1,36
Patricia Guayasamín	48	1,25

Tabla 3.15 Número de tareas por minuto por cada Usuario

De los resultados obtenidos en la **Tabla 3.15,** puede decirse que el usuario avanzado en promedio puede realizar 1 tarea, y un 36% de una segunda en un 1 minuto. Considerando el grupo de usuarios analizados tres de ellos llegan a igualar al usuario avanzado; por otro lado el usuario más bajo llega a un valor de 1 tarea y un 18% de ella. Como puede verse en **Figura 3.28**.

Los factores como la rapidez al digitar y la utilización de un mouse influyen en los resultados obtenidos. Dichos factores se adquieren con el uso constante del computador, de este modo se justifica el rendimiento más bajo de algunos usuarios.

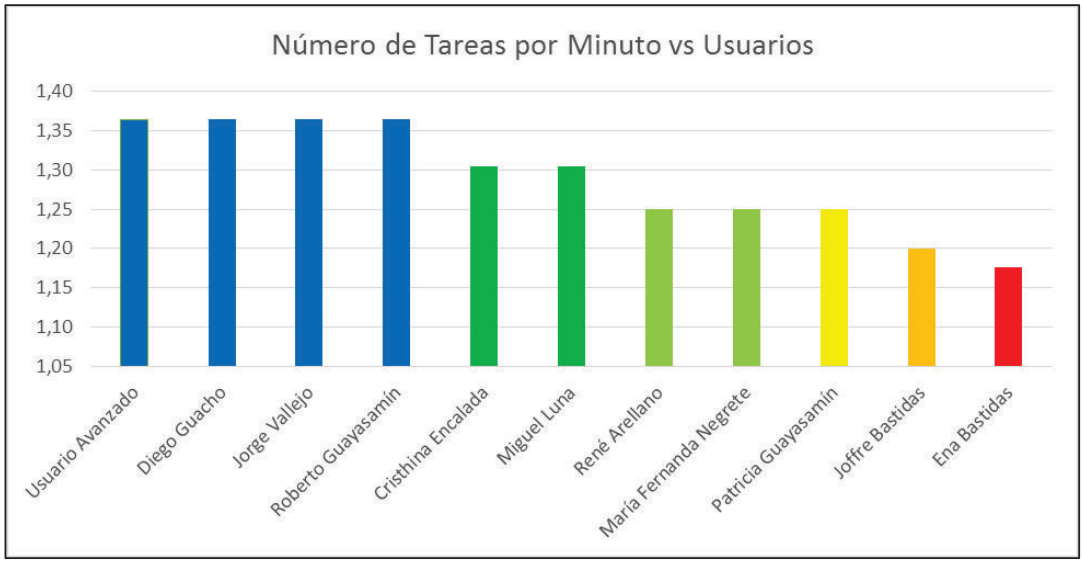

Figura 3.28 Gráfica de Tareas por Minuto Vs Usuarios

Sin embargo, frente a todos los parámetros mencionados, el sistema se considera eficiente puesto que usuarios que no utilizan con frecuencia el computador puedan llegar a tener un desempeño cercano a un usuario avanzado. Este hecho queda demostrado al considerar el caso más bajo de 1.18 tareas por minuto, el cual equivale a un 86.3% del valor del usuario avanzado tomado como referencia.

3.9.3 MEDIDA DE LA SATISFACCIÓN

La satisfacción es una métrica evaluada mediante la opinión que tiene el usuario frente a la aplicación, la cual puede ser evaluada utilizando cuestionarios [36].

El mismo grupo analizado realizó un cuestionario para poder evaluar la satisfacción del sistema. El cuestionario contiene 14 preguntas orientadas a la perspectiva del usuario frente a: la localización de varios elementos en la página web, la retroalimentación de la aplicación, la información presentada por cada una de las páginas, la facilidad de realizar ciertas tareas y finalmente la satisfacción del sistema.

