Desarrollo de la mensajería múltiple en el correo de voz de la telefonía fija

Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas.

Autor: Rosalina Sayú Savón

Tutor(es): Ing. Jommy Alexis Barbán Pérez
Ing. José Manuel Santos Alonso

La Habana, Junio 2014

"Año 56 de la Revolución"

DECLARACIÓN DE AUDITORÍA.

Declaro que soy la única autora del trabajo "**Desarrollo de la mensajería múltiple en el correo de voz de la telefonía fija**" y autorizo a la Facultad 4 de la Universidad de Ciencias Informáticas y al Ministerio de Comunicaciones a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los días___del mes_____del año _____ .

_____

Firma del Autor

Rosalina Sayú Savón.

_____ _____

Firma del Tutor.
Ing. Jommy Alexis Barbán Pérez.

Firma del tutor.
Ing. José Manuel Santos Alonso.

_____

Firma del Consultor:

Ing. Fabiannys Gámez Abad.

"La independencia depende de la tecnología, depende de la ciencia en el mundo de hoy"

Fidel

DEDICATORIA

Dedicado a:

Mis padres por ser los mejores del mundo…

A mi hermana por estar siempre a mi lado…

A mi sobrino por existir…

AGRADECIMIENTOS

A mi mamá por estar apoyándome en todos los momentos de mi vida, siempre supo cómo demostrar su amor y orgullo, sé que aunque no puedo estar presente, está pensando en mí y en este momento tan esperado. Sé que siempre tuvo la certeza de que yo podía llegar y le agradezco por darme fuerzas desde lejos, te adoro con el alma mami.

A mi papito lindo por ser el mejor padre del mundo al cual debo mi vida y ser la persona que soy, no te puedes imaginar lo orgullosa que estoy de tener un padre como tú, es algo que no sé cómo explicarlo, por ti y mi mama soy fuerte no sé lo que haré cuando ya no estén, te quiero mucho papi gracias a ti.

A mi hermana del alma, espero que nunca nos distanciemos y seamos como una sola persona como siempre lo hemos sido, tú eres una de las personas más importantes de mi vida, gracias por soportarme y quererme todos estos años.

A mi familia por creer en mí, en especial a mis abuelos Emna, Victor, Gonzalina y Pedro, sé que me apoyan desde el cielo y desde el lejano Guantánamo, a mis tutores por todo el apoyo que me han

dado todo este tiempo, a mi oponente por ayudarme y ser paciente y al tribunal.

Agradecer a las primeras mejores amigas que tuve en esta universidad Gloria y Lisbeth, con ellas aprendí a tener confianza en las personas, el significado de la palabra amistad y abrirme a otras gentes que no fueran mi familia, gracias por eso.

A Dalianne y Leslie por su cariño y amistad a Yudismary por comportarse como una tutora todos estos días, con ustedes pase los mejores momentos de alegrías y fiestas, me causa dolos no haberlas conocido en 1er año y que nuestra amistad fuera tan corta, ustedes me han enseñado mucho, espero que nunca perdamos el contacto, las quiero a todas, ustedes mis amigas son mi otra familia nunca olvidaré todo lo que han hecho por mí, gracias de todo corazón a Liena e Irainis.

A mi amigo Carlos por su cariño y consejos, a los fonis (Dainier, Pavel y Yander), a Cosme por ser como mi hermano, al antiguo grupo 8105 y 4504 gracias por ser los mejores y estar presentes en mi vida.

A Leonel, Chaviano, Jorge, Amet, Edwin, Pepe, Virgilio, Rodiel, Rene, Felo muchas gracias por su ayuda y por sus cuentas de Internet a todos mis amigos en general, los que no he mencionado no se preocupen los quiero igual.

A María Elena por ayudarme cuando lo necesité, al profe Desagües, Aliannys, muchas gracias a Faviannys y Duany por su ayuda, ustedes fueron el eslabón fundamental que posibilitó que yo estuviera parada aquí.

A Victor, Yoandrys, Yuli y a todos los hermanos, a Dios por escucharme y a la Revolución por hacerse sentir, a todos los amigos que no pudieron llegar al final de la carrera: Kirenia, Erenia, Karina, Luis, a todo mi grupo de 1er año, gracias por haber sido mis amigos nunca los olvidaré.

En general a todas las personas que me quieren y los que no también ya que me enseñaron a ser fuerte y a pasar todos los obstáculos de la vida.

Muchas gracias por existir los quiero mucho.

RESUMEN

El creciente desarrollo de las telecomunicaciones en Cuba, ha provocado que las empresas busquen soluciones informáticas para asegurar un mejor desempeño. El Ministerio de Comunicaciones (MICOM) y la Empresa de Telecomunicaciones (ETECSA) no se encuentran ajenos a estas soluciones, uno de los servicios que se desea valorar para su puesta en práctica por parte de estas entidades es el envío de mensajes de voz, mediante un sistema que permita de forma simultánea transferir información de voz utilizando un teléfono fijo. Para la puesta en práctica de este servicio se hace necesaria la creación de una Respuesta de Voz Iterativa (IVR), la cual mediante grabaciones sirva de guía para interactuar con el sistema, además se requiere una aplicación *web* que permita visualizar y gestionar las listas de destinatarios. Para guiar el proceso de desarrollo del software se seleccionó la metodología XP, UML como lenguaje de modelado y como *framework* se utilizó Symfony 2.3.7 apoyado del lenguaje de programación PHP 5.3.13. También fueron seleccionadas las tecnologías del lado del cliente y del lado del servidor, siendo estas: HTML 5, JavaScript en su versión 9.7 y CSS 3, el SGBD MySQL 5.5.24, el servidor *web* Apache 2.2.22, como central telefónica y servidor de comunicación *Asterisk* 1.8 para crear el IVR, se utilizó el NetBeans 7.3 como Entorno de Desarrollo Integrado y *Visual Paradigm* for UML como herramienta CASE. Se concluye la investigación con las pruebas unitarias y de aceptación del software, las cuales demostraron la correcta implementación de las funcionalidades.

Palabras claves: *Asterisk*, IVR, mensajería página *web*.

ÍNDICE

INTRODUCCIÓN 15
CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA 20
1.1. Marco Conceptual 20
1.1.1. Telefonía fija 20
1.1.2. *Asterisk* 21
1.1.3. Mensaje de voz 22
1.1.4. *Interactive Voice Response* (IVR) 22
1.2. Estudio de soluciones similares 23
1.2.1. *Phone Broadcast* 23
1.2.2. *Ifbyphone* 23
1.2.3. *DialMyCalls* 24
1.2.4. Conclusiones acerca de los sistemas similares en el mundo. 24
1.3. Situación en Cuba 25
1.4. Ambiente de desarrollo 26
1.4.1. Metodologías de desarrollo de software 26
1.4.2. *Framework* de desarrollo en PHP 29
1.4.3. Herramienta CASE para el modelado 31
1.4.4. Lenguaje de modelado 32
1.4.5. Lenguajes de programación 33
1.4.6. Entorno Integrado de Desarrollo NetBeans 7.3 35
1.4.7. Sistemas de Gestores de Bases de Datos (SGBD) 35
1.4.8. Servidores para aplicación web 36
1.5. Conclusiones del capítulo 37
CAPÍTULO 2. CARACTERÍSTICAS DEL SISTEMA. EXPLORACIÓN Y PLANIFICACIÓN 38
2.1. Objeto de automatización 38
2.2. Propuesta del sistema 39
2.2.1. Sistema de Servicio Telefónico Iterativo 40
2.2.2. Sistema de Gestión de Listas Telefónicas 42
2.2.3. Flujo de procesos del sistema propuesto1 43
2.2.4. Actores involucrados en el sistema 43
2.3. Funcionalidades del Sistema de Gestión de Listas de Mensajes Telefónicos (SIGLIMT) 44
2.4. Lista de Reserva del producto 45

2.5. Arquitectura 47
2.5.1. Propuesta de Arquitectura del sistema 48
2.5.2. Arquitectura Cliente-Servidor 48
2.5.3. Estilos arquitectónicos 49
2.6. Fase de exploración 50
2.6.1. Historias de usuario 50
2.7. Fase de Planificación 54
2.7.1. Estimación por historia de usuario 54
2.7.2. Plan de la Iteración. 55
2.7.3. Iteración 1. 55
2.7.4. Iteración 2. 56
2.7.5. Iteración 3. 56
2.7.6. Plan de duración de las iteraciones. 56
2.7.7. Plan de Entrega. 56
2.8. Conclusiones del capítulo 57
CAPÍTULO 3. DISEÑO DEL SISTEMA 58
3.1. Patrones Generales para la Asignación de Responsabilidades. 58
3.1.1. Patrones GOF (*Gang Of Four*, Banda de Cuatro). 58
3.1.2. Patrones generales de *software* para la asignación de responsabilidades (GRASP). 58
3.2. Tarjetas Clase- Responsabilidad-Controlador (CRC) 62
3.3. Modelo físico de la Bases de datos 62
3.4. Módulos del Sistema de Servicios Telefónico Iterativo 62
3.4.1. Módulos para la conexión con la Base de Datos 65
3.5. Conclusiones del capítulo 65
CAPÍTULO 4. IMPLEMENTACIÓN Y PRUEBA 66
4.1. Fase de implementación 66
4.1.1. Iteración #1. 66
4.1.2. Tareas de las Historias de Usuario implementadas en la iteración #1. 66
4.1.3. Iteración #2 87
4.1.4. Tareas de las Historias de Usuario implementadas en la iteración #2. 87
4.1.5. Iteración #3. 88
4.1.6. Tareas de las Historias de Usuario implementadas en la iteración #3. 89
4.2. Fase de Pruebas 67
4.2.1. Pruebas unitarias 68

4.2.2. Pruebas de aceptación 72
4.2.3. Resultado de las pruebas realizadas 74
4.3. Conclusiones del capítulo 75
CONCLUSIONES 76
RECOMENDACIONES 77
REFERENCIAS BIBLIOGRÁFICAS 78
GLOSARIO DE TÉRMINOS 82
ANEXOS 84

ÍNDICE DE FIGURAS

Figura 1. Escenarios de uso de Asterisk. 21
Figura 2. Respuesta de voz iterativa (fuente: Página oficial de TecnoVoz S.A) 22
Figura 3. Modelación del proceso del sistema diseñado. 39
Figura 4. Propuesta del sistema (fuente: Elaboración propia). 40
Figura 5. Proceso del Sistema de Servicio Telefónico Iterativo. 41
Figura 6. Proceso del Sistema de Gestión de Listas Telefónicas 43
Figura 6. Propuesta de la arquitectura del sistema. 48
Figura 7. Uso del patrón experto y creador en el controlador ListaController. 59
Figura 8. Uso del patrón creador en el controlador ListaController. 60
Figura 9. Uso del patrón controlador en las clases SecurityController y DefaultControler. 61
Figura 10. Modelo del diseño de la bases de datos. 62
Figura 11. Ejemplo de un SFLphone. 69
Figura 12. Resultado de las pruebas de aceptación. 75

ÍNDICE DE TABLAS

Tabla 1. Comparación entre las metodologías robustas y ágiles [16]. 27
Tabla 2. Tabla comparativa de algunos *framework* de desarrollo. 31
Tabla 3. Actores relacionados con el sistema. 44
Tabla 4. HU1: Añadir lista de destinatario. 51
Tabla 5. HU2: Gestionar destinatario 51
Tabla 6. HU3: Autenticar 52
Tabla 7. HU4: Eliminar lista de destinatario. 52
Tabla 8. HU5: Grabar mensajes de voz 53
Tabla 9. HU6: Enviar mensajes de voz 53
Tabla 10. HU7: Integración Web-IVR. 54
Tabla 11. Estimación del esfuerzo por HU. 55
Tabla 12. Plan de duración de las iteraciones. 56
Tabla 13. Plan de entregas del software 57
Tabla 14. Tiempo de las tareas abordadas en la iteración #1. 66
Tabla 15. Tarea #1 de la Historia de Usuario: Añadir listas de destinatario. 67
Tabla 16. Tarea #2 de la Historia de Usuario: Añadir listas de destinatario. 67
Tabla 17. Tarea #1 de la Historia de Usuario: Gestionar destinatario. 67
Tabla 18. Tarea #2 de la Historia de Usuario: Gestionar destinatario. 86
Tabla 19. Tarea #1 de la Historia de Usuario: Autenticar. 86
Tabla 20. Tarea #2 de la Historia de Usuario: Autenticar. 86
Tabla 21. Tarea #1 de la Historia de Usuario Eliminar listas de destinatario. 87
Tabla 22. Tarea #2 de la Historia de Usuario Eliminar listas de destinatario. 87
Tabla 23. Tiempo de las tareas abordadas en la iteración #2. 87
Tabla 24. Tarea #1 de la Historia de Usuario Grabar mensajes de voz. 88
Tabla 25. Tarea #2 de la Historia de Usuario Grabar mensajes de voz. 88
Tabla 26. Tarea #1 de la Historia de Usuario Enviar mensaje de voz. 88
Tabla 27. Tiempo de las tareas abordadas en la iteración #3. 89
Tabla 28. Tarea #1 de la Historia de Usuario Integración Web- IVR 89
Tabla 29. CP Verificar el acceso al servicio. 70
Tabla 30. CP Verificar que no se introduzcan números telefónicos iguales. 70
Tabla 31. CP Verificar el tiempo máximo de inactividad. 71
Tabla 32. Cantidad de pruebas unitarias realizadas. 72

Tabla 33. CP Añadir lista de destinatario....73
Tabla 34. CP Añadir lista de destinatario....74
Tabla 35. Gestionar destinatario....74
Tabla 36. Gestionar destinatario....89
Tabla 37. CP Autenticar. 89
Tabla 38. CP Eliminar lista de destinatarios. 90
Tabla 39. CP Eliminar lista de destinatarios. 90
Tabla 40. CP Grabar mensajes de voz. 91
Tabla 41. Tarjeta CRC: ListaController....84
Tabla 42. Tarjeta CRC: PersonController....84
Tabla 43. Tarjeta CRC: UserController....85
Tabla 44. Tarjeta CRC: SecurityController 86

INTRODUCCIÓN

Las telecomunicaciones son calificadas como las herramientas transformadoras del mundo actual, "*son el resultado de la fusión del computador con las redes de comunicaciones"… "Esta nueva tecnología tiende no solo a ofrecer mayor velocidad, capacidad y versatilidad en la transferencia de información sino también menor costo"* [1].

Actualmente están vinculadas en la vida de las personas, algo que sería inexplicable su impacto si dejara de existir, ya que brindan una mayor calidad de vida y entretenimiento. Por otro lado, permite que los empresarios controlen mejor su organización entre empleados y stakeholders[1] y sus actividades a nivel internacional [2].

Estas se basan en un conjunto de técnicas, las cuales propician la comunicación a distancia, muy utilizadas en empresas y administraciones públicas y su importancia se simplifica con el número de usuarios que la priorizan cada día. La telecomunicación incluye diversas tecnologías como la radio, televisión, teléfono y telefonía móvil, Internet, entre otras [3], entre los servicios brindados se puede encontrar la red telefónica, la cual es el sistema más completo del que dispone la humanidad ya que posee una mayor cobertura geográfica y gran número de usuarios. La misma permite la interacción entre dos usuarios, mediante una llamada en cualquier parte del planeta, de manera distribuida, automática e instantánea.

Existen dos tipos de redes, las redes públicas (ejemplo de estas redes se encuentra la red pública móvil y la red pública fija) y las redes telefónicas privadas (PBX[2]). Dichas redes poseen una serie de servicios en beneficio de la sociedad, entre ellos el conocido buzón de voz definido como: *"sistema informático que permite a los usuarios y abonados, el intercambio de mensajes de voz personales, para seleccionar y entregar información de voz, y para procesar las transacciones relacionadas con las personas, las organizaciones, los productos y servicios, utilizando un teléfono ordinario"* [4].

En años iniciales el manejo eficiente de las telecomunicaciones convencionales en los negocios, eran realizadas de forma manual, es decir, existían operadores los cuales eran los encargados de responder todas las llamadas entrantes. Estos sistemas eran llamados "centros de mensajes", los cuales presentaban desventajas ya que según los informes realizados se pudieron constatar que numerosas llamadas se presentaban en los horarios de almuerzo de los

[1] Hace referencia a los clientes externos e internos de una empresa.

[2] **P**rivate **B**ranch **E**xchange

operadores, dejando a los asistentes poco tiempo para tomar el mensaje quedando estos como perdidos, garabateados e imprecisos. En muchos de los casos los nombres de los remitentes estaban mal escritos y los números telefónicos eran equivocados.

Actualmente este sistema utiliza el almacenamiento digital y normalmente se almacena en la reserva de datos informáticos. Gracias a la innovación, se han integrado los buzones con los centros de llamadas, y como resultado se obtuvo un sistema avanzado auxiliado por las Respuestas de Voz Iterativas (Interactive Voice Response, IVR), los cuales son sistemas telefónicos que posibilitan la iteración de las personas con el teléfono mediante grabaciones de voz que sirven de guía para la correcta utilización del sistema. Estos usan la información digital almacenada en bases de datos corporativas para seleccionar palabras pregrabadas y frases almacenadas en un vocabulario de correo de voz y para formar oraciones que se entregan a la persona que llama [3], generalmente los IVR son puestos en funcionamiento para grandes empresas que reciben constantemente numerosas llamadas, con el fin de reducir costos y el tiempo de espera de sus clientes.

El buzón de voz en conjunto con la operadora automática, significan una revolución que dio solución a los problemas que se presentaban en los centros de mensajes. Con el desarrollo de la telefonía unificada se han vuelto a abrir oportunidades para los operadores, ya que cuentan con servicios compartidos para todas las formas de convergencia de las telecomunicaciones IP y los servicios de correo de voz. En cuanto a las empresas, se ha incrementado el flujo de las comunicaciones y se ahorra una enorme cantidad de dinero [4].

Cuba no está totalmente aislada de estos servicios sino que poco a poco venido dando pasos de avance en este sentido, ejemplo de esto se evidencia en el desarrollo del call-center o central telefónica PLATEL, la cual a través de la plataforma *Asterisk* [3] como su núcleo ha propiciado que en el país se brinde muchos servicios telemáticos para garantizar el rendimiento, supervisión y administración de los distintos Centros de Llamada ubicados en varias zonas de la nación.

"… La Empresa de Telecomunicaciones de Cuba, SA (ETECSA) es la entidad encargada de la prestación de los servicios públicos de telecomunicaciones mediante la operación, instalación, explotación, comercialización y mantenimiento de sus redes públicas en todo el territorio cubano…" [7]. Sin embargo, el buzón de voz no presenta mucha difusión entre las personas, ya que es utilizado en empresas y en pocas zonas de la isla, donde no se ha llegado a la máxima

[3] Central telefónica que actualmente es utilizada para la creación de IVR y otros servicios de comunicación.

explotación de los mismos y no se cuenta con un sistema eficiente, que permita el envío masivo de mensajes de voz, a través de los teléfonos fijos.

Los mensajes en el correo de voz de la telefonía fija únicamente se pueden “dejar” a un solo destinatario (propietario del buzón de voz), por parte del que realiza la llamada. El servicio podría ampliarse, brindando la posibilidad de enviar un mensaje a varias personas o destinatarios, lo cuál sería muy útil en la telefonía fija (por ejemplo en el área empresarial, o en el dominio de las organizaciones). Este servicio sería el equivalente a la mensajería múltiple en la telefonía móvil, donde se pueden distribuir mensajes de texto cortos (SMS) a varios destinatarios.

Por lo anteriormente expresado surge el siguiente **problema de investigación**: ¿Cómo facilitar el envío de mensajes de voz a múltiples destinatarios de forma simultánea, mediante la telefonía fija?

Partiendo de este problema se tiene como **objetivo general**: Desarrollar un sistema informático que permita el envío de mensajes de voz a múltiples destinatarios de forma simultánea de la telefonía fija.

