SISTEMA PARA LA GESTIÓN DE SERVICIOS DE RED INTELIGENTE SOPORTADOS EN INTERNET

PEDRO JOSÉ CAMACHO OJEDA

ISABEL CRISTINA HERNÁNDEZ PINO

ANEXO B

TECNOLOGIAS Y HERRAMIENTAS PARA LA IMPLEMENTACION DEL SERVICIO Y DEL SISTEMA DE GESTION

UNIVERSIDAD DEL CAUCA
FACULTAD DE INGENIERÍA ELECTRÓNICA Y TELECOMUNICACIONES
DEPARTAMENTO DE TELEMÁTICA
POPAYÁN
2005

TABLA DE CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

B1. INTRODUCCIÓN

El presente anexo tiene la finalidad de exponer las tecnologías, herramientas y plataformas que se utilizaron para implementar el prototipo del *servicio de lectura de correo electrónico a través de la linea telefónica,* así como del sistema de gestión para el servicio.

Para el proceso de selección de las diferentes tecnologías, plataformas y herramientas, se procuró responder a los siguientes requisitos generales:

- Ser preferiblemente *software* libre.
- Tener el mínimo costo posible.
- Ejecutarse en alguna distribución para el sistema operativo Linux.
- Contar con una buena documentación de instalación y uso.
- Facilitar ejemplos de aplicaciones previamente realizadas.
- Contar idealmente con personas dentro de la Universidad que hayan utilizado tal tecnología, plataforma o herramienta.

Es importante aclarar con referencia a los dos primeros puntos de esta lista, que a veces el software libre no es gratuito, pues su adquisición puede tener algún costo. Además en los costos que se podrían asumir al decidirse por alguna herramienta *software,* se debe tener en cuenta algunos factores como el valor de la instalación en los diferentes equipos donde funcionará, la capacitación de las personas que lo van a usar, el valor del soporte técnico, entre otros.

En la sección B2 se describen los detalles de la implementación del prototipo del servicio de *lectura de correo electrónico a través de la línea telefónica*, y en la sección B3 se incluyen los del sistema de gestión de configuración del servicio.

B1.1 LENGUAJE DE PROGRAMACIÓN

Como lenguaje de programación se escogió a Java, por ser un lenguaje de desarrollo de propósito general válido para realizar todo tipo de aplicaciones, donde una de sus características más importantes es que los programas creados por el compilador de Java se pueden ejecutar sobre una gran variedad de sistemas operativos. Algunas de sus características incluyen:

- Ser intrínsecamente orientado a objetos.
- Funcionar perfectamente en red.

- Aprovechar características de la mayoría de los lenguajes modernos evitando sus inconvenientes (En particular los del C++).
- Tener una gran funcionalidad gracias a sus librerías (clases).
- No tener punteros manejables por el programador, y en su lugar manejarlos de manera interna y transparente.
- Gestionar el manejo de la memoria.
- Generar aplicaciones con pocos errores posibles, principalmente porque la gestión de memoria y punteros la realiza el propio lenguaje.
- Incorporar el concepto de multi-hilo para permitir la ejecución de tareas concurrentes dentro de un mismo programa.

La clave para el desarrollo universal de Java es su arquitectura de dos niveles, la cual consta por un lado, de la Máquina Virtual de Java (JVM) y por otro, de las clases de Java y las APIs.

Al escoger a Java como lenguaje de programación para el desarrollo del servicio, se tuvo la ventaja adicional de contar con tecnologías basadas en Java que soportaban las funciones requeridas para el servicio, lo cuál facilitó su integración.

B1.2 SISTEMA DE GESTIÓN DE BASE DE DATOS

Se escogió a PostgreSQL[1] como sistema de gestión de bases de datos por ser un poderoso sistema de bases de datos relacional de código abierto, liberado bajo la licencia BSD (Berkeley Software Distribution, Distribución de Software Berkeley), que corre sobre la mayoría de sistemas operativos, incluyendo Linux, Unix, BeOS y Windows.

Algunas de las ventajas que presenta son:

- No hay un costo de licencia asociado por el software.
- Presenta un mejor soporte que los vendedores propietarios.
- Permite que haya un ahorro significativo en los costos de soporte.
- Ofrece confiabilidad y estabilidad.
- Es extensible.
- Proporciona soporte multiplataforma.
- Está diseñado para entornos con altos volúmenes de datos.
- Cuenta con herramientas gráficas para la administración y el diseño de Bases de Datos.

Tomando en cuenta esas características, muchas empresas e instituciones lo están adoptando como servidor de Bases de Datos.

1 Mayor información sobre PostgreSQL, se encuentra en http://www.postgresql.org/ y el comparativo con otros sistemas de Bases de Datos en: http://www.nextec.com.ar/postgres/comparacion.html

B2. IMPLEMENTACIÓN DEL SERVICIO

La implementación del *servicio de lectura de correo electrónico a través de la línea telefónica* se dividió en dos partes, a fin de agrupar las funciones del servicio según la red sobre la cuál se ejecutan. Estas partes son:

- La encargada de soportar las funciones de interacción con el usuario, para prestar los servicios de reproducción de mensajes, reconocimiento de voz y síntesis de voz, lo cuál ocurre sobre el dominio de la red telefónica involucrando a la central telefónica y específicamente a su módulo de IVR[2].

- La responsable de las funciones del servicio que se llevan a cabo sobre el dominio de la red IP, específicamente las necesarias para obtener y manejar los correos electrónicos del usuario, atender las solicitudes generadas desde el Cliente SPIRITS en la central telefónica, y soportar el inicio/terminación del servicio por parte del Administrador del servicio.

La división del servicio se bosqueja en la Figura B1:

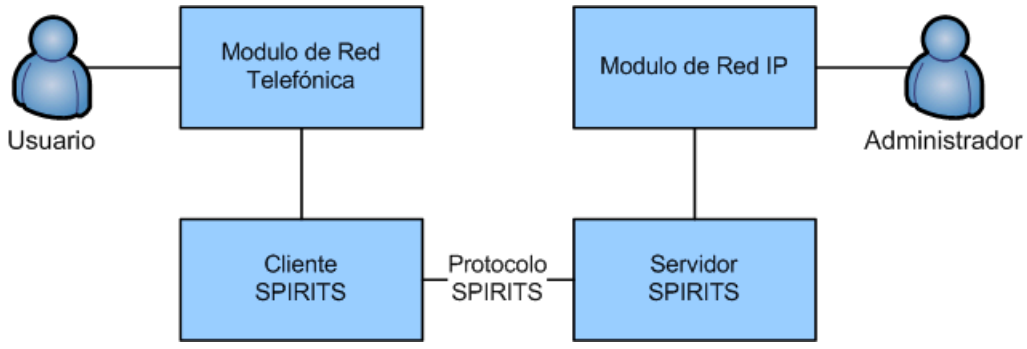

Figura B1 : División del Servicio según su funcionalidad

Como se puede observar en la Figura 1, la comunicación entre la parte que se ejecuta sobre la red telefónica y la red IP, se logra a través de la implementación de la arquitectura SPIRITS. Los elementos de la arquitectura que se están teniendo en cuenta son: el Cliente SPIRITS (sobre la red telefónica) y el Servidor SPIRITS (sobre la red IP), comunicándose a través del protocolo SPIRITS. Es así como en este capítulo, se incluyen además los detalles de implementación de la arquitectura SPIRITS.

2 IVR: Reconocimiento Interactivo de Voz. Sirve de interfaz de comunicación con el usuario, al recibir las órdenes que da el usuario mediante comandos de voz, y responderle mediante la reproducción de mensajes, incluida la conversión del correo electrónico de texto a voz.

B2.1 PARTE DE LA RED TELEFÓNICA

Para implementar las funciones encargadas de servir de interfaz para el usuario del servicio, era necesario establecer:

- Una API para el manejo del módem capaz de: recibir la llamada del usuario, detectar y reconocer los tonos del teclado cuando se oprimen las teclas, reproducir mensajes de audio a través de la linea telefónica y capturar el audio cuando el usuario da las órdenes de voz.

- Una tecnología y una herramienta encargada de realizar la conversión de texto a voz y el reconocimiento de comandos sencillos de voz.

