

Universidad de las Ciencias Informáticas

Facultad 2



Título: Generador Gráfico de IVR

Trabajo de Diploma para optar por el título de
Ingeniero en Ciencias Informáticas

Autores: Irina Valdes Meneses

Mario Iván Cid Vázquez

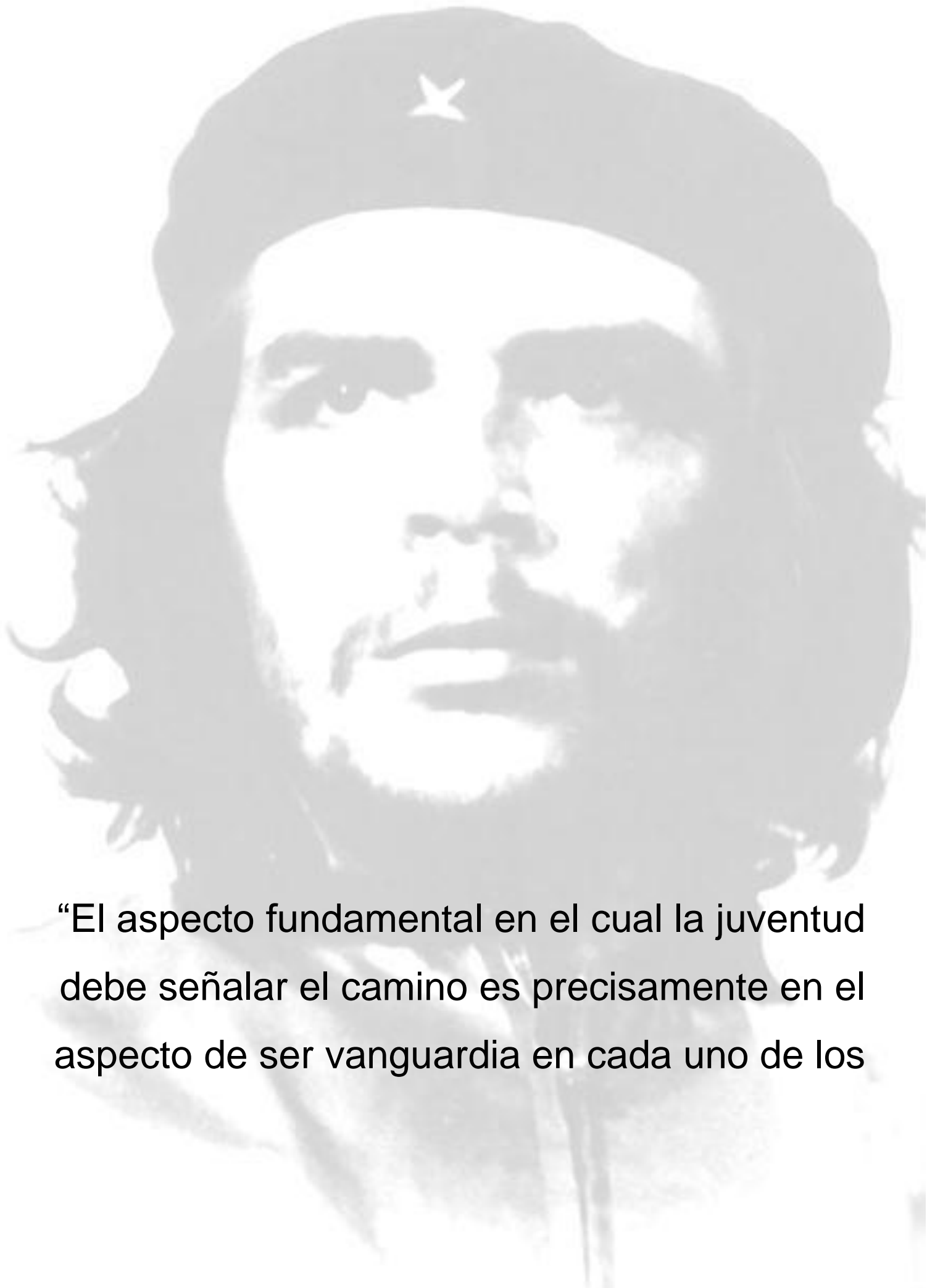
Tutor: Ing. David Rodríguez Rodríguez

Co-tutor: Ing. Yanerys Gourrié Fernández

Ciudad de La Habana, Cuba

“Año 54 de la Revolución”

Junio, 2012



“El aspecto fundamental en el cual la juventud debe señalar el camino es precisamente en el aspecto de ser vanguardia en cada uno de los

trabajos que le compete.”

Che

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo al centro de telemática perteneciente a la facultad 2 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Irina Valdes Meneses

Firma de Autor

Mario Iván Cid Vázquez

Firma de Autor

Ing. David Rodríguez Rodríguez

Firma del Tutor

DATOS DE CONTACTO

Tutor:

Nombre y Apellidos: Ing. David Rodríguez Rodríguez.

Sexo: M. **Institución:** UCI.

Dirección de la institución: Carretera a San Antonio de los Baños, km 2 ½, Boyeros, Ciudad de La Habana.

Correo electrónico: drrodriguez@uci.cu **Teléfono del trabajo:** 8378143

Título de la especialidad de graduado: Ingeniero en Ciencias Informáticas.

Año de graduación: 2008.

Institución donde se graduó: Universidad de las Ciencias Informáticas.

Co - tutor:

Nombre y Apellidos: Ing. Yanerys Gourrié Fernández.

Sexo: F. **Institución:** UCI.

Dirección de la institución: Carretera a San Antonio de los Baños, km 2 ½, Boyeros, Ciudad de La Habana.

Correo electrónico: ygourrie@uci.cu **Teléfono del trabajo:** 8373768

Título de la especialidad de graduado: Ingeniero en Ciencias Informáticas.

Año de graduación: 2010.

Institución donde se graduó: Universidad de las Ciencias Informáticas.

DEDICATORIA

Irina

El Presente trabajo se lo dedico en especial a mi abuela y a mis padres quienes han estado ahí para apoyarme en los buenos y malos momentos, quienes me han dado amor, dedicación y entrega, quienes me han enseñado a ser mejor persona cada día, a luchar por mis sueños ,a no darme por vencida nunca. Gracias

Mario

A mis padres, a mi hermano, a mi novia y a mis amigos.

AGRADECIMIENTOS

Irina

A mi mamá por el amor incondicional que me ha dado, por saberme guiar, por darme su ejemplo y su protección, por confiar en mí, por ser mi amiga y por su inagotable paciencia.

A mi papá por darme su amor, su sacrificio, su entrega, su apoyo, por saber que puedo contar con él cuando lo necesite. Gracias por estar ahí siempre para mí.

A mi abuela por ser mi estrella, por darme su eterno amor y su experiencia.

A mi hermano que ser mi soporte y mi vida.

A mi novio Reinier y su familia por cuidar de mí, darme cariño y apoyo.

A mi amigos Sacha, Adis, Raiko, Alejandro, Frank, Ricardo, Eduardo, Yenia, Rafael, que han compartido conmigo todos los momentos tanto buenos como malos de la universidad y han quedado grabados por siempre en mi corazón.

A mis compañeros de apartamento por ayudarme y brindarme su cariño y paciencia.

A todos mis profesores en especial a los que de una forma dejaron una huella en mí como profesionales y como amigos, Adrián, Lester, Maidelis, Maida, Aimé, Rosa, David

AGRADECIMIENTOS

Mario

A mis padres Ramona y Guillermo, por todo el amor y la confianza, por educarme para convertirme en la persona que soy y sobre todo por estar siempre ahí para apoyarme.

A mi hermano Marco, por el ejemplo, por estar siempre junto a mí todos estos años y por el cariño que siempre compartimos.

A mi tío Tato por apoyarme y estar siempre a mi lado en las buenas y en las malas.

A mi novia Elizabeth por todo el amor, por hacerme el hombre más feliz del mundo, te amo.

A mis suegros Hector e Isabel por acogerme en su familia.

A mis amigos Jorgito y Elizabeth por acogerme como hijo adoptivo desde el inicio, a Jorge por entenderme y empujarme siempre hacia adelante, a Janier por mostrarme que una Universidad de Informática no es solo acerca de informática, a Alberto por los momentos que compartimos juntos, a Gregorio por su amistad.

A todos mis amigos del 16, Reinaldo, Reinier, Yosbany, Firi, Rafa, Neyke, Yury...

A la gente de la ACM en especial a Alkaid y Char por el DreamTeam.

A todos los que de una forma u otra han contribuido a realizar este sueño.

RESUMEN

El trabajo con tecnología IVR (Respuesta de Voz Interactiva) se ha intensificado en el mundo de las telecomunicaciones. Consiste en un sistema telefónico que es capaz de recibir una llamada e interactuar con una persona a través de grabaciones de voz y el reconocimiento de respuestas simples. Utilizar tecnología IVR ayuda a reducir los turnos de los operadores, los costes asociados, el tiempo de espera de los clientes y la tasa de llamadas perdidas. Este sistema permite la disponibilidad de la información las veinticuatro horas del día y el manejo de un gran número de llamadas. La Línea de Desarrollo de Servicios para Call Center perteneciente al Centro de Telemática de la Facultad 2 requiere constantemente crear IVR. Este trabajo se vuelve cada vez más engorroso y complejo, ya que se hace introduciendo código en un fichero de texto, sin que medie una interfaz gráfica. Esta dificultad provoca que se dedique esfuerzo y tiempo por parte de los desarrolladores, necesitando conocer la lógica de interacción de una IVR con el usuario y la lógica de programación, así como el lenguaje en el que se va a implementar. Para hacer mucho más sencillo el trabajo con esta tecnología se desarrolló una herramienta capaz de generar el código de una IVR a través del uso de componentes visuales, para que este sea interpretado por Asterisk. La misma garantizará la posibilidad de dedicar personal especializado en otras tareas de interés para el proyecto.

Palabras Claves: Asterisk, Componentes Visuales, IVR, Sistema Telefónico.

ÍNDICE

DECLARACIÓN DE AUTORÍA	3
DATOS DE CONTACTO.....	4
DEDICATORIA	5
AGRADECIMIENTOS	6
AGRADECIMIENTOS	7
RESUMEN.....	8
ÍNDICE	9
INTRODUCCIÓN.....	13
CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA	18
Introducción.....	18
1.1. Conceptos Relacionados	18
1.1.1. IVR.....	18
1.1.2. Entorno gráfico de Desarrollo	18
1.1.3. Código Fuente.....	18
1.1.4. Generador de código.....	19
1.1.5. Metodología de desarrollo	19
1.1.6. Asterisk	19
1.1.7. Comunicaciones Unificadas	19
1.2. Análisis de soluciones existentes en el mundo actual	19
1.2.1. Quickfuse	20
1.2.2. CT Developer Studio	20
1.2.3. VBVoice	21
1.2.4. Entorno de Desarrollo IVR (EDI).....	21

1.3. Metodología de Desarrollo	22
1.3.1. Extreme Programming (XP).....	23
1.4. Lenguaje de Modelado	24
1.4.1. IDEF0.....	24
1.5. Lenguaje de Desarrollo.....	25
1.5.1. Java	25
1.6. Librerías.....	25
1.6.1. DockingFrames	25
1.7. Herramientas	26
1.7.1. Dia Project.....	26
1.7.2. NetBeans	26
1.8. Conclusiones	27
CAPÍTULO 2: EXPLORACIÓN Y PROPUESTA DEL SISTEMA	28
Introducción.....	28
2.1. Objeto de automatización	28
2.2. Propuesta del Sistema.....	28
2.3. Modelo del Negocio	31
2.4. Requisitos funcionales del Sistema.....	34
2.5. Requisitos no funcionales del Sistema.....	35
2.5.1. Usabilidad	36
2.5.2. Eficiencia.....	36
2.5.3. Hardware	36
2.5.4. Software.....	36
2.5.5. Interfaz de Usuario	36

2.5.6. Soporte	36
Manual de Usuario	36
Capacitación	37
2.5.7. Restricciones de Diseño e Implementación	37
2.6. Componentes de la IVR	37
2.7. Personas relacionadas con el Sistema	37
2.8. Fase de exploración.....	37
2.9. Planificación	47
2.10. Plan de Iteraciones.....	49
2.10.1. Plan de duración de las Iteraciones	49
2.11. Plan de Entregas.....	51
2.12. Conclusiones.....	51
CAPÍTULO 3: DISEÑO DEL SISTEMA.....	53
Introducción.....	53
3.1. Arquitectura del sistema	53
3.1.1. Patrón de Arquitectura utilizado.....	53
Modelo Vista Controlador (MVC).....	54
3.2. Patrones de Diseño	55
3.2.1. Patrón para Asignar Responsabilidades (GRASP)	55
Patrón Expert	55
Patrón Crator	56
Patrón Controller	56
3.2.2. Patrón Visitor.....	56
3.2.3. Patrón Observer.....	57

3.2.4. Patrón Singleton.....	57
3.3. Diagrama de Paquetes o Subsistemas de Diseño	57
3.4. Tarjetas CRC.....	61
3.5. Conclusiones.....	63
CAPÍTULO 4: IMPLEMENTACIÓN Y PRUEBA	64
Introducción.....	64
4.1. Tareas de la Ingeniería	64
4.2. Pruebas	71
4.3. Conclusiones	73
CONCLUSIONES GENERALES.....	74
RECOMENDACIONES	75
REFERENCIAS BIBLIOGRÁFICAS.....	76
GLOSARIO	79

INTRODUCCIÓN

Desde sus primeros pasos, el hombre ha sentido la necesidad de comunicarse ya sea por señales visuales o acústicas. Antiguamente se dificultaba la comunicación a larga distancia, por lo que se otorgó singular importancia a las telecomunicaciones, cuyo origen se remontan a la primera mitad del siglo XIX con la invención del telégrafo eléctrico, el cual permitió enviar mensajes cuyo contenido eran letras y números. Más tarde se desarrolló el teléfono, con el que fue posible comunicarse utilizando la voz, y posteriormente, las ondas de radio que dieron lugar a la revolución de la comunicación inalámbrica. En 1936 se inició un proyecto de síntesis de voz en los Laboratorios Bell, con la confección de un dispositivo llamado “el Voder” y en los años 60 se comenzó a utilizar las telecomunicaciones en el campo de la informática. La siguiente década se caracterizó fundamentalmente por la aparición de las redes de computadoras, los protocolos y arquitecturas, surgiendo la ARPANET, que dio origen a la Internet. Ya a finales de los años setenta surgen las redes de área local o LAN (del inglés Local Area Network). (1)

Hoy la necesidad de comunicarse es inminente, grandes industrias y empresas de las telecomunicaciones de todo el mundo se enfrentan a la competencia y a la lucha por su prevalencia en el mercado y requieren de grandes sistemas de comunicación que les facilite su trabajo y minimice los costos. Ya no es suficiente comunicarse dentro de un horario determinado o con un operador de su empresa, ahora se requiere de un sistema moderno que permita mayor satisfacción para los clientes al reducir los tiempos de espera y permitir la disponibilidad de la información las veinticuatro horas del día.

La automatización de determinados servicios, gracias a la tecnología, ayuda en el ahorro de costes y aumenta la eficiencia, ya que el personal tradicionalmente designado a estos, puede ser destinado a gestionar tareas que generan valor añadido. Una alternativa para suplir estas necesidades es el uso de menús de respuesta de voz interactiva, conocidas también por sus siglas en inglés IVR (Interactive Voice Response).