El trabajo enfoca su **objeto de estudio** en los procesos de desarrollo de servicios para la telefonía fija y el **campo de acción** estaría enfocado en los procesos de desarrollo de servicios de mensajería de voz en la telefonía fija.

Se plantean además como **objetivos específicos**:

- Analizar el estado del arte de los sistemas de mensajerías masivas.
- Construir una aplicación IVR capaz de gestionar listas de destinatarios telefónicos.
- Construir una aplicación *web* capaz de gestionar visualmente las listas de destinatarios telefónicos.

Para dar cumplimiento a los objetivos específicos se trazaron las siguientes **tareas**:

- Análisis del estado del arte.
    - Revisión bibliográfica
    - Análisis de la existencia de soluciones similares que contribuyan al desarrollo del software.
- Elaboración del diseño teórico de la investigación.
- Selección de las herramientas, tecnologías y metodología a emplear.

    - Revisión y análisis de productos realizados en la UCI cuyo cliente sea ETECSA, para identificar la arquitectura a utilizar en el proceso de desarrollo.
- Análisis de los requisitos funcionales y no funcionales.
- Elaboración de la propuesta de solución.
- Realización de pruebas para detectar anomalías y realizar validaciones.

Este trabajo de diploma posee como **idea a defender:** Con el desarrollo de la mensajería múltiple en el correo de voz de la telefonía fija, se garantiza un sistema que permita ampliar los servicios de mensajería de voz de la telefónica fija en Cuba para cualquier empresa y para personas naturales.

Con la presente investigación se obtendrá como **posible resultado** un sistema informático que implemente el servicio de correo de voz múltiple para la telefonía fija.

Para la correcta elaboración del presente trabajo se harán uso de los siguientes **métodos científicos** de investigación trazando la meta de alcanzar una mayor profundidad y organización en el mismo:
A continuación se muestran los **métodos teóricos** utilizados:

- **Histórico-Lógico:** Permite estudiar la evolución y trayectoria de los sistemas de mensajería múltiples para correos de voz en la telefonía fija, su funcionamiento y condicionamiento en los diferentes períodos de la historia basado en el estudio histórico, poniendo de manifiesto la lógica interna de desarrollo, de su teoría y esencia.
- **Analítico-Sintético:** Permite de manera profunda la recopilación y estudio de la información de los sistemas de mensajería múltiples para correos de voz en la telefonía fija mediante el análisis de teorías, tendencias, documentos y *web* relacionados con el tema, para expresar de manera resumida la posición del investigador
- **Modelación:** Permite el desarrollo de artefactos como diagramas, figuras, etc. con el objetivo de crear abstracciones para explicar la realidad.

A continuación se muestran los **métodos** empíricos utilizados:

- **Observación:** Permite obtener información acerca de los sistemas de mensajería de voz, mediante varias visitas al personal encargado de los servicios, al cliente y a las personas que intervienen directa o indirectamente, con el objetivo de recopilar la

información necesaria de los requisitos funcionales del sistema a desarrollar. Además se realizaron diferentes pruebas con el propósito de determinar la fiabilidad y el mejoramiento de la usabilidad del sistema, a través de la detección de errores.

El documento se encuentra estructurado de la siguiente forma:

- **Capítulo 1.** Fundamentación teórica: En este capítulo se abordaran los aspectos teóricos y técnicos de la investigación, incluye un estudio del estado de arte del tema tratado. Se realizará un estudio de las aplicaciones que permiten los servicios de mensajería múltiples en correo de voz en países extranjeros. Propone un análisis de las metodologías, tecnologías y herramientas a utilizar durante el desarrollo de la solución.
- **Capítulo 2.** Características del sistema, Exploración y Planificación: En este capítulo definirán los requerimientos necesarios para el desarrollo de la aplicación, así como los artefactos generados durante las fases de Exploración y Planificación del proyecto.
- **Capítulo 3.** Diseño del sistema: En el transcurso de este capítulo se evidenciarán las principales características y artefactos del flujo de diseño que propone la metodología seleccionada para el desarrollo.
- **Capítulo 4.** Implementación y prueba: Se detalla en este capítulo todo lo relacionado con los procesos de implementación y realización de pruebas de software.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

Introducción

En la actualidad el tema del envío de mensajes masivos integrado a empresas que lo utilizan, se ha propagado por todo el mundo, propiciando el desarrollo de los buzones de voz y las contestadoras automáticas. En este capítulo se brinda un panorama general acerca de los aspectos teóricos y técnicos a utilizar durante el desarrollo del sistema de mensajería, se realiza un estudio de los principales conceptos y definiciones para una mejor comprensión y desarrollo de la investigación, así como de las aplicaciones que permiten enviar mensajes múltiples por medio de los correos de voz en el mundo. Se describen algunos de los sistemas existentes utilizados para almacenar y gestionar mensajes de voz, además se refleja la situación cubana respecto a esto y los beneficios que traería consigo la alternativa de una aplicación implementada en software libre. Se hace énfasis en las metodologías, herramientas y tecnologías a utilizar para el desarrollo de la aplicación, de los mismos se tienen sus características más importantes y de qué forma serán utilizados durante el desarrollo de la solución.

1.1. Marco Conceptual

1.1.1. Telefonía fija

Con solo mencionar el término ¨telefonía fija¨, se suele pensar que se está hablando de los teléfonos fijos que regularmente son vistos en hogares o en centros de trabajo. Diversas teorías han clasificado la telefonía fija o convencional como aquella que hace referencia a las líneas y equipos, que se encargan de la comunicación entre terminales telefónicos no portables y generalmente enlazados entre ellos o con la central por medio de conductores metálicos [4], según los realizadores del sitio oficial de *Wikitel*, el servicio de telefonía fija realiza el transporte de voz en tiempo real entre dos terminales, estando ambos terminales, o al menos el terminal de origen (que realiza la llamada), conectados a una red conmutada de telecomunicaciones en una ubicación fija. Dicha red de telecomunicaciones es la red telefónica conmutada [7].

Después de vistas estas definiciones, se concluye que la telefonía fija, es el sistema encargado de enviar señales de voz, a través de medios telefónicos fijos, que se encuentran conectados entre sí y a gran distancia, por medio de un conmutador. Esta significó el primer paso que dio el hombre para poder comunicarse sin necesidad de realizarlo personalmente.

1.1.2. *Asterisk*

Con el auge de las nuevas tecnologías y el surgimiento de la telefonía IP, *Asterisk* es la PBX digital que está siendo usada por la mayoría de las empresas en el mundo, utiliza el concepto de software libre (GPL) y es promovida por la empresa *Digium*, la cual invierte en aspectos como el desenvolvimiento de código fuente y en *hardware* de telefonía de bajo costo que funciona con *Asterisk* [5]. Esta PBX corre en plataforma Linux y otras plataformas *Unix* con o sin *hardware* conectado a la red pública de telefonía [35].

Asterisk provee servicios de *voicemail* con directorios, conferencias, respuesta de voz interactiva (IVR), llamadas en espera, entre otros. Además, no necesita ningún *hardware* adicional para el VoIP y permite conectar oficinas, dar a todos los empleados casillas de voz integradas con *Internet* o red IP privada y construir aplicaciones interactivas de voz, que conecten el sistema ordinario o alguna otra aplicación en casa [36].

Asterisk es capaz de trabajar con prácticamente todos los estándares de telefonía tradicional:

- Líneas analógicas
- Líneas digitales

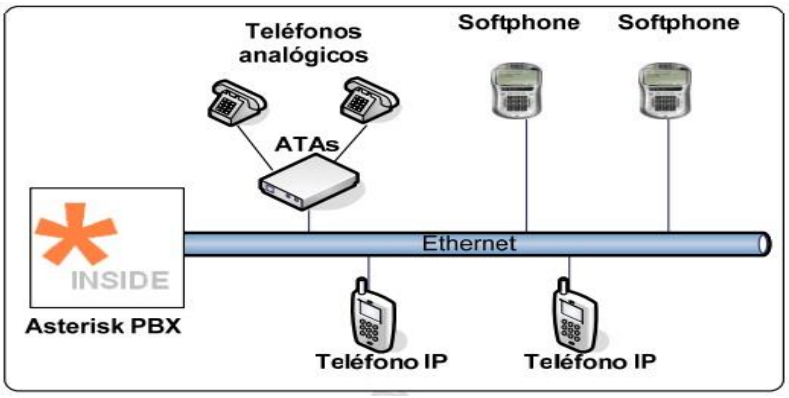


Figura 1.Escenarios de uso de Asterisk.

Para realizar configuraciones en *Asterisk* se agregan funcionalidades en el dialplan o plan de marcado, el cual es el área de configuración más importante y es el responsable por el funcionamiento de la conmutación de las llamadas. En este se instalan todas las acciones y situaciones que el PBX debe manejar, además se configuran contextos que funciones solo en una parte del día o la noche [35]. Cada llamada recorre una secuencia en la cual se ejecutan acciones (aplicaciones), se llaman macros o se salta a otros bloques de ejecución [37]. Para el desarrollo de la solución se empleó la versión 1.8 de *Asterisk*.

1.1.3. Mensaje de voz

Luego de una investigación recogida y simplificación de los datos más importante se define el mensaje de voz como un mensaje electrónico con un contenido primario de audio digitalizado [8], el *WordReference*[4] lo define, como un recado de palabra que una persona envía a otra [6], por otro lado, se plantea que "*un mensaje de voz en inglés llamado voicemail, es un mensaje que ha sido dejado de forma oral. Cualquier mensaje grabado es un mensaje de voz. Hoy en día se le llama mensaje de voz al mensaje que dejan en el celular, pero igualmente un mensaje de voz es aquel que dejan en un teléfono fijo (de casa) en la contestadora (sin importar que el medio de transmisión o recepción sea digital o análogo)"* [10].

1.1.4. *Interactive Voice Response* (IVR)

El IVR consiste en un sistema telefónico que es capaz de recibir una llamada e interactuar con el humano a través de grabaciones de voz y el reconocimiento de respuestas simples, como "sí", "no" u otras. Es un sistema automatizado de respuesta interactiva, orientado a entregar y/o capturar información a través del teléfono, permitiendo el acceso a servicios de información u otras operaciones [11] (véase la figura 2).

Figura 2.Respuesta de voz iterativa (fuente: Página oficial de TecnoVoz S.A)

Tecnologías utilizadas por un IVR

El IVR para brindar mejores servicios involucra otras tecnologías como:

- ***Dual Tone Multi Frequency* (DTMF):** Sistema Multifrecuencial para la Marcación de Tonos. Es la tecnología de tonos utilizada para el marcado, este sistema supera al de marcación por pulsos por cuanto disminuye la posibilidad de errores de marcación, al no depender de un dispositivo mecánico. Por otra parte, es mucho más rápido ya que

[4] Diccionario Ingles Español: 175.000 palabras con 240.000 significados, disponible en : *www.wordreference.com/es/*

no hay que esperar tanto tiempo para que la central detecte las interrupciones, según el número marcado [12].

- ***Text To Speech* o conversión de texto a voz (TTS):** iniciada en la informática, da la capacidad de transformar texto a audio que escucha el operador. Es la generación de redes inalámbricas por medios automáticos de una voz artificial que genera el sonido producido por una persona al leer un texto cualquiera en voz alta o una voz artificial. Es decir, son sistemas que permiten la conversión de textos en voz sintética [13].
- **Reconocimiento de Voz (ASR):** iniciada por la informática. Permite reconocer las palabras del usuario y aceptarlas como órdenes [10].

1.2. Estudio de soluciones similares

Para el desarrollo de esta aplicación se hizo necesario realizar una investigación de los sistemas similares que permitan la mensajería múltiple en correo de voz para la telefonía fija. Un ejemplo de estos sistemas son las aplicaciones que a continuación se mencionan:

1.2.1. *Phone Broadcast*

El software *Phone Broadcast* de la empresa *CallFire*, es una solución automatizada basada en la *web*, que permite realizar llamadas de voz automáticas, con el fin de dejar mensajes de voz pre-grabados en el buzón de los destinatarios. Algunos de los usos que puede tener este software son:

- Comunicar con una lista de contactos mediante el servicio de transmisión de voz.
- Realizar promociones periódicas.
- Enviar alertas de emergencia a través de mensajes telefónicos automáticos.
- Enviar información empresarial o alertas a los empleados a través del servicio de trasmisión de voz [51].

1.2.2. *Ifbyphone*

Ifbyphone 2.5 (conocido como "La voz de la *web* ") es claramente único en el mercado, ya que proporciona una solución completa para la integración de la *web* y los teléfonos de una Empresa Mediana o Pequeña (SMB por sus siglas en inglés). Los servicios de la versión 2.5 integra los servicio de "clic para llamada" en la *web* y de emisión de voz en una potente herramienta para las SMB. Estas aplicaciones facilitan conversaciones telefónicas entre un negocio y sus clientes; lo que aumenta las ventas y reduce los gastos en el servicio de envío. Esta versión de la plataforma incluye:

- "Clic para llamar" con diálogos personalizados
- Encuéntrame
- Correo de Voz Virtual
- Recepcionista Virtual
- Configuración de diálogos de voz vía *web*.
- Integración con sistemas de bases de datos
- Una API completa para la gestión de llamadas

Todas las soluciones basadas en *Ifbyphone* 2.5 funcionan con cualquier sistema telefónico y son totalmente portátiles; si una empresa pasa de proveedor de transporte telefónico, esto no supone un gran problema [52].

1.2.3. *DialMyCalls*

DialMyCalls es una solución *web* que permite el envío masivo de mensajes de texto y voz. El usuario puede administar las listas de destinatarios previamente o en el momento de enviar el mensaje. El flujo para su uso es el siguiente:

- Grabar un mensaje, que será reproducido en la llamada. Si ya se ha usado el servicio previamente, se puede escoger uno de los mensajes grabados.
- Crear una lisa con los destinatarios a los que se les enviará el mensaje o escoger una de las listas previamente creadas.
- Para finalizar, se establecen algunas opciones, tales como mostrar el "*caller* ID" y el momento hacer la llamada, y eso es todo, su mensaje será entregado a los buzones de los destinatarios.

DialMyCalls hace uso de *AccurateAMD*™, que es una tecnología privada para la detección de buzón de voz y contestadoras. Esta se asegura de dejar el mensaje en el buzón de voz si la llamada no es respondida. Además, genera un reporte de qué ocurrió con cada una de las llamadas justo después de que se lleven a cabo [53].

1.2.4. Conclusiones acerca de los sistemas similares en el mundo.

Los sistemas analizados anteriormente presentan las características necesarias para dar solución al problema de investigación, permiten el envío de mensajes masivos mediante listas de distribución definidas por el usuario, empleando la telefonía IP y analógica. Sin embargo, presentan como dificultad para su uso el alto costo de adquisición y soporte, debido a que estos son sistemas informáticos privados, por lo cual no se pueden modificar funcionalidades ni

agregar nuevas al no poder acceder al código, lo que trae consigo que sus características no se puedan adecuar a una empresa.

1.3. Situación en Cuba

Alcatel, Ericson y Mitel son varias de las plantas telefónicas privadas con que cuenta actualmente la empresa de telecomunicaciones del país, las cuales brindan el servicio IVR. Estas son adquiridas a un precio superior de su costo neto en el mundo, Cuba debe comprar estos servicios a terceros países como Francia e Israel porque al ser privados traen como consecuencia que se denieguen todos los privilegios de configuración, lo que propicia que no se resuelvan todos los problemas estatales y sociales, al ser la innovación un derecho exclusivo del fabricante [14]. Sin embargo Cuba está ensayando nuevas tecnologías para la implementación de la telefonía IP y la televisión digital con la finalidad de servir de soporte para profundizar la informatización de la sociedad y la introducción masiva de la computación con fines sociales [54].

En la UCI la telefonía IP no se implementará para sustituir a la telefonía tradicional brindada por ETECSA, sino que será un servicio complementario al existente. El servicio en estos momentos ya está disponible para determinados lugares de la universidad, que luego se irá expandiendo. Cuentan con el servicio, los Centros de Desarrollo, los Decanos y el Centro de Soporte, entre otros directivos [55]. *Asterisk* es uno de los software libres más importantes utilizados por esta universidad capaz de brindar comunicación a través de email, mensajería instantánea, IVR, etc., además de brindar la posibilidad de desarrollar servicios para empresas como ETECSA. En años anteriores fue utilizado por centros como Telemática y el Xetid, brindando soluciones de interés expuestas en las distintas ferias informáticas.

Actualmente no existe una infraestructura telefónica que sostenga y provea al país de estos servicios, se carece de sistemas para la transferencia de mensajes de voz telefónicos a listas de destinatarios utilizando la telefonía fija, medio que es muy utilizado por todas las empresas de la isla, además se presenta un bajo desarrollo en comparación con la comunicación convencional del mundo.

Hay que destacar el problema ético de que en el momento en que se está contratando servicios para la creación de un sistema, dicho sistema debería de ser libre y basado en herramientas libres, que respete la libertad de los usuarios sobre su producto y por lo tanto una vez adquirido pueda ser usado, copiado, estudiado, cambiado y retribuido libremente, con lo cual se puede ayudar al beneficio de otros, ya que se pagó por él y se tiene derecho a tener

acceso a esta información. Tanto Cuba como otros países latinoamericanos se han trazado la meta de lograr una independencia tecnológica, donde no existan las llamadas ¨puertas traseras¨ ni el pago elevado de licencias, un sistema propio, seguro y con visión sostenible que pueda ser utilizado por las nuevas generaciones. En consecuencia se requiere el empleo en los desarrollos en el país de las aplicaciones de telecomunicaciones de las plataformas de código abierto.

1.4. Ambiente de desarrollo

Tras haber analizado el estado del tema a nivel internacional y la situación en Cuba, se llega a la conclusión de que los sistemas automatizados presentan como principal característica su desarrollo sobre plataformas propietarias, debido a esto se realizó un estudio y búsqueda de información acerca de las tendencias y tecnologías que se usarán para el desarrollo del presente trabajo, comparando y eligiendo las más eficientes en este caso.

1.4.1. Metodologías de desarrollo de software

Las metodologías de desarrollo de software surgen ante la necesidad de utilizar una serie de procedimientos, técnicas, herramientas y soporte documental a la hora de desarrollar un producto (*software*). Según Lourdes García en su trabajo sobre las metodologías plantea: [13] "*las metodologías guían el proceso de desarrollo y la experiencia ha demostrado que la clave del éxito de un proyecto de software es la elección correcta de la metodología, que puede conducir al programador a desarrollar un buen sistema de software. La elección de la metodología adecuada es más importante que utilizar las mejores y más potentes herramientas*". Se clasifican en dos tipos: robustas y ágil, las más utilizadas son: La Programación Extrema (XP), Scrum y el Proceso Unificado de Software (RUP) [15].

1.4.1.1. Metodologías robustas (tradicionales)

Las metodologías denominadas tradicionales hacen referencia al conjunto de prácticas que se aplican con cierto éxito desde hace muchos años y en las cuales se encuentra la tendencia a ocuparse y centrar esfuerzos en la documentación, las prácticas bien realizadas, los avances o progresos prefijados. Estas metodologías intentan reducir el riesgo mediante una fuente de colección de requisitos y una planificación detallada para dejar lugar a los imprevistos. Ejemplo de este tipo de metodología se encuentran: RUP, Marco de Trabajo de Solución de *Microsoft* (MSF), Modelo en Espiral *Win-Win* e *Iconix* [16].

1.4.1.2. Metodologías ágiles

Las metodologías ágiles representan un nuevo enfoque en el desarrollo de software, mejor aceptado por los desarrolladores de proyectos que las metodologías tradicionales. Esto se debe a la simplicidad de sus reglas y prácticas, su orientación a equipos de desarrollo de pequeño tamaño, su flexibilidad ante los cambios y su filosofía colaborativa. Ejemplos de esta metodología son: XP, SCRUM, *Crystal Clear* (Cristal Transparente), Desarrollo de Software Adaptativo (ASD), *XBreed*, SXP[5] [16].