Las preguntas planteadas para medir la satisfacción del cliente, se basaron en una la guía presente en el Anexo E [37]. La gran mayoría de respuestas obtenidas presentan una buena aceptación de la aplicación desarrollada. En concreto la pregunta relacionada a la satisfacción de la aplicación, indica que el 100% de las

personas califican como 5 al grado de satisfacción, el equivalente a Muy Satisfecho, según se muestra en la **Figura 3.29**.

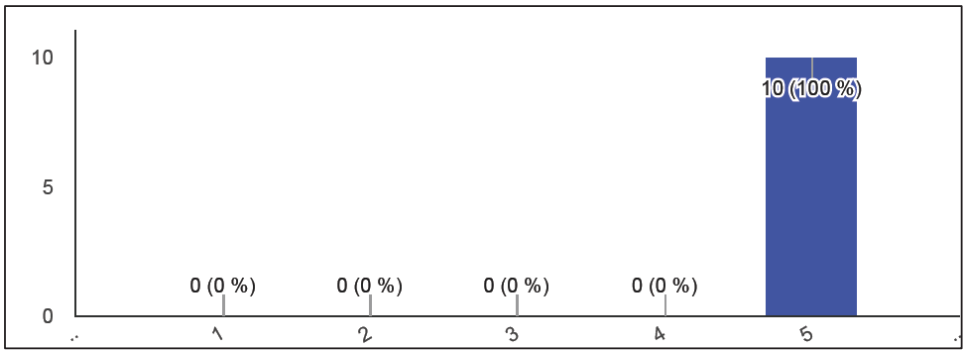

Figura 3.29 Grado de Satisfacción de la Aplicación

Las preguntas que se realizaron en el cuestionario se encuentran en el Anexo D y las respuestas provenientes de la encuesta de Satisfacción se pueden ver en el Anexo E.

CAPÍTULO 4

4 CONCLUSIONES Y RECOMENDACIONES

4.1 CONCLUSIONES

- En base a la encuesta presente en el Anexo A, realizada para determinar los requisitos de telefonía en un entorno doméstico, un 74.6% de los padres de familia encuestados indican que les gustaría poder controlar el uso de llamadas de sus hijos en el entorno doméstico, por lo cual se justifica los beneficios de implementar el prototipo desarrollado en el presente proyecto de Titulación.

- Según los resultados de la encuesta presente en el Anexo A, una de las molestias más comunes que los padres de familia presentan, es la falta de un registro de llamadas realizado en su hogar. Por este motivo, el prototipo implementado permite llevar un control a través de un registro visual sobre el número de llamadas que realizan los miembros del hogar, así como también la duración de las llamadas realizadas en el entorno doméstico.

- El prototipo desarrollado está basado en la herramienta *Open Source* Asterisk, muy conocida y difundida en lo referente a software para Telefonía IP, debido a su flexibilidad al permitir la implementación de servicios de Telefonía IP como buzón de voz y contestadora automática. Los servicios implementados y los requisitos, fueron obtenidos en función de la encuesta presente en el Anexo A.

- Las herramientas de gestión de Asterisk como Elastix y FreePBX fueron en sus inicios orientadas a ser utilizadas en un entorno empresarial, específicamente a usuarios con conocimientos técnicos. Por lo tanto, dichas herramientas no son viables para ser utilizadas en un ambiente doméstico, por este motivo, se escogió implementar Asterisk puro y desarrollar una aplicación de gestión más fácil de utilizar dirigida a cualquier tipo de

personas, sin la necesidad de tener conocimientos técnicos avanzados de telefonía IP.

- Las pruebas realizadas en la **sección 3.5** indican que el servicio telefónico implementado en el prototipo funciona correctamente. Puesto que permite realizar llamadas dentro y fuera del hogar, bloquear llamadas salientes de manera muy granular(por fecha y por horario), bloquear llamadas por destinos e implementar los servicios de buzón de voz y contestadora automática (IVR), todo acorde a los requisitos manifestados por los padres de familia según la encuesta realizada presente en el Anexo A.

- Las pruebas realizadas en la **sección 3.6.1** y **sección 3.6.2** indican que Raspberry Pi garantiza su funcionamiento para el escenario más crítico en un ambiente doméstico con una familia típica de 4 personas, realizando 2 llamadas simultáneas y utilizando la aplicación de gestión del servicio telefónico. Por lo tanto se concluye que el hardware es lo suficientemente robusto como para ejecutar la aplicación desarrollada.