IVR es una tecnología que permite a una computadora interactuar con los seres humanos a través de un teclado del teléfono o por reconocimiento de voz. Los sistemas IVR pueden responder a usuarios con pregrabados o generados de audio dinámicamente, guiándoles sobre cómo proceder. Están dimensionados para manejar grandes volúmenes de llamadas, reduciendo así el tiempo de las últimas y

evitando la necesidad de hacer transferencias entre agentes¹. Las ventajas que proporcionan no residen solamente en la reducción de costes y en el incremento de la eficiencia de un Centro de Llamadas, sino que también ayudan a reducir los turnos de los operadores y sus costes asociados, incrementan las horas de servicio, disminuyen la tasa de llamadas perdidas, incrementan la disponibilidad de los operarios y mejoran la flexibilidad para responder a las necesidades del cliente o a picos de llamadas.

Existen en el mundo muchas herramientas que permiten la creación de IVR y que proporcionan una interfaz gráfica y un fácil manejo de las funcionalidades, posibilitando crearlas en corto tiempo y completamente personalizables. Pero no es menos cierto que la gran mayoría son privativas y por tanto menos accesibles, por lo que normalmente, la principal alternativa consiste en la introducción directa de código en un fichero, lo cual se considera engorroso para el programador.

Estas limitantes también están presentes en la Línea de Desarrollo de Servicios para Call Center perteneciente al Centro de Telemática de la Facultad 2 de la Universidad de las Ciencias Informáticas, donde se desarrollan diferentes sistemas y aplicaciones para potenciar el desarrollo de las comunicaciones en Cuba. Dicha línea destina esfuerzos en la programación de IVR las cuales serán interpretadas por un sistema Asterisk (sistema que simula una planta telefónica), las mismas benefician, entre otros, a la Empresa de Telecomunicaciones de Cuba S.A. Los desarrolladores de este centro dedican parte importante del tiempo en su implementación, labor ardua, si se tiene en cuenta que se realiza manualmente. Esto puede resultar complejo, pues durante su desarrollo hay que considerar la secuencia de eventos de interacción del usuario con la IVR, la lógica de la propia programación y tener conocimiento del lenguaje en que se implementa.

Por todo lo anteriormente expuesto se plantea como **problema** a resolver:

¿Cómo generar el código de una IVR para un sistema Asterisk, en la línea de desarrollo de servicios Call Center, a partir de realizar el diseño visual de la misma?

Se define como **objeto de estudio** para esta investigación:

La generación de código IVR.

¹ Personal que trabaja en telecomunicaciones dedicado a atender a los abonados.

El **campo de acción** de esta investigación es:

La generación de código IVR para Asterisk en la línea de desarrollo de servicios Call Center.

El **objetivo general** que como propósito se aborda en este trabajo es:

Desarrollar una aplicación que permita generar el código fuente de una IVR para un sistema Asterisk en la línea de desarrollo de servicios Call Center.

Como **idea a defender** se propone:

Con la elaboración del Generador Gráfico de IVR se obtendrá como ventaja principal: Crear el código de una IVR a través de componentes visuales, disminuyendo tiempo y complejidad.

Para cumplir con el objetivo propuesto se desarrollan las siguientes **tareas de la investigación**:

- Analizar y estudiar aplicaciones similares existentes en el mundo actual para identificar procesos afines con el sistema a desarrollar.
- Estudiar herramientas y tecnologías adecuadas para el diseño y desarrollo del sistema.
- Estudiar los lenguajes y las metodologías existentes a emplear en el desarrollo de la aplicación.
- Analizar qué funcionalidades de las IVR se incluirán entre los componentes gráficos del sistema.
- Estudiar qué patrones de diseño aplicar para lograr un producto robusto, flexible y acabado.
- Analizar qué arquitectura diseñar para implementarla de tal forma que garantice la interoperabilidad y la extensibilidad de la aplicación.
- Estudiar el sistema para elaborar y aplicar pruebas que permitan comprobar que cumpla con los requisitos funcionales.

Se utilizaron **como métodos de investigación** en la elaboración de este trabajo, los siguientes:

Teóricos:

- *Analítico-Sintético*: Se distinguen los elementos de un fenómeno y se procede a revisar ordenadamente cada uno de ellos por separado. Consiste en la extracción de las partes de un todo, con el objeto de estudiarlas y examinarlas por separado, para ver, por ejemplo las relaciones entre las mismas. (2) Se hizo uso de él para comprender mejor el funcionamiento de los sistemas IVR

dividiéndolos en varias partes, estudiando cuáles son sus principales características, sus ventajas y desventajas, su utilidad e importancia para las empresas cubanas, haciendo uso de una bibliografía previamente revisada y seleccionada.

Teóricos:

- *Experimentación:* Implica alteración controlada de las condiciones naturales, de tal forma que el investigador creará modelos, reproducirá condiciones, abstraerá rasgos distintivos del objeto o del problema. (2) Se utilizó este método para crear alteraciones y modificaciones controladas y analizar su comportamiento, como modo de prueba para determinar si cumple o no con los requisitos y objetivos trazados.
- *Entrevista:* Se utiliza para la recopilación de información mediante una conversación profesional, con la que se adquiere información acerca de lo que se investiga. (2) Se hizo uso de él para determinar qué funcionalidades necesitaban los programadores de IVR e incorporarlas a la aplicación.

El presente documento tiene la siguiente estructura:

- *Capítulo 1: “Fundamentación teórica”:* Este capítulo contiene la base teórica para entender el problema planteado, donde se abordan algunos conceptos asociados al dominio del problema para facilitar su entendimiento. Además de un estudio de las soluciones existentes. Se presenta el análisis realizado para seleccionar la metodología que se utiliza para el desarrollo de la ingeniería del Generador Gráfico de IVR. Finalmente se proponen los lenguajes de desarrollo y las herramientas para la implementación del sistema.
- *Capítulo 2: “Exploración y Propuesta del Sistema”:* Este capítulo contiene la propuesta del sistema, además de los requerimientos y las funcionalidades de la aplicación, a través de las Historias de Usuario. También se plantean las posibles funcionalidades que van a tener las IVR.
- *Capítulo 3: “Diseño del Sistema”:* Este capítulo contiene todo el diseño del sistema propuesto. Se define la arquitectura y los patrones de diseño así como las clases del negocio a través de las Tarjetas de Cargo o Clase, Responsabilidad y Colaboración y se presentan los Diagramas de Paquetes o Subsistemas de Diseño.

- *Capítulo 4: “Implementación y Prueba”*: En este capítulo se realizan las Tareas de la Ingeniería. Además cuenta con el diseño de casos de prueba planteado por la metodología XP, para probar el correcto funcionamiento de los requisitos funcionales de la aplicación.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Introducción

En este capítulo se proporciona una breve explicación de todos los conceptos relacionados con el desarrollo de este trabajo. Se lleva a cabo un estudio acerca de las diferentes herramientas y métodos que existen en el mundo, en Cuba y en la Universidad de las Ciencias Informáticas para la creación de IVR. Además se especifica la metodología de desarrollo escogida, así como los lenguajes y las herramientas que se proponen para llevar a cabo el desarrollo de la aplicación.

1.1. Conceptos Relacionados

1.1.1. IVR

Es una solución de última generación para operadores de telefonía fija y móvil destinada a ofrecer sistemas inteligentes de pre atención, autogestión y diálogo interactivo con los abonados. Es un sistema automatizado de respuesta interactiva orientado a entregar y/o capturar información a través del teléfono, permitiendo el acceso a determinados servicios. Los sistemas automatizados de reconocimiento de voz permiten a los llamantes usar directamente palabras o frases que estos sistemas reconocen y convierten en comandos que se comunican de forma interactiva con aplicaciones informáticas. (3)

1.1.2. Entorno gráfico de Desarrollo

Un entorno gráfico de desarrollo es un conjunto de funcionalidades integradas para facilitar a los programadores el desarrollo de aplicaciones utilizando componentes visuales. (4)

1.1.3. Código Fuente

Es un programa en su forma original, tal y como fue escrito por el programador, no es ejecutable directamente por el computador, debe convertirse en lenguaje de máquina mediante compiladores, ensambladores o intérpretes. (5)

1.1.4. Generador de código

Un generador de código permite agilizar el desarrollo de aplicaciones, consiste fundamentalmente en realizar acciones para obtener funcionalidades concretas mediante componentes visuales y obtener el código fuente equivalente. (6)

1.1.5. Metodología de desarrollo

Una metodología de desarrollo de software es un conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores, guiando la realización del nuevo software. No existe una metodología estándar a utilizar en todos los procesos de desarrollo de software. (7)

1.1.6. Asterisk

Proyecto desarrollado por Mark Spencer, miembro fundador de la compañía Digium. Es un programa de software libre que convierte un ordenador en una central telefónica multifuncional. Se encarga de integrar funcionalidades de telefonía clásica con nuevas capacidades derivadas de su flexible y potente arquitectura, pudiendo conectar un número determinado de teléfonos para hacer llamadas entre sí. (8)

1.1.7. Comunicaciones Unificadas

La palabra clave para definir las Comunicaciones Unificadas es la colaboración entre diferentes aplicaciones tecnológicas con el fin de incidir en la productividad de la empresa y en el usuario final. Las Comunicaciones Unificadas representan un estado superior en el que se coordinan múltiples canales de comunicaciones y se agrega valor a las tecnologías existentes. Se trata de soluciones que integran voz, vídeo, web y datos sobre una infraestructura IP, al objeto de hacer interactuar y converger diferentes soluciones sobre una misma plataforma. (9)

1.2. Análisis de soluciones existentes en el mundo actual

Los sistemas IVR se utilizan normalmente para servicios de altos volúmenes de llamadas, para reducir costes y mejorar la experiencia del cliente. Como ejemplos de aplicaciones IVR típicos se pueden mencionar: la banca telefónica, el televoto, servicios de tarjetas de crédito, entre otros, haciendo a estos sistemas necesarios para el desarrollo de la comunicación. Sin embargo su implementación y

configuración se torna engorrosa para la mayoría de las empresas. En el mundo existen escasas aplicaciones que responden a la necesidad de crear IVR de forma sencilla y personalizada.

1.2.1. Quickfuse

Permite tan solo arrastrando componentes crear las IVR. Esta herramienta ofrece variadas opciones como cargar y grabar audio, hacer y recibir llamadas entre otras. Tiene como desventaja fundamental que no es una herramienta libre. Además después de diseñada solo brinda la opción de salvarla, una vez hecho esto, el sistema no permite mostrar el código generado y se desconoce su destino, ya que la aplicación solo necesita el diseño y se encarga de toda la infraestructura de telecomunicaciones. Su servicio es en línea.

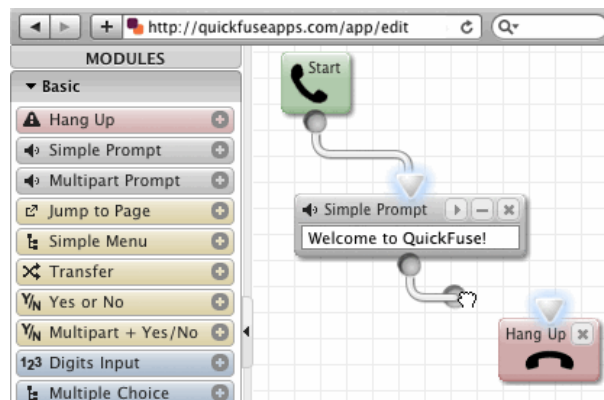


Figura 1: Aplicación Web que crea IVR gráficamente.

1.2.2. CT Developer Studio

CT Developer Studio es un software diseñado para crear, ejecutar y mantener IVR. Su desarrollo y diseño se implementan en forma gráfica. Utiliza una interfaz basada en íconos que se arrastran, elimina la complejidad de la creación de centros avanzados de llamada, scripts IVR y aplicaciones de enrutamiento de llamadas. Es fácil de utilizar pero tiene la desventaja de que no es multiplataforma ya que solo se encuentra disponible para el sistema operativo Windows, y requiere además de una computadora con una tarjeta de telefonía compatible como son Ai-Logix, Intel / Dialogic - Dialogic SR 5.1.1 y NMS. Actualmente existe una versión que se encuentra en modo de evaluación, pero no es una herramienta libre.

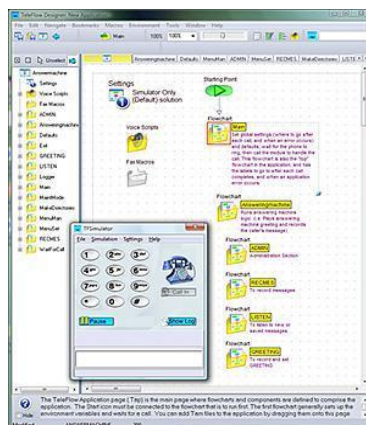


Figura 2: CT Developer Studio para crear IVR gráficamente.

1.2.3. VBVoice

VBVoice es un paquete de herramientas IVR para el desarrollo rápido de aplicaciones que se integra con Microsoft® Visual Studio®. Con él se pueden implementar aplicaciones de telefonía y voz usando una interfaz de usuario con la función de arrastrar y soltar. VBVoice permite desarrollar prototipos de IVR, evaluarlos, incorporar cambios según la respuesta del usuario y ponerlos en funcionamiento. Tiene como desventaja que no es multiplataforma y genera el código en los lenguajes VB6, C# y VB.NET los cuales no son compatibles con Asterisk. También es una herramienta privativa.

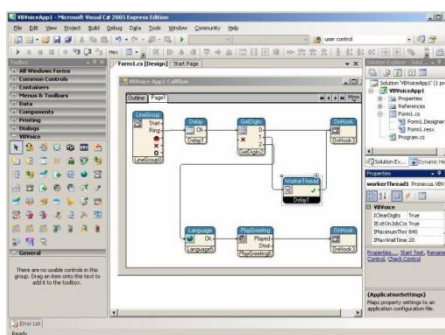


Figura 3: VBVoice para crear IVR gráficamente.

1.2.4. Entorno de Desarrollo IVR (EDI)

El Entorno de desarrollo de aplicaciones IVR es la solución más reciente del portafolio de productos desarrollado por Nimbus Systems. Permite gestionar el aumento de la complejidad de las tareas

implicadas en el desarrollo de aplicaciones de voz a través de la optimización de los procesos de diseño y despliegue de dichas aplicaciones. Así como acometer y coordinar el diseño, documentación, simulación, generación y despliegue de aplicaciones de voz. Es una herramienta integradora que permite obtener el código de una IVR a partir de su diseño. Pero aunque la generación de código está basada en plantillas, lo que le permite incluir otras y obtener el código de la IVR en varios lenguajes, este sigue siendo específico para la plataforma de voz, por lo que habría que hacer ajustes y modificaciones. Además es una herramienta privativa.

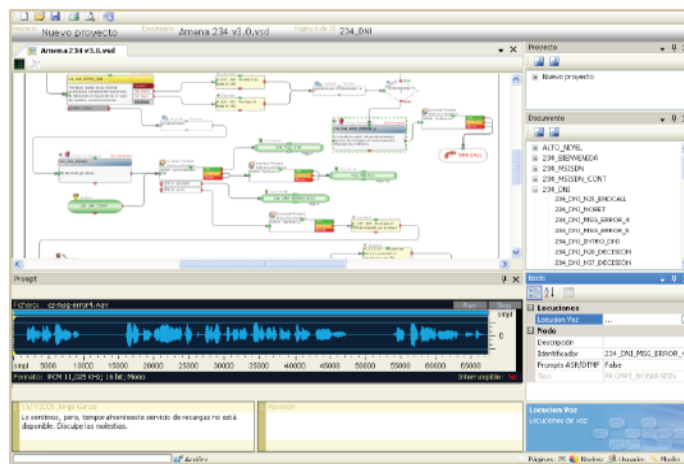


Figura 4: EDI para generar código IVR gráficamente.

Del estudio de sistemas homólogos se puede concluir que no existe una herramienta que sea libre, multiplataforma, y permita diseñar una IVR flexible y personalizada, así como obtener su código listo para utilizarse sin modificaciones adicionales.