B2.1.1 Módem

Uno de los problemas que se presenta para prestar el servicio es determinar cómo se va a permitir que la línea telefónica del usuario se conecte con el IVR y así el usuario pueda escuchar los mensajes pregrabados, o escuchar cuando se realiza el proceso de conversión de texto a voz, además de capturar y analizar el audio cuando el usuario pronuncia las órdenes sencillas de voz. Para hacer esto posible, el sistema debe contar con un módem de voz que facilite la reproducción y captura de audio a través de la línea telefónica, además de permitir la recepción y detección de tonos DTMF (Dual Tone Multi Frequency, Tono dual multifrecuencia) para la captura de los diferentes dígitos, necesarios para el ingreso del número telefónico de suscripción al servicio y el PIN, en el proceso de validación del usuario ante el sistema.

Actualmente, algunas de las tecnologías que permiten realizar control sobre el módem son los comandos AT Hayes y a través del API JTAPI. A continuación se estudiarán estas dos tecnologías.

B2.1.1.1 Comandos AT Hayes

Este tipo de comandos permite controlar y soportar al módem siendo un conjunto de comandos estándar que soportan la gran mayoría de modems. Estos mismos comandos son utilizados por cualquier “software” de comunicaciones que sirve para el envío de datos, fax o voz.

Un módem siempre funciona en uno de estos modos: el modo de comando o el modo en línea:

- **Modo Comando:** Se usa para cambiar la configuración del módem, para marcar un número telefónico, recibir llamadas, recibir tonos DTMF, entre otras funciones. Usualmente en este modo, después de ejecutar un comando el módem regresará un código que confirma el resultado de la operación.

- **Modo en linea:** Se pasa automáticamente a este modo cuando se establece una comunicación con otro módem o con una máquina de fax remota, a menos que al

configurar las opciones de marcado se especifique de otra manera. En este modo, el módem recibe caracteres o datos desde la computadora, convierte los datos en señalas analógicas y luego trasmite estas señales a través de la línea telefónica. Es también útil para la trasmisión y recepción de audio.

Un módem se puede utilizar principalmente para la transmisión de datos, fax y voz, por esto los comandos AT también se dividen usualmente en tres grupos para cubrir estos tipos de funciones.

Para la utilización de comandos AT se debe cumplir con algunas normas básicas[3]:

- Los comados AT se pueden introducir en mayúsculas, minúsculas o ambas a la vez.
- Todos los comandos AT empiezan con las letras AT, a excepción de A/ y +++
- La terminación del ingreso de un comando AT se hace con un carácter especial previamente definido, por defecto es el carácter de retorno o “enter”
- Una linea de comandos puede incluir uno o mas comandos AT siendo separados por espacios, pero su máximo es 40 caracteres luego del comando AT (Este valor puede variar de un módem a otro).

Ahora bien, la comunicación con el módem se hace a través de un puerto serial del computador, para lo cuál se utiliza el API de comunicaciones de Java, que unifica el modelo de programación para un amplio de rango de dispositivos como lo son impresoras, cámaras de video, escáneres, entre otros. Este API soporta el puerto serial (RS232/434) y el puerto paralelo.

Dentro del desarrollo del prototipo del servicio, se implementó el caso de uso validar usuario usando tanto el API de comunicación de Java, como los comandos AT Hayes, y aunque se realizaron pruebas de funcionamiento del servicio conectando el módem a la tarjeta de audio, como se muestra en la Figura B2, estos resultados no fueron los esperados pues el rango de variación del volumen es muy limitado y el volumen era, o demasiado bajo para reconocer los comandos de voz, o demasiado alto distorsionando la señal. No se insistió demasiado en este tipo de pruebas porque esta no es la forma como se implementaría esta clase de servicios en un entorno comercial.

3 Para consultar una lista completa de comandos AT puede ver el siguiente enlace en internet: http://modemhelp.org/setdef.html

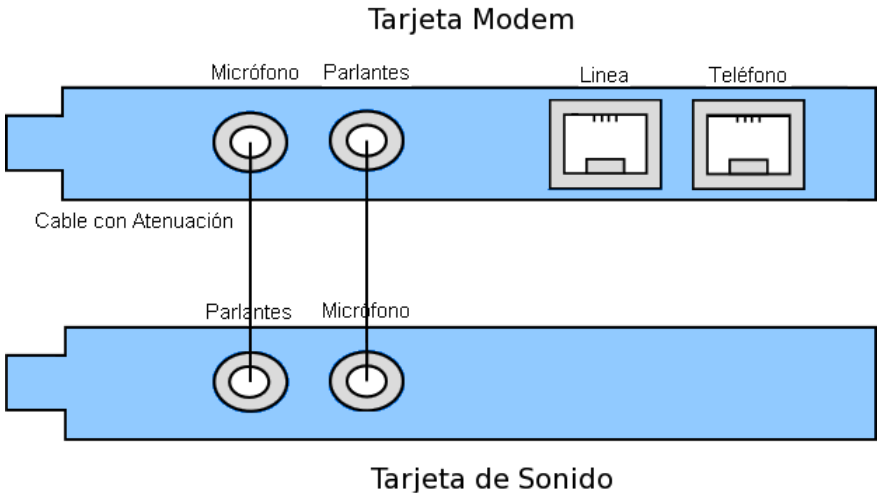

Figura B2 : Conexión Tarjeta de Sonido - Módem

Aunque los comandos AT Hayes son muy potentes y ampliamente difundidos presentan problemas porque algunos tipos de fabricantes soportan comandos AT que no son soportados por otros fabricantes y esto hace que los desarrollos sean particulares para un tipo especifico de módem. Como solución a este problema se han definido APIs de un nivel más alto como lo es TAPI y UNIMODEM, donde cada fabricante especifica en un archivo el conjunto de comandos AT Hayes que soporta las diferentes funcionalidades estandarizadas por el API.

B2.1.1.2 TAPI (Telephony Application Programming Interface, Interfaz de Programación de Aplicaciones Telefónicas)

Es un API que permite usuar los servicios telefónicos a los computadores que usan sistemas operativos de la familia windows. La versión 2.0 de esta API fue un paso hacia delante en el soporte de las funcionalidades prestadas por los ACD (Automatic Call Distributor, Distribuidores de Llamada Automáticos) y los PBX. La versión 3.0 habilita la telefonía IP proporcionando métodos simples y genéricos para hacer conexiones entre dos o más computadores, y ahora también ofrece la habilidad para acceder a cualquier flujo de audio implicado en la conexión.

B2.1.1.3 UNIMODEM

Es un proveedor de servicios TAPI que fue desarrollado por microsoft. Este proveedor de servicios está diseñado para soportar un amplio rango de modems y faxmodems bajo el sistema operativo windows. La desventaja que tiene es que la funcionalidad TAPI que proporciona es muy limitada, dando solamente el soporte necesario para aplicaciones relativamente simples, pues no proporciona ningún soporte para detectar y responder las llamadas que ingresan o para detectar cuándo se desconectó una llamada, tampoco puede detectar DTMF, ni reproducir o captar archivos de audio sobre la línea telefónica. En la versión Unimodem V se han incluido funcionalidades como reproducción y grabación de archivos en formato wav, identificador de llamada y reenvío de llamada.

Lamentablemente TAPI se desarrolló sólo para windows y no tiene soporte para Java, por lo tanto sólo es posible hacer uso de este API en lenguajes como C. Otra solución que se impone en el mercado y que soporta Java es JTAPI.

B2.1.1.4 JTAPI (Java Telephony API)

Este es un API extensible que ofrece una interfaz para todo el control de servicios telefónicos, donde los servicios incluyen desde aquellas necesidades en un dispositivo de usuario final, hasta aquellos "call centers" empresariales.

El propósito de JTAPI es servir como interfaz entre una aplicación Java y un sistema telefónico, donde el grado de control de la aplicación depende de donde este ubicada dicha interfaz. Por ejemplo en un escenario donde la interfaz este localizada en un terminal, la aplicación tiene el mismo grado de control que un usuario normal de telefonía, mientras que dentro de otro escenario donde el control de la llamada este localizado dentro del sistema telefónico (en el sistema de conmutación), la aplicación tendrá mas capacidades de control que le proporcionan acceso interno al sistema telefónico.