- Las pruebas planteadas en la **sección 3.9.1** indican que la aplicación desarrollada es usable utilizando la métrica de facilidad de aprendizaje, puesto que los usuarios se demoran cada vez menos tiempo en completar las tareas propuestas en un escenario dado. En consecuencia, se puede garantizar que los usuarios alcanzarán un nivel avanzado, en el uso de la aplicación, de manera muy rápida.

- En base a los resultados obtenidos en las pruebas de usabilidad mediante la métrica de eficiencia, se concluye que el sistema es eficiente. Puesto que permite a usuarios con un manejo del computador muy poco continuo, acercarse al desempeño de un usuario avanzado, a pesar de factores como la rapidez al teclear y el manejo del mouse, las cuales son características propias de cada usuario.

- Los resultados obtenidos en la encuesta del Anexo E indican que la aplicación desarrollada es usable bajo la métrica de Satisfacción del Cliente.

Los resultados reflejan un elevado grado de aceptación por parte de los usuarios.

- Raspberry Pi es un computador de placa reducida, de fácil instalación y configuración. Este dispositivo permite implementar un servicio de Telefonía IP con una interfaz de administración Web a un bajo costo, si se compara con los costos que representa implementar un Central Telefónica IP. Las pruebas realizadas sobre el prototipo demostraron que a pesar de la limitada capacidad de memoria y procesamiento de Raspberry Pi, el funcionamiento de la Central Telefónica IP para un entorno doméstico queda garantizada.

- Para el desarrollo de la aplicación web se escogió el framework Django el cual es muy recomendable debido a que presenta ventajas como la programación rápida y organizada puesto que se basa en la arquitectura Modelo Vista Controlador MVC, lo que hace que las modificaciones en un módulo no afecten a los módulos restantes. Además utiliza el lenguaje de Programación Python que es muy sencillo, fácil de aprender y además es flexible.

4.2 RECOMENDACIONES

- En el desarrollo de la aplicación web se recomienda en lo posible mantener por separado el archivo de la página web (.html) de los archivos que proporcionan los estilos (.css) y de los archivos que dan dinamismo a la página web (.js), con el objetivo de tener una mejor organización.

- Durante el tiempo que duren las llamadas telefónicas se recomienda no hacer cambios en el sistema en lo que respecta a la creación, edición y borrado de usuarios, aplicación de bloqueos y desbloqueos, y edición de grupos comunes; debido a que estas tareas requieren un reinicio del sistema de Telefonía IP y puede ocasionar el fin de una llamada telefónica. En este sentido se podría implementar en la aplicación web un mensaje de alerta que notifique al padre de familia, que existe una llamada en curso.

- Se recomienda designar a una única persona y un único computador para acceder a la aplicación de gestión para configurar los parámetros necesarios en el hogar, puesto que varias sesiones iniciadas en el servidor podrían traer problemas con la edición de archivos.

- El sistema incluye un manual de utilización de la aplicación, el cual se recomienda revisarlo antes de utilizar el prototipo para tener una guía sobre todas las funcionalidades que presenta el prototipo. De igual manera se recomienda utilizar los iconos y frases de ayuda con los cuales cuenta la aplicación, puesto que podrían colaborar con el usuario para que cumpla con una tarea que requiera realizar.

- Dado que un sistema perfecto no existe, se recomienda aplicar nuevas encuestas de satisfacción de uso de la aplicación al usuario padre de familia, a fin de identificar cambios que se pueden implementar en la interfaz para facilitar su uso.

- Se recomienda considerar el prototipo realizado como base para futuros trabajos e investigaciones que estén orientados a desarrollar soluciones de telefonía IP innovadoras y de bajo costo.

REFERENCIAS BIBLIOGRÁFICAS

[1] C. S. V. Murthy, “IP TELEPHONY FINDS ITS VOICE,” in *Data Communication and Networking.*, no. September, Mumbai: Himalaya Publishing House, 2010, p. 467.