A continuación se muestra un cuadro comparativo sobre estos dos tipos de metodologías:

Tabla 1. Comparación entre las metodologías robustas y ágiles [16].

Metodologías robustas	Metodologías ágiles
Más artefactos	Pocos artefactos
Más roles	Pocos roles
Existe un contrato prefijado	No existe un contrato tradicional o al menos es bastante flexible
El cliente interactúa con el equipo de desarrollo mediante reuniones	El cliente es parte del equipo de desarrollo
Grupos grandes	Grupos pequeños y trabajando en el mismo sitio
La arquitectura es esencial	Menos énfasis en la arquitectura

Luego del estudio y comparación de las metodologías se decide seleccionar para el desarrollo del software una metodología ágil debido a lo antes planteado, además estas presentan una curva de aprendizaje rápida posibilitando la disminución de los costos y brindando flexibilidad a los proyectos de software donde la incertidumbre esté presente. A continuación se describen algunas de las más importantes.

1.4.1.3. *Extreme Programming* (XP)

Extreme Programming es una metodología ágil de desarrollo de software basada en la simplicidad, la comunicación y la retroalimentación. Es la más destacada de los procesos ágiles de desarrollo de software formulada por Kent Beck. La misma se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad. Es la más adaptable a equipos pequeños y de corto plazo, además de generar

[5] ***S**equenced **P**acket **E**xchange*, **I**ntercambio de **P**aquetes **S**ecuenciados

poca documentación y artefactos, mientras que las robustas como RUP, no están preparadas para el cambio, por lo cual, no ofrecen una buena solución cuando el proyecto se realiza en un entorno volátil [16].

1.4.1.4. Scrum

Scrum es un proceso ágil y liviano que sirve para administrar y controlar el desarrollo de *software* que se realiza en forma iterativa e incremental (una iteración es un ciclo corto de construcción repetitivo). Cada ciclo o iteración termina con una pieza de *software* ejecutable que incorpora nueva funcionalidad, estas tienen una duración entre 2 y 4 semanas. Scrum se utiliza como marco para otras prácticas de ingeniería de *software* como RUP o Extreme Programming, ademas de presentar el inconveniente de no adaptarse a un plan, no ser acta para todos los proyectos y en muchas ocasiones es necesario completarlo con otros procesos de XP [17], el equipo es autodirigido, no hay rol de manager o en este caso jefe de proyecto, solo existe un rol llamado Scrum k para resolver problemas, no para manifestar que se debe cumplir con algo [18].

1.4.1.5. Selección de la metodología

Finalizada la investigación, se seleccionó la metodología ágil XP para la implementación del *software*, por ser más adaptable a equipos pequeños y de corto plazo. El equipo de desarrollo del sistema propuesto está integrado por una persona, por lo cual, tiende a sufrir con mayor peso los cambios realizados al existir incertidumbres, el proyecto está caracterizado por ser pequeño, sin mucha complejidad a la hora de implementar y diseñar las interface o artefactos necesarios para la solución. Existe una retroalimentación entre el cliente y el equipo de desarrollo, al ser este parte del mismo, existiendo a su vez una comunicación fluida entre todos los participantes.

Para el desarrollo de la solución se tuvo en cuenta las siguientes fases de la metodología:

Fase de exploración

- Se realizaron las actualizaciones regulares.
- Se definieron los requisitos funcionales en las historias de usuario.

Fase de planificación

- Se realizó una estimación del esfuerzo de las historias de usuario.
- Se priorizó con su debido orden, las historias de usuarios según el esfuerzo de realización.
- Se realizó el plan de entregas.

- Se realizó el plan iteraciones.

Fase de análisis y diseño

- Se tuvo en cuenta la simplicidad en las soluciones implementadas.
- Se realizó un análisis del diseño de la solución.
- Revisión continúa.

Fase de desarrollo del código

- Disponibilidad del cliente.
- Programación dirigida por las pruebas.
- Propiedad colectiva del código.
- Ritmo sostenido.

Fase de pruebas

- Pruebas unitarias.
- Detección y corrección de errores.
- Pruebas de aceptación.

En vista del análisis de esta metodología, se evidencia que presenta varios roles, de estos solo van a ser utilizados los siguientes, debido a la cantidad de integrantes con que cuenta el equipo de desarrollo y a la necesidad real de cada uno de ellos:

- **Programador:** Es el encargado de definir las pruebas y mantener el código del programa tan simple como le sea posible, es uno de los roles más importantes.
- **Cliente:** Es el encargado de definir los requisitos y decide cuándo deben ser logrados, además de las pruebas funcionales.
- **Verificador:** Es el encargado de realizar las pruebas y comunicar al programador los resultados, permitiendo que la culminación de la solución quede con la mayor calidad posible.
- **Consultor técnico:** Es una persona externa al proyecto, con conocimientos técnicos, la cual sirve como guía para resolver problemas específicos.

1.4.2. *Framework* de desarrollo en PHP

Un *Framework* es un conjunto de herramientas, librerías, convenciones y buenas prácticas que pretenden encapsular las tareas repetitivas en módulos genéricos fácilmente reutilizables [19]. Su objetivo es proporcionar una estructura común, de modo que los desarrolladores no tienen que hacer el código desde cero y pueden volver a utilizar la gran mayoría [20].

1.4.2.1. *Framework* CodeIgniter

CodeIgniter es un *framework* para la creación rápida de aplicaciones web, es un producto de código libre de uso para cualquier aplicación. Contiene una serie de librerías que sirven para el desarrollo de aplicaciones web y marca una manera específica de codificar las páginas y clasificar sus diferentes scripts, que sirve para que el código esté organizado y sea más fácil de crear y mantener. CodeIgniter implementa el proceso de desarrollo llamado Modelo Vista Controlador (MVC), que es un estándar de programación de aplicaciones, utilizado tanto para hacer sitios web como programas tradicionales. Este *framework* es capaz de trabajar en la mayoría de los entornos o servidores, incluso en sistemas de alojamiento compartido, donde se tiene un acceso por FTP para enviar los archivos al servidor, además no se permite el acceso a su configuración [49].

1.4.2.2. *Framework* Symfony

Symfony es un *framework* creado con PHP 5 para desarrollar aplicaciones web. Añade una nueva capa por encima de PHP y proporciona herramientas que simplifican el desarrollo de las aplicaciones, separa la lógica de negocio, la lógica de servidor y la presentación. Es de tipo *backend developer* ya que permite conectar la base de datos con el contenido del sitio.

Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. Symfony es compatible con la mayoría de gestores de bases de datos, como MySQL, PostgreSQL, Oracle y SQL Server de *Microsoft*. Proporciona acceso a librerias como Doctrine, además a plantillas, seguridad, formularios, validación y traducción, permite que las URL sean flexibles gracias al componente Routing (enrutamiento). Se puede utilizar tanto en plataformas Linux como en Windows, puede ser completamente personalizado para cumplir con los requisitos de las empresas que disponen de sus propias políticas y reglas para la gestión de proyectos y la programación de aplicaciones.

Symfony está publicado bajo licencia de *software* libre con el objetivo de donar su trabajo a la comunidad y aprovechar la respuesta de los usuarios, asegurando el mantenimiento y las ampliaciones futuras de la aplicación, está recomendado para todo aquel que desee desarrollar aplicaciones de forma muy rápida además de ser maduro, bien documentado y con una gran comunidad que lo apoya [21].

1.4.2.3. *Framework* Yii

Yii es un *framework* PHP basado en componentes de alta *performance* para desarrollar aplicaciones web de gran escala. El mismo permite la máxima reutilización en la programación

web y puede acelerar el proceso de desarrollo. Es liviano y está equipado con soluciones de cacheo sofisticadas, por lo cual es adecuado para desarrollar aplicaciones de gran tráfico como portales, foros, sistemas de administración de contenidos (CMS), Sistemas de comercio electrónico (*e-commerce*), etc. Está liberado bajo licencia BSD[6] (Cláusula 3 de la licencia), esto significa que es posible utilizarlo de forma gratuita para el desarrollo de cualquier aplicación web de código abierto. Hace uso del estilo arquitectónico Modelo Vista Controlador [50].

1.4.2.4. Selección del *framework*

Luego de realizado el estudio de algunos de los *framework* más importantes, se decidió utilizar Symfony en su versión 2.3.7**,** debido a que permite enfocarse directamente al proyecto minimizando la cantidad de código y agilizando el proceso de desarrollo. Dentro de las características principales que ayudaron a su selección, se destaca la publicación bajo licencia libre, además de poseer herramientas rápidas y una extensa comunidad de usuarios, posibilitando la comunicación de estos con los desarrolladores del sistema. Otro de los motivos de la selección es la familiarización por parte del equipo de desarrollo con dicho *framework,* además de poseer una extensa documentación de fácil acceso. A continuación se muestra una tabla comparativa para mayor entendimiento.

Tabla 2. Tabla comparativa de algunos *framework* de desarrollo.

Framework	Documentación	ORM	Código Abierto	Generador de Vistas	Experiencia del equipo
CodeIgniter	x	x	x		
Symfony	x	x	x	x	x
Yii	x	x	x		

1.4.3. Herramienta CASE para el modelado

Herramientas CASE (*Computer Aided Software Engineering*, Ingeniería de S*oftware* Asistida por Computadoras). Son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de *software* reduciendo el costo de las mismas en términos de tiempo y de dinero. Estas herramientas pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del *software* en tareas como el diseño de proyectos, cálculo de costos,

[6] **B**erkeley **S**oftware **D**istribución**:** licencia que permite utilizar el marco de trabajo de forma gratuita para desarrollar cualquier aplicación web de código abierto o software privativo.

implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores entre otras. También pueden mejorar la productividad en el desarrollo de una aplicación de bases de datos. Y por productividad se entiende tanto la eficiencia en el desarrollo, como la efectividad del sistema desarrollado.

La eficiencia se refiere al costo, tanto en tiempo como en dinero, de desarrollar la aplicación y la efectividad se refiere al grado en que el sistema satisface las necesidades de los usuarios. Para obtener una buena productividad, subir el nivel de efectividad puede ser más importante que aumentar la eficiencia [22].

1.4.3.1 *Visual Paradigm* 8.0

Visual Paradigm 8.0 es una potente multiplataforma y sin embargo, el modelado más fácil para usar UML visual. Facilita excelente interoperabilidad con otras herramientas CASE y la mayoría de los principales IDEs que destaca todo el proceso de desarrollo [23].

Es una herramienta diseñada para usuarios como Ingenieros de S*oftware*, Analistas de Sistemas, Arquitectos de Sistemas y otros que estén interesados en el diseño de *software* orientado a objetos. Con VP-UML se pueden crear los diferentes diagramas de UML con solo realizar la operación de arrastrar y soltar [32].

Proporciona al usuario una gran ayuda para el desarrollo de aplicaciones informáticas, desde la planificación, el análisis y diseño y la generación del código fuente de los programas. Se utiliza además de todas las características planteadas anteriormente, por ser muy fácil de usar y con un ambiente gráfico agradable para el usuario, además permite transformar diagramas de Entidad-Relación a tablas de base de datos, posibilitando la creación y diseño del modelo de datos y realizar el flujo de procesos del sistema. Existe una alternativa libre y gratuita de este *software*, la versión *Visual Paradigm* UML 6.4 *Community Edition* (ya que existe la *Enterprise, Professional*, etc.). Fue diseñado para una amplia gama de usuarios interesados en la construcción de sistemas de *software* de forma fiable a través de la utilización de un enfoque orientado a objetos [24].

1.4.4. Lenguaje de modelado

En el proceso de desarrollo de *software*, el modelado es de vital importancia y tiene una notable repercusión en la etapa de diseño e implementación, por proveer al ingeniero de un conjunto de notaciones, herramientas y prácticas, que le permiten "visualizar" el sistema a construir, logrando un nivel de abstracción que organice la lógica del mismo.

1.4.4.1. UML

UML son las siglas de *Unified Modeling Language* (Lenguaje Unificado de Construcción de Modelos), notación (esquemática en su mayor parte) con que se construyen sistemas por medio de conceptos orientados a objetos. Se define como: "lenguaje que permite especificar, visualizar y construir los artefactos de los sistemas de *software*...".El lenguaje UML estandariza los artefactos y la notación, pero no define un proceso oficial de desarrollo. Aumenta las probabilidades de una aceptación generalizada de la notación estándar del modelado sin la obligación de adoptar un proceso oficial. La esencia de un proceso apropiado admite mucha variación y depende de las habilidades del personal, de la razón investigación-desarrollo, de la naturaleza del problema, de las herramientas y de muchos otros factores [25].

Es fundamental aclarar que UML no es un lenguaje de programación, sino un lenguaje de modelado de propósito general, que ha demostrado su efectividad sobre todo en el área del análisis y diseño de sistemas de cómputo. [26].

1.4.5. Lenguajes de programación

Los lenguajes de programación han ido evolucionando según las necesidades de las plataformas. Expresan procesos los cuales son utilizados para crear aplicaciones que permitan controlar el procesamiento de la máquina, facilitando a su vez el trabajo de los desarrolladores. Se clasifican en lenguajes del lado cliente y lenguajes del lado servidor.

1.4.5.1. Lenguaje del lado del cliente

Son aquellos lenguajes que son asimilados directamente por el navegador y no necesitan pre tratamiento.

1.4.5.1.1. HTML 5

HTML, acrónimo inglés de *HyperText Markup Language* (lenguaje de marcas hipertextuales), lenguaje de marcación diseñado para estructurar textos y presentarlos en forma de hipertexto, que es el formato estándar de las páginas web. HTML5 es una colección de estándares para el diseño y desarrollo de páginas web. Esta colección muestra la forma en que se presenta la información y la manera de interactuar con ella. Permite una mayor interacción entre páginas web y contenido media (video, audio, entre otros) así como una mayor facilidad a la hora de codificar el diseño básico. Esta nueva versión se basó en el diseño más común de las páginas web alrededor del mundo, para llegar a un estándar de etiquetas que realicen las mismas tareas

de manera más rápida y eficiente, además ocupa menos recursos en la computadora del cliente y es compatible con las versiones anteriores de HTML [27].

1.4.5.1.2. JavaScript 9.7

JavaScript es un lenguaje "orientado a objetos", para el desarrollo de aplicaciones cliente-servidor dentro de *Internet*. Su uso está más difundido en el ámbito del cliente y de las páginas web, aunque se ha extendido a aplicaciones de servidor. Dentro de sus principales aplicaciones están:

- La validación de formularios dentro de una página.
- La personalización de la página por el usuario, que le permitirá tener una página web a su medida.
- La inclusión de datos del propio sistema, como son la hora y la fecha.
- El dar respuesta a eventos locales dentro de la página, como apretar un botón.
- La realización de cálculos en tiempo real [29].

Al igual que Java, es una de las múltiples maneras que han surgido para extender las capacidades del lenguaje HTML. Al ser la más sencilla, es por el momento la más extendida. Es un lenguaje script u orientado a documento, como pueden ser los lenguajes de macros que tienen muchos procesadores de texto [28].

1.4.5.1.3. Hojas de Estilo en Cascada

Las Hojas de estilo en Cascada, en inglés *Cascading Style Sheets* (CSS), fueron diseñadas y desarrolladas por la *World Wide Web Consortium* (W3C). Una hoja de estilos CSS es el tipo de documento que utiliza un navegador web para redefinir las propiedades de los distintos elementos y las etiquetas en el código HTML [19]. CSS3 ha venido desarrollándose desde 1999. Esta nueva especificación viene con interesantes novedades que permitirán hacer webs más elaboradas y dinámicas, con mayor separación entre estilos y contenidos. Dará soporte a muchas necesidades de las webs actuales, sin tener que recurrir a trucos de diseñadores o lenguajes de programación [30].

1.4.5.2. Lenguajes del lado servidor

Los lenguajes del lado del servidor son enviados al cliente en un formato claro para un mejor entendimiento, además de que se ejecutan en el mismo servidor.

1.4.5.2.1. PHP 5.3.13

PHP, acrónimo de "*PHP: Hypertext Preprocessor*", es un lenguaje de 'scripting' de propósito general y de código abierto que está especialmente pensado para el desarrollo web y que puede ser embebido en páginas HTML. Su sintaxis recurre a C, Java y Perl, y es fácil de aprender. La meta principal de este lenguaje es permitir a los desarrolladores web escribir dinámica y rápidamente páginas web generadas; aunque se puede hacer mucho más con PHP [31].

1.4.6. Entorno Integrado de Desarrollo NetBeans 7.3

El NetBeans es un entorno modular basada en estándares de desarrollo integrado (IDE), escrito en el lenguaje de programación Java. Es de código abierto con todas las funciones escritas en el lenguaje de programación Java y una plataforma de aplicaciones de cliente enriquecido, que puede ser utilizado como un marco genérico para crear cualquier tipo de aplicación [32].

Este IDE ha alcanzado un considerable grado de madurez y fiabilidad y se ha convertido en una sofisticada herramienta para el desarrollo de aplicaciones de escritorio, aplicaciones cliente-servidor, aplicaciones móviles y aplicaciones web a través de diversas tecnologías como Java, PHP, HTML5 y C/C++. NetBeans es gratuito, de código abierto y hay una comunidad de programadores detrás de él por todo el mundo que lo han convertido en lo que es [33].

El NetBeans 7.3, versión utilizada para la implementación, presenta versiones en español y es de código libre, este último dato es importante para cumplir con los objetivos propuestos de la solución. Puede integrarse con *Symfony*, lo cual constituye una ventaja, ya que fue seleccionado como *framework* para el desarrollo de la aplicación web. Posee un editor y depurador de código PHP, presenta los lenguajes necesarios a utilizar como HTML, JavaScript y CSS y permite generar la documentación del código fuente. Es la herramienta con que más ha interactuado el equipo de desarrollo.

1.4.7. Sistemas de Gestores de Bases de Datos (SGBD)

Sistemas de gestión de bases de datos (en inglés *database management system*, abreviado DBMS) o SGBD es el *software* que permite la utilización y/o la actualización de los datos almacenados en una (o varias) base(s) de datos por uno o varios usuarios desde diferentes puntos de vista y a la vez, con el objetivo de suministrar al usuario las herramientas que le

permitan manipular, en términos abstractos, los datos, o sea, de forma que no le sea necesario conocer el modo de almacenamiento de los datos en la computadora, ni el método de acceso empleado.

Presenta varias características de ellas aprecia que:

- Permite crear y gestionar base de datos de forma fácil, cómoda y rápida.
- Ofrece una gran flexibilidad para el trabajo con base de datos relacionales.
- Ofrece un ambiente agradable dado por su interfaz gráfica.

Para administrar la base de datos, se utiliza la interfaz web phpMyAdmin, que es bastante potente, amigable y una de las más extendidas. Es un programa de libre distribución en PHP, muy completa que permite acceder a todas las funciones típicas de la base de datos MySQL a través de una interfaz web muy intuitiva [34].
A continuación se detalla el SGBD escogido.

1.4.7.1. MySQL 5.5.24

MySQL 5.5.24 es el servidor de bases de datos relacionales más popular, desarrollado y proporcionado por MySQL AB. Es un sistema de administración de bases de datos relacionales donde se almacenan los datos en tablas separadas en lugar de poner todos los datos en un solo lugar, lo cual agrega velocidad y flexibilidad. La parte SQL de "MySQL" significa "Lenguaje Estructurado de Consulta", y es el lenguaje más usado y estandarizado para acceder a bases de datos relacionales [34].
El servidor de bases de datos MySQL es muy rápido, seguro y fácil de usar. Además de ser el recomendado para el trabajo con *Asterisk* (*software* empleado para la telecomunicación) ya que las Bases de Datos de este sistema están desarrolladas en este servidor.