En consecuencia JTAPI proporciona un modelo de un sistema de conmutación y un modelo de las llamadas telefónicas, el cuál corresponde a una vista general del segundo escenario que se explicó anteriormente.

- Principios

La base del diseño de JTAPI es un modelo de llamadas el cual es una abstracción de alto nivel independiente de la tecnología. Este modelo describe la llamada como un conjunto de máquinas de estado finito que suceden cuando una llamada está en curso. El modelo de llamada es muy general con el fin de cubrir diferentes escenarios de llamadas, entre los cuales estan:

- Llamadas entre dos partes.
- Múltiples llamadas simultaneas en un mismo terminal.
- Una conferencia entre múltiples partes.
- El establecimiento de una llamada que alerta a múltiples terminales.

Luego de estudiar sobre estos APIs que soportan el desarrollo de aplicaciones para telefonía, es posible escoger como la mejor opción para el desarrollo del prototipo de servicio a JTAPI, siendo necesario buscar una implementación de esta especificación. Al buscar una implementación se encuentran varios inconvenientes, la mayoría de implementaciones son dependientes del "harware" y usualmente son dispositivos que han sido diseñados para dar soluciones empresariales particulares, siendo dificil acceder a ellas. Para los módem para los que se tiene fácil acceso, sólo se consiguen las siguientes implementaciónes: GJTAPI2, XTapi3

Estas implementaciones hacen uso de TAPI pero lamentablemente, si bien la primera llegó a un estado de desarrollo estable o de producción y su ultima implementación es del 24 de mayo del 2005, aún no soporta la reproducción y captura de audio. En cuanto a la segunda, sólo llegó al nivel beta de desarrollo y el proyecto fue cerrado, siendo publicada la última

versión en el año 2002, donde sólo soportaba algunos controladores de módem.

Una solución a este problema seria pensar en realizar una implementación de JTAPI, pero este proyecto es amplio y ambicioso para llevarlo a cabo dentro de este proyecto de trabajo de grado. Pues involucra tanto el manejo de dispositivos *hardware*, como el manejo de APIs como TAPI por medios de lenguajes de programación que estén soportados por el API, interfaces de comunicación entre dicho lenguaje y Java y el desarrolla de una gran cantidad de código en Java para la implementación de JTAPI. Por lo anterior no se piense en dicha implementación pero se deja la propuesta para un trabajos futuros.

Por las razones anteriores se decidió hacer las pruebas solamente con los comandos AT Hayes, descartándo la parte de captura y reproducción de audio, pues la carencia de éste módulo no afecta los objetivos perseguidos en la construcción del prototipo de servicio. Dado el caso que el servicio se implementara con fines comerciales, los fabricantes del módulo "hardware" encargados de esto, como Intel, Avaya, NortelNetworks, Novell, entre otros, se encargan usualmente de entregar la implementación de JTAPI u otro API donde es fácil la interconexion entre el IVR y la línea telefónica del usuario.

B2.1.2 Tecnologías y herramientas de voz

El primer proceso de selección que se debió enfrentar fue escoger una tecnología para el desarrollo de aplicaciones basadas en voz, específicamente, de aplicaciones que posibilitaran la conversión de texto a voz y el reconocimiento de voz. Como resultado de la búsqueda se encontró que la tecnología que se está imponiendo actualmente en el mercado es VoiceXML, y que otra tecnología también muy importante en el campo del desarrollo de aplicaciones de voz es JSAPI. A continuación se realizará una breve descripción de estas dos tecnologías.

B2.1.2.1 VoiceXML

VoiceXML es una especificación propuesta por la W3C[4] que tiene como objetivo crear archivos siguiendo las reglas sintácticas de XML. Es así como un documento VoiceXML esta formado por etiquetas o marcas que permiten: reproducir sonido digitalizado, sonido sintetizado usando la tecnología de texto a voz (TTS[5]), reconocer información ingresada por el usuario mediante tonos DTMF[6], y reconocer palabras o frases pronunciadas por una persona, siendo posible realizar todo esto a través de un dispositivo de entrada/salida como un teléfono fijo, móvil o a través de VoIP[7]. Esta tecnología busca crear y manejar conversaciones hombre-maquina que le permita a los usuarios interactuar con aplicaciones informáticas, donde en lugar de utilizar una combinación de monitor, teclado y ratón para

4 W3C: Es una organización que produce estándares para la World Wide Web. Mayor información en: http://www.w3c.org/

5 TTS: Text to Speech. Tecnología mediante la cual es posible convertir teóricamente cualquier texto, en formato escrito, a palabras, en formato oral.

6 DTMF: Dual Tone Multi-Frecuency, Tono dual multifrecuencia. Es un sistema utilizado por los teléfonos para asignar un tono (frecuencia) a cada tecla de un teléfono. Es por esta razón que uno escucha un sonido distinto cuando presiona cada tecla de un teléfono.

7 VoIP: Voice over IP, Voz sobre IP.

interactuar con el usuario, se emplee un navegador basado en la voz.

VoiceXML se basa en la misma filosofía de HTML, un grupo de formularios que presentan una gramática que es el conjunto de palabras y expresiones que el sistema reconocerá y un audio previamente grabado o generado mediante la tecnología TTS. Así, el usuario podrá elegir las opciones para ir realizando transiciones entre los diferentes estados del diálogo, y la aplicación puede contar con diferentes ayudas para la recuperación en casos de error en el reconocimiento de las palabras dichas por el usuario, u otros errores.[8]

Algunas plataformas y herramientas que soportan el desarrollo de aplicaciones voiceXML son :

- voicegenie: http://www.voicegenie.com/
- nuance: www.nuance.com
- BeVocal: www.bevocal.com
- loquendo: http://www.loquendo.com
- con palabras: http://www.conpalabras.com

Algunos de los problemas que se presentan en el desarrollo de aplicaciones con voiceXML son:

- No se identificaron plataformas de desarrollo para VoiceXML disponibles en Colombia, sólo en España o en Estados Unidos, y para realizar pruebas se deben hacer llamadas telefónicas de larga distancia internacional, lo que supone un alto costo para el desarrollo de una aplicación de este tipo.
- Una vez la aplicación haya sido desarrollada, probada y esté en condiciones para ser explotada, se debe buscar un ISP (Internet Server Provider, Proveedor de Servicio de Internet) que preste el servicio de voiceXML, y en Colombia no existe por el momento.

Otra opción que no necesita de llamadas a la plataformas de VoiceXML consiste en bajar el módulo Voice Toolkit de IBM, que se integra con el entorno de desarrollo WebSphere y usa el motor de voz del programa IBM ViaVoice. El entono de desarrollo cuenta con un editor VoiceXML y las diferentes herramientas para el desarrollo y la realización de pruebas, sin embargo, la desventaja de utilizar esta aplicación es el costo de la licencia del entorno WebSphere, el cuál está al rededor de 4335 dólares. Otra desventaja es que las aplicaciones desarrolladas serían dependientes del entorno WebSphere. [9]

Hay otros motores de voz que soportan VoiceXML, pero que sólo soportan reconocimiento de voz en ingles, y conversión de texto a voz sólo en español. Sin embargo, se buscó que las herramientas soportaran conversión y reconocimiento de voz en español.

Se podría decir que la utilización de VoiceXML es una opción dependiente de las plataformas que existen en el mercado y éstas tienen un costo muy alto, razón que hizo que se descartara.

8 Se puede encontrar un Ejemplo de entorno para creación de aplicaciones de voz.: http://www.verbio.com/productes.php?id=3

9 Se puede consultar una noticia sobre el evolución de las tecnologías de voz en: http://www.javahispano.org/news.item.action?id=184114925

B2.1.2.2 JSAPI (Java Speech API)

El JSAPI es la especificación de Sun Microsystems que permite a las aplicaciones Java incorporar la tecnología de habla dentro de las interfaces de usuario. Este API contiene paquetes para el reconocimiento y la síntesis de voz, haciéndolo un API multiplataforma que puede soportar ingreso de comandos a través del reconocimiento del habla, sistemas de dictado y síntesis de voz. Esta especificación fue liberada el 26 de octubre de 1998, pero en la actualidad es una tecnología que aún está en desarrollo, pues las implementaciones y el desarrollo de aplicaciones para el idioma español son recientes.