1.3. Metodología de Desarrollo

Una metodología de desarrollo de software brinda a los desarrolladores un conjunto de procedimientos, técnicas y herramientas para guiarlos durante el proceso de creación del software de forma organizada y con la documentación apropiada. Las mismas se clasifican en tradicionales y ágiles.

Las tradicionales resaltan el uso exhaustivo de documentación durante todo el ciclo del proyecto, imponen una disciplina de trabajo sobre el proceso de desarrollo del software, con el fin de conseguir un software más eficiente. Las metodologías ágiles ponen vital importancia en la capacidad de respuesta a los

cambios y en mantener una buena relación con el cliente para llevar al éxito el proyecto. Estas últimas se utilizan para equipos de desarrollo que lo integran pocas personas y para proyectos de corto plazo. (10)

Para seleccionar una metodología hay que tener en cuenta las características del proyecto que se va a desarrollar. Dado que la aplicación se desarrollará en corto tiempo y con solo dos personas se decide utilizar una metodología ágil.

1.3.1. Extreme Programming (XP)

Una de las metodologías ágiles de desarrollo de software más exitosa en la actualidad es Extreme Programming (XP). La programación se basa en la simplicidad, la comunicación y la realimentación o reutilización del código desarrollado. Esta metodología consiste en una programación rápida o extrema, cuya peculiaridad es tener como parte del equipo al usuario final, ya que constituye uno de los requisitos para alcanzar el éxito del proyecto. Se simplifica el diseño para agilizar el desarrollo y facilitar el mantenimiento. En la misma todo el proceso está orientado a conseguir el objetivo, basándose para esto en las relaciones interpersonales y en la velocidad de reacción para implementar y responder ante los cambios que puedan surgir durante el proceso de desarrollo. (10)

XP se caracteriza porque los individuos y sus interacciones son más valiosos que los procesos y herramientas en sí. Es más útil y eficaz un software que funcione que la documentación detallada, por lo tanto los documentos deben ser cortos y centrados en lo verdaderamente importante. Para el desarrollo del software es indispensable la colaboración con el cliente más que la negociación de contratos. La planificación debe ser flexible y abierta para proporcionar una respuesta rápida y eficiente ante el cambio. (10)

Precisamente por ser una metodología tan sencilla como ágil y caracterizarse por su utilización para proyectos de corto plazo, pequeño equipo y breve tiempo de entrega, además de la ardua experiencia que poseen miembros del proyecto utilizándola, es adoptada para el desarrollo de esta herramienta.

El ciclo de vida de XP cuenta con seis fases:

- Exploración: Los clientes plantean en las historias de usuarios lo que constituye la principal prioridad para la primera versión del producto.

- Planificación de la Entrega (Release): El cliente establece la prioridad de cada historia de usuario, datos que son usados por los programadores para realizar una estimación del esfuerzo necesario de cada una de ellas.
- Iteraciones: Incluye varias iteraciones sobre el sistema antes de ser entregado. El Plan de Entrega está compuesto por iteraciones de no más de tres semanas. Los elementos que deben tomarse en cuenta durante la elaboración del Plan de la Iteración son: 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.
- Producción: En esta fase se realizan pruebas adicionales y revisiones de rendimiento antes de que el sistema sea llevado al entorno del cliente.
- Mantenimiento: Se plantean las tareas de soporte para el cliente. La fase de mantenimiento puede requerir adicionar personal al equipo y realizar cambios en su estructura.
- Muerte del Proyecto: Esta etapa es de suma importancia pues es cuando el cliente no tiene más historias para ser incluidas en el sistema. Lo que requiere que se satisfagan las necesidades del cliente en otros aspectos como rendimiento y confiabilidad del sistema. (11)

1.4. Lenguaje de Modelado

Las técnicas y notaciones son utilizadas fundamentalmente para lograr una mejor descripción de los procesos de negocio, ya que permite conocer los objetivos del negocio y plasmarlos en un modelo. Una técnica simple que posibilita representar de forma estructurada y jerárquica las actividades que conforman un sistema y los objetos o datos que soportan la interacción de esas actividades es IDEF0 (Definición de la integración para la modelización de las funciones). (12)

1.4.1. IDEF0

IDEF0 ha sido elegido para el desarrollo de la solución propuesta por ser capaz de figurar gráficamente una amplia variedad de negocios, por ser flexible y sencillo, además de intuitivo, útil para representar con menos complejidad los sistemas. También permite, al igual que la metodología XP, la participación activa del cliente para lograr acelerar el tiempo de modelado y garantizar una mayor calidad.

1.5. Lenguaje de Desarrollo

Existen muchos lenguajes de programación que despuntan en el mundo del software. Algunos más que otros con una madurez demostrada por sus años de implantación en el mismo. Para su utilización encontramos dentro de los que más se destacan C#, Java y Python.

1.5.1. Java

Como lenguaje de programación se ha elegido Java, ya que dentro de sus características fundamentales se destaca el hecho de que es multiplataforma y de código abierto. Además, Java es orientado a objetos, donde los objetos se agrupan en estructuras encapsuladas, tanto sus datos como las funciones que manipulan esos datos. El sistema de ejecución de java es dinámico, o sea, las clases solo se enlazan a medida que son necesitadas, esto trae consigo la rapidez y un menor consumo de recursos. Por otra parte Java es un lenguaje bastante robusto, lo cual está dado por numerosas comprobaciones en compilación y en tiempo de ejecución. Posee un recolector de basura por lo cual el programador no tiene que invocar a una subrutina para liberar memoria. (13)

1.6. Librerías

1.6.1. DockingFrames

DockingFrames es un framework de Java Swing para la creación de ventanas flotantes. DockingFrames es open source y está licenciado bajo LGPL 2.1. La filosofía de DockingFrames es tener la mayor flexibilidad posible, incluso si eso significa sacrificar la simplicidad o claridad. La mayoría de los módulos pueden ser reemplazados por los desarrolladores sin cambiar el código de DockingFrames. Sus características principales son: MultiSplitLayout, Área Principal del Editor con múltiples paneles, paneles dentro de los paneles, arrastrar y soltar los paneles, cada zona puede tener más de un panel organizado en pestañas, Minimizar a la barra lateral, ventanas corredizas en el puntero del ratón sobre las barras laterales, Maximizar una zona, por lo que esta ventana a de ocupar todo el espacio, soporta atajos de teclado para las acciones importantes (máximo, ir a la siguiente pestaña, ir a la siguiente zona), guardar y restaurar el diseño, mostrar / ocultar el botón de cierre (de nivel de aplicación), desacoplar de apoyo un panel independiente de Windows. (14)

Se escoge DockingFrames ya que es un marco de acoplamiento agradable y fresco, soporta muchas características de usabilidad, como Ctrl + Mayús + E para activar una ventana emergente para enfocar. También es compatible con varios temas y muelles de animaciones, etc. dentro de un panel acoplado.

1.7. Herramientas

Un IDE (Entorno de Desarrollo Integrado), cuando se usa correctamente, puede ahorrar gran cantidad de dinero, tiempo y, finalmente, reducir el riesgo de la tecnología. Ellos pueden hacer la vida de un desarrollador mucho más agradable a través de características tales como motores de refactorización, optimizadores de código, soporte de código, potentes funciones de navegación de código, depuradores integrados, y mucho más.

1.7.1. Dia Project

Día Project es un software desarrollado por la GNOME Foundation, se utiliza para el modelado de diagramas y fue liberado bajo la Licencia Pública General (GPL) del proyecto GNU (acrónimo recursivo que significa GNU No es Unix).

Puede ser utilizado para dibujar diferentes tipos de diagramas. Actualmente cuenta con objetos especiales para ayudar a dibujar diagramas entidad relación, diagramas UML, organigramas, diagramas de red, entre otros. Esta herramienta permite además, cargar y guardar los diagramas en un formato XML (comprimido con gzip, por defecto, para ahorrar espacio), así como, exportar diagramas a una serie de formatos, incluyendo EPS, SVG, xfig, WMF y PNG. (15)

Se utilizará esta herramienta ya que permite modelar procesos de negocio a través de la notación IDEF0, es multiplataforma, libre y exporta los diagramas en diferentes formatos.

1.7.2. NetBeans

NetBeans IDE es un entorno de desarrollo visual de código abierto, libre y gratuito sin restricciones de uso. Es utilizado para aplicaciones programadas mediante Java, uno de los lenguajes de programación más poderosos del momento, aunque puede ser utilizado para programar en otros lenguajes. Dentro de sus características se pueden encontrar, que es multiplataforma y con él es posible desarrollar desde

aplicaciones para la Web, para dispositivos portátiles, como móviles o Pocket PC, hasta potentes aplicaciones de Escritorio. (16)

Dado que la aplicación será de Escritorio con un fuerte contenido visual, el equipo de desarrollo cuenta, además, con una breve experiencia en el trabajo con dicho entorno de desarrollo y por lo anteriormente expuesto, se escoge como IDE, NetBeans.

1.8. Conclusiones

En este capítulo se realizó un estudio del arte de las diferentes herramientas y métodos que existen en la actualidad para crear IVR. Se abordaron sus principales características y funcionalidades, así como se concluyó a partir de este análisis, la necesidad de desarrollar una herramienta accesible que proporcione ventajas sustanciales para la confección de las mismas. Para el desarrollo de esta nueva solución se propusieron como herramientas, metodologías y lenguajes a utilizar para el desarrollo de la aplicación las siguientes: metodología XP, lenguaje de programación Java unido a NetBeans 6.0.9 como Entorno de Desarrollo Integrado, lenguaje para modelado IDEF0 en conjunto con Dia Project como herramienta para el modelado del negocio, finalmente la librería DockingFrames para enriquecer el trabajo visual de la aplicación.

CAPÍTULO 2: EXPLORACIÓN Y PROPUESTA DEL SISTEMA

Introducción

En el presente capítulo se describen las principales características del sistema a desarrollar, se tratan los temas relacionados con las fases de exploración y planificación de la metodología de desarrollo XP, donde se lleva a cabo la confección de las historias de usuarios para una mayor comprensión del sistema.

2.1. Objeto de automatización

Al ser la creación de IVR un proceso que se realiza de forma manual introduciendo código en un fichero, provoca que el desarrollador emplee notable tiempo y esfuerzo en su elaboración, de ahí la importancia de su automatización. Por lo que se pretende desarrollar una aplicación que permita crear el código de una IVR con tan solo arrastrar componentes visuales representativos de las funcionalidades más importantes que las mismas poseen, dígame Reproducir un sonido, transferir llamadas, entre otras.

La herramienta a desarrollar será una herramienta completamente **libre** para el uso de las empresas y proyectos cubanos, sin tener que pagar licencias ni actualizaciones. Además contará con un **entorno gráfico de programación** que se caracterizará por tener una interfaz personalizable con ventanas que se ocultan cuando no son utilizadas. Más limpio, con una mayor integración y funcionalidad de todos los elementos. Diseñará fácilmente el diálogo de interacción entre el usuario y el sistema automático, accederá más rápido a los recursos utilizados en el desarrollo. Permitirá **simplificar el trabajo para los desarrolladores** con su sencillo y amigable entorno de desarrollo proporcionando facilidades para generar IVR con calidad y en el menor tiempo posible. **Integrará funcionalidades variadas para poder crear IVR personalizadas** que se ajusten a las necesidades del cliente. Finalmente **generará automáticamente todos los ficheros** necesarios para la utilización de la IVR en una central telefónica, como son los ficheros con el código de la IVR y los ficheros de sonido.

2.2. Propuesta del Sistema

La solución propuesta está trazada para las instituciones que utilicen Asterisk como software que proporciona todas las funcionalidades de una central telefónica. IVRCreator es una herramienta que permite la gestión de IVR de forma visual para generar el código correspondiente a esta, durante ese

proceso se podrá gestionar además los archivos de audio ya generados que serán utilizados en las mismas, así como chequear que el código obtenido sea completamente funcional y responda a la necesidad para el que fue previsto.

El sistema está ideado básicamente con dos grupos fundamentales (Fig. 5). El primero viene enfocado a la Gestión de IVR que incluye diseño, creación y simulación de IVR (Fig. 6). El segundo se encuentra relacionado con la Gestión del Proyecto, proceso encargado de facilitar el manejo y la organización de los ficheros para obtener un producto con un mejor acabado (Fig. 7).

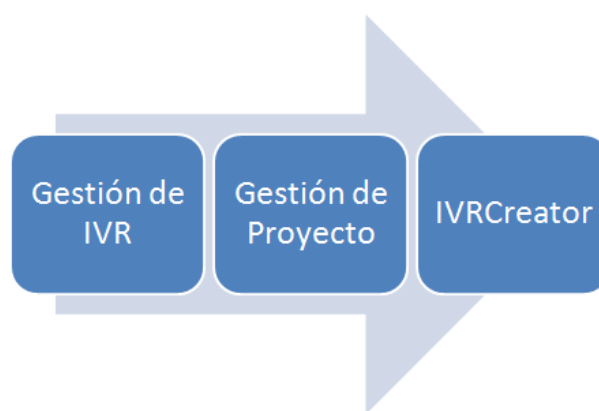


Figura 5: Generador Gráfico de IVR.

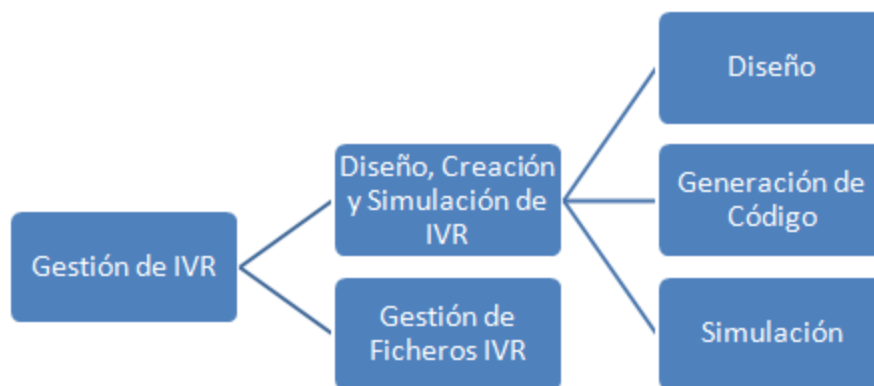


Figura 6: Gestión de IVR.



Figura 7: Carpeta Proyecto.

Gestión de IVR

La gestión de IVR se encuentra a su vez dividida en dos subgrupos: Diseño, Creación y Simulación de IVR y Gestión de Ficheros IVR como se ilustra en la figura 6.

Diseño, Creación y Simulación de IVR

Este subgrupo está conformado por: Diseño, Generación de Código y Simulación.

El **diseño** está relacionado con la estructura y organización de los componentes que conformarán la IVR, sus interrelaciones y la lógica que esta implica.

La **generación de código** se encuentra relacionada con la obtención del código PHP equivalente al diseño hecho previamente por el usuario y listo para ser interpretado por Asterisk.

La **simulación** es la encargada de simular la IVR, utilizando el diseño hecho por el usuario, para que este pueda comprobar la lógica que empleó durante su desarrollo y qué tanto cumple con lo que desea obtener.

Gestión de Ficheros IVR

La gestión de ficheros es el subgrupo encargado de tramitar los ficheros que se puedan generar, eliminar, modificar o utilizar para el desarrollo de la IVR, dígame tanto los ficheros de audio como los ficheros que contienen el diseño de la IVR.

Gestión de Proyecto

La solución contiene una carpeta llamada Proyecto que contiene, como se ilustra en la figura 7 la estructura jerárquica para almacenar los ficheros necesarios, con el objetivo de obtener el código completamente funcional de la IVR.

La carpeta Proyecto

La carpeta Proyecto guarda en su interior dos subcarpetas, la primera llamada Archivos IVR y la segunda Archivos de Sonido.