1.4.8. Servidores para aplicación web

Un servidor web es un programa que está diseñado para transferir hipertextos, páginas web o páginas HTML. Se ejecuta en un ordenador manteniéndose a la espera de peticiones por parte de un cliente (un navegador web) y responde a estas peticiones adecuadamente mediante una página web que se exhibirá en el navegador o mostrando un mensaje si se detectó algún error [43]. Se decidió utilizar el servidor Apache 2.2.22 para alojar la aplicación web.

1.4.8.1. Apache 2.2.22

Apache es principalmente usado para servir páginas web estáticas y dinámicas en la www. Es el servidor web del popular sistema XAMP (Wamp para Windows y Lamp para Linux), junto con MySQL y los lenguajes de programación PHP/Perl/Python [41].

El objetivo principal de Apache es proporcionar un servidor eficiente, seguro y extensible que proporcione servicios HTTP en sincronización con los estándares actuales. Es un esfuerzo de desarrollo de *software* de colaboración cuyo objetivo es crear la implementación de un código fuente de un servidor HTTP (Web), de calidad comercial con muchas características y de libre disposición. La versión 2.2.22 tiene correcciones de errores [42].

1.5. Conclusiones del capítulo

El análisis y estudio apoyado de los métodos científicos de investigación, permitió alcanzar una mayor profundidad y organización a la hora de elaborar el marco teórico que da base y sustenta la investigación realizada. Dando paso al conocimiento y definición de los principales conceptos que complementan la investigación. El estudio de sistemas similares permitió obtener características útiles para el desarrollo de la solución propuesta.

Se realizó un análisis de las metodologías, herramientas y tecnologías utilizadas para la solución, por lo cual se ha llegado a la conclusión de que es importante realizar un sistema teniendo en cuenta las facilidades de acceso a la misma desde distintas partes. Es de vital importancia el dominio de estas herramientas seleccionadas y analizar mejor forma de aplicarlas, para desarrollar un sistema de máxima calidad que cumpla con los requisitos propuestos y de al cliente una versión que satisfaga sus intereses.

CAPÍTULO 2. CARACTERÍSTICAS DEL SISTEMA. EXPLORACIÓN Y PLANIFICACIÓN

Introducción

En este capítulo se realizará un análisis de las características del sistema a desarrollar, teniendo como aspecto principal la propuesta del sistema, explicando sus principales características, se especificarán los requisitos funcionales y no funcionales. Se hará alusión a las fases de Exploración y Planificación de la metodología seleccionada mostrando las Historias de Usuarios (HU) en cada iteración definida.

2.1. Objeto de automatización

El uso de los buzones de voz para el envío de los mensajes en Cuba, está casi obsoleto, debido a la gran cantidad de recursos y esfuerzos que se necesitan para desarrollar estos sistemas automáticos. Actualmente no se cuenta con una infraestructura que sustente los servicios prestados, obligando la realización de citas o la llegada de información en un tiempo inapreciable.

Debido a que no se pueden enviar mensajes de voz a varias personas al mismo tiempo, la llegada de información es realizada de forma presencial (reuniones, etc.) o a través del teléfono y a una persona a la vez. Por cuanto ETECSA no oferta en estos momentos, un servicio telefónico convencional de mensajería colectiva, el desarrollo de la mensajería múltiple en el correo de voz de la telefonía fija, debe garantizar la automatización de este proceso mediante un sistema diseñado para las empresas que necesiten de este servicio a través del correo de voz, con el objetivo de minimizar gastos y tiempo, al ser este muy utilizado para la transmisión de información.

Este sistema permite que los usuarios, después de haberse identificado con su código de seguridad, poder enviar mensajes de voz y administrar sus listas (crear listas, eliminar listas, añadir o eliminar destinatario) a través del teléfono, guiados por una operadora, que proporciona los flujos a seguir para la realización de las acciones antes mencionadas, la misma debe ser clara y precisa, con el fin de satisfacer las necesidades del usuario y que este pueda interactuar con el sistema sin complicaciones.

A continuación se muestra la modelación del sistema analizado:

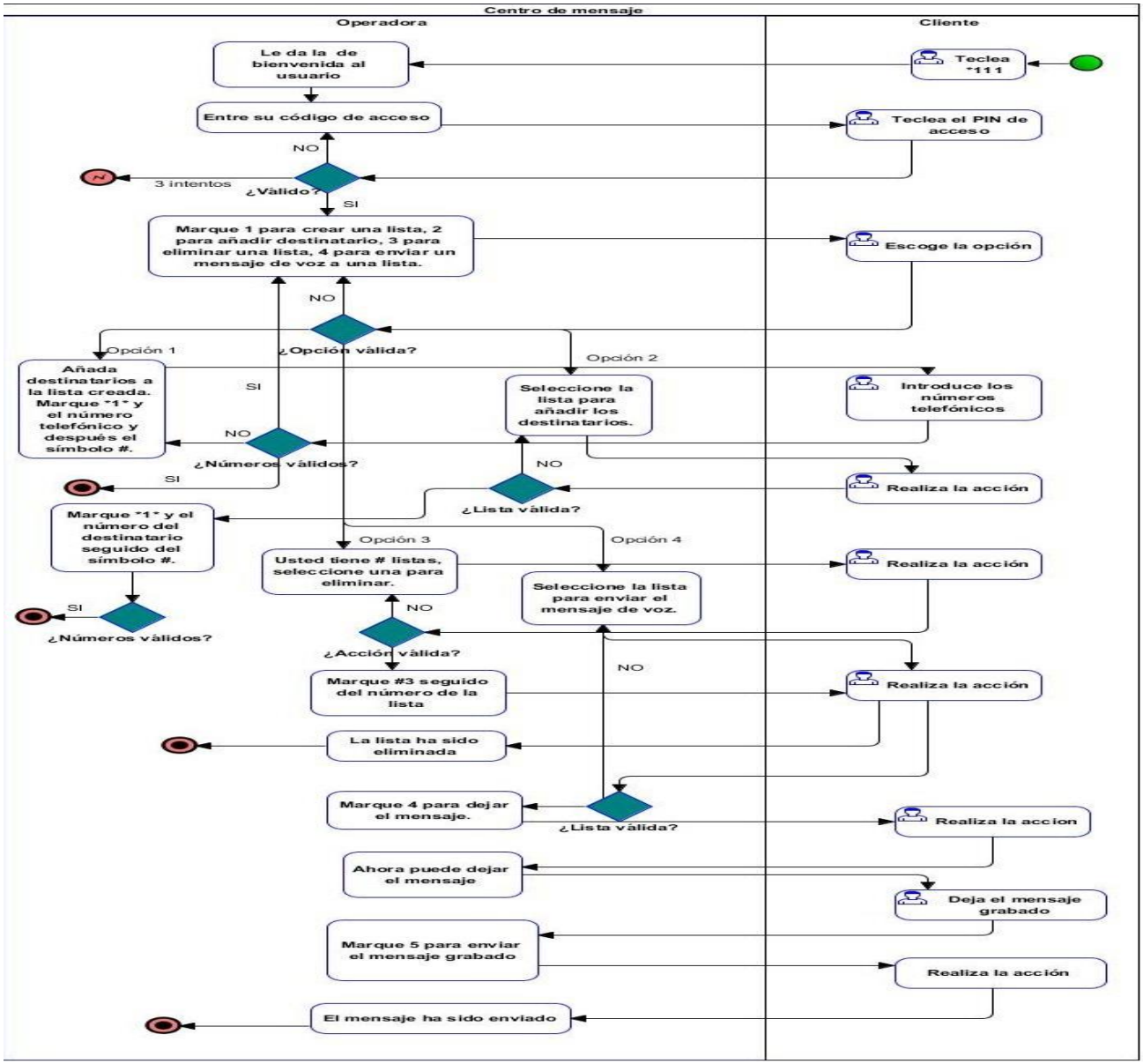


Figura 3. Modelación del proceso del sistema diseñado.

2.2. Propuesta del sistema

Para la creación de este sistema se va a desarrollar una aplicación informática, la cual permite mediante la web, que un usuario registrado con su número telefónico y contraseña, pueda administrar sus listas (crear listas, eliminar listas, añadir o eliminar destinatario de la lista), así como visualizar el resultado de la acción realizada. Además, la aplicación facilita la inserción y/o modificación de los números telefónicos de los destinatarios.

Se desarrollará un servicio de respuesta iterativa, el mismo contará con grabaciones que le permiten al usuario interactuar con el sistema.

Figura 4.Propuesta del sistema (fuente: Elaboración propia).

2.2.1. Sistema de Servicio Telefónico Iterativo

El Sistema de Servicio Telefónico Interactivo (SSTI) permite la interacción del usuario con el sistema, a través del teléfono fijo y mediante grabaciones que indican los pasos a seguir, para el envío de mensajes de voz masivos. Estas deben ser claras y precisas con el fin de satisfacer las necesidades del usuario.

A continuación se modela el proceso del SSTI antes mencionado.

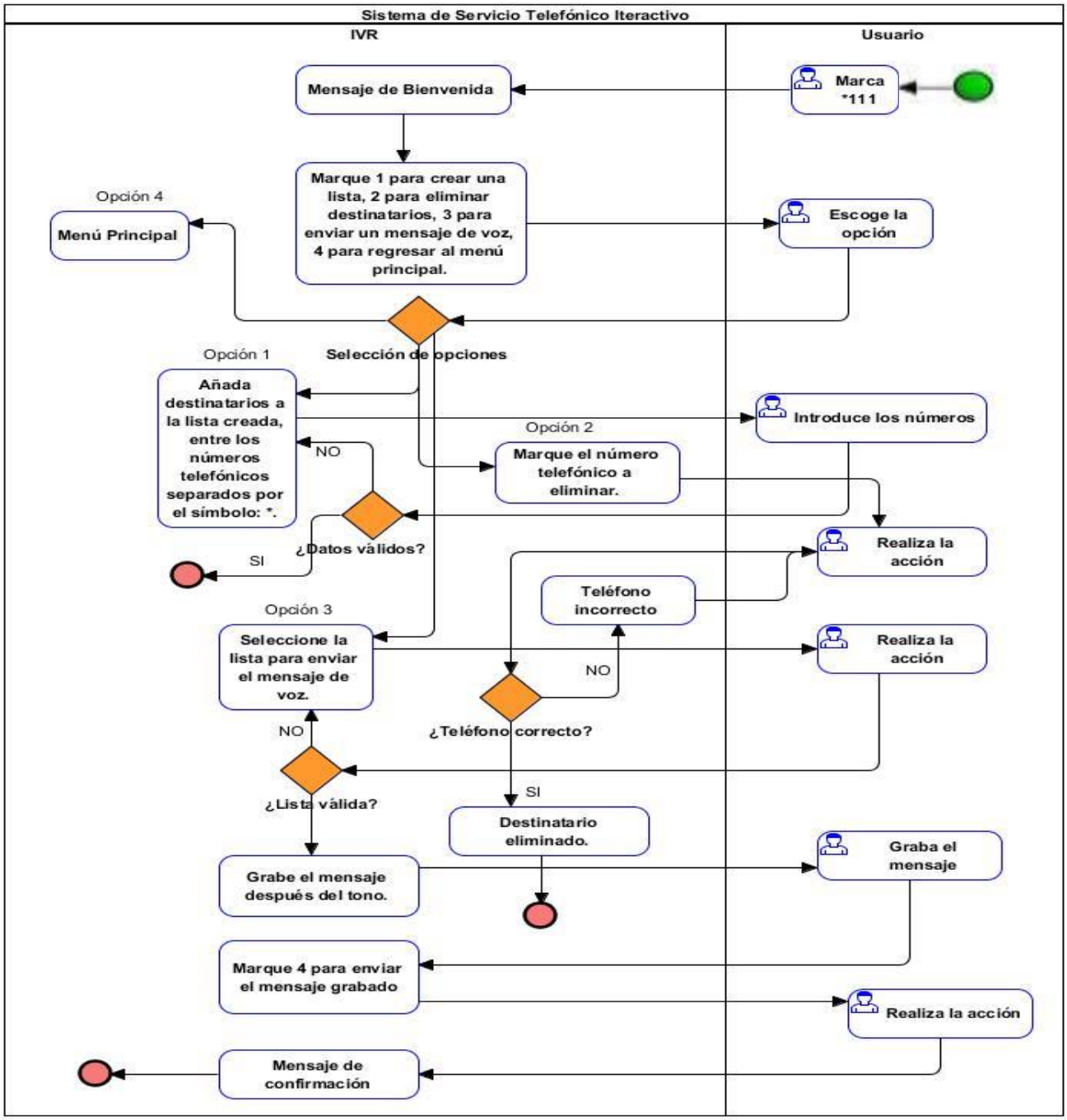


Figura 5.Proceso del Sistema de Servicio Telefónico Iterativo.

Al ser efectuada una llamada a *Asterisk* mediante el número *111, se reproduce el siguiente mensaje de bienvenida al usuario: "*Bienvenido al Sistema de Mensajería Masiva para la Telefonía Fija*", el sistema le brinda al usuario 4 opciones (1, 2, 3, y 4), mediante un menú de voz (menú de voz principal)

Si el usuario escoge la opción 1, el sistema reproduce el siguiente mensaje: *"Añada destinatarios a la lista creada"*, el usuario introduce los números telefónicos, el sistema valida los datos insertados, si los números son válidos el usuario puede terminar el flujo colgando el teléfono o cuando se llegó al tiempo máximo de inactividad del sistema, si los datos son inválidos el sistema reproduce el mensaje anterior.

Si el usuario escoge la opción 2, el sistema reproduce el siguiente mensaje*: "Marque el número telefónico a eliminar"*, el usuario realiza la acción, el sistema valida la acción, de no existir el número, el sistema reproduce el mensaje: *"Teléfono incorrecto"*, de lo contrario reproduce el mensaje: *"Destinatario eliminado"*, el usuario termina el flujo colgando el teléfono o cuando se llegó al tiempo máximo de inactividad del sistema.

Si el usuario escoge la opción 3, el sistema reproduce el siguiente mensaje: *" Seleccione la lista para enviar el mensaje de voz "*, el usuario realiza la acción, el sistema verifica si la lista es válida, si la lista no existe entonces el sistema reproduce el mensaje anterior, de lo contrario reproduce el mensaje: *"Grabe el mensaje después del tono"*, el usuario graba el mensaje, el sistema reproduce el siguiente mensaje: *"Marque 4 para enviar el mensaje grabado"*, el usuario realiza la acción y después el sistema reproduce el mensaje de confirmación, el flujo termina cuando el usuario cuelga el teléfono o cuando se llegó al tiempo máximo de inactividad del sistema.

Si el usuario selecciona la opción 4, el sistema reproduce el menú de voz principal.

2.2.2. Sistema de Gestión de Listas Telefónicas

El Sistema de Gestión de Listas Telefónicas (SGLT) será capaz de administrar las listas de distribución, mediante una aplicación web, en el cual, el usuario una vez identificado con su número telefónico y contraseña, será capaz de consultar la información relacionada con el uso del servicio, así como gestionar sus listas (crear lista, eliminar lista y visualizar) y gestionar sus destinatarios (adicionar, modificar y eliminar destinatario, visualizar). A continuación se modela el proceso del negocio antes mencionado.

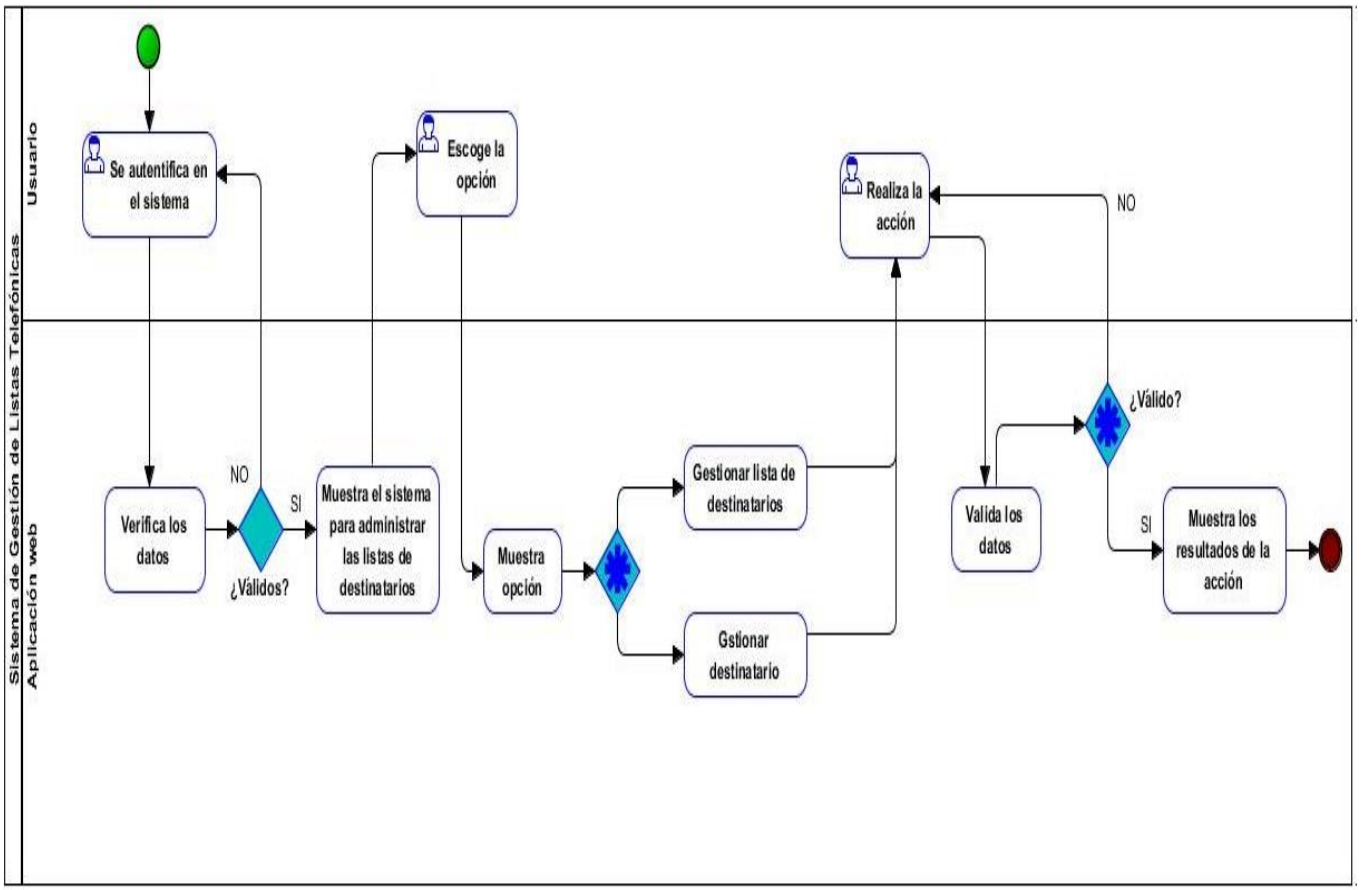


Figura 6. Proceso del Sistema de Gestión de Listas Telefónicas.

2.2.3. Flujo de procesos del sistema propuesto1

El proceso comienza con la introducción de datos por parte del cliente, introduciendo en el proceso su número de teléfono y contraseña. Una vez autenticado contará con todo los privilegios de la aplicación, posibilitándole crear o eliminar listas y añadir o eliminar destinatarios, así como modificar números telefónicos y visualizar los datos, además de analizar los datos de las listas guardadas en la base de datos.

2.2.4. Actores involucrados en el sistema

Los actores involucrados en el sistema son aquellas entidades, en este caso las personas, que interactúan con el sistema añadiendo o consumiendo información necesaria para el manejo de los requisitos.

Tabla 3. Actores relacionados con el sistema.

Actores involucradas en el sistema	Justificación
Cliente	Es la persona que se encarga de interactuar con el SGLT y con el SSTI.
Administrador	Es el encargado de integrar el SGLT con el SSTI. Además de gestionar los usuarios registrados en la aplicación.