Esta tecnología presenta las bondades que brinda el lenguaje de programación Java, y aunque aún está en desarrollo, tiene una buena documentación y cuenta con motores de voz en español que soportan esta tecnología. Otra ventaja es que por basarse en Java, permite la fácil integración con otros módulos desarrollados en java para el servicio. Por estas razones ésta fue la tecnología que se escogió para soportar el desarrollo del servicio en el presente trabajo de grado.

La siguiente tarea consistía en encontrar una herramienta TTS que soportara esta tecnología, para el idioma español, e idealmente, soportara también otros idiomas. En la Tabla B1 se presentan las principales herramientas de TTS y reconocimiento de voz, con las características que se deben tener en cuenta en este proceso de selección:

Nombre de la Herramienta	Sistema Operativo	Soporta síntesis de voz en español	Soporta Reconocimiento de voz en español	La licencia tiene un costo
FreeTTS	Windows/ Linux	No	No	No
IBM's Speech for Java	Windows	Si	Si	No[10]
IBM's Speech for Java	Linux	No	No	descontinuado
The Cloud Garden	Windows	Si	Si	Si[11]
Festival	Linux	Si	No	No

Tabla B1 : Herramientas para TTS y Reconocimiento de Voz

Teniendo en cuenta los criterios de selección previamente fijados, la mejor herramienta para el proceso de síntesis de voz es Festival, debido a que soporta tanto el idioma español como otros idiomas, cuenta con una buena documentación, hay personas que han trabajado con

10 No existe ningún valor para el IBM's Speech for Java, pero si tiene valor el motor de síntesis y reconocimiento de voz que viene con el programa IBM Via Voice. El IBM's Speech for Java se retiró de IBM y en la página de la empresa ya no es posible conseguirla.

11 Debido a que utiliza cualquier motor de TTS o reconocimiento que cumpla con el estándar de Microsoft

esta herramienta dentro de la Universidad, y es software libre, pero no soporta reconocimiento de voz en español. De las otras herramientas, se descarta el FreeTSS pues sólo maneja el idioma ingles, el IBM's Speech for Java para Linux es un proyecto que no se continuó y que se basa en el programa IBM Via Voice, cuya consecución de instaladores para Linux no es fácil, además sólo soporta el idioma ingles. The Cloud Garden fue una herramienta que se utilizó para aprender sobre JSAPI pero que no se utilizó, pues se debe comprar una licencia si se quiere usar para periodos superiores a 30 días. La última opción, el IBM's Speech for Java para windows, cumple con algunos de los requisitos pero su problema radica en el valor de la licencia del motor de voz y el sistema operativo sobre la cuál corre.

En la Tabla B2 se presentan algunas herramientas libres para el reconocimiento de voz que se pueden utilizar:

Nombre de la Herramienta	Sistema Operativo	Soporta Reconocimiento de voz en español	La licencia tiene un costo
Sphinx4	Linux	No	No
Open Mind Speech	Linux	No	No

Tabla B2 : Herramientas para Reconocimiento de Voz

La mayoría de las aplicaciones para reconocimiento de voz tienen el limitante de soportar solamente el idioma ingles, y aunque se pueden hacer pruebas, se estaría agregando más dificultades a las que ya presentan los sistemas de reconocimiento de voz. Además es importante tener en cuenta que la aplicación perderá cierta calidad de audio pues esta limitada al ancho de banda que soporta el sistema telefónico.

Es así como después de revisar las diferentes herramientas de reconocimiento de voz se escogió el IBM's Speech for Java con el motor de reconocimiento y síntesis que presta la herramienta IBM VIA Voice, por las siguientes razones:

- Una sola aplicación soporta tanto el reconocimiento como la síntesis de voz
- Es posible desarrollar aplicaciones usando la tecnología JSAPI
- Cuenta con varias fuentes de documentación: En la página del fabricante se encuentran ejemplos de cómo desarrollar aplicaciones con JSAPI, el Javadoc de JSAPI, además de la documentación de propia del producto que indica la manera de instalarlo.

Pero las limitaciones de esta herramienta son que:

- Corre sobre sistemas operativos de Microsoft, teniendo que pagar el valor de la licencia del sistema operativo
- Se debe pagar alrededor de 50 dólares por la licencia del IBM ViaVoice, edición estándar.

Si bien, se buscaba una herramienta que no tuviera costo alguno, el valor de esta licencia es bastante económico y se asegura una aplicación de buena calidad, ya que IBM cuenta con un amplio recorrido en aplicaciones de voz.

Es importan tener en cuenta que para una implementación comercial del servicio es necesario que el motor de voz soporte varias peticiones de reconocimiento o síntesis al tiempo, para lo cuál se necesita que la herramienta elegida cuente con componente adicional, o se puede utilizar otro software como el IBM Speech Recognition SDK for ViaVoice Dictation and/or Command & Control, u otros que incluyen el hardware adicional necesario para soportar los puertos telefónicos que atienden las llamadas simultáneamente.

Por ultimo, cabe mencionar que la empresa fabricante de IBM's Speech for Java a finales del 2004 decidió retirar el soporte y la distribución de este producto, y liberó parte del código a la Fundación de software Apache.

B2.2 PARTE DE LA RED IP

Dejando de lado la forma de acceder al servicio SPIRITS que se escogió como prototipo a ser desarrollado, éste consiste en un servicio de lectura de correo electrónico y por tanto, necesita tener acceso al servidor de correo electrónico del usuario para descargar los correos que tiene en su bandeja de entrada, discriminar cuáles de esos correos son nuevos, y demás consideraciones que dependen de los parámetros de configuración que haya fijado el usuario. De modo que la forma de abordar este problema es implementar un cliente de correo electrónico que supla estas necesidades.

Una arquitectura de correo electrónico típica está compuesta por cuatro elementos:

- Oficina Postal (Postal Office): donde los mensajes de salida se almacenan temporalmente antes de la transmisión y donde se almacenan los mensajes entrantes. En la “Oficina Postal” corre el software servidor capaz de enrutar los mensajes (un agente de transferencia de mensajes) y mantener la base de datos de la oficina postal.

- Agentes de Transferencia de Mensajes: Para el reenvío de mensajes entre oficinas postales y a los clientes destinatarios. El software puede residir en la oficina postal local o en un servidor separado físicamente.

- Pasarelas (Gateways): Proporcionan parte de la funcionalidad a los agentes de transferencia de mensajes. Se encargan de hacer las traducciones necesarias entre diferentes sistemas de correo electrónico, diferentes esquemas de direccionamiento y protocolos de mensajería.

- Clientes de correo electrónico: Es normalmente el computador que conecta a la oficina postal. Contiene 3 partes:

    - El API de correo electrónico, como MAPI, VIM, MHS y CMC.
    - Protocolo de Mensajería, como el SMTP.

- Protocolo de transporte de red, como Ethernet, etc.

Vista en su forma más simple, una arquitectura de correo electrónico se puede ver como una arquitectura cliente/servidor como la que se aprecia en la Figura B3:

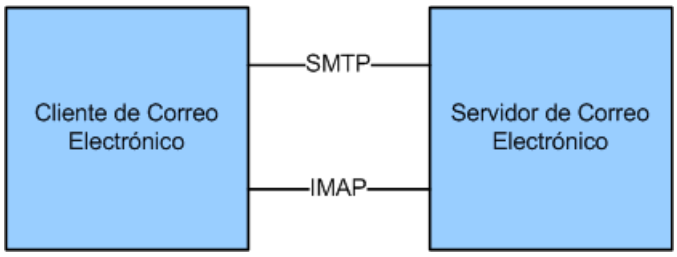

Figura B3 : Arquitectura cliente/servidor del correo electrónico

Debido a que se escogió a Java como lenguaje para desarrollar el servicio, se tiene que éste presenta un API para el manejo de correos electrónicos llamado JavaMail, el cuál se presenta como una alternativa para construir el cliente de correo electrónico requerido en el servicio.