La subcarpeta Archivos IVR

Esta subcarpeta contiene los archivos con los componentes de la IVR, sus relaciones y propiedades.

La subcarpeta Archivos de Sonido

Esta es la subcarpeta que contiene los archivos de sonido previamente grabados y que son importados para la aplicación. Los mismos son los únicos utilizados dentro del diseño y obtención de la IVR.

2.3. Modelo del Negocio

Se realiza un modelado detallado de qué debe hacer el Sistema, utilizando la notación IDEF0, teniendo como proceso base, Crear IVR.

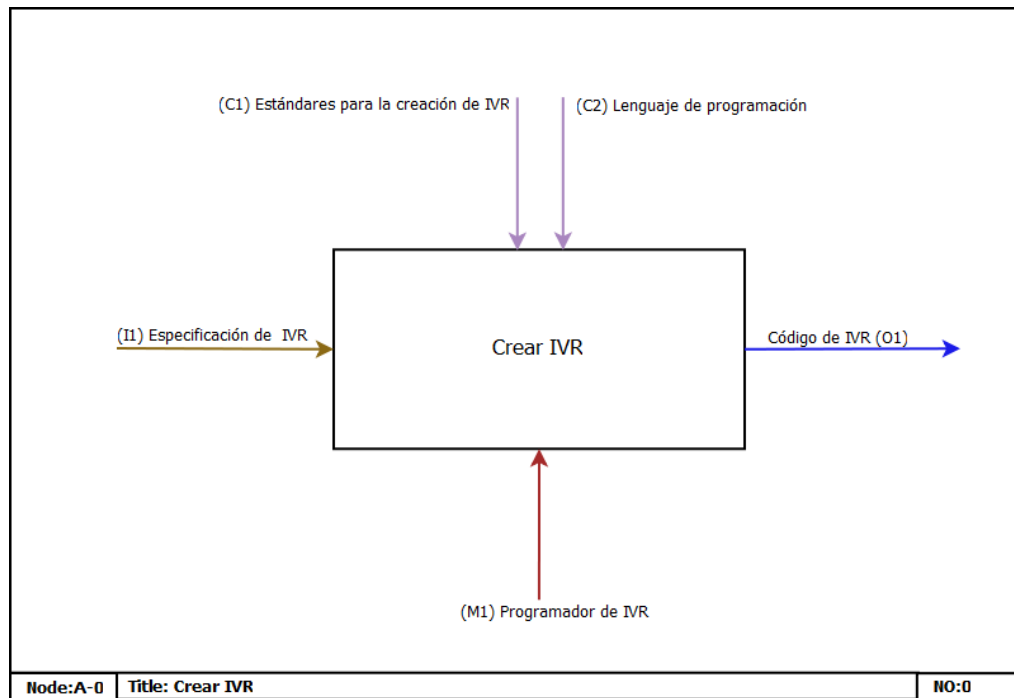


Figura 8: Proceso base "Crear IVR".

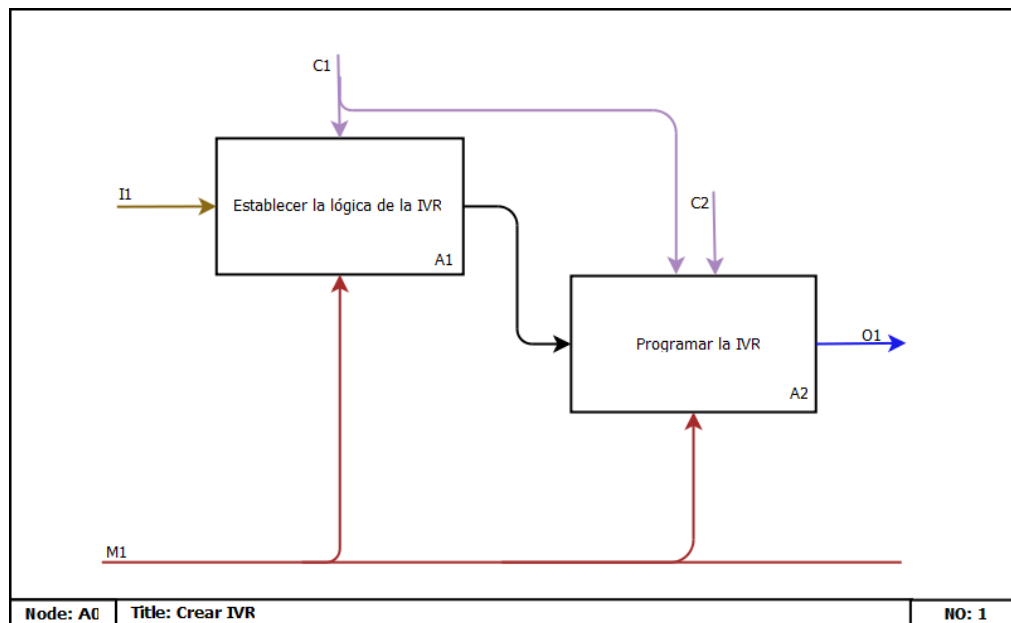


Figura 9: Modelo detallado del proceso base "Crear IVR".

Nombre	Justificación
Programador de IVR	Es la persona que conoce la lógica de programación de IVR y el lenguaje para programar la IVR y por tanto crearla.

Tabla 1: Persona que interviene en el proceso de negocio.

Descripción de los Procesos de Negocio

Los procesos de negocio identificados para la generación del código de una IVR son:

A1- Establecer la lógica de la IVR

Entrada

- Especificación de IVR: Documento formal entregado por el cliente al programador con las descripciones necesarias para la implementación de la IVR.

Salida

- Lógica de la IVR: Conjunto de pasos lógicos que respondan al funcionamiento de la IVR.

Mecanismos

- Programador de IVR: Miembro del proyecto que conoce la lógica y el lenguaje de programación.

Controlador

- Estándares para la creación de IVR: Conjunto de reglas acotadas por el sistema que simula una planta telefónica, para interpretar el código de una IVR.

Descripción

- El proceso consiste en dada las especificaciones de qué clase de IVR se desea construir, establecer el orden lógico de pasos a seguir para crearla y que estos respondan organizadamente al objetivo propuesto.

A2- Programar la IVR

Entrada

- Lógica de la IVR

Salida

- Código de la IVR: Fichero que contiene el código de la IVR.

Mecanismos

- Programador de IVR

Controlador

- Estándares para la creación de IVR.
- Lenguaje de programación: Lenguaje de programación específico en el que se va a implementar.

Descripción

- El proceso consiste en conocida la lógica de la IVR y el lenguaje en que se va implementar traducir esa lógica al lenguaje

2.4. Requisitos funcionales del Sistema

Un requisito funcional define el comportamiento interno del software: cálculos, detalles técnicos, manipulación de datos y otras funcionalidades específicas que muestran cómo los casos de uso serán llevados a la práctica. Los requisitos funcionales establecen los comportamientos del sistema. (17)

Para la metodología XP se aplica este concepto de igual forma pero enfocado a las historias de usuarios. El Generador Gráfico de IVR debe permitir:

Gestionar IVR

1. Crear una nueva IVR
2. Agregar componente Responder Llamada
3. Agregar componente Colgar Llamada
4. Agregar componente Transferir Llamada

5. Agregar componente Selección Múltiple
6. Agregar componente Base de Datos
7. Agregar componente Reproducir Sonido
8. Agregar componente Reproducir Dato
9. Agregar componente Capturar Dato
10. Agregar componente Introducir Código
11. Agregar componente Reproducir Sonido Continuo
12. Agregar conexión entre componente
13. Eliminar conexión entre componentes
14. Eliminar componente
15. Eliminar una IVR
16. Modificar una IVR
17. Cargar y mostrar una IVR

Importar Archivo de Voz

Gestionar Proyecto

1. Crear un nuevo proyecto
2. Exportar un proyecto
3. Abrir un proyecto
4. Guardar un proyecto

Generar el código fuente de una IVR

Compilar la IVR

Simular la IVR

2.5. Requisitos no funcionales del Sistema

Un requisito no funcional o atributo de calidad es en la ingeniería de software, un requisito que especifica criterios que pueden usarse para juzgar la operación de un sistema en lugar de sus comportamientos

específicos, ya que éstos corresponden a los requisitos funcionales. Por tanto, se refieren a todos los requisitos que no describen información a guardar, ni funciones a realizar. (18)

Para un correcto funcionamiento del sistema se deben tener en cuenta los siguientes requisitos no funcionales:

2.5.1. Usabilidad

La aplicación contará con figuras intuitivas y diferenciadas para cada tipo de objeto. Las mismas tendrán un menú propiedades para establecer y modificar el contenido de los datos. Se mostrará una vista del código asociado a cada objeto del diseño. Además posee una barra de herramientas con botones que indican su funcionalidad al pasarle el mouse por encima.

2.5.2. Eficiencia

La eficiencia estará determinada en su mayoría por la velocidad de compilación de la aplicación, la cual estará por debajo de los 20 segundos.

2.5.3. Hardware

Para la instalación de la aplicación se debe disponer de una computadora de 512 GB de RAM o superior, 100 GB de disco duro o superior.

2.5.4. Software

Se requiere la instalación de la Máquina Virtual de Java.

2.5.5. Interfaz de Usuario

La interfaz estará representada por los colores azul, gris y blanco y seguirá el estilo de los IDE de desarrollo.

2.5.6. Soporte

Manual de Usuario

Al finalizar el proyecto se entregará al cliente en conjunto con la aplicación y sus documentos, un Manual de Usuario, explicándoles sobre cómo proceder para manejar correctamente la aplicación.

Capacitación

Se impartirá una capacitación por parte del equipo de desarrollo a los usuarios sobre cómo trabajar con la aplicación. Además de una breve explicación sobre la lógica de interacción de una IVR con los abonados, organizada por los líderes del proyecto.

2.5.7. Restricciones de Diseño e Implementación

Se hace uso del estándar de codificación que propone el framework DockingFrame.

2.6. Componentes de la IVR

- Responder una llamada
- Reproducir un sonido
- Reproducir un mensaje
- Capturar dato
- Base de datos
- Transferir llamada
- Introducir Código
- Reproducir Sonido Continuo
- Colgar una llamada

2.7. Personas relacionadas con el Sistema

Las personas relacionadas con el sistema serán aquellas encargadas de la creación de IVR, las cuales tendrán conocimiento sobre cómo funcionan las mismas. Puede ser un miembro del proyecto, no necesariamente el programador.

2.8. Fase de exploración

La fase de Exploración es la primera fase definida por la metodología XP, especificándose el alcance del sistema. Además permite que el equipo de desarrollo alcance con respecto a los procesos, herramientas y

tecnologías un grado de familiarización significativo para el desarrollo de la solución. Durante esta fase se crean las Historias de Usuario (HU) las cuales definen mediante su redacción la necesidad del cliente, otorgando detalles acerca de la estimación del riesgo y el tiempo empleado en su implementación. (11)

Historias de Usuario

Las HU definen lo que se debe construir en el proyecto de software. El cliente es el responsable de asignarle una prioridad a cada una de ellas. Las mismas serán divididas en tareas y su tiempo será estimado por los desarrolladores, que se traduce a la cantidad de semanas que se emplearán para su implementación. (19)

Clasificación de las Historias de Usuario atendiendo a dos criterios:

La prioridad en el negocio:

- **Alta:** Se le otorga a las HU que resultan funcionalidades fundamentales en el desarrollo del sistema, a las que el cliente define como principales para el control integral del sistema.
- **Media:** Se le otorga a las HU que resultan para el cliente como funcionalidades a tener en cuenta, sin que estas tengan una afectación sobre el sistema que se esté desarrollando.
- **Baja:** Se le otorga a las HU que constituyen funcionalidades que sirven de ayuda al control de elementos asociados al equipo de desarrollo, a la estructura y no tienen nada que ver con el sistema en desarrollo. (19)

El riesgo en su desarrollo:

- **Alta:** Cuando en la implementación de las HU se consideran la posible existencia de errores que lleven la inoperatividad del código.
- **Media:** Cuando pueden aparecer errores en la implementación de la HU que puedan retrasar la entrega de la versión.
- **Baja:** Cuando pueden aparecer errores que serán tratados con relativa facilidad sin que traigan perjuicios para el desarrollo del proyecto. (19)

Las HU serán representadas mediante tablas divididas por las siguientes secciones:

Número: número de la historia de usuario incremental en el tiempo.

- **Nombre de Historia de Usuario:** el nombre de la historia de usuario sería para identificarlas mejor entre los desarrolladores y el cliente.
- **Modificación de Historia de Usuario Número:** si sufrió alguna modificación anterior.
- **Usuario:** involucrados en el desarrollo de la HU.
- **Iteración Asignada:** número de la iteración.
- **Prioridad en negocio:** Alta, Media o Baja.
- **Riesgo en Desarrollo:** Alta, Media o Baja.
- **Puntos estimados:** Tiempo estimado que se demorará el desarrollo de la HU.
- **Puntos Reales:** tiempo que se demoró en realidad el desarrollo de la HU.
- **Descripción:** breve descripción de la HU.
- **Observaciones:** señalamiento o advertencia del sistema.
- **Prototipo de interfaz:** prototipo de interfaz si aplica. (19)

A continuación se muestran las HU de prioridad alta:

Historia de Usuario	
Número: 1	Nombre de Historia de Usuario: Crear una nueva IVR
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Irina Valdes Meneses Mario Iván Cid Vázquez	Iteración Asignada: 1
Prioridad en negocio: Alta	Puntos estimados: 2/3
Riesgo en Desarrollo: Bajo	Puntos Reales: 1
Descripción: Esta opción le permite al usuario crear una IVR nueva para posteriormente agregarle componentes y relacionarlos.	
Observaciones: La nueva IVR se creará en blanco, esta aparecerá en el árbol jerárquico situado en el panel superior izquierdo de la ventana.	

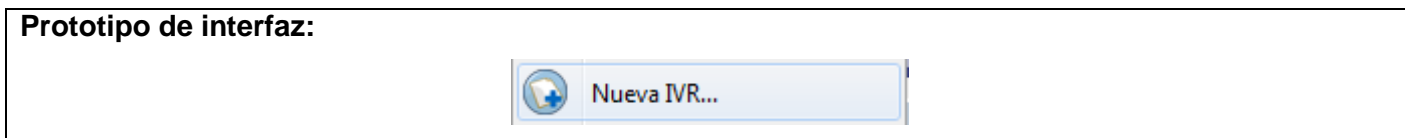


Tabla 2: Historia de usuario: Crear una nueva IVR.


Historia de Usuario	
Número: 2	Nombre de Historia de Usuario: Agregar componente Responder Llamada
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Irina Valdes Meneses Mario Iván Cid Vázquez	Iteración Asignada: 1
Prioridad en negocio: Alta	Puntos estimados: 1/2
Riesgo en Desarrollo: Medio	Puntos Reales: 1
Descripción: Esta opción le permite al usuario arrastrar el componente Responder Llamada hacia el panel visual para confeccionar el cuerpo de la IVR. Además de conocer sus propiedades.	
Observaciones: Las propiedades del componente no podrán ser cambiadas solo consultadas. El usuario podrá arrastrar más de un componente de este tipo pero lógicamente una IVR solo debe tener uno.	
Prototipo de interfaz:	
	

Tabla 3: Historia de usuario: Agregar componente Responder Llamada.