2.3. Funcionalidades del Sistema de Gestión de Listas de Mensajes Telefónicos (SIGLIMT)

Los requisitos funcionales son declaraciones de los servicios que debe proporcionar el sistema. Especifican la manera en que este debe reaccionar a determinadas entradas, así como debe de comportarse el sistema en situaciones particulares y pueden declarar específicamente lo que el sistema no debe hacer [44]. A continuación se muestran las funcionalidades que presentes en el sistema:

R1. **Añadir listas de destinatarios:**

Subsistema Gestión telefónica (portal web):

R1.1.Mostrar formulario para el nombre de la lista.

R1.2. Validar datos.

Subsistema Servicio telefónico (IVR):

R1.3. El sistema reproduce los pasos a seguir.

R1.4.Valida los datos.

R2. Gestionar destinatario:

Subsistema Gestión telefónica (portal web):

R2.1. Mostrar formulario para modificar número telefónico del destinatario.

R2.2. Mostrar formulario para añadir destinatario.

R2.3. El usuario debe marcar en la lista de destinatarios el destinatario que desee eliminar.

R2.4. Validar datos.

Subsistema Servicio telefónico (IVR):

R2.5. El sistema reproduce los pasos a seguir

R2.6. Valida los datos.

R3. Autenticar:

Subsistema Gestión telefónica (portal web):

R3.1. Mostrar el formulario de autenticación en la página.

R3.2. Validar los datos introducidos por el usuario.

R3.3. Mostrar al usuario las opciones que tiene para interactuar con el sistema.

R4. Eliminar listas de destinatario*:*

Subsistema Gestión telefónica (portal web):

R4.1. Mostar la opción para eliminar la lista seleccionada.

R5. Grabar mensajes de voz.

Subsistema Servicio telefónico (IVR):

R5.1. Nombre y extensión de los archivos grabados.

R6. Enviar mensajes de voz.

2.4. Lista de Reserva [7]del producto

La lista de reserva del producto, recoge todas las características no funcionales que todo sistema debe poseer.

Para la correcta elaboración y redacción de estos requisitos se realizó un estudio de la norma NC ISO/IEC 9126-1:2005, logrando mejorar la calidad interna y externa del producto. A continuación se evidencian estas características y sub-características.

R7. Confiabilidad:

R7.1. El tiempo máximo de inactividad será de 7 segundos, una vez que el sistema IVR quede inactivo el usuario deberá de seleccionar de nuevo el servicio.

[7] Artefacto definido por el equipo, para describir los requisitos no funcionales del sistema.

R7.2. El sistema deberá estar disponible 24 horas.

R7.3. El sistema mediante una grabación, le indicará al usuario los pasos que debe seguir para trabajar con el servicio brindado.

R8. Usabilidad:

R8.1. Comprensibilidad y Cognoscibilidad: el sistema estará basado en una interfaz de diseño común, de forma que el usuario pueda comprenderlo y operarlo con facilidad, evitando la desorientación.

R8.2. Tipo de usuario: el sistema será de utilidad para las diferentes empresas y organizaciones que necesiten este servicio así como para todos los abonados de las centrales telefónicas digitales del país.

R8.3. Operabilidad: el *software* cuenta con funcionalidades de fácil comprensión para el usuario, de modo que este pueda operarlo y controlarlo sin dificultad.

R8.4. Atracción: el sistema presenta una interfaz de fácil aprendizaje por parte del usuario evitando la sobrecarga de información y la utilización en exceso de imágenes además del buen uso de los colores sin dejar de ser atractiva.

R9. Eficiencia:

R9.1. El tiempo máximo de respuesta deberá ser menor de 5 segundos.

R10. Mantenibilidad:

R10.1. Flexibilidad y Estabilidad: como es un *software* de código libre, un usuario administrador que cuente con conocimientos previos podrá aplicar los cambios que desee al *software*, con el objetivo de satisfacer las necesidades que se presenten minimizando los efectos inesperados de las modificaciones.

R11. Requerimientos de restricciones de diseño:

R11.1. La duración máxima de los mensajes será 30 segundos.

R11.2. Este sistema solo deberá ser aplicable para la telefonía fija.

R11.3. La lista deberá tener como máximo 12 destinatarios.

R12. Características del *hardware/ software*

R12.1. Servidor de aplicaciones web

Procesador: Intel Pentium III, 700 Mhz

Memoria RAM: 1GB

Sistema Operativo: Ubuntu Server

Tamaño de la instalación: 50MB

Disco duro: 160GB

Servidor web: Apache 2.2.22

R12.2. Servidores de Bases de Datos

Procesador: Intel Pentium IV, 700 Mhz

Memoria RAM: 2GB

Sistema Operativo: Ubuntu Server

Disco duro: 160GB

Gestor de Bases de Datos: MySQL 5.5.24

R12.3. Servidor de Telecomunicación

RAM mínima: 64 MB

Espacio en Disco: no menos de 500 GB

Procesador: Intel Pentium III, 700 Mhz

Servidor de telecomunicación: *Asterisk* 1.8

R12.4. PC cliente

Memoria RAM: 512MB

Navegador web: *Firefox, Chrome, Safari, Opera, Internet Explorer*, etc.

2.5. Arquitectura

La arquitectura de *software* es un conjunto de patrones que proporcionan un marco de referencia necesario para guiar la construcción de un *software*, permitiendo a los programadores, analistas y todo el conjunto de desarrolladores compartir una misma línea de trabajo y cubrir todos los objetivos y restricciones de la aplicación [32].

Es una representación que permite que un ingeniero del *software*: analice la efectividad del diseño para cumplir con los requisitos establecidos, considere opciones arquitectónicas en una etapa en que aún resulta relativamente fácil hacer cambios al diseño y reduzca los riesgos asociados con la construcción del *software* [45].

2.5.1. Propuesta de Arquitectura del sistema

El cliente será el que interactúe con el IVR y a la misma vez el que defina las listas de destinatarios en la web o en el teléfono. Tanto la aplicación web como el servidor *Asterisk* deben lograr una conexión con el Gestor de Base de Datos MySQL, con el propósito de que el mismo pueda leer los datos almacenados de la interacción con el cliente, así como modificar los ya existentes en la Base de Datos.

Figura 6.Propuesta de la arquitectura del sistema.

2.5.2. Arquitectura Cliente-Servidor

Se utilizó la arquitectura Cliente-Servidor para la realización del sistema ya que consiste en un cliente que realiza peticiones a otro programa (el servidor) que le da respuesta. Es la arquitectura utilizada por el servidor PBX del sistema desarrollado.

El remitente de esta arquitectura conocido como el cliente es el que inicia las solicitudes a través de una interfaz web o física esperando a su vez la respuesta del servidor, el mismo interactúa directamente con los usuarios finales. El receptor de estas solicitudes es conocido

como el servidor, el cual al iniciar espera a que le lleguen las solicitudes de los clientes para posteriormente procesarla y enviar la respuesta, no es frecuente que interactúe directamente con los usuarios finales.

Debido al resultado del estudio de esta arquitectura se propuso dos capas, con el fin de separar las responsabilidades, centralizar los recursos, mejorar la seguridad, etc. Estas capas están estrechamente relacionadas logrando una correcta comunicación entre ellas.

- Capa de presentación: Está relacionada con la presentación de la información al usuario y con toda la interacción con él [47], ya que puede visualizar y realizar todas las peticiones que desee, en este caso es conocida como interfaz gráfica y física.
- Capa de negocio: Es la que realiza todo lo relacionado con las respuestas a las peticiones del usuario, es donde residen los programas que se ejecutan, en este caso es conocido como lógica del negocio ya que se establecen las reglas que deben cumplirse.

2.5.3. Estilos arquitectónicos

Un estilo arquitectónico es la transformación impuesta al diseño de todo un sistema, con el objetivo de establecer una estructura para todos los componentes. Estos difieren de los patrones arquitectónicos en varios elementos fundamentales: 1) el alcance de un estilo es mayor, ya que no se concentra en un aspecto sino en toda la arquitectura; 2) un patrón impone una regla sobre la arquitectura, pues describe la manera que el *software* manejará algún aspecto de su funcionalidad al nivel de su infraestructura y 3) los patrones arquitectónicos tienden a abarcar aspectos específicos del comportamiento dentro del contexto de la arquitectura. Los estilos arquitectónicos se usan junto con los patrones para determinar la forma de la estructura general de un sistema [46].

2.6.1. Modelo Vista Controlador

Para el desarrollo de la aplicación web SGLT se eligió el estilo arquitectónico Modelo Vista Controlador debido a la organización en partes que posee: separa los datos, la interfaz de usuario y la lógica de control en tres componentes distintos:

Modelo: Es la representación especifica de la información con la cual el sistema opera, es la encargada de todo el acceso a los datos y las funciones (lógica del negocio). Lleva un registro de datos de las vistas y controladoras en función individual garantizando que si se cambia de

gestor de base de datos ocurra un daño mínimo a la aplicación, maximizando la adaptabilidad a los cambios [48].

Vista: Se encarga de la interacción del usuario con la aplicación web mostrando la información del modelo al usuario. Permite poder realizar cambios sin tener que tocar una parte delimitada del código [48].

Controlador: Se encarga de unir las capas mencionadas anteriormente, esta capa es la que escucha los cambios en la vista y se los envía al modelo, el cual le regresa los datos a la vista. Para recoger los datos se usan los métodos GET y POST establecidos en los formularios HTML [48].

En la solución propuesta, un ejemplo de este estilo se observa en el *Bundle* Listabundle donde se encuentra los directorios: *Controller* (en este están presentes las clases *DefautlController*, ListaController, *PersonController*, *SecurityController* y *UserController*), *Resources* (están presentes los directorios que contienen las clases de las vistas, así como las plantillas .twig, los archivos javascript y css) y Entity (contiene las clases que representan el modelo: Listas, Person y User).

2.6. Fase de exploración

En esta fase los clientes plantean a grandes rasgos los elementos a reflejar en las historias de usuario que son de interés para la entrega del producto. Permite que el equipo de desarrollo se familiarice con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto, se prueba la tecnología además la fase de exploración toma pocas semanas a pocos meses, dependiendo del conocimiento que tengan los programadores con la tecnología [48].

2.6.1. Historias de usuario

Las historias de usuario identifican acciones relacionadas con el comportamiento del sistema, en ella se emplea la terminología del cliente excluyendo el lenguaje técnico. Estas son muy empleadas para estimar el tiempo, el lenguaje debe ser corto y lo más sencillo posible. Durante la fase de exploración de identificaron 7 historias de usuario, las cuales se detallan a continuación:

Tabla 4. HU1: Añadir lista de destinatario.

Historia de Usuario	
Número: 1.	**Usuario:** Cliente.
Nombre de la historia: Añadir lista de destinatario.	
Prioridad en negocio: Media.	**Riesgo en desarrollo:** Medio.
Puntos estimados: 2.	**Iteración asignada:** 1.
Programador responsable: Rosalina Sayú Savón	
Descripción: El cliente escoge la opción: Añadir lista destinatario y completa los formularios en la aplicación web. En el IVR el usuario después de seleccionar el servicio y la opción 1 introduce los números telefónicos, las listas se crean con un nombre de destinatario predeterminado (default), el cual podrá ser modificado en la web. El sistema verifica si existe otra lista con el mismo nombre o si existe el número telefónico, de ser así se muestra un mensaje, sino, se mostrará la lista creada (en el caso de la aplicación web).	
Observaciones: Al crear la lista en la página web, esta se presentará vacía inicialmente. En el IVR es obligatorio introducir números telefónicos para crear las listas.	

Tabla 5. HU2: Gestionar destinatario

Historia de Usuario	
Número: 2.	**Usuario:** Cliente.
Nombre de la historia: Gestionar destinatario.	
Prioridad en negocio: Media.	**Riesgo en desarrollo:** Medio.
Puntos estimados: 1.	**Iteración asignada:** 1.
Programador responsable: Rosalina Sayú Savón	
Descripción: En la aplicación web el cliente deberá introducir los datos del destinatario (nombre, el número telefónico y seleccionar la lista a la cual pertenece), así como eliminarlo o modificar su número de teléfono y nombre. En el IVR el cliente podrá eliminar el destinatario introduciendo su número telefónico, guiado por las grabaciones de voz.	
Observaciones: La lista debe estar creada para realizar estas operaciones.	

Tabla 6. HU3: Autenticar

Historia de Usuario	
Número: 3.	**Usuario:** Cliente.
Nombre de la historia: Autenticar.	
Prioridad en negocio: Media.	**Riesgo en desarrollo:** Medio.
Puntos estimados: 1.	**Iteración asignada:** 1.
Programador responsable: Rosalina Sayú Savón	
Descripción: Se brinda la posibilidad de que la persona que acceda al sistema introduzca sus datos: número telefónico y contraseña en el caso de la aplicación web, en el caso del IVR solo se podrá realizar la llamada una vez registrado en el protocolo de configuración (sip.conf) de *Asterisk*.	
Observaciones: No procede.	

Tabla 7. HU4: Eliminar lista de destinatario.

Historia de Usuario	
Número: 4.	**Usuario:** Cliente.
Nombre de la historia: Eliminar lista de destinatario.	
Prioridad en negocio: Media.	**Riesgo en desarrollo:** Medio.
Puntos estimados: 2.	**Iteración asignada:** 1.
Programador responsable: Rosalina Sayú Savón	
Descripción: El cliente deberá escoger la opción para eliminar la lista deseada en la aplicación web.	
Observaciones: Debe de estar creado al menos una lista de destinatario para poder eliminar.	

Tabla 8. HU5: Grabar mensajes de voz

Historia de Usuario	
Número: 5.	**Usuario:** Cliente.
Nombre de la historia: Grabar mensajes de voz.	
Prioridad en negocio: Alto.	**Riesgo en desarrollo:** Alto.
Puntos estimados: 2.5.	**Iteración asignada:** 2.
Programador responsable: Rosalina Sayú Savón	
Descripción: El usuario deberá escoger la lista para enviar el mensaje (listas: 1, 2,…,12), luego graba el mensaje para su posterior envío guiado por el sistema IVR.	
Observaciones: Debe estar creada la lista seleccionada. El sistema muestra el nombre de la lista escogida. El mensaje se guarda con un nombre predeterminado y una extensión (mensaje.wav).	

Tabla 9. HU6: Enviar mensajes de voz

Historia de Usuario	
Número: 6.	**Usuario:** Cliente.
Nombre de la historia: Enviar mensaje de voz.	
Prioridad en negocio: Alto.	**Riesgo en desarrollo:** Alto.
Puntos estimados: 2.5.	**Iteración asignada:** 2.
Programador responsable: Rosalina Sayú Savón	
Descripción: El usuario debe escoger la opción para enviar el mensaje guiado por el IVR.	
Observaciones: Debe de estar creado al menos una lista de destinatario para poder enviar el mensaje.	

Tabla 10. HU7: Integración Web-IVR.

Historia de Usuario	
Número: 7.	**Usuario:** Administrador.
Nombre de la historia: Integración Web- IVR.	
Prioridad en negocio: Alto.	**Riesgo en desarrollo:** Alto.
Puntos estimados: 1.	**Iteración asignada:** 2.
Programador responsable: Rosalina Sayú Savón	
Descripción: Luego de contar con la aplicación web y la IVR, se realizará la integración de estas dos partes. Una vez finalizada se podrá contar con el nuevo servicio de mensajería múltiple para la telefonía fija.	
Observaciones: No procede	

2.7. Fase de Planificación

En esta fase el cliente establece la prioridad de cada historia de usuario, y correspondientemente, los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente. Las estimaciones del esfuerzo asociado a la implementación de las historias las establecen los programadores utilizando como medida el punto de función [50].

2.7.1. Estimación por historia de usuario

Para el óptimo desarrollo del sistema propuesto se ha realizado una estimación de esfuerzo por cada una de las historias de usuario identificadas, además se estiman los puntos de función para cada funcionalidad, estos no son más que la estimación por parte del equipo de la cantidad de líneas que se necesitarán para realizar el código, cada punto de función equivale a media semana de trabajo de programación.

El equipo mantiene un registro de la “velocidad” de desarrollo establecida en puntos por iteración, la misma es utilizada para establecer cuántas historias se pueden implementar para una fecha determinada o cuánto tiempo tomará implementar un conjunto de historias. Los puntos a completar se determinan al multiplicar el número de iteraciones por la velocidad del proyecto, esto es realizado cuando se desea planificar el tiempo. El número de iteraciones

necesarias se obtiene al dividir la suma de puntos de las historias de usuario seleccionadas para una iteración entre la velocidad del proyecto, y sumándolas luego hasta el total de las historias. Esto se realiza cuando se desea planificar la duración de programación del sistema [50]. A continuación se muestra la tabla de estimación del esfuerzo para cada Historia de Usuario.

Tabla 11. Estimación del esfuerzo por HU.

Historia de usuario	Puntos de función	Puntos de estimación (semana)
Añadir listas de destinatario.	4	2
Gestionar destinatario.	2	1
Autenticar.	2	1
Eliminar listas de destinatario.	4	2
Grabar mensajes de voz.	5	2.5
Enviar mensaje de voz.	5	2.5
Integración Web- IVR.	2	1

2.7.2. Plan de la Iteración.

Contiene las Historias de Usuario a implementar en cada fase de iteración, las cuales se traducen en tareas específicas de programación [51]. Durante la elaboración del Plan de la Iteración se tienen en cuenta: historias de usuario no abordadas, velocidad del proyecto, pruebas de aceptación no superadas en la iteración anterior y tareas no terminadas en la iteración anterior. Todo el trabajo de la iteración es expresado en tareas de programación, cada una de ellas es asignada a un programador como responsable [50].

Luego de haber identificado las historias de usuario y estimado el esfuerzo dedicado a la realización de cada una de ellas se da paso a la realización de la planificación de la etapa de implementación del SIGLIMT.

2.7.3. Iteración 1.

En esta primera iteración se lleva a cabo la implementación de las HU[8] del número 1 al 4, por su media complejidad en el negocio. Durante la misma se comienza el trabajo conforme a la arquitectura de la propuesta para el sistema, finalizando con una primera versión de la aplicación web y del sistema IVR.

[8] **H**istoria de **U**suario

2.7.4. Iteración 2.

El objetivo de esta iteración es la implementación de las HU número 5 y 6, que tienen un mayor grado de complejidad. Al finalizar esta iteración se obtendrá una versión final de la aplicación web y del sistema IVR.

2.7.5. Iteración 3.

Esta última iteración tendrá el objetivo de implementar la HU número 7, la cual es la encargada de fusionar el resultado de lo implementado en las iteraciones anteriores. Obteniendo al final de la misma una primera versión de prueba del Sistema de Gestión de Listas de Mensajes Telefónicos (SIGLIMT), el cual se pondrá a prueba para verificar su funcionamiento.

A continuación se muestra el plan de duración de las iteraciones que se tuvo en cuenta para representar y ordenar las 3 iteraciones definidas en las HU planteadas con anterioridad.

2.7.6. Plan de duración de las iteraciones.

El plan de duración de las iteraciones es el encargado de exponer cada una de las historias de usuario que serán implementadas en las iteraciones antes mencionadas, además se realizará una estimación del tiempo en que estas demorarán en ser desarrolladas. Para la realización de este plan se ha tomado la decisión de seguir un orden de implementación, el cual se muestra a continuación.

Tabla 12. Plan de duración de las iteraciones.

Iteración	Orden de las HU a implementar	Duración total de la iteración
1	Añadir listas de destinatario.	6 semanas
	Gestionar destinatario.	
	Autenticar.	
	Eliminar lista de destinatario.	
2	Grabar mensajes de voz.	5 semanas
	Enviar mensaje de voz.	
3	Integración Web- IVR.	1 semana

2.7.7. Plan de Entrega.

El Plan de Entrega es usado por el equipo de desarrollo para finalizar los entregables a tiempo, es consultado por el cliente para conocer las fechas pactadas de entregas mediante una

reunión con los programadores, para ello se tiene en cuenta dos parámetros: tiempo de desarrollo ideal y el grado de importancia para el cliente [50].