B2.2.1 API JavaMail

B2.2.1.1 Introducción al API JavaMail

El API JavaMail es una extensión estándar de Java (paquete opcional) para construir aplicaciones de cliente de correo electrónico basadas en la tecnología Java, proporcionando así las facilidades para leer, componer y enviar correos electrónicos.

Este API está diseñado para proporcionar acceso independiente del protocolo para enviar y recibir mensajes, dividiéndose en dos partes:

- La primera parte del API está relacionada con el envío y recepción de mensajes independientemente del proveedor/protocolo.
- La segunda parte habla de lenguajes especificos del protocolo como SMTP, POP, IMAP, y NNTP. Con el API JavaMail se necesitan proveedores de protocolos para comunicarse con un servidor particular.

Sun Microsystems proporciona una implementación de referencia del API en forma binaria, que los desarrolladores pueden utilizar y distribuir. La implementación incluye los paquetes núcleo de JavaMail y los proveedores de servicio para IMAP, POP3 y SMTP.

JavaMail interactúa con el contenido de los mensajes a través del JAF (JavaBeans Activation Framework, Marco de Activación de JavaBeans), el cuál proporciona una forma uniforme de determinar el tipo de mensaje y de encapsularlo.

B2.2.1.2 Protocolos Relacionados

Los protocolos usados con el API son básicamente cuatro: SMTP, POP, IMAP y MIME, aunque también se encuentran el NNTP, entre otros. Aunque el API sea independiente del protocolo, no se pueden evitar las limitaciones que suponen los protocolos subyacentes.

- SMTP (Simple Mail Transfer Protocol, Protocolo de Transferencia de Correo Simple, está definido por la RFC 821): Define el mecanismo para enviar correos electrónicos. La aplicación que utiliza el API JavaMail se comunica con el servidor SMTP del proveedor de servicios (ISP), y éste se encarga de dejar el correo en el servidor SMTP del destinatario, para que sea posteriormente recogido por el destinatario a través de POP o IMAP.

- POP (Post Office Protocol, Protocolo de Oficina Postal, actualmente en la versión 3, también conocido como POP3 se define en el RFC 1939): Es un protocolo muy limitado que permite a los usuarios tener acceso a los correos que se encuentran en la bandeja de entrada (mailbox) de un servidor de correo electrónico. Si bien tiene menos capacidades que IMAP, POP se utiliza ampliamente. Los mensajes se suben al servidor de correo electrónico mediante SMTP y se descargan mediante POP.

- IMAP (Internet Message Access Protocol, Protocolo de Acceso a Mensajes en Internet, definido en el RFC 2060): IMAP es un protocolo más avanzado para accesar mensajes de correo electrónico almacenados en un servidor de correo. En otras palabras, le permite a un programa de correo electrónico "cliente" acceder a correos remotos como si fueran locales[12], siempre y cuando el servidor de correo soporte IMAP. A pesar de que IMAP presenta capacidades más avanzadas que POP, éste no se emplea tanto debido a que sobrecarga mucho el servidor de correo, requiriendo que el servidor reciba los nuevos mensajes, los entrege a los usuarios cuando sean solicitados, y los mantenga en las distintas carpetas de cada usuario. Con POP, los mensajes recuperados se eliminan del servidor de correo.

- MIME (Multipurpose Internet Mail Extension, Extensión multipropósito de Correo en Internet, Hay varios documentos relacionados: las RFC 822, RFC 2045, RFC 2046, y RFC 2047): Permite la transmisión y recepción de correo que contiene varios tipos de datos, tales como: texto, imágenes y video. Define el contenido de lo que se está transfiriendo: el formato de los mensajes, la información adjunta, etc.

- NNTP (Network News Transport Protocol, Protocolo de Transporte de Redes Nuevas) y Otros: el API JavaMail soporta fácilmente protocolos adicionales. Sun Mantiene una lista de proveedores de tercer órden que dan soporte a los protocolos que no han sido considerados por él. Ahí se encuentra soporte para NNTP, S/MIME (Secure MIME, MIME Seguro), entre otros.

B2.2.1.4 Cliente de correo electrónico para el servicio

La aplicación que se desarrolló con la funcionalidad del cliente de correo electrónico, se encarga de descargar los correos del servidor de correos y darles el formato necesario para

12 Mayor información en http://www.imap.org/

su tratamiento, así como de dar respuesta a las órdenes que recibe desde la central telefónica (específicamente desde el Cliente SPIRITS) a través de la clase encargada de la comunicación (relacionadas específicamente con el Servidor SPIRITS). Las órdenes que recibe son de dos tipos:

- Órdenes relacionadas con la administración del buzón de correo electrónico como: ir al primer correo, ir al último correo, ir al correo anterior, ir al correo siguiente, solicitar ayuda y obtener el número de correos disponibles.
- Órdenes relacionadas con la administración del correo electrónico como: leer correo, ampliar la información del correo, responder correo y borrar correo.

B2.3 IMPLEMENTACIÓN DE LA ARQUITECTURA SPIRITS

Para implementar la arquitectura SPIRITS, era necesario la construcción de los componentes funcionales que la componen y también la implementación del protocolo SPIRITS. Particularmente para éste último, era necesario conseguir una plataforma o API que diera la posibilidad de modificar el protocolo SIP (base del protocolo SPIRITS) para así cumplir con las especificaciones del protocolo. A continuación se relacionan algunas de éstas plataformas.

B2.3.1 JAIN SIP

JAIN SIP es una especificación para transacciones de propósito general para el protocolo SIP que se basa en interfaces Java. Es rico tanto en semántica como en la definición del protocolo SIP, y se desarrollo como una interfaz estándar para el protocolo que puede ser usado independientemente del nivel de programación y el entorno. JAIN SIP puede ser usado de multiples formas:

- Como una especificación para plataformas J2SE que permiten el desarrollo de aplicaciones “stand alone” de agentes de usuario, aplicaciones de registro y proxy.
- Una base para la implementación SIP para contenedores de SIP Servlets que permite el desarrollo de agentes de usuario, aplicaciones de registro y proxy dentro de un entorno basado en Servlets.
- Una base para la implementación SIP para contenedores de EJBs (Enterprise JavaBeans) que permiten el desarrollo de agentes de usuario, aplicaciones de registro y proxy en un entorno EJB.

JAIN SIP proporciona una implementación de referencia con una funcionalidad completa de la implementación SIP, la cuál puede ser útil para los desarrolladores que se pueden comunicar usando SIP desde el entorno Java. JAIN SIP es útil para la comunidad de desarrolladores que están familiarizados con el protocolo SIP y requieren control de transacciones sobre la implementación de SIP.

- Información General de la Arquitectura

JAIN SIP soporta la funcionalidad del RFC 3261 y las siguientes extensiones SIP, el método INFO (RFC 2979), la fiabilidad para proporcionar respuestas (RFC 3262), el marco para la

notificación de eventos (RFC 3265), el método UPDATE (RFC 3311), el encabezado Reason (RFC 3326) y el método Message (RFC 3428) definido para la mensajería instantánea.

JAIN SIP estandariza la interfaz para el modelo de transacciones genéricas definido por el protocolo SIP, proporciona acceso para la funcionalidad de diálogo desde la interfaz de transacciones. La arquitectura se desarrolla para el entorno J2SE por tanto se basa en eventos utilizando el modelo de eventos Listener / Provider. La especificación es asíncrona usando identificadores de transacciones para los mensajes correlacionados, también define varias clases "factory" para la creación de mensajes Request, Response y encabezados SIP.

JAIN SIP define una interfaz para cada encabezado que soporta, los cuáles se pueden adicionar a mensajes Request y Response. Estos mensajes se pasan al SipProvider con una transacción para ser enviados a traves de la red, mientras el SipListener escucha los eventos que llegan y que encapsulan los mensajes, los cuáles pueden ser respuesta a los diálogos iniciados o nuevos diálogos que comienzan.