[2] D. D.-G. Margit Brandl - Karl Franzens - UNI Graz and R. G. G.-U. of P. Erik Dobbelsteijn - SURFnet, “Technological Background,” in *IP Telephony Cookbook*, TERENA, Ed. 2004, pp. 11–13.

[3] N. J. Muller, “Historical Perspective,” in *LANs to WANs : The Complete Management Guide*, no. June, A. House, Ed. Norwood, Massachusetts, 2003, p. 149.

[4] “Conference Bridge | Asterisk.org.” [Online]. Available: http://www.asterisk.org/get-started/applications/conference. [Accessed: 22-Oct-2015].

[5] W. Flanagan, *Understanding VoIP and Unified Communications : Internet Telephony and the Future Voice Network*, no. October. 2012.

[6] D. Dempster, B., Gomillion, D., & Merel, “Introduction to Asterisk,” in *Asterisk 1.6 : Build Feature-Rich Telephony Systems with Asterisk*, Olton, Birmingham, 2009, pp. 9–14.

[7] R. B. Leif Madsen, Jim Van Meggelen, *Asterisk The Definitive Guide*, Third Edit., vol. XXXIII, no. 2. 2011.

[8] “Elastix - Overview,” 2015. [Online]. Available: http://www.elastix.org/index.php/es/informacion-del-producto/informacion.html. [Accessed: 22-Oct-2015].

[9] A. Robar, *FreePBX 2. 5 Powerful Telephony Solutions : Configure, Deploy, and Maintain an Enterprise-Class VoIP PBX*. 2009.

[10] R. P. I. FOUNDATION, “WHAT IS A RASPBERRY PI?,” 2015. [Online]. Available: https://www.raspberrypi.org/help/what-is-a-raspberry-pi/. [Accessed: 03-Nov-2015].

[11] G. Upton, E., & Halfacree, “Meet the Raspberry Pi,” in *Raspberry Pi User Guide (2nd Edition)*, 2nd ed., no. November, I. John Wiley & Sons, Ed. John Wiley & Sons, Incorporated, 2013, pp. 15–17.

[12] P. K. J. Mohapatra, “Write the Requirements,” in *Software Engineering*, no. October, N. A. International, Ed. Delhi: New Age International, 2010.

[13] Roger S. Pressman, *Ingeniería del Software un Enfoque Práctico*, 7th ed., vol. 33. Mexico, 2012.

[14] “Diagramas de Casos de Uso.” [Online]. Available: http://datateca.unad.edu.co/contenidos/200609/exeuml/leccin_17_diagramas_de_casos_de_uso.html. [Accessed: 08-Nov-2015].

[15] J. M. Beas, “Historias de Usuario.” .

[16] I. Sommerville, *Ingeniería del software*, 7th ed. Madrid, 2005.

[17] K. Beck, M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. C. Martin, S. Mellor, K. Schwaber, J. Sutherland, and D. Thomas, “Principios del Manifiesto Ágil.” [Online]. Available: http://agilemanifesto.org/iso/es/principles.html. [Accessed: 05-Nov-2015].