Se presenta en este epígrafe el plan de entregas estimado para la fase de implementación, como resultado de esto se realizaran versiones entregables del sistema al finalizar cada iteración.

Tabla 13. Plan de entregas del software

Sistema	Final de la iteración 1 (21 de abril del 2014)	Final de la iteración 2 (25 de mayo del 2014)	Final de la iteración 3 (30 de mayo del 2014)
SIGLIMT	SGLT v0.1	SSTI v0.1	SIGLIMT v1.0

2.8. Conclusiones del capítulo

La elaboración de los artefactos permitió realizar una descripción de la propuesta del sistema que se desea automatizar, tomando estos como guía para la posterior implementación de la solución. Se realizó un análisis sobre el estilo arquitectónico utilizado y se pudo constatar para quienes va dirigido el sistema delimitando la existencia de dos tipos de roles. Se identificaron un total de 7 HU, especificando la prioridad de cada cual para establecer el orden y el número de iteración en que serán desarrolladas según la estimación propuesta, estas permitieron la confesión del plan de entregas.

CAPÍTULO 3. DISEÑO DEL SISTEMA

Introducción

Durante este capítulo se logrará un diseño sencillo y fácil de implementar según lo que describe la metodología seleccionada en este caso XP, el cual requiere menor tiempo y esfuerzo en la obtención del producto final. En la fase de diseño se van a definir los patrones de diseño utilizados, las tarjetas CRC[9] de cada una de las iteraciones como herramienta de reflexión en el diseño de *software*. Se propone el diseño de la bases de datos utilizada.

3.1. Patrones Generales para la Asignación de Responsabilidades.

Los patrones de diseño expresan esquemas para definir estructuras de diseño (o sus relaciones) con las que construir sistemas de *software*. Los patrones de arquitectura expresan un esquema organizativo estructural fundamental para sistemas de *software* [36].

3.1.1. Patrones GOF (*Gang Of Four*, Banda de Cuatro).

Los patrones GOF son soluciones concretas con el propósito de resolver un determinado problema, se encargan de la creación de instancias de los objetos (son creacionales), plantean las relaciones entre las clases combinándolas y formando estructuras mayores (son estructurales) y expresan la interacción e interrelación entre las clases (comportamiento).

3.1.1.1. Decorador

Este patrón plantea la utilización de una o varias plantillas globales de diseño base, que guardan el código que es usual para varias platillas del sistema, para no tener que repetirlo en cada interfaz. Para la presente investigación las plantillas decoradas son las clases base, *default, index, login y register*.

3.1.1.2. Singleton

Está diseñado para restringir la creación de objetos pertenecientes a una clase o el valor de un tipo a un único objeto. Su intención consiste en garantizar que una clase sólo tenga una instancia y proporcionar un punto de acceso global a ella [41]. Este patrón se ve evidenciado en las clases controladoras del sistema propuesto.

3.1.2. Patrones generales de *software* para la asignación de responsabilidades (GRASP).

GRASP es un acrónimo que significa General Responsibility Asignment Software Patterns (patrones generales de *software* para asignar responsabilidades), representa parejas de

[9] **C**ontenido **R**esponsabilidad **C**olaboración

problema solución con un nombre, que codifican buenos principios y sugerencias relacionados frecuentemente con la asignación de responsabilidades [39].

A continuación se enuncian los utilizados en el desarrollo del sistema:

3.1.2.1. Experto

Consiste en asignar una responsabilidad (métodos) al experto en información, que a su vez es la clase que cuenta con la información necesaria para cumplir una responsabilidad. En la misma se encuentran las clases que se encargan de interactuar con la Base de Datos a través de ORM Doctrine

Problema: ¿Cómo lograr que cada clase cumpla con las funcionalidades que le corresponde?

Solución: Asignarle a una clase la responsabilidad de crear instancias de otra clase, establecer un método a la clase que tiene la información necesaria para cumplirla.

Ejemplo: Este patrón se ve evidenciado en las clases controladoras las cuales son expertas en información y permiten crear formularios que permiten mostrar los campos requeridos en las vistas. A continuación de muestra un ejemplo de la utilización de este patrón.

```
public function createAction(Request $request) {
    $entity = new Lista();
    $form = $this->createCreateForm($entity);
    $form->handleRequest($request);

    if ($form->isValid()) {
        $em = $this->getDoctrine()->getManager();
        $entity->setUser($this->getUser());
        $em->persist($entity);
        $em->flush();

        return $this->redirect($this->generateUrl('lista_show', array('id' => $entity->getId())));
    }

    return $this->render('ListaBundle:Lista:new.html.twig', array(
                'entity' => $entity,
                'form' => $form->createView(),
    ));
}
```

Figura 7.Uso del patrón experto y creador en el controlador ListaController.

3.1.2.2. Creador

Consiste en crear una instancia con información de otras clases, asigna a una clase B la responsabilidad de crear una instancia de una clase A. Si A tiene información necesaria para crear B entonces A es el perfecto creador de B.

Problema: ¿Cómo lograr relacionar una clase con la clase responsable de realizar la conexión a la base de datos?

Solución: Establecer a una clase la responsabilidad de crear una instancia de otra. En el desarrollo de la solución, la implementación de este patrón se evidencia cuando las clases controladoras las cuales crean formularios tomando la responsabilidad de crear instancias de la clase ListaType, además de crear entidades asociadas a un determinado formulario. Un ejemplo de este patrón se puede encontrar en la figura 7 y a continuación.

```
private function createEditForm(Lista $entity) {
    $form = $this->createForm(new ListaType(), $entity, array(
        'action' => $this->generateUrl('lista_update', array('id' => $entity->getId())),
        'method' => 'PUT',
    ));

    $form->add('submit', 'submit', array('label' => 'Editar', 'attr' => array('class' => 'btn btn-primary')));

    return $form;
}
```

Figura 8.Uso del patrón creador en el controlador ListaController.

3.1.2.3. Bajo acoplamiento

Asignar una responsabilidad para mantener un bajo acoplamiento, fomenta el aumento de la reutilización y la eliminación de la redundancia, creando clases más independientes.

Problema: ¿Cómo lograr que una clase no dependa de las otras?

Solución: Este patrón se basa en la idea de tener las clases lo menos entrelazadas posible, de tal forma que al producirse una modificación entre ellas tenga la mínima repercusión posible disminuyendo la dependencia. En la aplicación de concibió mantener separadas las clases pertenecientes al modelo de datos, las vistas de usuario y las clases controladoras.

3.1.2.4. Alta cohesión

Una clase de alta cohesión posee una importante funcionalidad relacionada y poco trabajo por hacer. Este patrón plantea que las clases se deben apoyar en el funcionamiento con otras para lograr su objetivo y no incluir en ella todo el proceso de negocio.

Problema: ¿Cómo lograr que las clases trabajen en su misma área de aplicación manteniendo la complejidad manejable?

Solución: Asignarle a una clase la responsabilidad de muchas cosas en áreas muy heterogéneas, asignarle funcionalidades muy moderadas en un área funcional y que colabore con otras para llevar a cabo sus tareas.

3.1.2.5. Controlador

Representa el sistema entero. Los controladores de fachada son adecuados cuando es imposible redirigir los mensajes o los eventos del sistema a otros controladores, como sucede con un sistema de procesamiento de mensajes.

Problema: ¿Cómo lograr atender un evento del sistema?

Solución: Asignarle a una clase la responsabilidad del manejo de mensajes de los eventos del sistema.

En la estructura de Symfony este patrón se evidencia dentro de la carpeta *Controller* que se encuentra en los paquetes de cada Bundle, en la misma se localizan las clases controladoras para el monitoreo de las aplicaciones web.
En la siguiente figura se muestra un ejemplo de la utilización de este patrón.

Clase SecurityController

```
class SecurityController extends Controller{

    /**
     * @return \Symfony\Component\HttpFoundation\Response
     */
    public function loginAction(){...}

    public function chpasswordAction(){}
}
```

Clase DefaultController

```
class DefaultController extends Controller {

    public function indexAction() {...}

}
```

Figura 9. Uso del patrón controlador en las clases SecurityController y DefaultControler.

3.2. Tarjetas Clase- Responsabilidad-Controlador (CRC)

Para el desarrollo del SIGLIMT se hizo necesario la utilización de la metodología XP, la cual no necesita la realización de diagramas de clases, debido a que esta contiene una variable denominada tarjetas de Cargo o Clases, Responsabilidad y Colaboración (CRC), las cuales determinan el comportamiento de cada actividad del sistema. Las tarjetas CRC identificadas durante el desarrollo se encuentran en el **Anexo 1.**

3.3. Modelo físico de la Bases de datos

La construcción de la Base de Datos es una de las tareas más importantes en el diseño de aplicaciones, ya que reflejan las relaciones y datos necesarios para el correcto funcionamiento de la aplicación. El funcionamiento de un IVR, parte de la creación de una Base de Datos a la que debe accederse para consultas [38], para ello el desarrollo del presente proyecto se ha creado una Base de Datos que contenga los registros de listas de destinatarios, estas están presentes en las tablas: lista_destinatarios, usuario y teléfono.

La Base de Datos se llama **Lista**, a continuación se muestra su diseño.

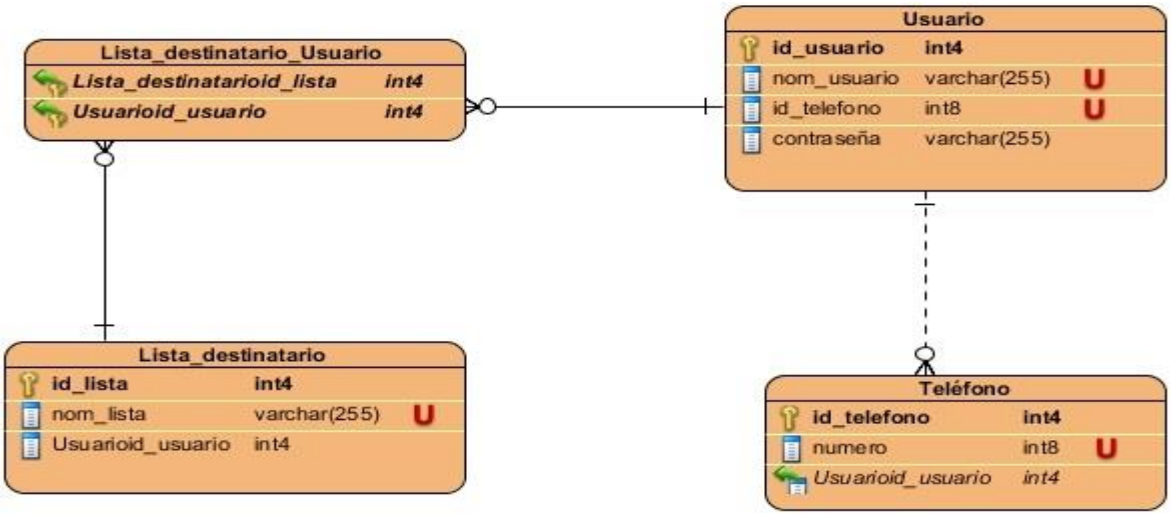


Figura 10. Modelo del diseño de la bases de datos.

3.4. Módulos del Sistema de Servicios Telefónico Iterativo

En este epígrafe se explicará a breves rasgos que componentes se necesitan y como estará estructurado el Sistema de Servicios Telefónico Iterativo, se detallan los módulos del IVR encontrados dentro de los archivos de configuración del *Asterisk*.

- **sip.conf:** es el archivo de configuración del canal correspondiente al protocolo SIP (Session Initiation Protocol, Protocolo de Inicialización de Sección) el cual es el

protocolo de señalización más popular en la actualidad. Es donde se definen los contextos y usuarios a utilizar [39]. El mismo presenta sub-módulos los cuales se definen a continuación con algunos ejemplo de las configuraciones realizadas:

- General: sub-módulo que define los puertos, contextos y direcciones ip.

Port = 5060: puerto que utiliza el servidor *Asterisk*.
Bindaddr = 0.0.0.0: especifica la IP donde estará *Asterisk* (si un equipo tiene más de una IP, 0.0.0.0 vale para cualquiera).
Language = es: lenguaje especificado para el servidor, en este caso es el español.

- *111: sub-módulo para la creación de la extensión del servicio, permite que un usuario acceda a las opciones que le brinda el sistema a través del teléfono.

[*111]: se específica la extensión.
Type = friend: tipo de extensión. Puede ser friend (para recibir y realizar llamadas), user (recibir llamadas) o peer (realizar llamadas).
Context = usuario: contexto donde entrará la llamada al dialplan.
Host = dynamic: si se conecta la extensión a un host que cambia de IP.

- 1100x: sub-módulo para la creación de la extensión de los usuarios con sus respectivas propiedades (x=1, 2, es la identificación del usuario). A través de esta extensión el usuario puede comunicarse con la persona que desee.

[1100x]
type=friend
username = 11001: nombre de usuario.
secret= 123: contraseña del usuario.
context = usuario: contexto al que pertenece el usuario.
mailbox=11001@default: buzón de voz del usuario
host= dynamic
allow = g729: códec permitidos por la extensión**.**
Allow = alaw: códec permitidos por la extensión**.**
Callgroup = 1: grupo de llamada al que pertenece el usuario.
Pickupgroup = 1: grupo de captura de llamada a la que pertenece el usuario.

- **extensions.conf:** es el archivo más importante del *Asterisk* y tiene como misión principal definir el dialplan o plan de marcado que seguirá la centralita para cada contexto y por tanto para cada usuario [40]. A continuación se definen los sub-módulos utilizados correspondientes a este archivo.

    - General: sub-módulo donde están definidas las propiedades del script.

    - Usuario: sub-módulo para procesar los requerimientos de las extensiones definidas en el archivo **sip.conf**. Aquí se define la extensión que se debe marcar para seleccionar el servicio.

      [usuario]: contexto que dirige el plan de marcado para los usuarios
      include => voicemail: para incluir el contexto de buzón de voz.
      exten => *111,1,Goto(ivr_tesis,s,1): permite que cuando se seleccione el servicio, el usuario pueda acceder al árbol de rutas del IVR.
      exten => 11001,1,Dial(SIP/11001,10): permite llamar a un usuario.
      exten => 11001,2,Voicemail(11001@default,u): permite acceder al buzón de voz del usuario, cuando este se encuentre ocupado.
      exten => 11002,1,Dial(SIP/11002,10)
      exten => 11002,2,Voicemail(11002@default,u)

- **IVR:** es el archivo donde se implementa las acciones a realizar por el usuario y la centralita, este no se encuentra definido en *Asterisk* por lo cual debe crear e incluir al archivo **extensions.conf** (#include IVR) donde generalmente son realizadas las implementaciones, este módulo fue creado con el objetivo de alcanzar una mejor organización y estructura del sistema. A continuación de describen los sub-módulos implementados.

    - ivr_tesis: sub-módulo donde se procesa el árbol de rutas del IVR.

    - crear y eliminar lista: sub-módulo que permite leer lo entrado por el teléfono para posteriormente realizar la conexión a la Base de Datos para crear o eliminar lista.

    - eliminar_destinatario: sub-módulo que permite leer lo entrado por el teléfono luego realiza la conexión a la Base de Datos para eliminar destinatarios seleccionados.

- grabar_mensaje: sub-módulo que permite grabar y enviar los mensajes de voz.

- **Use lista_de_destinatario:** módulo que hace uso de la Base de Datos para gestionar las listas.

- **voicemail.conf**: módulo que permite configurar el buzón de voz del usuario, el mismo deberá de poseer el identificador del usuario, así como la contraseña definida para el buzón.

3.4.1. Módulos para la conexión con la Base de Datos

- **odbcinst.in:** módulo que permite configurar el driver para la conexión con MySQL.

- **odbc.ini:** módulo que permite crear el identificador que *Asterisk* usará para realizar la conexión con MySQL. Este archivo permite especificar la Bases de Datos utilizada, la misma se debe especificar (Driver = MySQL o Postgres, etc.). En el mismo se especifican los parámetros de la BD (Usuario, Contraseña, Nombre de la Bases de Datos, Servidor, Puerto, etc.)

- **func_odbc.conf:** módulo que permite realizar las consultas que serán definidas en el dialplan.

- **cdr_mysql:** módulo en el cual se definen las tablas de la Base de Datos de *Asterisk*, este módulo es utilizado para realizar la integración con otras Bases de Datos. El mismo permitió la integración del IVR con la aplicación web conjuntamente con el módulo RealTime.

3.5. Conclusiones del capítulo

Para realizar un adecuado diseño e implementación del sistema se identificaron los patrones arquitectónicos a desarrollar y se representaron las clases más relevantes mediante las tarjetas CRC. Se realizó el diseño de las tablas de la base de datos de la aplicación web que serán consumidas por *Asterisk*, además de la definición de los módulos que fueron utilizados para el correcto funcionamiento del sistema.

CAPÍTULO 4. IMPLEMENTACIÓN Y PRUEBA

Introducción

En este capítulo se hace evidencia la transformación que da el *software* de ser algo abstracto a algo concreto logrando su materialización, ya que se hace referencia a las dos fases más importantes en el desarrollo de cualquier aplicación: la Implementación y la Prueba. Se realizaran las pruebas necesarias para comprobar si fueron cumplidos los objetivos trazados durante la realización del SIGLIMT.

4.1. Fase de implementación

Para dar continuidad a la metodología ágil escogida durante el inicio de cada iteración se revisa el plan de iteraciones y se expresan las tareas de programación, donde en cada una de ellas se nombra al responsable de su implementación, las mismas están escritas en un lenguaje técnico que no tienes por qué ser entendibles por el cliente. A continuación se detallan algunas de las iteraciones y las historias de usuario, las demás se encuentran en el **Anexo 2**.

4.1.1. Iteración #1.

En esta iteración se implementaron todas las historias de usuario referentes a las principales funcionalidades de la página web y parte del sistema IVR con motivo de mostrarle al cliente el primer prototipo funcional del proyecto SIGLIMT.

Tabla 14. Tiempo de las tareas abordadas en la iteración #1.

Historia de usuario	Estimación (días)	Real (días)
Añadir listas de destinatario.	2	2
Gestionar destinatario.	1	1
Autenticar.	1	1
Eliminar listas de destinatario	2	2
Total	**6**	**6**

4.1.1. Tareas de las Historias de Usuario implementadas en la iteración #1.

Tabla 15. Tarea #1 de la Historia de Usuario: Añadir listas de destinatario.

Tarea de ingeniería	
Número Tarea: 1	**Número Historia de Usuario:**1
Nombre Tarea: Diseño de la interfaz Añadir listas de destinatario.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 2
Fecha Inicio: 10/4/2014	**Fecha Fin:** 12/4/2014
Programador Responsable: Rosalina Sayú Savón	
Descripción: Se crea una interfaz capaz de gestionar los destinatarios.	

Tabla 16. Tarea #2 de la Historia de Usuario: Añadir listas de destinatario.

Tarea de ingeniería	
Número Tarea: 2	**Número Historia de Usuario:** 1
Nombre Tarea: Implementar la funcionalidad Añadir listas de destinatario.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 2
Fecha Inicio:10/4/2014	**Fecha Fin:** 12/4/2014
Programador Responsable: Rosalina Sayú Savón	
Descripción: Se crea la funcionalidad que permite añadir las listas de distribución.	

Tabla 17. Tarea #1 de la Historia de Usuario: Gestionar destinatario.

Tarea de ingeniería	
Número Tarea: 1	**Número Historia de Usuario:** 2
Nombre Tarea: Diseño de la interfaz Gestionar destinatario.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 1
Fecha Inicio: 14/4/2014	**Fecha Fin:** 15/4/2014
Programador Responsable: Rosalina Sayú Savón	
Descripción: Se diseñó la interfaz visual para gestionar los destinatarios.	