El diseño JAIN SIP es extensible, definiendo una interfaz genérica de encabezados que se pueden usar para aplicaciones que utilicen encabezados que no son soportados directamente por JAIN SIP. El diseño también define mecanismos que soportan métodos de creación de diálogos futuros en un entorno JAIN SIP, usando las propiedades Java. JAIN SIP puede ser manejado estáticamente considerando las direcciones IP y la función de encaminamiento, y la especificación dinámica para puertos y metodo de transporte (udp, tcp).

El manejo por defecto de la retransmisión de mensajes en JAIN SIP es dependiente de la aplicación. Aplicaciones basadas en proxy Stateful no necesitan encargarse de las retransmisiones ya que estas son manejadas por JAIN SIP. Típicamente las aplicaciones de agentes de usuario deben manejar las retransmisiones de ACK's y las respuestas de tipo 2xx, que se encargan de informar cuando una solicitud ha sido recibido de manera exitosa pero esta pendiente de su tratamiento, o cuando la solicitud fue recibido de forma adecuada, comprendida y aceptada. Sin embargo JAIN SIP proporciona una función conveniente que asegura que todas las retransmisiones son manejadas por la implementación JAIN SIP, reduciendo la complejidad de las aplicaciones que actúen como agentes de usuario. Esta función puede también ser útil si JAIN SIP es utilizado como una base para la implementación de contenedores SIP Servlet o una implementación EJB.

En la Figura B4 se puede apreciar la Arquitectura de Mensajes SIP:

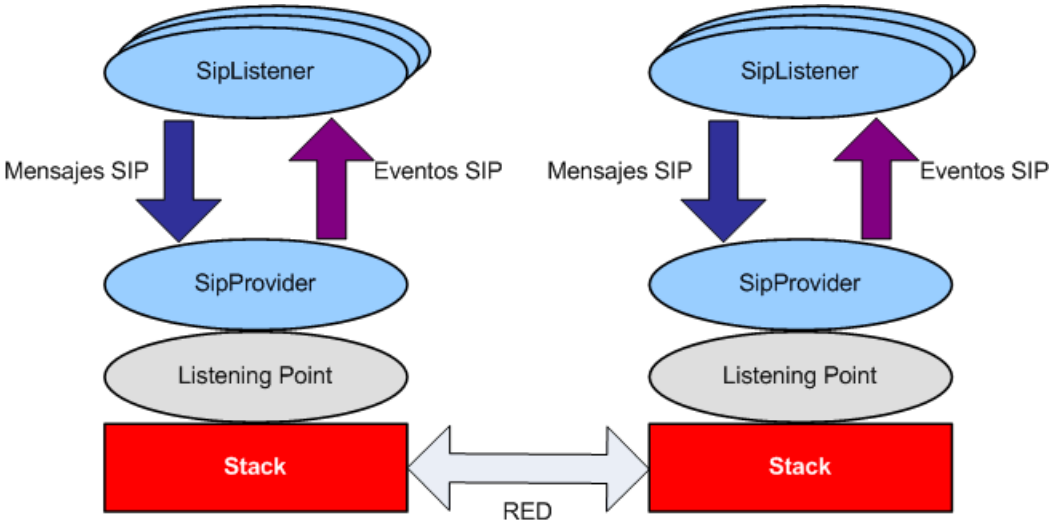

Figura B4 : Arquitectura de Mensajes SIP.

B2.3.2 SIP LITE

La especificación SIP Lite es una vista abstracta del protocolo SIP, que proporciona un entorno de programación SIP para desarrolladores que no conocen de SIP. La especificación del API está principalmente desarrollada para la plataforma J2SE, sin embargo como la especificación es realmente pequeña también se puede implementar en las plataformas J2ME. La motivación detrás de SIP Lite en la plataforma J2ME es proporcionar un modelo de objetos rico que pueda ser adecuado para dispositivos de tamaño mediano con mayor procesamiento y batería que los teléfonos móviles, como por ejemplo las PDA's y los teléfonos SIP. SIP Lite es muy común como base para agentes de usuarios o programas de interfaz para un teléfono SIP en la plataforma J2SE.

- Información General de la Arquitectura

SIP Lite soporta la funcionalidad definida en el RFC 3261. SIP Lite puede ser implementado en ambas plataformas la J2SE y la J2ME, por eso no es obligatorio el modelo de eventos Listener /Provider como en JAIN SIP, sin embargo, SIP Lite utiliza la convención de nombres Listerner / Provider. SIP Lite define una arquitectura de tres capas, donde el concepto de Listener existe para un dialogo (Dialog), un Call y un CallProvider. Estas tres interfaces en esencia escuchan los mensajes que llegan, los diálogos y las respectivas llamadas.

SIP Lite no define especificamente interfaces Request y Response; sino que son combinadas usando una sola interfaz Message. Los mensajes se identifican basándose en constantes de Request y Response definidas dentro de la especificación del API. Los mensajes son creados y envíados para un diálogo especifico, donde la respuesta se crea con base en un mensaje de Request especifico. Una petición es creada especificamente por un método, y con encabezados content type y content body. También se define una interfaz genérica para los encabezados SIP, que contiene los valores y parámetros SIP genéricos.

La arquitectura SIP Lite define el concepto de las interfaces Call y Dialog, dentro de la cual Call puede contener múltiples Dialogs. Éste no es un modelo de transacciones dentro de un Dialog; las transaciones son manejadas internamente por la implementación, oculto del desarrollador de la aplicación. SIP Lite define una sola clase factory la cual es usada para crear direcciones, usuarios, encabezados y parámetros, mientras CallProvider puede ser vista como una clase factory para Calls, y Call puede ser vista como una clase factory para los Dialogs.

El API SIP Lite es una especificación diseñada especificamente para aplicaciones de agentes de usuarios. Está no define o soporta ninguna capacidad de proxy, y siempre tendrá un comportamiento "statefully". Toda la semántica de retransmisión sera manejada por la implementación, ademas simplificando el entorno de programación para el desarrollador de aplicaciones.

B2.3.3 SIP Servlet

La especificación del API SIP Servlet define un entorno para la ejecución de aplicaciones de red basadas en SIP. Es implementado en un servidor de aplicaciones que soporta SIP y opcionalmente HTTP y la plataforma J2EE. Está construido en la especificación del API Servlet HTTP y como el Servlet HTTP se define un API y formato de archivos para los paquetes de aplicaciones. SIP Servlet soporta la base de SIP definida en el RFC 3261 y también soporta las siguientes extensiones, el entorno de notificación de eventos (RFC 3265) y el método de mensajes para mensajería instantánea. (RFC 3428)

En el centro del SIP Servlet da la capacidad a las aplicaciones para ejecutar la señalización SIP, o como un punto final (agente de usuario) o como un proxy SIP. La especificación de API pretende permitir a las aplicaciones el control completo sobre la señalización SIP, mientras al mismo tiempo esconde mucho de la complejidad no esencial de SIP, la cual no es relevante para los desarrolladores de aplicaciones.

La principal diferencia entre un no servidor de aplicaciones y un servidor de aplicaciones ambos basados en la especificación del API SIP es que el servidor de aplicaciones dentro de si mismo crea y maneja recursos, mientras que el se menciono de primero no lo hace. El contenedor de SIP Servlet administra recursos como puertos de escucha, los hilos, transacciones y dialogos, el estado de la sesión y los componentes de la aplicación.

- Información General de la Arquitectura

Las aplicaciones pueden actuar como un agente de usuario y como proxy. El proxy puede ser del tipo *stateless*[13] o *stateful*[14]. El contenedor SIP Servlet mantiene en un nivel inferior las transacciones SIP tal como el establecimiento de diálogos (Dialogs). Este no intenta proporcionar un API de señalización independientemente de SIP; la filosofía es que las

13 Los servidores proxy stateless, son los que procesan un mensaje SIP y entonces "olvidan" todo lo referente a la llamada hasta que vuelven a recibir otro mensaje SIP asociado a la misma. Esto se refiere al "estado" de la llamada, sin embargo, pueden mantener un "estado" para una simple transacción SIP, lo que es denominado "minimal state".