Historia de Usuario	
Número: 3	Nombre de Historia de Usuario: Agregar componente Colgar Llamada
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Irina Valdes Meneses Mario Iván Cid Vázquez	Iteración Asignada: 1
Prioridad en negocio: Alta	Puntos estimados: 1/2

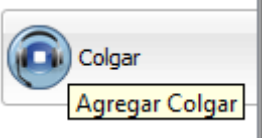
Riesgo en Desarrollo: Medio	Puntos Reales: 1
Descripción: Esta opción le permite al usuario arrastrar el componente Colgar Llamada hacia el panel visual para confeccionar el cuerpo de la IVR. Además de conocer sus propiedades.	
Observaciones: Las propiedades del componente no podrán ser cambiadas solo consultadas. El usuario podrá arrastrar más de un componente de este tipo.	
Prototipo de interfaz:	
	

Tabla 4: Historia de usuario: Agregar componente Colgar Llamada.

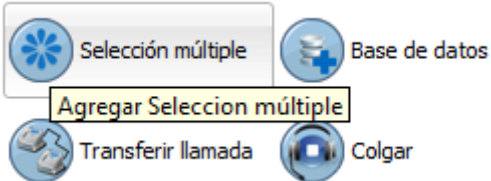
Historia de Usuario	
Número: 5	Nombre de Historia de Usuario: Agregar componente Selección Múltiple
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Irina Valdes Meneses Mario Iván Cid Vázquez	Iteración Asignada: 1
Prioridad en negocio: Alta	Puntos estimados: 2/3
Riesgo en Desarrollo: Alto	Puntos Reales: 1
Descripción: Esta opción le permite al usuario arrastrar el componente Selección Múltiple hacia el panel visual para confeccionar el cuerpo de la IVR. Además de establecer sus propiedades como las condiciones para seleccionar los diferentes caminos que puede tomar la IVR.	
Observaciones: Las propiedades del componente podrán ser cambiadas con excepción del nombre que solo será consultado. El usuario podrá arrastrar más de un componente de este tipo.	
Prototipo de interfaz:	
	

Tabla 5: Historia de usuario: Agregar componente Selección Múltiple.

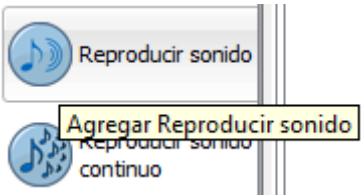
Historia de Usuario	
Número: 7	Nombre de Historia de Usuario: Agregar componente Reproducir
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Irina Valdes Meneses Mario Iván Cid Vázquez	Iteración Asignada: 1
Prioridad en negocio: Alta	Puntos estimados: 1/2
Riesgo en Desarrollo: Alto	Puntos Reales: 1
Descripción: Esta opción le permite al usuario arrastrar el componente Reproducir Sonido, hacia el panel visual para confeccionar el cuerpo de la IVR. Además de establecer sus propiedades como fichero de audio que será reproducido.	
Observaciones: Las propiedades del componente podrán ser cambiadas con excepción del nombre que solo será consultado. El usuario podrá arrastrar más de un componente de este tipo.	
Prototipo de interfaz:	
	

Tabla 6: Historia de usuario: Agregar componente Reproducir.

Historia de Usuario	
Número: 8	Nombre de Historia de Usuario: Agregar componente Reproducir Dato
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Irina Valdes Meneses Mario Iván Cid Vázquez	Iteración Asignada: 1
Prioridad en negocio: Alta	Puntos estimados: 1/2
Riesgo en Desarrollo: Alto	Puntos Reales: 1
Descripción: Esta opción le permite al usuario arrastrar el componente Reproducir Dato, hacia el panel	

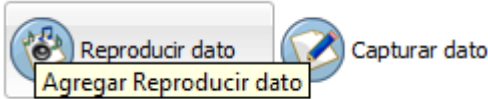
visual para confeccionar el cuerpo de la IVR. Además de establecer sus propiedades como el dígito que se va a reproducir.
Observaciones: Las propiedades del componente podrán ser cambiadas con excepción del nombre que solo será consultado. El usuario podrá arrastrar más de un componente de este tipo.
Prototipo de interfaz:


Tabla 7: Historia de usuario: Agregar componente Reproducir Dato.

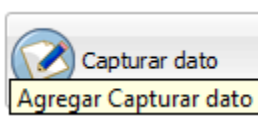
Historia de Usuario	
Número: 9	Nombre de Historia de Usuario: Agregar componente Capturar Dato
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Irina Valdes Meneses Mario Iván Cid Vázquez	Iteración Asignada: 1
Prioridad en negocio: Alta	Puntos estimados: 1/2
Riesgo en Desarrollo: Alto	Puntos Reales: 1
Descripción: Esta opción le permite al usuario arrastrar el componente Capturar Dato, hacia el panel visual para confeccionar el cuerpo de la IVR. Además de establecer sus propiedades como la condición de parada.	
Observaciones: Las propiedades del componente podrán ser cambiadas con excepción del nombre que solo será consultado. El usuario podrá arrastrar más de un componente de este tipo.	
Prototipo de interfaz:	
	

Tabla 8: Historia de usuario: Agregar componente Capturar Dato.

Historia de Usuario	
Número: 12	Nombre de Historia de Usuario: Agregar conexión entre componentes

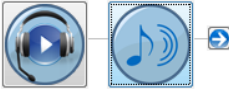
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Irina Valdes Meneses Mario Iván Cid Vázquez	Iteración Asignada: 1
Prioridad en negocio: Alta	Puntos estimados: 1/2
Riesgo en Desarrollo: Alto	Puntos Reales: 1
Descripción: Esta opción le permite al usuario arrastrar desde un componente, un botón representado por una flecha y soltarlo encima de otro componente para establecer una conexión lógica entre los componentes que integran la IVR.	
Observaciones: Para el caso del componente Selección Múltiple podrá establecer tantas conexiones como condicionales haya definido en las propiedades, para los demás será solo una y en el caso de Colgar Llamada y Transferir Llamada será cero conexiones.	
Prototipo de interfaz:	
	

Tabla 9: Historia de usuario: Agregar conexión entre componentes.

Historia de Usuario	
Número: 18	Nombre de Historia de Usuario: Generar el código fuente de una IVR
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Irina Valdes Meneses Mario Iván Cid Vázquez	Iteración Asignada: 1
Prioridad en negocio: Alta	Puntos estimados: 2/3
Riesgo en Desarrollo: Alto	Puntos Reales: 1
Descripción: Esta funcionalidad permite generar el código en lenguaje PHP para cada componente de la IVR que ha sido insertado.	
Observaciones: El código solo se genera si los componentes visuales están relacionados mediante una línea.	

Tabla 10: Historia de usuario: Generar el código fuente de una IVR.

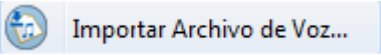
Historia de Usuario	
Número: 19	Nombre de Historia de Usuario: Importar Archivo de Voz
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Irina Valdes Meneses Mario Iván Cid Vázquez	Iteración Asignada: 1
Prioridad en negocio: Alta	Puntos estimados: 1/2
Riesgo en Desarrollo: Alto	Puntos Reales: 1
Descripción: La funcionalidad Importar Archivo de Voz brinda la posibilidad a los usuarios de buscar en algún a dirección específica el Archivo de Voz que desea utilizar en sus IVR, los importa hacia una carpeta perteneciente al proyecto y le cambia el nombre si así lo precisa. El procedimiento se aplica para todos los archivos que desee importar a la aplicación.	
Observaciones: El usuario solo podrá utilizar los archivos de voz que hayan sido importados.	
Prototipo de interfaz:	
	

Tabla 11: Historia de usuario: Importar Archivo de Voz.

Historia de Usuario	
Número: 21	Nombre de Historia de Usuario: Exportar un proyecto
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Irina Valdes Meneses Mario Iván Cid Vázquez	Iteración Asignada: 1
Prioridad en negocio: Alta	Puntos estimados: 2/3
Riesgo en Desarrollo: Alto	Puntos Reales: 1
Descripción: El sistema con esta funcionalidad le permite al usuario una vez creado el proyecto, seleccionar la opción Exportar Proyecto. De esta forma obtiene todos los documentos y recursos para hacer funcionar su IVR.	
Observaciones: El proyecto se exporta para generar la carpeta que contendrá el código de la IVR y los	

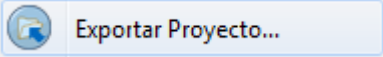
archivos de sonido, listos para trabajar. Este procedimiento se lleva a cabo una vez elaborado el proyecto conjuntamente con las IVR.
Prototipo de interfaz:


Tabla 12: Historia de usuario: Exportar un proyecto.

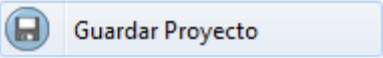
Historia de Usuario	
Número: 23	Nombre de Historia de Usuario: Guardar un proyecto
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Irina Valdes Meneses Mario Iván Cid Vázquez	Iteración Asignada: 1
Prioridad en negocio: Alta	Puntos estimados: 2/3
Riesgo en Desarrollo: Alto	Puntos Reales: 1
Descripción: Esta funcionalidad permite al usuario Guardar el proyecto con el que está trabajando. El proyecto se guarda en la dirección especificada por el usuario con una carpeta general que tiene dos subcarpetas, los archivos de voz importados y las IVR con extensión .imc para que después puedan ser cargadas y modificadas antes de generar el código definitivo.	
Observaciones:	
Prototipo de interfaz:	
	

Tabla 13: Historia de usuario: Guardar un proyecto.

Historia de Usuario	
Número: 25	Nombre de Historia de Usuario: Compilar la IVR
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Irina Valdes Meneses Mario Iván Cid Vázquez	Iteración Asignada: 1

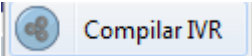
Prioridad en negocio: Alta	Puntos estimados: 2/3
Riesgo en Desarrollo: Alto	Puntos Reales: 1
Descripción: Esta funcionalidad le permite al usuario una vez compilada la IVR que está creando observar en la pestaña Código el código que se va a generar.	
Observaciones: El código que se genera es de consulta y este no se puede modificar directamente, para hacerlo es preciso modificar la IVR y volver a compilar.	
Prototipo de interfaz:	
	

Tabla 14: Historia de usuario: Compilar la IVR.

Las restantes tablas correspondientes a las historias de usuarios que no se presentaron anteriormente pueden encontrarse en el Anexo I.

2.9. Planificación

La actividad de planeación en la metodología XP comienza con la creación de una serie de HU que describen las características y funcionalidades necesarias para la construcción del software. Durante esta fase se lleva a cabo una estimación del esfuerzo que costará implementar cada una. El cual se expresa utilizando como unidad de medida el punto. Un punto se razona como una semana ideal de trabajo donde los miembros de los equipos de desarrollo trabajan el tiempo planeado sin ningún tipo de interrupción.

Estimación de esfuerzo por Historias de Usuario

A continuación se muestra la estimación del esfuerzo por cada HU propuesta para el desarrollo de la aplicación:

Historia de Usuario	Punto de Estimación
Crear una nueva IVR	2
Agregar componente Responder Llamada	1
Agregar componente Colgar Llamada	1

Agregar componente Transferir Llamada	2
Agregar componente Selección Múltiple	2
Agregar componente Base de Datos	2
Agregar componente Reproducir Sonido	1
Agregar componente Reproducir Dato	1
Agregar componente Capturar Dato	1
Agregar componente Introducir Código	1
Agregar componente Reproducir Sonido Continuo	1
Agregar conexión entre componentes	1
Eliminar conexión entre componentes	1
Eliminar componente	1
Eliminar una IVR	1
Modificar IVR	2
Cargar y mostrar una IVR	1
Generar el código fuente de una IVR	2
Importar Archivo de Voz	1
Crear un nuevo proyecto	1
Exportar un proyecto	2
Abrir un proyecto	1

Guardar un proyecto	2
Compilar la IVR	2
Simular la IVR	2

Tabla 15: Estimación por historia de usuario.

2.10. Plan de Iteraciones

Después de ser identificadas y descritas las HU, así como estimado el esfuerzo dedicado a la realización de las mismas, se procede a la planificación de la fase de implementación para lo cual se establecieron tres iteraciones.

Iteración 1

En la iteración 1 se llevará a cabo el desarrollo de las HU 1,2,3,5,7,8,9,12,18,19,21,23,24 las cuales pertenecen a las historias de usuarios que poseen alta prioridad para el cliente.

Iteración 2

En la iteración 2 se llevará a cabo el desarrollo de la HU 4,6,10,11,13,14,16,17,20,22 las cuales pertenecen a las historias de usuarios que poseen prioridad Media para el cliente.

Iteración 3

En la iteración 2 se llevará a cabo el desarrollo de la HU 15, 25 las cuales pertenecen a las historias de usuarios que poseen prioridad Baja para el cliente. Al finalizar esta iteración se obtendrá una versión 1.0 del producto final y a partir de aquí el sistema se pondrá en función para ser evaluado.

2.10.1. Plan de duración de las Iteraciones

El plan de duración de las iteraciones se encarga de mostrar las HU en el orden en que se implementarán en cada iteración así como la duración estimada de las mismas.

Iteración	Orden de las historias de usuarios a implementar	Duración Total
-----------	--	----------------

1	Crear una nueva IVR Agregar componente Responder Llamada Agregar componente Colgar Llamada Agregar componente Selección Múltiple Agregar componente Reproducir Sonido Agregar componente Reproducir Dato Agregar componente Capturar Dato Agregar conexión entre componentes Generar el código fuente de una IVR Importar Archivo de Voz Exportar un proyecto Guardar un proyecto Compilar la IVR	19 semanas
2	Agregar componente Transferir Llamada Agregar componente Base de Datos Agregar componente Introducir Código Agregar componente Reproducir Sonido Continuo Eliminar conexión entre componentes Eliminar componente	13 semanas

	Modificar IVR Cargar y mostrar una IVR Crear un nuevo proyecto Abrir un proyecto	
3	Eliminar una IVR Simular la IVR	3 semanas

Tabla 16: Plan de iteraciones.

2.11. Plan de Entregas

El plan de entrega detalla la fecha fin de cada iteración.

- Historias de usuarios con prioridad alta
- Historias de usuarios con prioridad media
- Historias de usuarios con prioridad baja

Aplicación	Fin de la Iteración 1 (22 de marzo de 2012)	Fin de la Iteración 2 (19 de abril de 2012)	Fin de la Iteración 3 (30 abril de 2012)
Generador gráfico de IVR	Historias de Usuarios con prioridad Alta.	Historias de Usuarios con prioridad Media.	Historias de Usuarios con prioridad Baja.

Tabla 17: Plan de entrega.

2.12. Conclusiones

En el presente capítulo se elaboró la propuesta del sistema que se va a desarrollar. De forma similar se identificaron las funcionalidades tanto de la aplicación, como de la IVR, para un total de 25 y 9 respectivamente, además de los 7 requisitos que el sistema debe cumplir. También se confeccionó las descripciones de las 25 HU y la planificación del esfuerzo dedicado a la realización de cada una de ellas, teniendo en cuenta el orden en que se les dará cumplimiento según las necesidades del cliente.

Finalmente se obtuvo la duración aproximada del proyecto que corresponde con 12 semanas como mínimo y 36 semanas como máximo.

CAPÍTULO 3: DISEÑO DEL SISTEMA

Introducción

En el presente capítulo se define la arquitectura de la aplicación, de similar manera los patrones de diseño que se van a utilizar para su desarrollo. Así como se identifican, a través de los Diagramas de Paquetes o subsistemas de Diseño, las clases más relevantes para dar cumplimiento a las funcionalidades del sistema. Finalmente se describen las Tarjetas de Clase, Responsabilidad y Colaboración.