4.2. Fase de Pruebas

Unos de los pilares de la programación extrema es el proceso de pruebas. Se recomienda experimentar tanto sea posible con el fin ir mejorando a su vez la calidad del *software* desarrollado, reduciendo el número de errores no detectados y disminuyendo el transcurso entre la aparición de un error y su detección.

En esta fase de aumenta la seguridad para evitar efectos secundarios no deseados a la hora de realizar modificaciones y refactorizaciones.

Esta metodología seleccionada divide el sistema en dos grupos: pruebas unitarias (son las encargadas de verificar el código) diseñadas por los programadores y las pruebas de aceptación o pruebas funcionales (evalúan si al final de una iteración se lograron las funcionalidades requeridas) diseñadas por el cliente final [52].

4.2.1. Pruebas unitarias

Las pruebas unitarias se encargan de probar una clase en concreto, testeando cada uno de sus métodos y viendo si dados unos parámetros de entrada, la salida es la esperada [53]. Están destinadas para el aseguramiento del desarrollador.

El TDD (*Test Driven Development*) son pruebas unitarias que siguen el principio "test-first", la prueba unitaria se crea antes de crear la propia clase ayudando a desarrollar antes un mejor diseño. De ésta manera, se asegura que no exista ninguna clase que no esté probada. El BDD (Behaviour Driven Development) es una evolución de TDD, con una forma de escribir las pruebas más centrada en el comportamiento que debe tener la clase. Code Coverage es la métrica que mide la cantidad de código que está probado unitariamente [53].

Estas pruebas no generan artefactos y no son directamente palpables por el cliente, las mismas se realizaron constantemente en cada conclusión de una funcionalidad probándola directamente en el sistema.

En el transcurso del desarrollo del sistema se aplicaron una serie de pruebas de este tipo, que ayudaron a corregir errores encontrados dentro del código.

Un ejemplo lo constituye esto fragmento de la implementación del IVR:

```
exten => s,2,NoOp(Entre el número de teléfono)
exten => s,3,Read(telef,beep,6)
exten => s,4,NoOp(telefono: ${telef})
exten => s,5,Set(telefono=${ODBC_select(${telef})})
exten => s,6,GotoIf($["${telefono}foo" = "foo"]?notfound)
exten => s,7,NoOp(El teléfono: ${telefono} ya existe)
exten => s,8,Hangup()
exten => s,9(notfound),Set(ODBC_list_insert()=${telef})
exten => s,10,NoOp(Teléfono insertado)
```

Este código permitía que la cantidad de teléfonos insertados en las listas fuera infinita, lo cual se pudo corregir mediante una sentencia while permitiendo llegar al máximo de números permitidos.

```
exten => s,2,Set(COUNT=1)
exten => s,3,While($[ ${COUNT} < 13 ])
exten => s,4,NoOp(Entre el número de teléfono)
exten => s,5,Read(telef,beep,6)
exten => s,6,NoOp(telefono: ${telef})
exten => s,7,Set(telefono=${ODBC_select(${telef})})
exten => s,8,GotoIf($["${telefono}foo" = "foo"]?notfound)
exten => s,9,NoOp(El teléfono: ${telefono} ya existe)
exten => s,10,Hangup()
exten => s,11(notfound),Set(ODBC_list_insert()=${telef})
exten => s,12,NoOp(Teléfono insertado)
exten => s,13,Set(COUNT=${COUNT}+1)
exten => s,14,EndWhile( )
```

Se configuró un softphone o SFLphone que permite simular un teléfono fijo posibilitando la realización de estas pruebas auxiliado de la interfaz de monitorio y control del flujo de los procesos realizados en el servidor *Asterisk* (CLI, a la misma se puede acceder mediante el comando *Asterisk* -r), el mismo se ilustra a continuación:

Figura 11. Ejemplo de un SFLphone.

A continuación se definen algunas de los Casos de Prueba realizados.

Tabla 18. CP Verificar el acceso al servicio.

Caso de Prueba unitaria #1
Nombre: Verificar el acceso al servicio.
Probador: Rosalina Sayú Savón.
Descripción/Pasos: Prueba para determinar que los usuario puedan acceder al servicio sin ningún error. El usuario deberá marcar *111, luego el sistema deberá monitorear las entradas del usuario y su interacción.

Código al que se aplica:
`exten => *111,1,Goto(ivr_tesis,s,1)`
`[ivr_tesis]`
`exten => s,1,Answer`
`exten => s,2,Ringing`
`exten => s,3,background(/etc/alternatives/asterisk-prompt-en/conf-adminmenu-162)`
`exten => s,4,WaitExten(7)`

Resultado de la prueba: -- Executing [*111@usuario:1] Goto("SIP/11002-00000000", "ivr_tesis,s,1") in new stack -- Goto (ivr_tesis,s,1) -- Executing [s@ivr_tesis:1] Answer("SIP/11002-00000000", "") in new stack -- Executing [s@ivr_tesis:2] Ringing("SIP/11002-00000000", "") in new stack -- Executing [s@ivr_tesis:3] NoOp("SIP/11002-00000000", "MENÚ PRINCIPAL.....") in new s -- Executing [s@ivr_tesis:4] BackGround("SIP/11002-00000000", "/etc/alternatives/asterisk-prompt-en/conf-adminmenu-162") new stack -- <SIP/11002-00000000> Playing '/etc/alternatives/asterisk-prompt-en/conf-adminmenu-162.slin' (language 'es') Ingreso al contexto Respuesta del sistema y menú principal
Evaluación: Satisfactoria.

Tabla 19. CP Verificar que no se introduzcan números telefónicos iguales.

Caso de Prueba unitaria #1
Nombre: Verificar que no se introduzcan números telefónicos iguales.
Probador: Rosalina Sayú Savón.
Descripción/Pasos: Prueba para determinar que los números de teléfonos sean únicos. El usuario después de seleccionar el servicio escoge la opción 1 y luego introduce los teléfonos, el sistema deberá añadir los números a la base de datos, reproduciendo los mensajes guías y de confirmación.

Código al que se aplica:

```
exten => s,1,Wait(1)
exten => s,2,Set(COUNT=1)
exten => s,3,While($[ ${COUNT} < 13 ])
exten => s,5,Read(telef,beep,6)
exten => s,6,NoOp(telefono: ${telef})
exten => s,7,Set(telefono=${ODBC_select(${telef})})
exten => s,8,GotoIf($["${telefono}foo" = "foo"]?notfound)
exten => s,10,Hangup()
exten => s,11(notfound),Set(ODBC_list_insert()=${telef})
exten => s,13,Set(COUNT=${COUNT}+1)
exten => s,14,EndWhile( )
exten => s,15,WaitExten(7)
exten => 1,1,Goto(crear_lista,s,3)
exten => 4,1,Goto(ivr_tesis,s,2)
exten => s,16,Hangup()
```

Resultado de la prueba:

Evaluación: Satisfactoria.

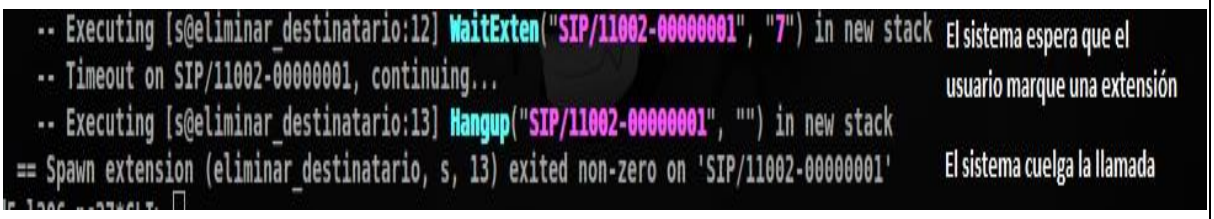
Tabla 20. CP Verificar el tiempo máximo de inactividad.

Caso de Prueba unitaria #1
Nombre: Verificar el tiempo máximo de inactividad.
Probador: Rosalina Sayú Savón.
Descripción/Pasos: Prueba para determinar que el sistema cuelgue la llamada cuando no se ha interactuado en un tiempo determinado.

Código al que se aplica:
`exten => s,8,WaitExten(7)`
`exten => 1,1,Goto(crear_lista,s,1)`
`exten => 2,1,Goto(eliminar_destinatario,s,1)`
`exten => 3,1,Goto(grabar_mensaje,s,1)`
`exten => 4,1,Goto(ivr_tesis,s,2)`
`;exten => t,1,goto(ivr_tesis,s,2)`

```
exten => h,1,Hangup
```

Resultado de la prueba:



Evaluación: Satisfactoria.

Tabla 21. Cantidad de pruebas unitarias realizadas.

Cantidad de pruebas realizadas	Cantidad de pruebas satisfactorias	Cantidad de errores encontrados	Cantidad de errores corregidos
15	**10**	**5**	**5**

Los errores se corrigieron a tiempo por lo que las pruebas arrojaron un resultado satisfactorio.

4.2.2. Pruebas de aceptación

Las pruebas de aceptación son creadas en base a las historias de usuarios. Una historia de usuario no se puede considerar terminada hasta tanto pase correctamente todas las pruebas de aceptación. La misma es ejecutada antes de que la aplicación sea instalada dentro de un ambiente de producción, se realizan documentos de planes de prueba de aceptación y especificación de los mismos, basados en los criterios de aceptación del cliente [54].

Esta prueba es la determinada por el cliente, el cual comprueba que las funcionalidades desarrolladas cumplan con todos los requisitos propuestos. Luego de ser superada esta prueba el sistema estará apto para su uso.

Se realizaron en correspondencia con el diseño de los casos de prueba teniendo en cuenta el orden establecido en las historias de usuario y la prioridad asignada en dependencia de la necesidad de desarrollo de la funcionalidad.

Las pruebas de aceptación propuestas a realizarse se encuentran divididas en las siguientes secciones para una mayor organización:

- Código Caso de Prueba: será el identificador de la prueba a la cual se hace referencia.
- Nombre Historia de Usuario: contendrá el nombre de la Historia de Usuario a la cual hace referencia.

- Nombre de la persona que realiza la prueba: nombre del miembro del equipo que realiza la prueba.
- Descripción de la prueba: se realiza una breve descripción de la funcionalidad que se desea probar.
- Condiciones de ejecución: describe las condiciones que debe cumplirse para realizar la prueba.
- Entradas/Paso de Ejecución: se describen cada uno de los pasos llevados a cabo durante el desarrollo de la prueba, teniendo en cuenta las entradas realizadas por el usuario, teniendo como propósito fundamental, observar si se tiene el resultado esperado.
- Resultado Esperado: se realiza una breve descripción del resultado esperado que se obtiene de la prueba realizada.
- Evaluación de la prueba:
    - Satisfactoria: cuando el resultado obtenido es exactamente el esperado por el cliente.
    - Parcialmente satisfactoria: cuando el resultado obtenido no es óptimo o esperado por el cliente.
    - Insatisfactoria: cuando el resultado genera un error de codificación en la aplicación [55].

A continuación se muestran algunos de los Casos de Prueba de aceptación correspondientes, los demás se encuentran anexados, específicamente en el **Anexo 3**.

Tabla 22. CP Añadir lista de destinatario.

Caso de Prueba de aceptación	
Código Caso de Prueba: SIGLIMT-01-01	**Nombre Historia de Usuario:** Añadir lista de destinatario.
Nombre de la persona que realiza la prueba: Rosalina Sayú Savón	
Descripción de la prueba: Adicionar una lista de destinatario para comprobar que es insertada correctamente en la aplicación web.	
Condiciones de ejecución: La aplicación web debe estar disponible y el servidor *Asterisk* debe estar activo.	
Entradas/Paso de Ejecución: El usuario accede al SGLT, escoge la opción para adicionar las listas y luego completas los formularios correspondientes.	
Resultado Esperado: Se adiciona la lista de destinatario.	

Evaluación de la prueba: Satisfactoria.

Tabla 23. CP Añadir lista de destinatario.

Caso de Prueba de aceptación	
Código Caso de Prueba: SIGLIMT-01-02	**Nombre Historia de Usuario:** Añadir lista de destinatario.
Nombre de la persona que realiza la prueba: Rosalina Sayú Savón	
Descripción de la prueba: Adicionar una lista de destinatario para comprobar que es insertada correctamente en la base de datos.	
Condiciones de ejecución: La aplicación web debe estar disponible, el servidor *Asterisk* debe estar activo y debe de existir un teléfono disponible.	
Entradas/Paso de Ejecución: El usuario luego de seleccionar el servicio y la opción 1, introduce los números telefónicos.	
Resultado Esperado: Se adiciona la lista de destinatario a la Base de Datos.	
Evaluación de la prueba: Satisfactoria.	

Tabla 24. Gestionar destinatario.

Caso de Prueba de aceptación	
Código Caso de Prueba: SIGLIMT-02-01	**Nombre Historia de Usuario:** Autenticar.
Nombre de la persona que realiza la prueba: Rosalina Sayú Savón	
Descripción de la prueba: Introducir los datos para gestionar el destinatario.	
Condiciones de ejecución: La aplicación web debe estar disponible y el servidor *Asterisk* debe estar activo.	
Entradas/Paso de Ejecución: El usuario después de autenticado debe escoger la opción eliminar o editar el destinatario seleccionado.	
Resultado Esperado: El sistema muestra los datos modificados.	
Evaluación de la prueba: Satisfactoria.	

4.2.3. Resultado de las pruebas realizadas

Los resultados obtenidos de las pruebas se representan en la siguiente gráfica, donde se obtuvo un total de 20 no conformidades, en la cual 10 fueron resueltas, 10 no procedían y no quedo ninguna pendiente por resolver.

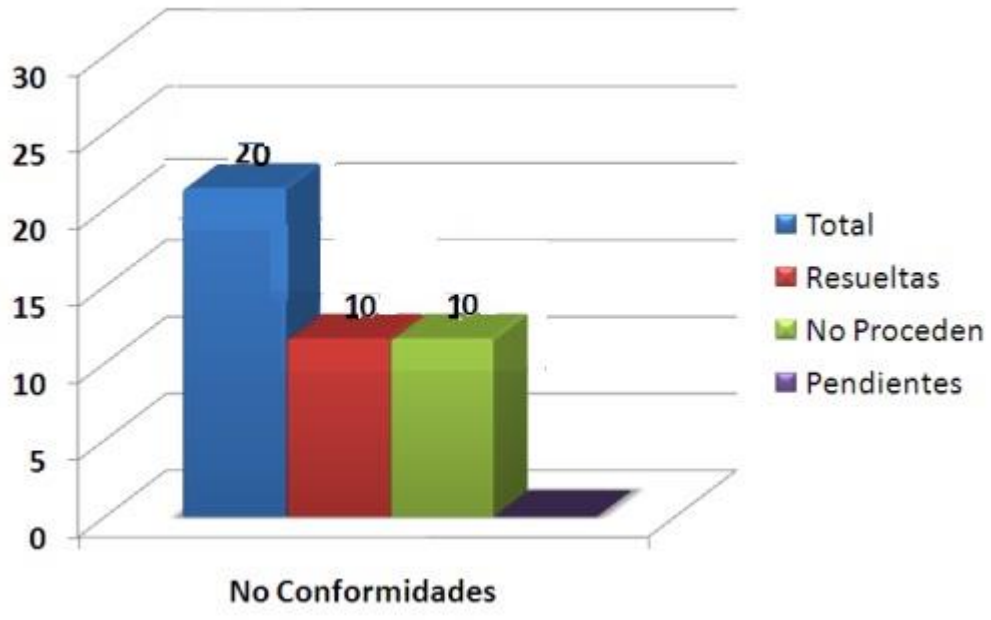


Figura 12. Resultado de las pruebas de aceptación.

4.3. Conclusiones del capítulo

En este capítulo se presentaron las tareas de ingeniería, las mismas favorecieron la representación de las tareas realizadas para implementar las funcionalidades. Durante la fase de implementación se realizaron las pruebas unitarias con las que se verificó el correcto funcionamiento del código, luego de la etapa de implementación se dio paso a crear el diseño de los casos de las pruebas de aceptación, que permitan evaluar el correcto funcionamiento de la aplicación. Con la culminación de este capítulo se finaliza la propuesta de solución del sistema.

CONCLUSIONES

Con la realización del Sistema de Gestión de Listas de Mensajes Telefónicos se determinó que los métodos utilizados pudieron fundamentar y definir las teorías y conceptos de gran importancia que complementan la investigación. La correcta selección de la metodología, herramientas y tecnologías ayudaron cumplir con los objetivos trazados por el equipo de desarrollo. Se implementó una IVR y una aplicación web, dando paso a la solución de la propuesta, se realizaron las tareas de investigación y las pruebas necesarias de forma satisfactoria, con el objetivo de verificar lo implementado y de esta forma eliminar los errores detectados.

La solución aportará a cualquier empresa beneficios como: disminución del tiempo de iteración para enviar mensajes importantes, posibilidad de reducir las operadoras en lugares más importantes donde la empresa lo requiera e introducirá esta entidad al mundo de *software* libre como parte del proceso de migración que lleva a cabo en el país hacia tecnologías libres.

RECOMENDACIONES

Como parte de la investigación realizada se recomienda:

- Continuar desarrollando la investigación a fin de incrementar y perfeccionar las funcionalidades de la herramienta para lograr una mayor adaptación en la empresa.
- Continuar con la investigación de nuevas herramientas informáticas para garantizar mejoras en futuras versiones del sistema.
- Al MICOM, generalizar el sistema de gestión de información que se desarrolló, al resto de sus organizaciones en aras de lograr la informatización de sus procesos.

REFERENCIAS BIBLIOGRÁFICAS

1. **Valera, P.** Ciencia y Tecnología. *Ciencia y Tecnología.* [En línea] [Citado el: 3 de Abril de 2014.] http://blog.pucp.edu.pe/item/40109/10-razones-por-las-cuales-las-telecomunicaciones-son-importantes.

2. **Armendáriz Sánchez, Saúl.** Universidad Nacional Autónoma de México. *Biblioteca del CICH.* [En línea] [Citado el: 3 de Abril de 2014.] http://www.dgbiblio.unam.mx/servicios/dgb/publicdgb/bole/fulltext/volVII3/redes.html.

3. **García Abert, Franks.** *Automatización de la obtención de la factura de los abonados.* Universidad de las Ciencias Informáticas. La Habana, Cuba : s.n., 2011. pág. 87, Tesis.

4. **Carballo Penton, Eufemia Sandra.** Ecured. *Ecured.* [En línea] 24 de Noviembre de 2013. [Citado el: 5 de Febrero de 2014.] http://www.ecured.cu/index.php/Teléfono.

5. **Server, Exchange.** Sitio oficial de Exchange. *Sitio oficial de Exchange.* [En línea] 22 de Abril de 2013. [Citado el: 1 de Abril de 2014.] http://technet.microsoft.com/es-es/library/ee633462.aspx.

6. **WordReference.** WordReference/Online Language Dictionaries. *WordReference/Online Language Dictionaries.* [En línea] [Citado el: 1 de Abril de 2014.] http://www.wordreference.com/definicion/mensajedevoz.

7. **Mas Cmacho, Rosa Maria y Febles Rodriguez, Juan Pedro.** Experiencias de la aplicación de la ingeniería en sistemas de gestión. *Revista Cubana de Informática Medica.* [En línea] [Citado el: 1 de Abril de 2014.] www.cecam.sld.cu/pages/rcim/revista_1/articulos_pdf/r0100a01.pdf.

8. **RODRIGUEZ Acosta, Dallana y ALVAREZ Maquerira, Sady.** *MoodleMin: Módulo de apoyo al proceso de instalación y configuración de la plataforma de telecomunicación Moodle.* Ingeniero en Ciencias Informáticas. La Habana, Cuba : s.n., 2013. pág. 58.