14 Los servidores proxy stateful retienen información de la llamada durante el tiempo que dure el establecimiento de esta.

aplicaciones deben frecuentemente necesitar acceder a especificos encabezado SIP. El SIP Servlet define un descriptor de implementación SIP y puede ser usado para la convergencia de aplicaciones, el cual es muy similar al despliegue del descriptor de implementaciones definido para la especificación del API HTTP Servlet. El rol más importante es declarar Servlets y definir un conjunto de reglas que controlan cuando la aplicación puede ser invocada. Las reglas son esencialmente atributos o cualidades sobre las peticiones SIP. Si una regla coincide con una petición, el contenedor puede invocar el Servlet correspondiente.

Como ultima parte en la descripción de plataformas Java para la implementación de plataforma SIP cabe mencionar la existencia de SIP for J2ME, que es la especificación que define una interfaz SIP para pequeñas plataformas, pero no es descripta en este documento pues no se acerca al API que se esta buscando

Después de estudiar estas plataformas se eligió trabajar con JAIN SIP por las siguientes razones:

- Proporciona una implementación de referencia con una funcionalidad completa de la implementación SIP.
- Estandariza la interfaz para el modelo de transacciones genéricas definidos por el protocolo SIP y proporciona acceso para la funcionalidad de diálogo desde la interfaz de transacciones.
- La arquitectura se desarrolla para el entorno J2SE por tanto se basa en eventos utilizando el modelo de eventos Listener / Provider, con el cuál se ha tenido experiencia en el desarrollo de aplicaciones anteriores.
- Define varias clases factory para la creación de mensajes Request, Response y encabezados SIP. Define un interfaz para cada encabezado que soporta, los cuáles pueden ser adicionados a mensajes Request, Response.
- El diseño es extensible, definiendo una interfaz genérica de encabezados que se puede usar para aplicaciones que utilicen encabezados que no son soportados directamente por JAIN SIP.

Las dos características anteriores permiten adaptar los encabezados de los mensajes según los requerimientos propios del protocolos SPIRITS.

- Se puede manejar estáticamente con almacenamiento de direcciones IP y la función de encaminamiento, y la especificación dinámica para puertos y transporte.

Esta última característica permite hacer una implementación punto a punto de la comunicación entre el cliente y servidor SPIRITS, evitando la implementación de servidores proxy, de redirección y registro, que hacen parte de la especificación SIP pero que no son necesarios en la construcción y validación de este prototipo de servicio.

Otra razón importante para escoger a Jain SIP es que al interior de la Universidad del Cauca se ha trabajado con una implementación de la plataforma, llamada NIST SIP, la cuál cuenta con una buena documentación sobre las clases e interfaces que componen el API, así como ejemplos de desarrollo.

Ahora es necesario elegir una implementación de esta plataforma que permita el desarrollo y se eligió NIST-SIP.

B2.3.4 NIST-SIP

La implementación NIST-SIP en su distribución 1.2 contiene las siguientes librerías:

- La implementación de referencia oficial del API JAIN SIP (JSR 32).
- El kit de compatibilidad tecnológica (TCK "Technology Compatibility Kit") para el JSR 32.
- Una implementación del API JAIN SDP[15] (JSR 141)

B2.3.4.1 Plataformas Soportadas

La especificación JAIN-SIP-1.2 necesita J2SE 1.4.x, siendo recomendable usar J2SE 1.4.2 o superior. Para adicionar soporte Swing es necesario instalar una librería adicional.

B2.3.4.2 Librerías y Grupo de Paquetes

En la Tabla B3 se listan los paquetes incluidos en la jerarquía gov.nist. y javax.sip que forman la implementación de referencia para JAIN-SIP 1.2

Paquete	Descripción
gov.nist.core	Contiene las clases núcleo, de las que dependen el resto de implementaciones.
gov.nist.core.net	Contiene las clases e interfaces de la red.
gov.nist.javax.sdp	Implementación anterior del paquete javax.sdp (ver JSR 141)
gov.nist.javax.sip	Esta es la raíz de la implementación JAIN de SIP.
gov.nist.javax.sip.address	Implementación del paquete address del API JAIN SIP.
gov.nist.javax.sip.header	Contiene la implementación de los encabezados SIP como se definen en JAIN-SIP-1.1, y una implementación de la "factory"[16] de encabezados JAIN-SIP.
gov.nist.javax.sip.parser	Traductores para los encabezados SIP, URL's y direcciones.
gov.nist.javax.sip.stack	Este paquete contiene las clases para la construcción de la pila SIP.
javax.sdp	Esta es una versión preliminar del paquete javax.sdp basado en la versión publica JSR141.
javax.sip	Este paquete contiene las principales interfaces del modelo de la arquitectura JAIN SIP desde el punto de vista del desarrollador de aplicaciones y del vendedor.

15 SDP: Session Description Protocol, Protocolo de Descripción de Sesión.
16 Se prefiere dejar el término original, para referirse a las "fábricas" de clases.

javax.sip.address	Este paquete contiene interfaces que representan el componentes de direccionamiento del protocolo SIP.
javax.sip.header	Este paquete contiene todas las interfaces de encabezados y extensiones soportados por esta especificación.
javax.sip.message	Este paquete contiene las interfaces que representan los mensajes SIP de Request y Response.

Tabla B3 : Paquetes JAIN-SIP 1.2

B2.3.4.3 "Factories"

Las "factories" que define JAIN SIP son cuatro, cada una con su respectiva responsabilidad, como se muestra en la Tabla B4:

Factory	Descripción
SipFactory	Esta interfaz define métodos para crear los nuevos objetos Stack y otros objetos factory.
AddressFactory	Esta interfaz define métodos para crear direcciones SIP y telefónicas.
HeaderFactory	Esta Interfaz define métodos para crear nuevos objetos Headers.
MessageFactory	Esta interfaz define métodos para crear nuevos objetos de mensajes SIP de Request y Response.

Tabla B4 : Factories de JAIN-SIP

B3. IMPLEMENTACIÓN DEL SISTEMA DE GESTIÓN

El desarrollo del Sistema de Gestión de configuración para servicios SPIRITS, da las facilidades de gestión para el *servicio de lectura de correo electrónico a través de la línea telefónica,* descrito anteriormente. En esta parte del documento se muestra la arquitectura que se escogió para la implementación del sistema de gestión, además de las tecnologías mediante las cuáles se desarrolló.

B3.1 ARQUITECTURA PARA EL SISTEMA DE GESTIÓN

Actualmente existe una nueva relación entre el operador/proveedor y el usuario de los servicios de telecomunicaciones, la cuál presenta dos características fundamentales:

- Una mayor participación del cliente (lo que facilita la obtención de una mayor satisfacción del mismo).
- Un aumento de la inversión del operador/proveedor en personal y en infraestructura para el soporte de las actividades de atención y asistencia al cliente (lo que implica mayores posibilidades de retención del cliente, permitiendo aumento de la base de clientes y por consiguiente reducción de las tarifas del servicio).

La implementación de este nuevo modelo comercial de los sistemas de gestión de servicios se basa en el logro de dos objetivos:

- Acceso Zero-Touch: Se busca eliminar o reducir la intervención de elementos intermedios como representantes de ventas, representantes de servicio al cliente (CSR´s), Call-Centers, etc. El cliente tiene acceso inmediato y directo al sistema de gestión.
- Venta One-Stop: Se pretende unificar el proceso de interacción con el cliente. El cliente puede obtener toda la información necesaria y realizar todas las funciones necesarias directamente en el sistema.

Al respecto, la interfaz Web tras su rápido despliegue en el mundo Internet, se ha revelado como un paradigma de interfaz de usuario gracias a sus características que la hacen ser amigable, intuitiva, independiente de la arquitectura y con una curva de aprendizaje rápida. Esta tecnología suele estar basada en el uso de lenguajes como Java y mecanismos de comunicación distribuida tales como DCOM[17], CORBA[18], RMI[19], o SOAP[20], que posibilitan que el usuario interactúe mediante clientes ligeros o páginas generadas dinámicamente, con servidores distribuidos de forma que se aprovechen los recursos eficientemente.