3.1. Arquitectura del sistema

La arquitectura del software alude a "la estructura global del software y a las formas en que la estructura proporciona la integridad conceptual de un sistema". En su forma más simple, la arquitectura es la estructura jerárquica de los componentes del programa (módulos), la manera en que los componentes interactúan y la estructura de datos que van a utilizar los componentes. Sin embargo, en un sentido más amplio, los "componentes" se pueden generalizar para presentar los elementos principales del sistema y sus interacciones. (20)

"El diseño arquitectónico define la relación entre los elementos estructurales principales del software, los patrones de diseño que se pueden utilizar para lograr los requisitos que se han definido para el sistema, y las restricciones que afectan a la manera en que se pueden aplicar los patrones de diseño arquitectónicos" (20)

3.1.1. Patrón de Arquitectura utilizado

Un patrón arquitectura de software se selecciona con base en objetivos y restricciones. Los objetivos son aquellos prefijados para el sistema de información, pero no solamente los de tipo funcional, también otros objetivos como la mantenibilidad, auditabilidad, flexibilidad e interacción con otros sistemas de información. Las restricciones son aquellas limitaciones derivadas de las tecnologías disponibles para implementar sistemas de información. Unas arquitecturas son más recomendables de implementar con ciertas tecnologías mientras que otras tecnologías no son aptas para determinadas arquitecturas. (15)

Los patrones expresan el esquema fundamental de organización para sistemas de software. Proveen un conjunto de subsistemas predefinidos; especifican sus responsabilidades e incluyen reglas y guías para organizar las relaciones entre ellos; así como ayudan a especificar la estructura fundamental de una aplicación. (21)

Modelo Vista Controlador (MVC)

El patrón de arquitectura de MVC divide una aplicación interactiva en tres partes: el Modelo contiene los datos y las funcionalidades esenciales, las Vistas despliegan la información al usuario y los Controladores manejan las entradas de datos. Las Vistas y los Controladores conforman la interfaz de usuario. Un mecanismo de propagación de cambios asegura la consistencia entre la interfaz y el modelo. La separación de la vista y el controlador permite tener múltiples perspectivas del mismo modelo. Si el usuario cambia el modelo a través del controlador de una vista, todas las otras vistas dependientes deben reflejar los cambios. Por lo tanto, el modelo notifica a todas las vistas siempre que sus datos cambien. Las vistas, en cambio, recuperan los nuevos datos del modelo y actualizan la información que muestran al usuario.



Figura 10: Arquitectura

El MVC es utilizado como patrón de arquitectura, debido a:

- Cada elemento del patrón está altamente especializado en su tarea (la vista en mostrar datos al usuario, el controlador en las entradas y el modelo en su objetivo de negocio), lo cual le proporciona una mayor cohesión a la aplicación.
- Desacopla las vistas del modelo, permitiendo un diseño flexible de la aplicación.

- Se pueden crear múltiples vistas de un modelo, lo cual es muy útil durante la implementación, ya que se puede utilizar cualquier vista, no necesariamente la del producto final, durante las pruebas de funcionalidad.
- Se puede cambiar el modo en que una vista responde al usuario sin cambiar su representación visual, es decir, con solo cambiar de Controlador podemos capturar de forma diferente las acciones realizadas en las Vistas, lo que permite agregar o modificar funcionalidades según las necesidades durante la implementación. (22)

3.2. Patrones de Diseño

Los patrones de diseño resuelven problemas concretos y crean diseños Orientados a Objeto más elegantes, flexibles y reutilizables. Describen un problema recurrente y una solución. Cada patrón nombra, explica, evalúa un diseño. Un patrón de diseño identifica: Clases, Instancias, Roles, Colaboraciones y la distribución de responsabilidades. (23)

Elementos principales:

- **Nombre:** Nombre del patrón
- **Problema:** Problema que lo impulsa a utilizarlo
- **Solución:** Descripción abstracta (23)

3.2.1. Patrón para Asignar Responsabilidades (GRASP)

Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. (24) Dentro de los patrones GRASP utilizados en el desarrollo del sistema se encuentran los siguientes:

Patrón Expert

Nombre: Expert

Problema: ¿Cómo lograr que cada clase cumpla con la responsabilidad que le corresponde?

Solución: Asignar una responsabilidad a la clase que tiene la información necesaria para cumplirla, ejemplo de las clases que cumplen con el patrón son Controller, IVRFile, Project, View, VisualPane, CodeGenerator, Compiler.

Patrón Crator

Nombre: Creator

Problema: ¿Quién debería ser el responsable de la creación de una nueva instancia de alguna clase?

Solución: Asignarle a una clase la responsabilidad de crear una instancia de otra clase. Ejemplo la clase Project es un experto de la clase IVRFile y es el responsable de crear una instancia de esta clase. Otras clases que implementan el patrón son CodeGenerator, IVRFile, Compiler.

Patrón Controller

Nombre: Controller

Problema: ¿Cómo lograr atender un evento del sistema?

Solución: Un Controlador es un objeto de interfaz no destinado al usuario que se encarga de manejar un evento del sistema. La clase Controller es la encargada de crear este objeto y contener todos los eventos que se manejan en la aplicación.

3.2.2. Patrón Visitor

Representa una operación sobre los elementos de una estructura de objetos. Permite definir una nueva operación sin cambiar las clases de los elementos sobre los que opera. (25)

Nombre: Visitor

Problema: ¿Cómo hacer para generar el código de la IVR, haciendo que cada componente sea responsable de aportar su propia implementación?

Solución: Se define una interfaz ComponentVisitor con todos los métodos Visit para visitar cada uno de los componentes, estos a su vez llaman al método específico de esta clase que se encuentra implementado en la clase PhpCodeGenerator el cual implementa dicha interfaz.

3.2.3. Patrón Observer

El patrón Observer tiene como propósito definir una dependencia 1: n de forma que cuando el objeto 1 cambie su estado, los n objetos sean notificados y se actualicen automáticamente. (26)

Nombre: Observer

Problema: ¿Cómo hacer para cuando se cambien un objeto en la clase Model la clase View sea notificada?

Solución: Se define una interfaz con un método Update al cual se le pasa como parámetro el objeto que será cambiado, La clase View implementa dicha interfaz y por consiguiente el método que será llamado desde la clase Model encargada de hacer las notificaciones a través del método notify.

3.2.4. Patrón Singleton

El patrón Singleton tiene como propósito asegurar que una clase tiene una única instancia y asegurar un acceso global. (27)

Nombre: Singleton

Problema: ¿Cómo hacer para que no se pueda crear una instancia desde otra clase, de la clase Controller pero a su vez se pueda acceder a sus métodos?

Solución: Hacer que sea la propia clase quien controle la creación de esa única instancia y quien proporcione un acceso global a la misma. Creando un constructor privado en la clase Controller, un atributo estático de ella misma y un método estático que permita crear la instancia y devolverla. Otra clase que implementa dicho patrón es la clase Model.

3.3. Diagrama de Paquetes o Subsistemas de Diseño

El diagrama de paquetes o subsistemas de diseño permite una agrupación lógica de los elementos del sistema. En el presente caso representa de manera independiente, cada una de las capas del MVC con la intención de organizar las clases.

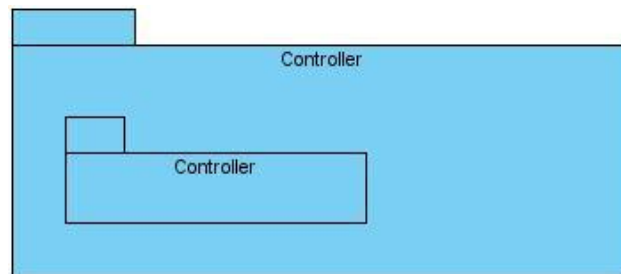


Figura 11: Paquete Controller.

En la clase Controller se encuentran agrupados todos los eventos que se capturan de las Vistas, esta clase es la encargada de manejar la interacción entre el usuario y el negocio de la aplicación. Está conformado por las clases Controller, ComponentMouseAdapter, ComponentMouseMotionAdapter.

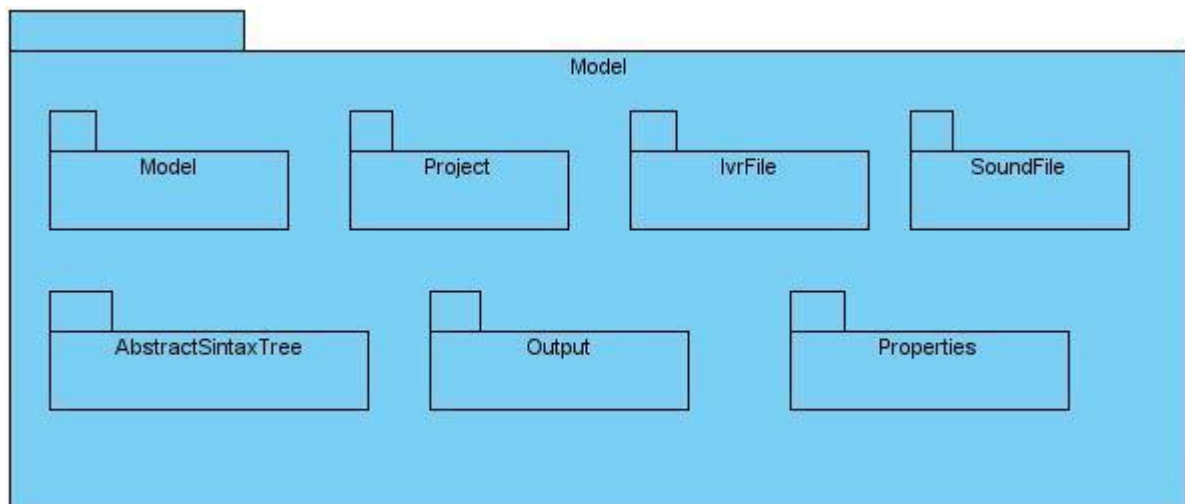


Figura 12: Paquete Model.

El paquete Model contiene el negocio de la Aplicación, está conformado por las clases Project, Model, IvrFile y SoundFile, así como contiene además los subpaquetes AbstractSintaxTree, Output y Properties.

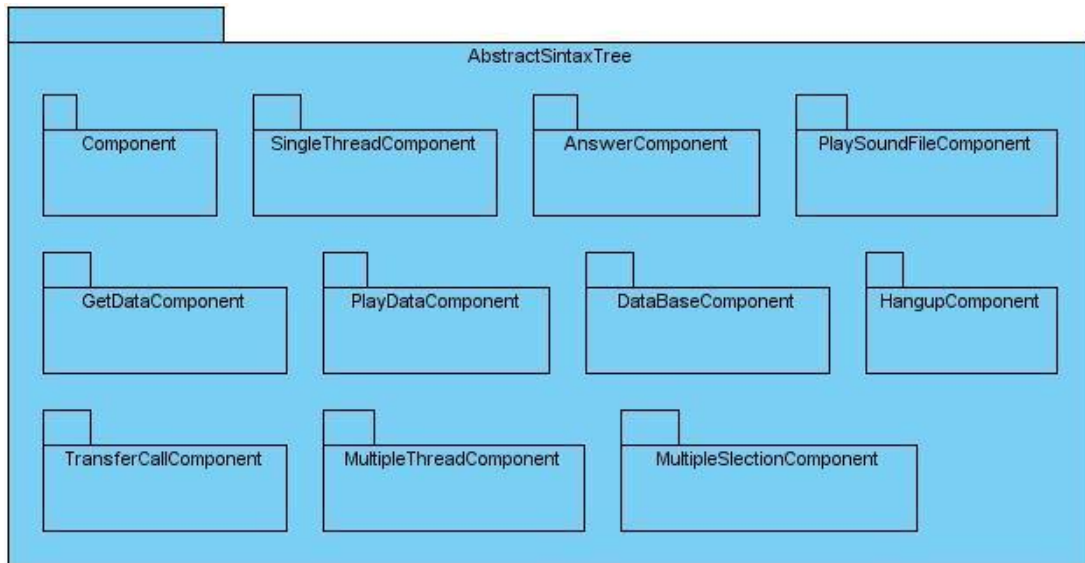


Figura 13: Paquete AbstractSintaxTree.

En el subpaquete AbstractSintaxTree se encuentran los componentes utilizados en la construcción de las IVR, agrupados según la clase base de cada uno, que puede ser SingleThreadComponent o MultiThreadComponent.

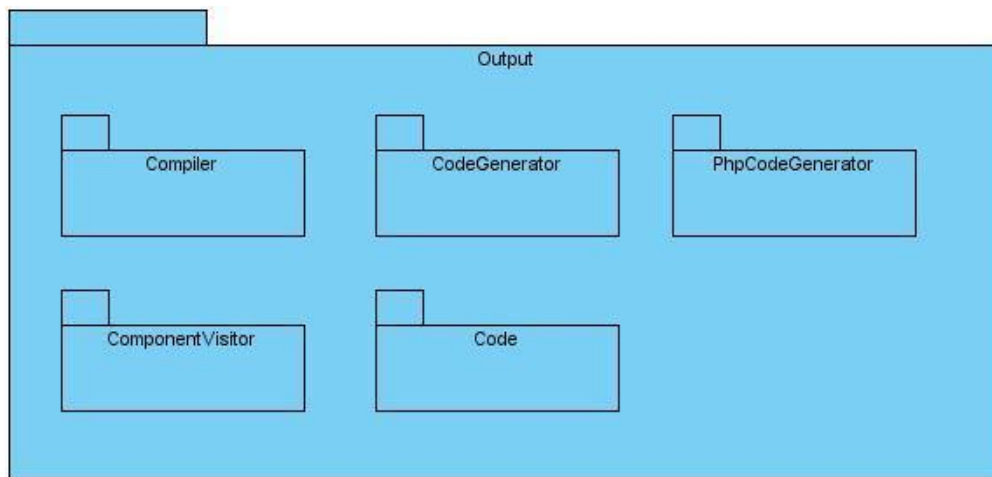


Figura 14: Paquete Output.

En el subpaquete Output se implementa el patrón Visitor para la generación de código, contiene un clase abstracta CodeGenerator de la cual heredaran cada una de las clases según el lenguaje en el cual se

quiera generar el código, en este caso solo existe la clase PhpCodeGenerator, pero también es posible agregar la generación de código en otros lenguajes en caso de que sea necesario.

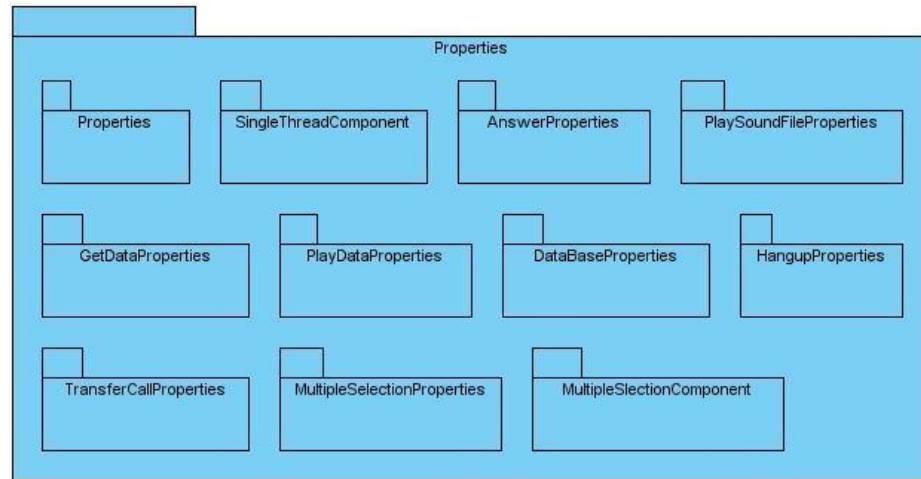


Figura 15: Paquete Properties.

El subpaquete Properties contiene una clase para cada una de las propiedades de cada uno de los componentes que se utilizan para crear la IVR.