[18] J. Tomayko and O. Hazzan, “Software Engineering Methods,” in *Human*

Aspects of Software Engineering, no. November, 204AD, pp. 18–22.
[19] E. Bascón Pantoja, “El patrón de diseño Modelo-Vista-Controlador (MVC) y su implementación en Java Swing,” *Acta Nov.*, vol. 2, no. Mvc, pp. 493–507, 2004.
[20] Y. Fernández and Y. González, “Patrón Modelo-Vista-Controlador,” *Revista Telemática*, vol. 11, no. 1, pp. 47–57, 2012.
[21] J. Játiva, “Informe de Actividades,” Quito, 2015.
[22] “¿Qué es un nivel de confianza?” [Online]. Available: http://support.minitab.com/es-mx/minitab/17/topic-library/basic-statistics-and-graphs/introductory-concepts/confidence-interval/confidence-level/. [Accessed: 16-Jun-2016].
[23] F. Egüez Delgado, *Estadística para Investigación, Manual Práctico*. Quito, 2007.
[24] D. Peláez and C. Tipantuña, “Servidor de comunicaciones unificadas con Raspberry Pi y Micro-Elastix,” pp. 293–302, 2014.
[25] “Win32DiskImager – Raspberry Pi Projects.” [Online]. Available: http://www.raspberry-projects.com/pi/pi-operating-systems/win32diskimager. [Accessed: 28-Apr-2016].
[26] “Usando Call ID en Asterisk y Grandstream usando como PSTN a Pacifictel - CNT Costa | EcuaLUG.” [Online]. Available: http://www.ecualug.org/?q=2010/03/19/forums/usando_call_id_en_asterisk_y_grandstream_usando_como_pstn_pacifictel_cnt_costa. [Accessed: 23-Mar-2016].
[27] “FrontPage - Python Wiki.” [Online]. Available: https://wiki.python.org/moin/. [Accessed: 08-Mar-2016].
[28] “Maestros de la WEB,” Jun-2012. [Online]. Available: http://www.maestrosdelweb.com/ventajas-python/. [Accessed: 08-Mar-2016].
[29] H. Boparai, “10 Reasons Why Python Scores Over PHP for Web Development,” Oct-2014. [Online]. Available: http://www.netsolutionsindia.com/blog/10-reasons-why-python-scores-over-php-for-web-development/. [Accessed: 08-Mar-2016].
[30] A. Holovaty and J. Kaplan-Moss, “Chapter 1: Introduction to Django.” [Online]. Available: http://www.djangobook.com/en/2.0/chapter01.html. [Accessed: 14-Mar-2016].
[31] “6. Models and Databases — How to Tango with Django 1.7.” [Online]. Available: http://www.tangowithdjango.com/book17/chapters/models.html. [Accessed: 14-Mar-2016].
[32] A. Holovaty and J. Kaplan-Moss, “Chapter 2: Getting Started.” [Online]. Available: http://www.djangobook.com/en/2.0/chapter02.html. [Accessed: 14-Mar-2016].
[33] A. Holovaty and J. Kaplan-Moss, “4. Django Basics — How to Tango with Django 1.7.” [Online]. Available: http://www.tangowithdjango.com/book17/chapters/setup.html#creating-a-django-application. [Accessed: 15-Mar-2016].
[34] “5. Templates and Static Media — How to Tango with Django 1.7.” [Online]. Available: http://www.tangowithdjango.com/book17/chapters/templates_static.html. [Accessed: 15-Mar-2016].
[35] INEC, “Encuesta Nacional de Ingresos y Gastos (ENIGHUR) 2011- 2012.”

2012.
[36] Hayser Jacquelín Beltré Ferreras, “APLICACIÓN DE LA USABILIDAD AL PROCESO DE DESARROLLO DE PÁGINAS WEB,” Universidad Politécnica de Madrid, 2008.
[37] F. Almazán and J. Camus, “Modelo de Test de Usuario,” *Test*, pp. 1–15.

ANEXOS

Todos los Anexos se encuentran presentes en el CD adjunto.

Anexo A: Encuesta de Requisitos de los Servicios de Telefonía

Anexo B: Bosquejo de Primeras Vistas de la Aplicación.

Anexo C: Manual de Instalación y Configuración del Sistema.

Anexo D: Escenarios planteados para: Pruebas de Aceptación y Pruebas de Usabilidad.

Anexo E: Resultados de las Encuestas de medida de Satisfacción del Usuario y Guía de Preguntas para medir la Satisfacción.

Anexo F: Manual de Utilización de la Aplicación.

Anexo G: Imágenes:

- Diagrama de Clases completo de la Aplicación.
- Gráfica de comportamiento de Intentos vs Tiempos en la Prueba de facilidad de aprendizaje.
- Tabla de Pruebas de Aceptación del Cliente.

Anexo H: Videos de Pruebas de Funcionalidad.

Anexo I: Datos de las mediciones de uso de CPU y memoria proporcionados por PRTG.

Anexo J: Código Fuente de la Aplicación.