17 **DCOM:** Distributed Component Object Model, Modelo de Objetos de Componentes Distribuidos.
18 **CORBA:** Common Object Request Broker Architecture, Arquitectura Común de Intermediarios de Peticiones de Objetos.
19 **RMI:** Remote Method Invocation, Invocación de Métodos Remotos.
20 **SOAP:** Simple Object Adapter Protocol, Protocolo Simple de Adaptadores de Objetos

Es así como se decide implementar el sistema de gestión mediante el desarrollo de una aplicación Web.

B3.2 TECNOLOGÍAS PARA EL DESARROLLO DE APLICACIONES WEB

En cuanto a tecnologías para el desarrollo de aplicaciones web, las tres más fuertes son ASP, PHP y JSP, sobre las cuáles se profundizara muy poco porque son tecnologías ampliamente difundidas y al interior de la FIET son tratadas en algunas de las asignaturas que están dentro del plan de estudios.

De ASP se puede decir que es la tecnología desarrollada por Microsoft y su desarrollo esta basado en lenguajes como VBScript o el lenguaje especifico de ASP. Al ser comparada con JSP, esta última es más poderosa y ofrece una mejor solución para aplicaciones complejas que requieren componentes reutilizables, por último JSP es portable a otros sistemas operativos y servidores Web.

PHP se basa en un lenguaje de *scripting*[21], es una tecnología gratuita y su código puede ser distribuido libremente. Sin embargo la ventaja que tiene JSP es que la parte dinámica está escrita en Java, el cuál es un lenguaje ampliamente difundido que tiene una gran cantidad de APIs que soportan acceso a bases de datos, programación para funcionalidades de redes y objetos distribuidos.

Por las razones anteriores se ha decido desarrollar el sistema de gestión usando JSPs, para la parte del desarrollo de las interfaces de usuario o vista, Servlets para la parte de control, y Java Beans para la parte de modelo.

Otras de las ventajas de usar tecnologías como JSPs, Servlets y Java Beans son:

- Aunque hayan N peticiones a un Servlet, éste se carga una sola vez en memoria.
- El código al ser desarrollado en Java se puede reutilizar y tiene un amplio soporte para acceder a bases de datos, fijar encabezados HTTP, manejar cookies y sesiones, entre otras utilidades.
- Permiten la implementación de políticas de seguridad de las aplicaciones y los servidores en los cuales corren.

B3.2.1 Servlets

Es una tecnología Java, que permite escribir programas que corren en un servidor Web y actúan como una capa intermedia entre las peticiones que llegan del navegador Web u otro cliente HTTP y las bases de datos o aplicaciones que están en el servidor HTTP. Su trabajo es:

21 Tipo de lenguaje de computador que se escribe en texto plano y se interpreta usualmente comando a comando en el orden de aparición en el archivo.

- **Leer cualquier dato enviado por el usuario:** Estos son ingresados usualmente por medio de un formulario en una página Web, pero puede ser también por medio de un Applet Java o un programa cliente HTTP.

- **Buscar cualquier otra información sobre la petición que esta embebida en la petición HTTP**: Esta información incluye detalles sobre las capacidades del navegador, Cookies, el nombre del equipo cliente, entre otros.

- **Generar resultados**: Estos procesos pueden requerir interactuar con una base de datos, ejecutar llamadas CORBA o RMI, o invocar a aplicaciones heredadas.

- **Dar formato a los resultados dentro de un documento:** En la mayoría de los casos involucra embeber la información dentro de páginas HTML.

- **Fijar los parámetros apropiados de respuesta HTTP:** Esto significa reportar al navegador que tipo de documento ha sido retornado por ejemplo HTML, instalando cookies y almacenado parámetros.

- **Enviar el documento al cliente:** Este documento puede ser enviado en formato HTML, formato binario (imágenes en formato GIF) o un formato de archivo comprimido como zip, que es la capa mas alta de algún otro formato que se encuentra abajo.

Usualmente muchas de las peticiones de los clientes pueden ser satisfechas con documentos previamente construidos que son manejadas por el servidor sin la invocación de Servlets. Pero en muchos casos, estos resultados no son suficientes y las páginas necesitan ser generadas de acuerdo a la petición. Algunas razones de por qué las páginas deben ser construidas dinámicamente son:

- Las páginas Web se basan en los datos enviados por el usuario, por ejemplo las páginas de resultados de una búsqueda y las páginas de confirmación de órdenes de almacenes en linea.
- Las páginas Web se derivan de datos que cambian con frecuencia, por ejemplo las páginas del reporte del tiempo y las páginas de noticias.
- Las páginas que usan información desde bases de datos corporativas u otras fuentes dentro del servidor, por ejemplo los sitios de comercio electrónico, para mostrar las listas con los precios actuales y la disponibilidad de los productos.

B3.2.2 JSP (JavaServer Pages)

Esta tecnología permite que dentro de una página HTML exista contenido estático con contenido dinámico generado por los Servlets. Los JSP permiten crear dos partes separadas, una parte consiste en código HTML regular, el cuál se envía al navegador del cliente sin ser cambiado. Otra parte se genera dinámicamente y se denota con etiquetas[22] especiales similares al código HTML que se introducen correctamente en la página. Así JSP permite a los desarrolladores generar dinámicamente HTML, XML o algún otro tipo de

22 Etiquetas es la traducción que se dio al término ingles “tags”, en otros documentos se traduce también como marcas.

páginas Web.

Los JSP son una extensión de la tecnología de los Servlet y es común el uso de ambas tecnologías en una misma aplicación Web.

B3.2.3 Java Beans

Son componentes *software* escritos en java, que se definen como componentes de *software* reutilizables

Para su implementación la clase debe cumplir con ciertas reglas como:

- La clase debe implementar la interfaz *Serializable*.
- Debe soportar un constructor por defecto sin argumentos.
- Sus propiedades se deben acceder usando métodos *get* y *set,* y seguir la convención de nombres estándar.
- Debe soporta introspección, así el IDE podría analizar el Bean.

B4. BIBLIOGRAFÍA

Sobre Tecnologías de Voz

BLACK, Alan W; TAYLOR, Paul and CALEY, Richard. The Festival Speech Synthesis System documentation. Edition 1.4, for Festival Version 1.4.017. Junio de 1999. Disponible en Internet: http://www.cstr.ed.ac.uk/projects/festival/manual/festival_toc.html

BRAÑA GUNDIN, Lluis Eloi. VUI (Voice User Interfaces) y VoiceXML. Curso de doctorado Avances en tecnologías Web. Mayo 2004.

CORREA CORNEJO, Deusdit A. La nueva era de la voz. Disponible en Internet: http://www.informatizate.net/articulos/la_nueva_era_de_la_voz_parte_02_12052004.html

Sobre el Desarrollo del Cliente de Correo Electrónico

API JavaMail. Sun. Traducción de Juan Antonio Palos. Disponible en Internet: http://www.programacion.com/java/tutorial/javamail/

Documentación del API JavaMail. Sun. Disponible en Internet: http://java.sun.com/products/javamail/javadocs/index.html

FAQ API JavaMail. Sun. Disponible en Internet: http://java.sun.com/products/javamail/FAQ.html

Sobre las Plataformas para la Implementación de SIP

Documentación sobre librerias y paquetes de NIST-SIP. Disponible en Internet: http://snad.ncsl.nist.gov/proj/iptel/src/nist-sip/jain-sip/docs/api/index.html

O'DOHERTY, Phelim. SIP Specifications and the Java Platforms, the look and feel of SIP. Sun Microsystems. 9 de Enero 2003. Disponible en Internet: www.cs.columbia.edu/sip/Java-SIP-Specifications.pdf

O'DOHERTY, Phelim; RANGANATHAN, Mudumbi. JAIN SIP Tutorial. Sun Microsystems, NIST. Disponible en Internet: http://java.sun.com/products/jain/JAIN-SIP-Tutorial.pdf

Página principal del Proyecto NIST-SIP: http://snad.ncsl.nist.gov/proj/iptel/

Sobre Servlets y JSP

Prentice Hall and Sun Microsystems. Overview of Servlets and JavaServer Pages. 2002. . Disponible en Internet: http://pdf.coreservlets.com/CSAJSP-Chapter1.pdf

CAICEDO, Oscar Mauricio. Presentación sobre Java Beans para la asignatura Aplicaciones y Servicios Telemáticos. Universidad del Cauca. 2004