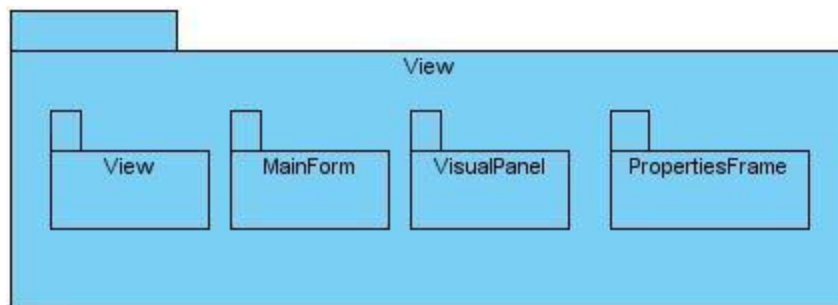


Figura 16: Paquete View.

El paquete View contiene la capa Vista de la aplicación, está compuesto por: el formulario principal, el panel visual donde aparecen los componentes y se dibujan las relaciones, y el paquete PropertiesFrame.

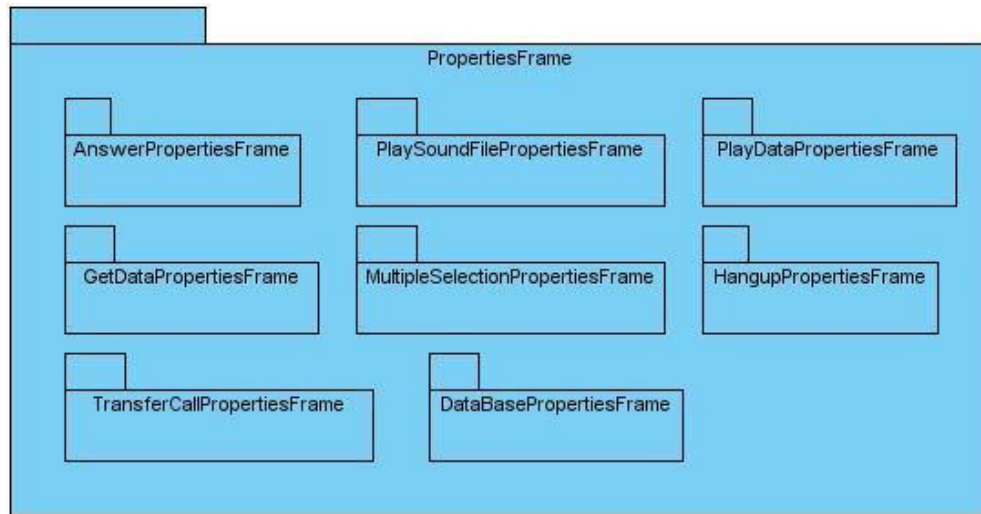


Figura 17: Paquete PropertiesFrame.

El paquete PropertiesFrame contiene cada uno de los frames utilizados para mostrar las propiedades de los componentes.

3.4. Tarjetas CRC

Las tarjetas CRC (Clase, responsabilidad y colaboración) constituyen una técnica para la representación de sistemas orientados a objetos, ya que permiten conocer el comportamiento de una clase en un alto nivel y no simplemente como un conjunto de datos aislados. Las tarjetas CRC están divididas en tres elementos importantes, el nombre de la clase, las responsabilidades de la clase (cualquier cosa que la clase sabe o hace) y los colaboradores (aquellas clases que se requieren para que una clase reciba la información necesaria para completar una responsabilidad). (19)

A continuación se muestran las tarjetas CRC correspondientes a las clases más importantes del Generador Gráfico de IVR.

Nombre de la Clase: View	
Responsabilidades:	Colaboradores:

<ul style="list-style-type: none"> • Crear la clase Controller • Crear el MainForm. • Capturar las notificaciones de la clase Model y comunicar al MainForm los cambios. 	<ul style="list-style-type: none"> • Controller • Model
---	---

Tabla 18: Tarjeta CRC: View.

Nombre de la Clase: Controller	
Responsabilidades:	Colaboradores:
<ul style="list-style-type: none"> • Capturar los eventos que se producen en el MainForm y expandir los cambios a la Model. 	<ul style="list-style-type: none"> • View • Model

Tabla 19: Tarjeta CRC: Controller.

Nombre de la Clase: Model	
Responsabilidades:	Colaboradores:
<ul style="list-style-type: none"> • Crear la clase View. • Mantener toda la información acerca del Proyecto actual y de las IVR y los archivos de sonidos que contiene el mismo. • Notificar los cambios que se realizan en los datos a las Vistas. 	<ul style="list-style-type: none"> • Controller • View

Tabla 20: Tarjeta CRC: Model.

Nombre de la Clase: PhpCodeGenerator	
Responsabilidades:	Colaboradores:
<ul style="list-style-type: none"> • Implementar el patrón Visitor, para crear el código a partir del Árbol de Sintaxis Abstracta creado por el usuario en el MainForm. 	<ul style="list-style-type: none"> • AnswerComponent • PlaySoundFileComponent

	<ul style="list-style-type: none">• PlayDataComponent• GetDataComponent• DataBaseComponent• HangupComponent• TransferCallComponent• MultipleSelectionComponent
--	---

Tabla 21: Tarjeta CRC: PhpCodeGenerator.

Las restantes tablas correspondientes a las tarjetas CRC que no se presentaron anteriormente pueden encontrarse en el Anexo II.

3.5. Conclusiones

Al finalizar este capítulo se alcanzó comprender mejor el sistema y cómo debe ser su implementación, a raíz de la definición del diseño, el cual estuvo basado en la selección del patrón arquitectónico Modelo Vista Controlador (MVC) y los diferentes patrones de diseño, los cuales en conjunto con la arquitectura facilitan la reutilización, organización, optimización y elegancia del código. Los patrones escogidos fueron los GRASP (Controlador, Experto, Creador), Visitante, Observador e Instancia Única. Por último se desarrollaron las tarjetas Clase, Responsabilidad, Colaboración para asignar las responsabilidades a cada clase, así como la colaboración entre ellas.

CAPÍTULO 4: IMPLEMENTACIÓN Y PRUEBA

Introducción

En el presente capítulo se elaborarán las Tareas de Ingeniería las cuales han sido utilizadas como base para la implementación del software. Finalmente se evaluará la calidad de la aplicación a través de las pruebas de software, derivadas de las Historias de Usuario y Tareas de la Ingeniería que se han implementado.

4.1. Tareas de la Ingeniería

Las tareas de la Ingeniería XP las define como una labor que llevan a cabo los desarrolladores para dar cumplimiento a las Historias de Usuarios. Cada historia de usuario se desglosa en varias de estas tareas. A su vez la implementación de las historias de usuarios, las cuales son definidas por el cliente, se agrupan en iteraciones que al finalizar cada una de ellas se obtiene un producto funcional el cual debe ser mostrado y aprobado por el cliente. (19)

Las tareas de la ingeniería serán representadas mediante tablas divididas por las siguientes secciones:

- **Número Tarea:** los números deben ser consecutivos.
- **Número Historia de Usuario:** número de la historia de usuario a la que pertenece la tarea.
- **Nombre Tarea:** nombre que identifica a la tarea.
- **Tipo de Tarea:** las tareas pueden ser de: Desarrollo, Corrección, Mejora, Otra (Especificar)
- **Puntos Estimados:** tiempo estimado en semanas que se le asignará a su desarrollo.
- **Fecha Inicio:** fecha en que inicia el desarrollo de la tarea.
- **Fecha Fin:** fecha en que finaliza el desarrollo de la tarea.
- **Programador Responsable:** nombre y apellidos del programador.
- **Descripción:** breve descripción de la tarea.

A continuación las tareas correspondientes a las HU de alta prioridad:

Tarea de Ingeniería

Número Tarea: 1	Número Historia de Usuario: 1
Nombre Tarea: Crear una nueva IVR	
Tipo de Tarea: Desarrollo	Puntos Estimados: 2/3
Fecha Inicio: 01/02/2012	Fecha Fin: 03/02/2012
Programadores Responsables: Mario Iván Cid e Irina Valdes Meneses.	
Descripción: Esta opción le permite al usuario crear una IVR nueva para posteriormente agregarle componentes y relacionarlos. La nueva IVR se creará en blanco, esta aparecerá en el árbol jerárquico situado en el panel superior izquierdo de la ventana.	

Tabla 22: Tarea de Ingeniería: Crear nueva IVR.

Tarea de Ingeniería	
Número Tarea: 2	Número Historia de Usuario: 2
Nombre Tarea: Agregar componente Responder Llamada	
Tipo de Tarea: Desarrollo	Puntos Estimados: 1/2
Fecha Inicio: 04/02/2012	Fecha Fin: 06/02/2012
Programadores Responsables: Mario Iván Cid e Irina Valdes Meneses.	
Descripción: Esta opción le permite al usuario arrastrar el componente Responder Llamada hacia el panel visual para confeccionar el cuerpo de la IVR. Además de conocer sus propiedades. Las propiedades del componente no podrán ser cambiadas solo consultadas.	

Tabla 23: Tarea de ingeniería: Agregar componente Responder Llamada.

Tarea de Ingeniería	
Número Tarea: 3	Número Historia de Usuario: 3

Nombre Tarea: Agregar componente Colgar Llamada	
Tipo de Tarea: Desarrollo	Puntos Estimados: 1/2
Fecha Inicio: 07/02/2012	Fecha Fin: 08/02/2012
Programadores Responsables: Mario Iván Cid e Irina Valdes Meneses.	
Descripción: Esta opción le permite al usuario arrastrar el componente Colgar Llamada hacia el panel visual para confeccionar el cuerpo de la IVR. Además de conocer sus propiedades. Las propiedades del componente no podrán ser cambiadas solo consultadas.	

Tabla 24: Tarea de ingeniería: Agregar componente Colgar Llamada.

Tarea de Ingeniería	
Número Tarea: 4	Número Historia de Usuario: 5
Nombre Tarea: Agregar componente Selección Múltiple	
Tipo de Tarea: Desarrollo	Puntos Estimados: 2/3
Fecha Inicio: 09/02/2012	Fecha Fin: 15/02/2012
Programadores Responsables: Mario Iván Cid e Irina Valdes Meneses.	
Descripción: Esta opción le permite al usuario arrastrar el componente Selección Múltiple hacia el panel visual para confeccionar el cuerpo de la IVR. Además de establecer sus propiedades como las condiciones para seleccionar los diferentes caminos que puede tomar la IVR. Las propiedades del componente podrán ser cambiadas con excepción del nombre que solo será consultado.	

Tabla 25: Tarea de ingeniería: Agregar componente Selección Múltiple.

Tarea de Ingeniería	
Número Tarea: 5	Número Historia de Usuario: 7

Nombre Tarea: Agregar componente Reproducir Sonido	
Tipo de Tarea: Desarrollo	Puntos Estimados: 1/2
Fecha Inicio: 16/02/2012	Fecha Fin: 17/02/2012
Programadores Responsables: Mario Iván Cid e Irina Valdes Meneses.	
Descripción: Esta opción le permite al usuario arrastrar el componente Reproducir Sonido, hacia el panel visual para confeccionar el cuerpo de la IVR. Además de establecer sus propiedades como fichero de audio que será reproducido. Las propiedades del componente podrán ser cambiadas con excepción del nombre que solo será consultado.	

Tabla 26: Tarea de ingeniería: Agregar componente Reproducir Sonido.

Tarea de Ingeniería	
Número Tarea: 6	Número Historia de Usuario: 8
Nombre Tarea: Agregar componente Reproducir Dato	
Tipo de Tarea: Desarrollo	Puntos Estimados: 1/2
Fecha Inicio: 18/02/2012	Fecha Fin: 20/02/2012
Programadores Responsables: Mario Iván Cid e Irina Valdes Meneses.	
Descripción: Esta opción le permite al usuario arrastrar el componente Reproducir Dato, hacia el panel visual para confeccionar el cuerpo de la IVR. Además de establecer sus propiedades como el dígito que se va a reproducir. Las propiedades del componente podrán ser cambiadas con excepción del nombre que solo será consultado.	

Tabla 27: Tarea de ingeniería: Agregar componente Reproducir Dato.

Tarea de Ingeniería

Número Tarea: 7	Número Historia de Usuario: 9
Nombre Tarea: Agregar componente Capturar Dato	
Tipo de Tarea: Desarrollo	Puntos Estimados: 1/2
Fecha Inicio: 21/02/2012	Fecha Fin: 24/02/2012
Programadores Responsables: Mario Iván Cid e Irina Valdes Meneses.	
Descripción: Esta opción le permite al usuario arrastrar el componente Capturar Dato, hacia el panel visual para confeccionar el cuerpo de la IVR. Además de establecer sus propiedades como la condición de parada.	

Tabla 28: Tarea de ingeniería: Agregar componente Capturar Dato.

Tarea de Ingeniería	
Número Tarea: 8	Número Historia de Usuario: 12
Nombre Tarea: Agregar conexión entre componentes	
Tipo de Tarea: Desarrollo	Puntos Estimados: 1/2
Fecha Inicio: 25/02/2012	Fecha Fin: 27/02/2012
Programadores Responsables: Mario Iván Cid e Irina Valdes Meneses.	
Descripción: Esta opción le permite al usuario arrastrar desde un componente, un botón representado por una flecha y soltarlo encima de otro componente para establecer una conexión lógica entre los componentes que integran la IVR.	

Tabla 29: Tarea de ingeniería: Agregar conexión entre componentes.

Tarea de Ingeniería	
Número Tarea: 9	Número Historia de Usuario: 18

Nombre Tarea: Generar el código fuente de una IVR	
Tipo de Tarea: Desarrollo	Puntos Estimados: 2/3
Fecha Inicio: 28/02/2012	Fecha Fin: 03/03/2012
Programadores Responsables: Mario Iván Cid e Irina Valdes Meneses.	
Descripción: Esta funcionalidad permite generar el código en lenguaje PHP para cada componente de la IVR que ha sido insertado.	

Tabla 30: Tarea de ingeniería: Generar el código fuente de una IVR.

Tarea de Ingeniería	
Número Tarea: 9	Número Historia de Usuario: 19
Nombre Tarea: Importar Archivo de Voz	
Tipo de Tarea: Desarrollo	Puntos Estimados: 1/2
Fecha Inicio: 05/03/2012	Fecha Fin: 06/03/2012
Programadores Responsables: Mario Iván Cid e Irina Valdes Meneses.	
Descripción: La funcionalidad Importar Archivo de Voz brinda la posibilidad a los usuarios de buscar en algún a dirección específica el Archivo de Voz que desea utilizar en sus IVR, los importa hacia una carpeta perteneciente al proyecto y le cambia el nombre si así lo precisa. El procedimiento se aplica para todos los archivos que desee importar a la aplicación.	

Tabla 31: Tarea de ingeniería: Importar Archivo de Voz.

Tarea de Ingeniería	
Número Tarea: 11	Número Historia de Usuario: 21
Nombre Tarea: Exportar un proyecto	

Tipo de Tarea: Desarrollo	Puntos Estimados: 2/3
Fecha Inicio: 07/03/2012	Fecha Fin: 10/03/2012
Programadores Responsables: Mario Iván Cid e Irina Valdes Meneses.	
Descripción: El sistema con esta funcionalidad le permite al usuario una vez creado el proyecto, seleccionar la opción Exportar Proyecto. De esta forma obtiene todos los documentos y recursos para hacer funcionar su IVR.	

Tabla 32: Tarea de ingeniería: Exportar un proyecto.

Tarea de Ingeniería	
Número Tarea: 12	Número Historia de Usuario: 23
Nombre Tarea: Guardar un proyecto	
Tipo de Tarea: Desarrollo	Puntos Estimados: 2/3
Fecha Inicio: 12/03/2012	Fecha Fin: 16/03/2012
Programadores Responsables: Mario Iván Cid e Irina Valdes Meneses.	
Descripción: Esta funcionalidad permite al usuario Guardar el proyecto con el que está trabajando. El proyecto se guarda en la dirección especificada por el usuario con una carpeta general que tiene dos subcarpetas, los archivos de voz importados y las IVR con extensión .mi para que después puedan ser cargadas y modificadas antes de generar el código definitivo.	