9. **Salazar Torrero Lirizandra, González Rodríguez Raúl.** *Solución informática para el portal web ¨Preparación para la Defensa Facultad #4.* Universidad de Ciencias Informáticas. La Habana Cuba : s.n., 2012. pág. 95.

10. **LibrosWeb.** *CSS avanzado.* Barcelona, España : s.n., 2014.

11. **Gómez, Edgar J.** Edgar J. Gómez. Servicios Informáticos. *Edgar J. Gómez. Servicios Informáticos.* [En línea] 15 de Mayo de 2013. [Citado el: 8 de Junio de 2014.] http://edgargomez.es/que-es-un-framework/.

12. **Fabien Potencier, François Zaninotto.** *Symfony, La guía definitiva.* 2008. pág. 425.

13. **Anónimo.** *VP-UML User's Guide.* pág. 1277.

14. **Ecured.** *Visual Paradigm.*

15. **Larman, Craig.** *UML y Patrones. Introducción al análisis y diseño orientado a objeto.* PRENTICE HALL, MCxico : s.n., 1999 . pág. 536.

16. **Ecured.** *Lenguaje UML.*

17. **Gauchat, Juan Diego.** *El gran libro de HTML5, CSS3 y Javascript.* pág. 350.

18. **Curso de JavaScript**. [En línea] [Citado el: 8 de Junio de 2014.]

19. ***Curso de JavaScript.*** documento word.

20. **Ojeda, Antonio Navajas**. *OpenLibra."La Biblioteca Libre online que estabas esperando".* s.l. : Autoedición, 2012. pág. 63.

21. **PHP, El grupo de. Manual de PHP**. [En línea] 2014. [Citado el: 8 de Junio de 2014.] http://www.php.net/manual/es/preface.php.

22. **NetBeans, Comunidad del. NetBeans IDE**. [En línea] 7 de Junio de 2013. [Citado el: 2014 de Junio de 9.] https://netbeans.org/community/releases/73/relnotes.html.

23. **Joanpaon. wordpress**. [En línea] 21 de Mayo de 2013. [Citado el: 9 de Junio de 2014.] http://joanpaon.wordpress.com/2013/05/21/instalacion-de-netbeans-7-3/.

24. **Bravo, Indira Martinez**. INFORMATICA.Trabajos Investigativos sobre Sistemas,Softwares,Tecnología Informática entre otros. [En línea] [Citado el: 9 de Junio de 2014.] http://indira-informatica.blogspot.com/2007/09/qu-es-un-sistema-de-gestin-de-base-de.html.

25. **Congalves, Flavio E**. *Asterisk PBX. Guía de la configuración.* Janeiro : s.n., 2007. pág. 362.

26. **Kris Sheets, Jimmy Terence Estrada,Marcelo Garcia,Dany Saavedra**. *Asterisk en Español* . 2005.

27. **Santamaria, Loris**. *Introducción aAsterisk Introducción a Asterisk.*

28. **López, Ernerto MatIas**. *Diseño y configuracion del IVR.* 2010. pág. 89.

29. **Jiménez, Ramón**. Mi experiencia plasmada. [En línea] 20 de Febrero de 2011. [Citado el: 10 de Junio de 2014.] http://jimenezra.blogspot.com/2011/02/telefonia-ip-asterisk-sipconf.html.

30. **ALEGSA**. Diccionario de informática. [En línea] 1998 - 2014. [Citado el: 11 de Junio de 2014.] http://www.alegsa.com.ar/Dic/apache.php.

31. **Foundation:, The Apache Software**. Apache HTTP Server for Windows. [En línea] [Citado el: 11 de Junio de 2013.] http://descargar.cnet.com/Apache-HTTP-Server-for-Windows/3000-2247_4-10803652.html.

32. **DIAZ Espinoza, Yasmin y SORI Echerri Ricardo, Ernesto.** *Sistema de gestión de encuestas telefónicas a través de Respuesta de Voz Iterativa.* Universidad de las Ciencias Informáticas. La Habana, Cuba : s.n., 2011. pág. 67, Tesis.

33. **NEGRET Pons, Yudith**. *Desarrollo del proceso Registro de Sucursales Extranjeras del módulo Registral del proyecto Sistema de Información Registral de la Cámara de Comercio.* Universidad de las Ciencias Informáticas. La Habana, Cuba : s.n., 2012. pág. 67.

34. **Bass, Clement y Kazman**. Pressman_10_Diseño_Arquitectónico_Parte_1. *Pressman_6ta_edición.*

35. **Pressman**. Estilos y Patrones arquitectónicos (epigrafe 10.3). *Pressman_6ta_Edición.*

36. **Sommerville_7ma_Edición**. *Sommerville_Parte_III_Diseño.*

37. **Rodríguez Pons, Felipe**. *Desarrollo del proceso de la Cámara de Comercio.* Universidad de las Ciencias Informáticas. La Habana, Cuba : s.n., 2011. pág. 54.

38. **Wells, J. Donvan**. Ciclo de vida de un proyecto XP. [En línea] 2009. [Citado el: 11 de Junio de 2014.] http://oness.sourceforge.net/proyecto/html/ch05s02.html.

39. **Patricio Letelier, Mª Carmen Penadés**. *Métodologías ágiles para el desarrollo de software:*. Valencia : s.n., 2010. pág. 17.

40. **Peña, Raúl Suares**. *Creación de un IVR en tiempo real.* Universidad de México. México : s.n., 2011. pág. 54.

41. **Artola, Luis**. Cuaderno de software. [En línea] 5 de Junio de 2006. [Citado el: 12 de Junio de 2014.] http://www.programania.net/.

42. **Ruiz, Armando Alvarez**. *Desarrollo del inventario de llamadas telefónicas.* Universidad de las Ciencias Informáticas. La Habana, Cuba : s.n., 2011. pág. 54, Tesis.

43. ***Tipos de prueba de aceptación***. La Habana, Cuba : s.n.

44. **Ecured**. [En línea] [Citado el: 30 de Marzo de 2014.] http://www.ecured.cu/index.php/Metodolog%C3%ADas_de_desarrollo_de_software.

45. —. [En línea] [Citado el: 16 de Febrero de 2014.] http://www.ecured.cu/index.php/Herramienta_CASE.

46. **voztovoice**. [En línea] 20 de Octubre de 2008. [Citado el: 10 de Junio de 2014.] https://voztovoice.org/?q=node/50.

47. **procesosoftware**. [En línea] 2014. [Citado el: 2014 de Junio de 12.] http://procesosoftware.wikispaces.com/Metodologíagil+XP.

48. **Ecured.** [En línea] [Citado el: 14 de Febrero de 2014.] http://www.ecured.cu/index.php/Metodologías_de_desarrollo_de_software.

49. **Álvarez, Miguel Angel**. Desarrolloweb. [En línea] 23 de Noviembre de 2009. [Citado el: 22 de Junio de 2014.] http://www.desarrolloweb.com/articulos/codeigniter.html.

50. **LLC, Yii Software**. Yiiframework. [En línea] 2014. [Citado el: 22 de junio de 2014.] http://www.yiiframework.com/doc/guide/1.1/es/quickstart.what-is-yii.

51. **CallFire**. [En línea] 2014. [Citado el: 22 de Junio de 2014.] https://www.callfire.com.

52. **Ifbyphone**. [En línea] 2014. [Citado el: 22 de Junio de 2014.] public.ifbyphone.com.

53. **DialMyCalls**. [En línea] 2014. [Citado el: 22 de Junio de 2014.] www.dialmycalls.com.

54. Informáticas, Universidad de las Ciencias. Investigaciones. [En línea] 25 de Noviembre de 2013. [Citado el: 23 de Junio de 2014.] investigaciones.uci.cu.

GLOSARIO DE TÉRMINOS

PBX: (*Private Branch Exchange*) es una central telefónica privada que permite interconectar los teléfonos internos de una empresa y seleccionar la línea saliente de forma automática. Permiten transferir llamadas, realizar conferencias y llevar un control de los números marcados. Es usado por empresas grandes y pequeñas para reducir costos [37].

Call Center: es una unidad funcional dentro de una empresa (o puede ser la misma empresa) diseñada para manejar grandes volúmenes de llamadas telefónicas entrantes y salientes desde y hacia sus clientes, con el fin de dar soporte a las operaciones cotidianas de la entidad.

Platel: software para centros de contactos basados en VoIP.

Dialplan: siglas correspondiente al plan de marcado de una plataforma telefónica, en Asterisk a la particularidad de edición del fichero de configuración extensión.conf.

GPL: la *GNU Genaral Public License* (Licencia Publica General) es una licencia creada por Free Software Foundation y orientada principalmente a los términos de distribución, modificación y uso del software Su propósito es declarar que el software cubierto por licencia en Software Libre.

ODBC: es un estándar de acceso a Base de Datos desarrollado por *Microsoft Corporation*, con el objetivo de acceder a cualquier dato desde cualquier aplicación.

VoIP: *Voice over Internet Protocol*, Voz sobre Protocolo de Internet. Es un grupo de recursos que hacen posible que la señal de voz viaje a través de Internet empleando un protocolo IP (Protocolo de Internet).

VoiceMail: es un programa informático conjuntamente con una contestadora automática que permite el envío de mensajes de voz hacia los buzones personales.

Iteración: se refiere a la acción de repetir una serie de pasos un cierto número de veces.

Softphone: (Software telefónico) hace una simulación de un teléfono convencional por computadora

Subsistema: conjunto de elementos o funciones relacionados que completan un sistema.

Bundle: Directorio que aloja todo aquello relativo a una funcionalidad determinada. Puede incluir clases PHP, plantillas, configuraciones, JavaScript y CSS.

ANEXOS

Anexo 1

Tabla 25. Tarjeta CRC: ListaController

Clase: ListaController	
Responsabilidad	**Colaboración**
indexAction() createAction(Request $request) createCreateForm(Lista $entity) newAction() showAction($id) editAction($id) createEditForm (Lista $entity) updateAction(Request $request, $id) deleteAction(Request $request, $id) createDeleteForm($id)	

Tabla 26. Tarjeta CRC: PersonController

Clase: PersonController	
Responsabilidad	**Colaboración**
indexAction() createAction(Request $request) createCreateForm(Person $entity) newAction() showAction($id)	

editAction($id) createEditForm(Person $entity) updateAction(Request $request, $id) deleteAction(Request $request, $id) createDeleteForm($id)	

Tabla 27. Tarjeta CRC: UserController

Clase: UserController	
Responsabilidad	**Colaboración**
indexAction() createAction(Request $request) createCreateForm(User $entity) userRegisterAction(Request $request) registerAction() createRegisterForm(User $entity) newAction() showAction($id) editAction($id) createEditForm(User $entity) updateAction(Request $request, $id) deleteAction(Request $request, $id) createDeleteForm($id)	

Tabla 28. Tarjeta CRC: SecurityController

Clase: SecurityController	
Responsabilidad	**Colaboración**
loginAction() chpasswordAction()	

Anexo 2

Tabla 29. Tarea #2 de la Historia de Usuario: Gestionar destinatario.

Tarea de ingeniería	
Número Tarea: 2	**Número Historia de Usuario:** 2
Nombre Tarea: Implementar la funcionalidad Gestionar destinatario.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 1
Fecha Inicio: 14/4/2014	**Fecha Fin:** 15/4/2014
Programador Responsable: Rosalina Sayú Savón	
Descripción: Se implementó la funcionalidad para gestionar destinatarios.	

Tabla 30. Tarea #1 de la Historia de Usuario: Autenticar.

Tarea de ingeniería	
Número Tarea: 1	**Número Historia de Usuario:** 3
Nombre Tarea: Diseño de la interfaz Autenticar.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 1
Fecha Inicio: 17/4/2014	**Fecha Fin:** 18/4/2014
Programador Responsable: Rosalina Sayú Savón	
Descripción: Se diseñó la interfaz para autenticar los destinatarios.	

Tabla 31. Tarea #2 de la Historia de Usuario: Autenticar.

Tarea de ingeniería	
Número Tarea: 2	**Número Historia de Usuario: 3**
Nombre Tarea: Implementar la funcionalidad Autenticar.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 1
Fecha Inicio: 17/4/2014	**Fecha Fin:** 18/4/2014

Programador Responsable: Rosalina Sayú Savón
Descripción: Se implementó la funcionalidad autenticar.

Tabla 32. Tarea #1 de la Historia de Usuario Eliminar listas de destinatario.

Tarea de ingeniería	
Número Tarea: 1	**Número Historia de Usuario:** 4
Nombre Tarea: Diseño de la interfaz Eliminar listas de destinatario.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 2
Fecha Inicio: 20/4/2014	**Fecha Fin:** 22/4/2014
Programador Responsable: Rosalina Sayú Savón	
Descripción: Se diseñó la interfaz para eliminar destinatarios.	

Tabla 33. Tarea #2 de la Historia de Usuario Eliminar listas de destinatario.

Tarea de ingeniería	
Número Tarea: 2	**Número Historia de Usuario:** 4
Nombre Tarea: Implementar la funcionalidad Eliminar listas de destinatario.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 2
Fecha Inicio: 20/4/2014	**Fecha Fin:** 22/4/2014
Programador Responsable: Rosalina Sayú Savón	
Descripción: Se implementó la funcionalidad Eliminar destinatario.	

4.3.1. Iteración #2

En esta iteración se implementaron las historias de usuario referentes a las principales funcionalidades del sistema IVR con el objetivo de mostrarle al usuario el segundo prototipo funcional del SIGLIMT.

4.3.2. Tareas de las Historias de Usuario implementadas en la iteración #2.

Tabla 34. Tiempo de las tareas abordadas en la iteración #2.

Historia de usuario	Estimación(días)	Real(días)
Grabar mensajes de voz.	2.5	2.5
Enviar mensaje de voz	2.5	2.5
Total	**5**	**5**

Tabla 35. Tarea #1 de la Historia de Usuario Grabar mensajes de voz.

Tarea de ingeniería	
Número Tarea: 1	**Número Historia de Usuario:** 5
Nombre Tarea: Implementar la funcionalidad Grabar mensajes de voz.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 2.5
Fecha Inicio: 1/5/201	**Fecha Fin:** 4/5/2014
Programador Responsable: Rosalina Sayú Savón	
Descripción: Se implementó la funcionalidad Grabar mensajes de voz.	

Tabla 36. Tarea #2 de la Historia de Usuario Grabar mensajes de voz.

Tarea de ingeniería	
Número Tarea: 2	**Número Historia de Usuario:** 5
Nombre Tarea: Almacenar el mensaje grabado.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 2.5
Fecha Inicio: 6/5/2014	**Fecha Fin:** 10/5/2014
Programador Responsable: Rosalina Sayú Savón	
Descripción: Se implementó la funcionalidad almacenar datos grabados.	

Tabla 37. Tarea #1 de la Historia de Usuario Enviar mensaje de voz.

Tarea de ingeniería	
Número Tarea: 1	**Número Historia de Usuario:** 5
Nombre Tarea: Implementar la funcionalidad Enviar mensajes de voz.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 2.5
Fecha Inicio: 12/5/2014	**Fecha Fin:** 15/5/2014
Programador Responsable: Rosalina Sayú Savón	
Descripción: Se implementó la funcionalidad de enviar mensajes de voz.	

4.3.3. Iteración #3.

En esta iteración se implementará la historia de usuario encargada de complementar el resultado de lo implementado en las iteraciones anteriores con el objetivo de mostrar al cliente el producto final.

4.3.4. Tareas de las Historias de Usuario implementadas en la iteración #3.

Tabla 38. Tiempo de las tareas abordadas en la iteración #3.

Historia de usuario	Estimación(días)	Real(días)
Integración Web- IVR	1	1
Total	**1**	1

Tabla 39. Tarea #1 de la Historia de Usuario Integración Web- IVR

Tarea de ingeniería	
Número Tarea: 1	**Número Historia de Usuario:** 5
Nombre Tarea: Implementar la funcionalidad que permita integrar la aplicación web con la IVR.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 1
Fecha Inicio:20/5/2014	**Fecha Fin:**21/5/2014
Programador Responsable: Rosalina Sayú Savón	
Descripción: Se integró la aplicación web con el IVR.	

Anexo 3

Tabla 1. Gestionar destinatario.

Caso de Prueba de aceptación	
Código Caso de Prueba: SIGLIMT-02-02	**Nombre Historia de Usuario:** Autenticar.
Nombre de la persona que realiza la prueba: Rosalina Sayú Savón	
Descripción de la prueba: Introducir el número telefónico para eliminar el destinatario.	
Condiciones de ejecución: La aplicación web debe estar disponible, el servidor *Asterisk* debe estar activo y debe de existir un teléfono disponible.	
Entradas/Paso de Ejecución: El usuario luego de seleccionar el servicio y la opción 2, introduce el número telefónico a eliminar.	
Resultado Esperado: El sistema elimina el destinatario de la Base de Datos.	
Evaluación de la prueba: Satisfactoria.	

Tabla 2. CP Autenticar.

Caso de Prueba de aceptación	
Código Caso de Prueba: SIGLIMT-03-01	**Nombre Historia de Usuario:** Autenticar.
Nombre de la persona que realiza la prueba: Rosalina Sayú Savón	

Descripción de la prueba: Introducir los datos para la autenticación en el SGLT
Condiciones de ejecución: La aplicación web debe estar disponible y el servidor *Asterisk* debe estar activo.
Entradas/Paso de Ejecución: El usuario escribe su número telefónico y contraseña en el formulario de autenticación en la web, el sistema valida los datos luego de mostrar la interfaz de administración si los mismos son válidos.
Resultado Esperado: El usuario accede al sistema si los datos son válidos.
Evaluación de la prueba: Satisfactoria.

Tabla 3. CP Eliminar lista de destinatarios.

Caso de Prueba de aceptación	
Código Caso de Prueba: SIGLIMT-04-01	**Nombre Historia de Usuario:** Eliminar lista de distribución.
Nombre de la persona que realiza la prueba: Rosalina Sayú Savón	
Descripción de la prueba: Comprobar que se eliminan las listas de destinatarios	
Condiciones de ejecución: La aplicación web debe estar disponible y el servidor *Asterisk* debe estar activo.	
Entradas/Paso de Ejecución: Luego de autenticado, el usuario deberá escoger la opción para eliminar la lista.	
Resultado Esperado: La lista de destinatarios es eliminada del sistema.	
Evaluación de la prueba: Satisfactoria.	

Tabla 4. CP Eliminar lista de destinatarios.

Caso de Prueba de aceptación	
Código Caso de Prueba: SIGLIMT-04-01	**Nombre Historia de Usuario:** Eliminar lista de distribución.
Nombre de la persona que realiza la prueba: Rosalina Sayú Savón	
Descripción de la prueba: Comprobar que se eliminan las listas de destinatarios.	
Condiciones de ejecución: La aplicación web debe estar disponible y el servidor *Asterisk* debe estar activo.	
Entradas/Paso de Ejecución: Luego de autenticado, el usuario deberá escoger la opción para eliminar la lista.	
Resultado Esperado: La lista de destinatarios es eliminada del sistema.	
Evaluación de la prueba: Satisfactoria.	

Tabla 5. CP Grabar mensajes de voz.

Caso de Prueba de aceptación	
Código Caso de Prueba: SIGLIMT-05-01	**Nombre Historia de Usuario:** Grabar mensaje de voz.
Nombre de la persona que realiza la prueba: Rosalina Sayú Savón	
Descripción de la prueba: Comprobar que el mensaje sea grabado.	
Condiciones de ejecución La aplicación web debe estar disponible, el servidor *Asterisk* debe estar activo y debe de existir un teléfono disponible.	
Entradas/Paso de Ejecución: Luego de seleccionar el servicio y la opción 3 el usuario pasa a grabar el mensaje de voz.	
Resultado Esperado: El sistema guarda los mensajes con un nombre y una extensión.	
Evaluación de la prueba: Satisfactoria.	