Tabla 33: Tarea de ingeniería: Guardar un proyecto.

Tarea de Ingeniería	
Número Tarea: 13	Número Historia de Usuario: 24
Nombre Tarea: Compilar la IVR	

Tipo de Tarea: Desarrollo	Puntos Estimados: 2/3
Fecha Inicio: 17/03/2012	Fecha Fin: 22/03/2012
Programadores Responsables: Mario Iván Cid e Irina Valdes Meneses.	
Descripción: Esta funcionalidad le permite al usuario una vez compilada la IVR que está creando observar en la pestaña Código el código que se va a generar.	

Tabla 34: Tarea de ingeniería: Compilar la IVR.

Las restantes tablas correspondientes a las tareas de ingeniería que no se presentaron anteriormente pueden encontrarse en el Anexo III.

4.2. Pruebas

Medir el buen funcionamiento y el correcto acabado del software no es algo que se debe tomar a la ligera y es el proceso de prueba el único instrumento adecuado para determinar la calidad de un producto de software. En este proceso se ejecutan pruebas dirigidas a componentes del software o al sistema de software en su totalidad, con el objetivo de medir el grado en que el software cumple con los requerimientos. Básicamente es una fase en el desarrollo de software que consiste en probar las aplicaciones construidas. En estas se usan casos de prueba, especificados mediante técnicas de prueba que permitan comprobar el grado de cumplimiento respecto de las especificaciones iniciales del sistema, para identificar posibles errores en la implementación, calidad, o usabilidad de un programa de software.

“[...] El testing puede probar la presencia de errores pero no la ausencia de ellos [...]”

Edsger Dijkstra

XP durante el desarrollo del software establece probar constantemente el código que se va elaborando para determinar en horas tempranas y durante la marcha la mayor cantidad de errores. Reduciéndolos en gran escala y posibilitando erradicarlos con la menor complejidad posible. Mientras antes se detecte una falla, más barata es su corrección.

La metodología XP lleva a cabo la Fase de Prueba en la que divide las pruebas en dos grupos:

Pruebas Unitarias: Las pruebas unitarias son pruebas dirigidas a probar clases aisladamente y están relacionadas con verificar el código y la responsabilidad de cada clase y sus fragmentos de código más críticos. La realización de estas pruebas permite asegurar la calidad del código entregado. Es la mejor forma de detectar errores tempranamente en el desarrollo, por ende son diseñadas por los programadores. Así como ayuda a definir los requerimientos y responsabilidades de cada método en cada clase probada. (28)

Pruebas de Aceptación: El objetivo de las pruebas de aceptación es validar que un sistema cumple con el funcionamiento esperado y permitir al usuario de dicho sistema que determine su aceptación, desde el punto de vista de su funcionalidad y rendimiento. Las pruebas de aceptación son definidas por el usuario del sistema y preparadas por el equipo de desarrollo, aunque la ejecución y aprobación final corresponden al usuario. Estas están diseñadas para asegurar que se satisfacen todos los requisitos funcionales antes especificados teniendo en cuenta también los requisitos no funcionales relacionados con el rendimiento, seguridad de acceso al sistema, a los datos y procesos, así como a los distintos recursos. Estas pruebas no se realizan durante el desarrollo, pues sería impresentable al cliente; sino que se realizan sobre el producto terminado e integrado o pudiera ser una versión del producto o una iteración funcional pactada previamente con el cliente. (29)

Las pruebas de aceptación correspondiente a cada una de las funcionalidades del Generador Gráfico de IVR serán representadas mediante tablas divididas por las siguientes secciones:

- **Entradas Válidas:** en esta sección se formulará la descripción de cada uno de los pasos seguidos durante el desarrollo de la prueba, se tendrá en cuenta cada una de las entradas válidas que hace el usuario con el objetivo de ver si se obtiene el resultado esperado.
- **Entradas Inválidas:** en esta sección se formulará la descripción de cada uno de los pasos seguidos durante el desarrollo de la prueba, se tendrá en cuenta cada una de las posibles entradas inválidas que hace el usuario con el objetivo de ver si se obtiene el resultado esperado y cómo responde el sistema.
- **Resultado Esperado:** en esta sección se concebirá una breve descripción del resultado esperado tanto para las entradas válidas como para las entradas inválidas.

- **Resultado de la Prueba:** en esta sección se hará una breve descripción del resultado que se obtiene.
- **Observaciones:** en este apartado se refleja algún señalamiento o advertencia que sea necesario hacerle a la sección que se está probando. (29)

El proceso que se llevará a cabo para aplicarle al software las pruebas de aceptación será estructurado de la siguiente manera. Primero se redactarán los casos de prueba, para ello se tendrá en cuenta el orden de las HU y la prioridad que ha sido asignada a las funcionalidades. A continuación se efectuará con el cliente la debida planificación para determinar cuáles serán las pruebas que se harán y en qué momento, de esta forma serán notificados los miembros del proyecto que cuentan con esta responsabilidad. Por último, se completarán cada uno de los campos de las tablas de las pruebas de aceptación con el resultado de la prueba. Las pruebas de aceptación correspondiente a las funcionalidades de la aplicación pueden encontrarse en el Anexo IV.

4.3. Conclusiones

Al concluir el presente capítulo se obtuvo el producto terminado. Se implementaron las 25 Tareas de la Ingeniería definidas para cada Historia de Usuario, las cuales proporcionaron organización al precisarse el orden lógico de tareas a implementar para dar cumplimiento a las funcionalidades. Posteriormente se elaboraron los diseños de los casos de prueba para comprobar que cada funcionalidad respondiera de acuerdo a las especificaciones del cliente. Luego de aplicadas todas las comprobaciones se derivaron un grupo de no conformidades que fueron resueltas y concluyeron en la total aceptación del producto.

CONCLUSIONES GENERALES

Con el presente trabajo de diploma se exhibe una aplicación que brinda a los usuarios o empresas que necesiten crear IVR, la posibilidad de minimizar tanto la complejidad de este proceso como su tiempo de desarrollo. Estas características son el producto de poder diseñar una IVR a través de funciones básicas de arrastrar y soltar componentes visuales. Así como simular y generar automáticamente el código fuente de la misma a partir de dicho diseño e importar ficheros de audio. Para su confección se realizó el análisis y estudio de las herramientas, metodologías y lenguajes a emplear. Se abogó por el uso de herramientas libres en todo el proceso de desarrollo, arribando a la utilización de DiaProject y NetBeans para el modelado y la programación respectivamente, así como la metodología ágil XP, la librería DockingFrame para enriquecer el trabajo visual y Java como lenguaje de programación. Además se le aplicaron diferentes pruebas al software para asegurar que el sistema responde a los requisitos especificados por el cliente. Se determinó que fueron cumplidos satisfactoriamente los objetivos planteados en el presente trabajo, al diseñar y generar el código de una IVR en 30 minutos por un analista del proyecto sin conocimiento del lenguaje PHP, mientras que esta fue desarrollada anteriormente por un programador con experiencia y conocimiento del lenguaje en 1 día. También durante el desarrollo de esta solución se tuvo en cuenta un análisis para incorporar otras funcionalidades que aporten usabilidad a la misma, como introducir otros lenguajes de programación a través del uso de plugins, deshacer y rehacer una operación y la gestión de ficheros de voz.

RECOMENDACIONES

Para mejorar en pos de alcanzar la más alta calidad en el desarrollo de la presente solución informática se recomienda:

- Agregar la funcionalidad Deshacer y Rehacer una operación.
- Adicionar la funcionalidad Grabar archivo de voz.
- Incorporar otros lenguajes de programación mediante el uso de plugins.

REFERENCIAS BIBLIOGRÁFICAS

1. **Landívar, Edgar.** *Comunicaciones Unificadas con Elastix Volumen 1.* 2009.
2. **Zayas, Dr. Carlos Alvarez de.** *METODOLOGIA DE LA INVESTIGACION CIENTIFICA.* SANTIAGO DE CUBA : CENTRO DE ESTUDIOS DE EDUCACION SUPERIOR, 1995.
3. Software Call Center. *Software Call Center.* [En línea] 22 de Marzo de 2011. [Citado el: 2011 de Noviembre de 20.] <http://www.softwarecallcenter.net/2011/03/%C2%BFque-es-un-ivr/>.
4. **Ellison R J, Feiler P H, Habermann A N.** *Software Development Environments.* s.l. : IEEE Computer, 1987.
5. Pergaminovirtual Buscador Hispano. *Pergaminovirtual Buscador Hispano Glosario.* [En línea] 2011. http://www.pergaminovirtual.com.ar/definicion/Codigo_Fuente.html. 1847110..
6. **Marcos M, Estévez E, Gangoti U, Portillo J.** *GENERADOR DE CÓDIGO IEC 61131 BASADO EN TECNOLOGÍAS XML.* s.l. : Departamento de Ingeniería de Sistemas y Automática. 37269.
7. **Isaías Carrillo Pérez, Rodrigo Pérez González, Aureliano David Rodríguez Martín.** *Metodología de Desarrollo del Software.* 2008.
8. **Costa, Yanelay Villamón.** *Creación Gráfica de IVR para Asterisk.* Ciudad Habana : s.n., 2011.
9. **Cabanillas, Marta.** Network World. *Network World.* [En línea] IDG COMMUNICATIONS, 1 de Diciembre de 2006. http://www.networkworld.es/El-autentico-valor-de-la-convergencia_Comunicaciones-unifica/seccion-/articulo-180573.g.
10. **Rios, Sergio Sánchez.** *Metodologías de Análisis y Diseño.* noviembre 2007.
11. **González, Carlos Sánchez.** Ciclo de vida de un proyecto XP. *Ciclo de vida de un proyecto XP.* [En línea] Septiembre de 2004. <http://ciclo%20de%20vida%20de%20xp.html>.
12. **Romero, Eduardo Alvarez.** *Integration Definition for Function Modeling.* 2005.

13. **Enjolras, Maryvonne.** Beneficios del uso de JAVA en las aplicaciones modernas de Bibliotecas. *Beneficios del uso de JAVA en las aplicaciones modernas de Bibliotecas.* [En línea] 2009. http://www.ciepi.org/fesabid98/Comunicaciones/m_enjolras.htm..
14. **Romero, Eduardo Álvarez.** Java x Ejemplo. *Java x Ejemplo.* [En línea] 2011. <http://javabyexample.wisd.com/plug.com/java-concepts/34-core-java/64-10-free-docking-frameworks-for-java.html>
15. The GNOME Project. *The GNOME Project.* [En línea] 2005-2011. [Citado el: 2011 de Noviembre de 15.] <http://projects.gnome.org/dia/>.
16. **Navarro, Juan.** Más que Código. *Más que Código.* [En línea] 9 de Agosto de 2003. [Citado el: 2011 de Diciembre de 17.] <http://www.juanjonavarro.com/masquecodigo/2003/12/09/netbeans-frente-a-eclipse>.
17. **Wiergers, Karl.** *Software Requirements 2: Practical techniques for gathering and managing requirements throughout the product development cycle.* 2003.
18. **Pressman, Roger S.** *Ingeniería de Software. Un enfoque práctico.* Madrid : 5ta Edición, 2001.
19. **Penadés, Patricio Letelier.** *Métodologías ágiles para el desarrollo de software :eXtreme Programming (XP).* Valencia : Universidad Politécnica de Valencia.
20. **David Garlan, Mary Shaw.** *Software Architecture.* New York, E.U.A : Editorial Prentic-Hall, 1996.
21. **Microsoft.** *Microsoft Application Architecture Guide 2nd Edition.* 2009. 9780735627109.
22. **Marquina, Ernesto.** *Guía de Patrones, Práctica y Arquitectura . NET.* 2008.
23. **Eric Freeman, Elisabeth Freeman.** *Head First Design Patterns O'Reilly.*
24. **Hernán, Marcello Visconti Astudillo y.** *Fundamentos de Ingeniería de Software Patrones GRASP.* s.l. : Universidad Técnica Federico Santa María.
25. **Juan Manuel Dodero, Camino Fernandez Llamas.** *Patrones de comportamiento: Visitor.* Madrid : Universidad Carlos III de Madrid, 2003.

26. —. *Patrones de comportamiento: Observer*. Madrid : Universidad Carlos III de Madrid, 2003.
27. **Kurniawan, Budi**. *Utilizando el patrón Singleton*. s.l. : Java Users Group Argentina, 2004.
28. **Rodriguez, Jorge**. *Pruebas unitarias*. 2006.
29. **Trabajo, AEN/CTN71/SC7/GT26 Grupo de**. *Pruebas de Software*. 2009.

GLOSARIO

IVR: Respuesta de Voz Interactiva de sus siglas en inglés Interactive Voice Response. IVR es una tecnología que permite a una computadora interactuar con los seres humanos a través del teclado del teléfono o por reconocimiento de voz. Utiliza pregrabados o generados de audio dinámicamente para guiar a los usuarios sobre cómo proceder.

Asterisk: Es una aplicación de software libre (bajo licencia GPL) que proporciona funcionalidades de una central telefónica (PBX). Como cualquier PBX, se puede conectar un número determinado de teléfonos para hacer llamadas entre sí e incluso conectar a un proveedor de voz sobre IP.

ETECSA: Empresa de Telecomunicaciones de Cuba S.A.

IP: Protocolo de Internet.

Lenguaje de Desarrollo: Un lenguaje de desarrollo es un idioma artificial que interpreta la computadora. Generalmente se utilizan para crear programas que controlan el funcionamiento de un fenómeno determinado. Estos lenguajes están formados por un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura.

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

Hardware: Incluye todas las partes físicas del computador, es decir, aquellos dispositivos que se conectan entre sí para formar una sola unidad de trabajo.

MVC: del inglés Model-View-Controller, en español Modelo-Vista-Controlador, patrón utilizado en el diseño y desarrollo web.

PBX-PABX: central telefónica privada utilizada dentro de una empresa, en la que los usuarios de la misma pueden realizar llamadas tanto internas como externas.

PHP: es un lenguaje de programación interpretado, diseñado originalmente para la creación de páginas web dinámicas. Es usado principalmente en interpretación del lado del servidor.

Software: Se refiere al equipamiento lógico o soporte lógico de un computador digital.

XP: es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo.

UCI: Universidad de las Ciencias Informáticas.

Comunicaciones Unificadas: El término Comunicaciones Unificadas es utilizado para designar la integración de los servicios de telefonía, mensajería unificada, mensajería instantánea corporativa, conferencias web y estado de disponibilidad del usuario en una sola e innovadora experiencia para los colaboradores y para el personal que administra y da mantenimiento a la infraestructura.

Historia de Usuario: Una historia de usuario es una representación de un requerimiento de software escrito en una o dos frases utilizando el lenguaje común del usuario.

Prototipo: Maqueta visual funcional o no de la futura aplicación. Este puede ser una imagen o una aplicación software que simule funcionalidades del software.

Software Libre: Se denomina software libre a la libertad que tienen los usuarios de ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el software. Significa que los usuarios de programas tienen las cuatro libertades esenciales ya sea para ejecutar el programa, para cualquier propósito, así como estudiar cómo trabaja y cambiarlo si se desea, por lo que se conoce el código fuente, además se puede redistribuir copias tanto de las originales como de las versiones modificadas.

Aplicación Informática: Un programa de ordenador que se compra ya realizado y listo para usar. Las hay de muy diversos tipos, según para qué propósito se hayan diseñado: procesadores de texto, bases de datos, programas de contabilidad, de facturación, etc.

Elastix: Es una distribución libre de Servidor de Comunicaciones Unificadas que dispone de varios servicios como Mensajería Instantánea, Correo Electrónico y PBX.

VoIP: Protocolo de Internet sobre Voz.